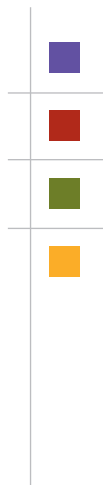




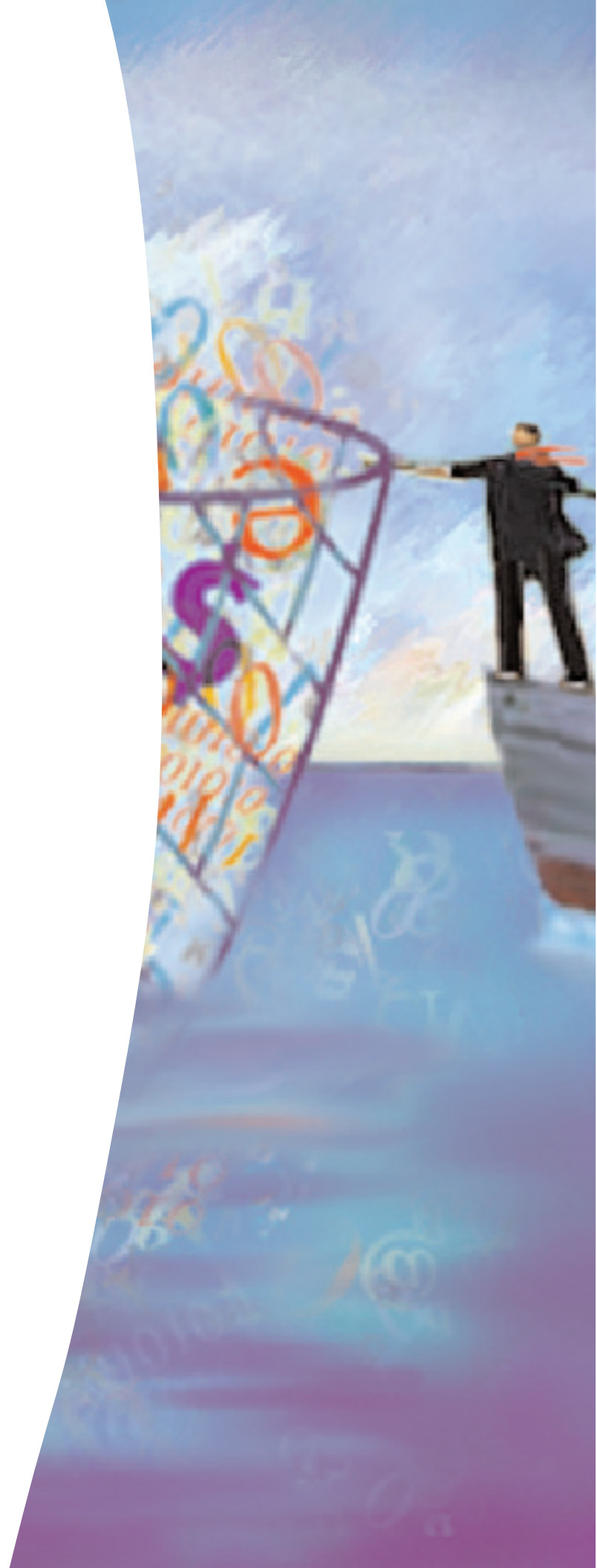
SAS Publishing

# What's New in SAS<sup>®</sup>, Release 8.1 and Release 8.2

- How to Use This CD-ROM
- License Agreement
- Table of Contents



*The Power to Know<sup>™</sup>*



## How to Use This CD-ROM

The ***What's New*** documents are available from the **Table of Contents**. Click to open any item on the list.

Each document includes a set of bookmarks, including a table of contents, to help you locate topics.

Use the Acrobat Reader navigation pane to access the document bookmarks. If the pane is not visible, click the navigation button (  ) on the Acrobat Reader menu bar.

To return to the ***What's New*** table of contents, click the first page button (  ) on the Acrobat Reader menu bar.

## How to Print from This CD-ROM

Please use the file pull-down menu and select Print ... to control the number of pages to print or to print an individual book, use the ***book\_files*** folder on this CD.

**Note:** Beware if you click on the print icon, you may generate a printout of 1784 pages.

## How to Load This CD-ROM to Your Intranet

***What's New*** can be supported on most Web servers to be served to your user community via your company's intranet.

Copy the ***What's New in SAS, Release 8.1 and Release 8.2, February 2001, PDF Format CD*** in a location known by your Web server.

Start the Web browser and point to:

`http://<Web domain>/<install root>/sasdoc/saspdf/common/mainpdf.htm`

In the above address, <Web domain> is your site's designated intranet domain and <install root> is the Web-served location where you have installed ***What's New***.

# What's New in SAS® Documentation License Agreement

**Important:** Read the following terms and conditions carefully before using What's New in SAS Documentation (the "Documentation"). Using the Documentation (or authorizing any other person to do so) indicates Licensee's acceptance of these terms and conditions. If Licensee does not agree to the terms and conditions of this License Agreement, return the Documentation to SAS Institute Inc. If Licensee paid for its copy of the Documentation, Licensee will be entitled to a full refund of that payment by returning proof of payment for the copy of the Documentation within thirty (30) days from the date of purchase. If Licensee received its copy of the Documentation at no charge as part of the annual license for SAS Software, then Licensee will not be entitled to a refund because no payment was made for the Documentation.

## 1. License

In this license agreement ("Agreement"), you, the purchaser and end-user of the license rights granted by this Agreement, are referred to as "Licensee". Subject to the terms and conditions set forth below, SAS Institute Inc. ("SAS") hereby grants to Licensee a non-exclusive, non-transferable and non-assignable license to use the Documentation.

## 2. Scope of Use

The use of and access to the Documentation shall be made only by persons authorized to use the related SAS software ("Users"). Licensee shall use the Documentation only for its own internal business use. SAS provides a single copy of the Documentation to Licensee. Licensee may: (1) copy and use the Documentation in the same manner and to the same extent as Licensee is authorized to copy and use the related SAS software and/or (2) copy the Documentation to a single hard disk of computer hardware that is owned or controlled by Licensee for use over the Licensee's internal network, and/or (3) copy the Documentation to an internal Web server, and/or (4) copy the Documentation to an external Web server provided that access is restricted to Users only. Licensee must retain the original media containing the Documentation during the term of this Agreement. Licensee may not duplicate the original media containing the Documentation except as necessary for backup purposes. Licensee shall not modify the Documentation. In the event of any modification, this Agreement shall be deemed terminated and all warranties and other obligations of SAS shall be deemed revoked by SAS and waived by Licensee.

## 3. Printing

Licensee may print portions or all of the Documentation. Users may duplicate such printed Documentation only for internal use. Licensee agrees to ensure that all printed or duplicated copies of the Documentation contain SAS' copyright notices. Licensee and Users may not sell such printed or duplicated copies of the Documentation.

## 4. Term

Licensee may terminate this Agreement at any time. SAS may terminate this Agreement if Licensee breaches any of the terms and conditions of this Agreement or breaches any of the terms and conditions of its software license for SAS software. Upon termination of this Agreement for any reason, Licensee shall destroy all copies of the Documentation and return the original copy of the Documentation to SAS. All provisions of this Agreement relating to confidentiality, proprietary rights, disclaimers of warranties, and limitation of liability, remedies or damages shall survive termination.

## 5. Proprietary Rights

The Documentation is copyrighted. Title to the Documentation shall be the sole and exclusive property of SAS or its licensors. Licensee shall not sell, transfer, publish (except as specifically set forth herein), rent, lease, loan or otherwise make available any portion of SAS OnlineDoc to others or export (except as specifically set forth herein) the Documentation. Licensee agrees to secure and protect the Documentation in a manner consistent with the maintenance of SAS' rights therein and to take appropriate action by instruction or agreement with all persons who are permitted access to the Documentation to satisfy its obligations hereunder.

## 6. Limited Warranty and Disclaimer

SAS warrants that the media on which the Documentation is furnished will be free from defects in material and workmanship under normal use for a period of ninety (90) days from the date of delivery of the Documentation. Licensee's exclusive remedy for breach of this warranty shall be replacement of the defective media by SAS. **SAS and its licensors disclaim all other warranties, express or implied, including, but not limited to, any implied warranties of merchantability and/or fitness for a particular purpose whether alleged to arise by law, by reason of custom or usage in the trade, or by course of dealing. Because some jurisdictions do not allow the waiver or exclusion of implied warranties, the limitations in this Section 6 may not be wholly enforceable; however, they are enforceable to the greatest extent permitted by law.**

## 7. Limitation of Liability

**SAS and its licensors shall have no liability with respect to their obligations under this Agreement or otherwise for special, incidental, consequential, punitive or exemplary damages even if advised of the possibility of such damages. In no event shall SAS' and its licensor's liability for all reasons and upon all causes of action in the aggregate exceed the lesser of (1) the fees paid by Licensee to SAS under this Agreement or (2) the then-current retail price of the Documentation. Because some jurisdictions do not allow the limitation of liability for incidental or consequential damages, the limitation in this Section 7 may not be wholly enforceable; however, they are enforceable to the greatest extent permitted by law.**

## 8. Government Use

Use, duplication or disclosure of the Documentation by the U.S. Government is subject to this license and the restrictions as set forth in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June 1987).

## 9. Export Restrictions

In addition to the restrictions set forth elsewhere in this Agreement, Licensee may not download or otherwise export or reexport the Documentation or any underlying information or technology except in full compliance with all laws and regulations of the United States of America and in full compliance with any other applicable laws and regulations. By accepting this Agreement and installing the Documentation, Licensee is agreeing to the foregoing. Licensee also affirms that Licensee is a lawfully admitted permanent resident of and located in a country to which the United States permits SAS to send the Documentation. Licensee further affirms that Licensee is a party to whom the United States allows SAS to license the Documentation, and Licensee is not on any United States Government list of parties to whom SAS may not license the Documentation, and Licensee will not allow any such party to access the Documentation. The current list of parties to whom SAS may not license the Documentation includes but is not limited to parties who are involved in terrorism or who will use the Documentation for design, development, or production of nuclear, chemical, or biological weapons, or missiles.

## 10. General

This Agreement and any invoices relating to the Documentation constitute the complete agreement between Licensee and SAS about the Documentation and supersede any purchase order, communications, or representations regarding the Documentation. If any provision of this Agreement shall be unenforceable, then that provision shall be deemed severable from this Agreement and shall not affect the enforceability of the remaining provisions. Any additions or modifications must be approved in writing signed by both parties. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement. This Agreement is governed by the laws of the United States of America and the State of North Carolina. Breach of SAS' intellectual property rights will lead to damages not adequately remedied by an award of money; therefore, the parties agree that SAS may protect those intellectual property rights through temporary restraining orders or injunctions, without the obligation of posting bond.



## **What's New in SAS Software, Release 8.1**

**SAS/ETS Software: Changes and Enhancements for Release 8.1**

**SAS/IML Software: Changes and Enhancements for Release 8.1**

**SAS/QC Software: Changes and Enhancements for Release 8.1**

**SAS/OR Software: Changes and Enhancements for Release 8.1**

**SAS/STAT Software: Changes and Enhancements for Release 8.1**

## **What's New in SAS Software, Release 8.2**

**SAS/ASSIST Software: Changes and Enhancements for Release 8.2**

**SAS/ETS Software: Changes and Enhancements for Release 8.2**

**SAS/IML Software: Changes and Enhancements for Release 8.2**

**SAS/QC Software: Changes and Enhancements for Release 8.2**

**SAS/OR Software: Changes and Enhancements for Release 8.2**

**SAS/STAT Software: Changes and Enhancements for Release 8.2**

# **What's New in SAS<sup>®</sup> Software for Release 8.1**

---

*The Power to Know<sup>™</sup>*



SAS Publishing

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *What's New in SAS® Software for Release 8.1*, Cary, NC: SAS Institute Inc., 2000.

**What's New in SAS® Software for Release 8.1**

Copyright © 2000 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-661-9

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, April 2000

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

IBM® and all other International Business Machines Corporation product or service names are registered trademarks or trademarks of International Business Machines Corporation in the USA and other countries.

Oracle® and all other Oracle Corporation product or service names are registered trademarks or trademarks of Oracle Corporation in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.

---

# Contents

Important Information About This Book ix

## **PART 1   Base SAS Software   1**

### **Chapter 1 ▲ Base SAS Software: General Features and Language Elements   3**

SAS Output: Output Delivery System   3

SAS I/O: Holding a SAS Data Set in Memory   4

SAS I/O: Generating and Processing XML Documents   4

SAS I/O: Improving I/O Performance for Access Methods with the BUFSIZE= Option   4

SAS I/O: Applying OBS= and FIRSTOBS= to WHERE Processing   5

SAS I/O: New Referential Action and Message Options for Integrity Constraints   5

SAS I/O: Locking a SAS File   5

Universal Printing   5

Windowing Environment: My Favorite Folders Window Commands   5

Windowing Environment: File Shortcut Assignment Window   6

Windowing Environment: Import/Export Wizard   6

Data Set Options   6

Formats   6

Functions and CALL Routines   6

Statements   7

System Options   7

### **Chapter 2 ▲ Base SAS Software: Procedures   9**

The CIMPORT Procedure   9

The COPY Procedure   9

The CPORT Procedure   9

The DATASETS Procedure   9

The EXPORT Procedure   10

The FREQ Procedure   10

The IMPORT Procedure   10

The PRINT Procedure   10

The REPORT Procedure   11

The SORT Procedure   11

The SQL Procedure   11

The UNIVARIATE Procedure   11

### **Chapter 3 ▲ SAS Software in the CMS Environment   13**

Statements   13

System Options   13

**Chapter 4 ▲ SAS Software in the Microsoft Windows Environment 15**

Graphical User Interface 15

Application Integration 15

SAS Language Elements 15

Enhancements for Running SAS under Windows NT and Windows 2000 16

**Chapter 5 ▲ SAS Software in the OpenVMS Environment 17**

Transport Files 17

SAS Language Elements 17

SAS Session Manager 18

**Chapter 6 ▲ SAS Software in the OS/2 Environment 19**

Functions and CALL Routines 19

System Options 19

**Chapter 7 ▲ SAS Software in the OS/390 Environment 21**

Statements 21

System Options 21

**Chapter 8 ▲ SAS Software in UNIX Environments 23**

Call Routines 23

Commands 23

System Options 23

**PART 2 Other SAS Software Products 25****Chapter 9 ▲ SAS/ACCESS Software 27**

All SAS/ACCESS Relational Interfaces 27

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts 27

SAS/ACCESS Interface to DB2 under VM (formerly SQL/DS) 27

SAS/ACCESS Interface to Microsoft SQL Server 27

SAS/ACCESS Interface to Oracle Rdb 28

SAS/ACCESS Interface to ODBC 28

SAS/ACCESS Interface to OLE DB 28

SAS/ACCESS Interface to ORACLE 28

SAS/ACCESS Interface to Teradata 28

SAS/ACCESS Interface to ADABAS (available on OS/390) 29

SAS/ACCESS Interface to CA-DATACOM/DB (available on OS/390) 29

SAS/ACCESS Software for PC File Formats 29

SAS/ACCESS Interface to Baan 29

SAS/ACCESS Interface to R/3 29

**Chapter 10 ▲ SAS/AF Software 31**

New Classes 31

**Chapter 11 ▲ SAS Component Language 33**

New Functions 33

Enhanced Functions 33

**Chapter 12 ▲ SAS/CONNECT Software 35**

Firewall Configuration 35

System Options 35

**Chapter 13 ▲ SAS/EIS Software 37**

General Functionality 37

**Chapter 14 ▲ SAS/ETS Software 39**

Investment Analysis System 39

SASECRSP Engine 39

QLIM Procedure 40

VARMAX Procedure 40

X12 Procedure 40

For Additional Information 40

**Chapter 15 ▲ SAS/GIS Software 41**

New Tutorial 41

PROC GIS Statement 41

Displayed Maps 41

Data Administration Window 42

**Chapter 16 ▲ SAS/GRAPH Software 43**

Using Image Files in SAS/GRAPH 43

FONTLIST Command 43

All Procedures 43

GCHART Procedures 43

GPLOT Procedure 44

GRADAR Procedure 44

SYMBOL Statement 44

**Chapter 17 ▲ SAS/IML Software 45**

Financial Functions 45

Robust Regression 45

Multivariate Time Series Analysis 45

**Chapter 18 ▲ SAS/INSIGHT Software 47**

Graphical Enhancements 47

Distribution Analyses 47

Fit Analyses 47

Multivariate Analyses 47

**Chapter 19 ▲ SAS Integration Technologies 49**

New Features 49

For Additional Information 49

**Chapter 20 ▲ SAS/IntrNet Software 51**

New Features and Enhancements in Application Dispatcher 51

New Features and Enhancements for SAS Java Tools 51

**Chapter 21 ▲ SAS OLAP Server Software 53**

Product Features 53

**Chapter 22 ▲ SAS/OR Software 55**

BOM Procedure 55

CPM Procedure 55

DTREE Procedure 55

GANTT Procedure 56

INTPOINT Procedure 56

NETDRAW Procedure 56

NETFLOW Procedure 56

PM Procedure 56

For Additional Information 56

**Chapter 23 ▲ SAS/QC Software 57**

SHEWHART Procedure 57

ADX Interface 57

**Chapter 24 ▲ SAS/SHARE Software 59**

Cross-Platform Catalog Access 59

Overlapping I/O 59

SAS/SHARE Data Provider 59

**Chapter 25 ▲ SAS/STAT Software 61**

CATMOD Procedure 61

FACTOR Procedure 61

FREQ Procedure 62

GAM Procedure 62

LOESS Procedure 62

LOGISTIC Procedure 63

MI Procedure 63

MIANALYZE Procedure 63

MIXED Procedure 64

MODECLUS Procedure 64

MULTTEST Procedure 64

NLMIXED Procedure 64

PHREG Procedure 64

SURVEYMEANS Procedure 65

For Additional Information 65

References 65

## **Chapter 26 ▲ SAS/Warehouse Administrator Software 67**

New Tutorial 67

Faster Loading of DBMS Tables 67

User Interface Enhancements 67

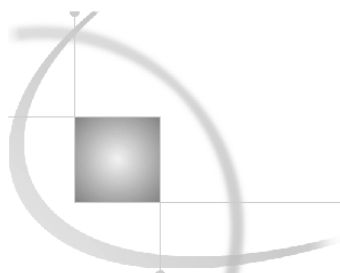
Metadata API Enhancements 68

Experimental Software: LDAP Security Interface 69

**Index 71**







## Important Information About This Book

---

*Where to Find More Information About New and Enhanced Features* ix  
*Get the Latest News* ix

---

### Where to Find More Information About New and Enhanced Features

This book gives you a quick, convenient overview of features that have been added or enhanced in base SAS software and in 17 other SAS software products for Release 8.1.

For additional details, see the SAS Help System for Release 8.1 and the following books:

- ❑ *SAS/ETS Software: Changes and Enhancements, Release 8.1*
- ❑ *SAS/IML Software: Changes and Enhancements, Release 8.1*
- ❑ *SAS/OR Software: Changes and Enhancements, Release 8.1*
- ❑ *SAS/QC Software: Changes and Enhancements, Release 8.1*
- ❑ *SAS/STAT Software: Changes and Enhancements, Release 8.1*

Note that the SAS/IML and SAS/QC documentation is available only as PDF files. See the SAS Publishing Web site ([www.sas.com/pubs](http://www.sas.com/pubs)) for more information.

---

### Get the Latest News

To stay up-to-date, you can subscribe to NEWDOCNEWS, a monthly electronic newsletter. This newsletter provides SAS software users with information about new books that were published during the previous month, notice of new Observations articles on our Web page, announcements of online documentation, notice of documentation updates, and general news items about SAS documentation.

Customers in the USA can subscribe to this electronic newsletter as follows:

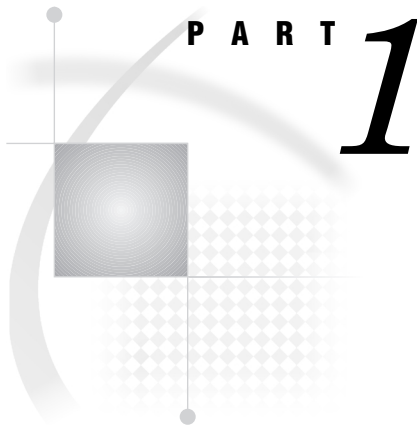
- 1 Send an e-mail message to `listserv@vm.sas.com`.
- 2 Leave the Subject line blank.
- 3 The text of the message should be `Subscribe NEWDOCNEWS-L John Doe` (replace John Doe with your name).
- 4 Send the message.

You will receive a message confirming that you have been added to the NEWDOCNEWS mailing list. The message will also contain instructions on how to unsubscribe from the mailing list, in case you choose to do so in the future.

International customers can subscribe to the newsletter as follows:

- 1** Send an e-mail message to **pubs@unx.sas.com**.
- 2** The subject line should be **Add to NEWDOCNEWS**.
- 3** The text of the message should be **Subscribe NEWDOCNEWS John Doe** (replace John Doe with your name).
- 4** Send the message.

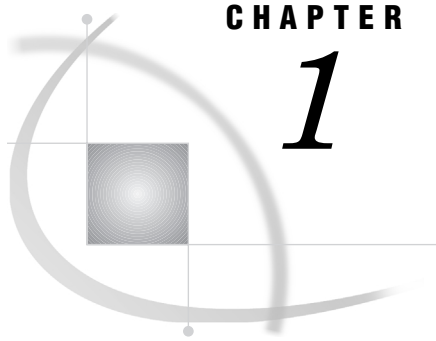
We will forward your name and e-mail address to your local SAS office, and they will add you to the mailing list for their electronic newsletter.



## **Base SAS Software**

|                        |                                                                  |           |
|------------------------|------------------------------------------------------------------|-----------|
| <i>Chapter 1</i> ..... | <b>Base SAS Software: General Features and Language Elements</b> | <b>3</b>  |
| <i>Chapter 2</i> ..... | <b>Base SAS Software: Procedures</b>                             | <b>9</b>  |
| <i>Chapter 3</i> ..... | <b>SAS Software in the CMS Environment</b>                       | <b>13</b> |
| <i>Chapter 4</i> ..... | <b>SAS Software in the Microsoft Windows Environment</b>         | <b>15</b> |
| <i>Chapter 5</i> ..... | <b>SAS Software in the OpenVMS Environment</b>                   | <b>17</b> |
| <i>Chapter 6</i> ..... | <b>SAS Software in the OS/2 Environment</b>                      | <b>19</b> |
| <i>Chapter 7</i> ..... | <b>SAS Software in the OS/390 Environment</b>                    | <b>21</b> |
| <i>Chapter 8</i> ..... | <b>SAS Software in UNIX Environments</b>                         | <b>23</b> |





## CHAPTER

## 1

# Base SAS Software: General Features and Language Elements

|                                                                                       |   |
|---------------------------------------------------------------------------------------|---|
| <i>SAS Output: Output Delivery System</i>                                             | 3 |
| <i>SAS I/O: Holding a SAS Data Set in Memory</i>                                      | 4 |
| <i>SAS I/O: Generating and Processing XML Documents</i>                               | 4 |
| <i>SAS I/O: Improving I/O Performance for Access Methods with the BUFSIZE= Option</i> | 4 |
| <i>SAS I/O: Applying OBS= and FIRSTOBS= to WHERE Processing</i>                       | 5 |
| <i>SAS I/O: New Referential Action and Message Options for Integrity Constraints</i>  | 5 |
| <i>SAS I/O: Locking a SAS File</i>                                                    | 5 |
| <i>Universal Printing</i>                                                             | 5 |
| <i>Windowing Environment: My Favorite Folders Window Commands</i>                     | 5 |
| <i>Windowing Environment: File Shortcut Assignment Window</i>                         | 6 |
| <i>Windowing Environment: Import/Export Wizard</i>                                    | 6 |
| <i>Data Set Options</i>                                                               | 6 |
| <i>Formats</i>                                                                        | 6 |
| <i>Functions and CALL Routines</i>                                                    | 6 |
| <i>Statements</i>                                                                     | 7 |
| <i>System Options</i>                                                                 | 7 |

## SAS Output: Output Delivery System

- Two new ODS statements are now available:

### ODS RTF

opens, manages, or closes the RTF destination. If the destination is open, you can create SAS output for Microsoft Word.

### ODS PROCTITLE

specifies whether to suppress the title that indicates which procedure produced the results.

- Aliases have been added to the following ODS statements:

| Statement       | Alias               |
|-----------------|---------------------|
| ODS VERIFY ON   | ODS VERIFY          |
| ODS VERIFY OFF  | ODS NOVERIFY        |
| ODS EXCLUDE ALL | ODS EXCLUDE DEFAULT |
| ODS SELECT ALL  | ODS SELECT DEFAULT  |

- The ODS PRINTER statement supports the following new options:

ANCHOR=

specifies the base name for the HTML anchor tag that identifies each output object in the current file.

AUTHOR=

writes the name that you specify to the header of an HTML file.

BASE=

specifies a string to use as the first part of all links and references that ODS creates in the HTML file.

KEYWORDS="*item(s)*"

enables your program to produce output based on the value of *item*.

PDFMARK

enables a program to insert special tags into a PostScript file.

PDFNOTE

enables you to add comments to a PDF file.

SUBJECT=

writes the subject that you specify to the header of an RTF file.

TITLE=

writes the title that you specify to the header of an RTF file.

- The ODS HTML statement supports a new option called **STYLESHEET**, which specifies the use of cascading style sheets.
- The **SOURCE** statement in **PROC TEMPLATE** supports a new option called **NOFOLLOW**, which specifies that the program not resolve links in the **PARENT=** option.
- The Results window now supports the **EXPAND ALL** and **COLLAPSE ALL** commands.

---

## SAS I/O: Holding a SAS Data Set in Memory

The new global **SASFILE** statement requests that a SAS data set be opened and that enough buffers be allocated to hold the entire data set in memory. Once read, data is held in memory, where it is available to subsequent **DATA** and **PROC** steps until a second **SASFILE** statement closes it and frees the buffers. The **SASFILE** statement can improve performance by taking advantage of large amounts of memory.

---

## SAS I/O: Generating and Processing XML Documents

The new XML engine generates and processes an XML document, which is an application- and machine-independent file. You invoke the engine by specifying the engine name **XML** in the **LIBNAME** statement.

---

## SAS I/O: Improving I/O Performance for Access Methods with the BUFSIZE= Option

To optimize the sequential access method, SAS now sets the default value for the **BUFSIZE=** system option and the **BUFSIZE=** data set option based on the operating environment.

---

## SAS I/O: Applying OBS= and FIRSTOBS= to WHERE Processing

SAS can now process a segment of data that is conditionally selected with a WHERE expression by applying OBS= and FIRSTOBS= processing. Processing only a segment of the subset of data can improve performance substantially.

---

## SAS I/O: New Referential Action and Message Options for Integrity Constraints

- A new referential action for foreign key integrity constraints, CASCADE, enables the data values in a primary key integrity constraint to be updated, and also updates matching values in the foreign key data file to the same values.
- The MESSAGE= option, which was introduced for the PROC DATASETS IC CREATE statement in Version 8, is now supported in PROC SQL. The MESSAGE= option enables you to associate a user-defined message with the SAS error message that is displayed for an integrity constraint.
- When you use the MESSAGE= option, a new MSGTYPE= option enables you to control the format of the integrity-constraint error message.

---

## SAS I/O: Locking a SAS File

You can issue the new LOCK statement to acquire an exclusive lock on an existing SAS file.

---

## Universal Printing

Universal Printing functionality is a new way of providing consistent printing support for all operating environments. It enables you to print or preview the contents of an active SAS window. You can produce output in either PostScript or PCL format.

Universal Printing is the default printing mechanism in the UNIX and OpenVMS operating environments. You can use the UNIVERSALPRINT system option to specify Universal Printing in other operating environments.

---

## Windowing Environment: My Favorite Folders Window Commands

You can use the following new commands to work with files in the My Favorite Folders window:

- COPYITEM copies files that appear in the My Favorite Folders window.
- MOVEITEM moves files that appear in the My Favorite Folders window.
- DELETEITEM deletes files that appear in the My Favorite Folders window.
- RENAMEITEM renames files that appear in the My Favorite Folders window.



---

## Windowing Environment: File Shortcut Assignment Window

The new File Shortcut Assignment window enables you to assign file shortcuts (filerefs) graphically. You can also use this window to make file shortcuts permanent, meaning that they will be reassigned during SAS startup until they are deassigned. The file shortcuts that you create with this window are added to the File Shortcuts folder of the SAS Explorer window.

To access this window, issue the DMFILEEASSIGN command.

---

## Windowing Environment: Import/Export Wizard

Depending on the operating environment, the Import/Export Wizard supports the following additional external data sources:

- Microsoft Access 2000 table
- Microsoft Excel Version 2000 spreadsheet.

For details, see the online Help that is available from the Wizard. To access the Wizard, select **File**, then either **Import Data** or **Export Data**.

---

## Data Set Options

- The default value for the BUFSIZE= data set option is now set by the operating environment to optimize sequential access.
- The OBS= and FIRSTOBS= data set options can now be used with a WHERE expression, which is supported for the WHERE statement, the WHERE= data set option, and the WHERE clause in the SQL procedure.

---

## Formats

- The new DTDATEw. format produces the same output as DATEw., but accepts a datetime value as input.

---

## Functions and CALL Routines

The following new functions and CALL routines have been implemented:

- The CALL SLEEP routine and the SLEEP function suspend the execution of a SAS DATA step for a specified period of time and return a value.
- The CALL STREAMINIT routine (Experimental) specifies a seed value to use for subsequent random number generation by the RAND function.
- The RAND function (Experimental) generates random numbers from a specified distribution.

---

## Statements

- The new SASFILE global statement requests that a SAS data set be opened and that enough buffers be allocated to hold the entire data set in memory.
- The WHERE statement can now be used with OBS= and/or FIRSTOBS= processing.
- The new LOCK statement acquires an exclusive lock on an existing SAS file.

---

## System Options

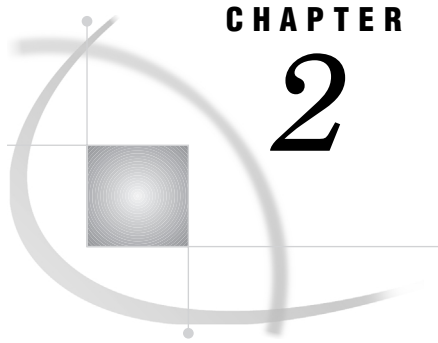
The following SAS system options have been added:

- The new EMAILHOST= system option specifies the Simple Mail Transport Protocol (SMTP) server that supports email access for your site.
- The new EMAILPORT= system option specifies the port to which the Simple Mail Transport Protocol (SMTP) server is attached.
- The new UNIVERSALPRINT system option (Experimental in CMS, MVS, OS/2, and Windows operating environments), turns Universal Printing on for an operating environment.

The following SAS system options have been enhanced:

- The default value for the BUFSIZE= system option is now set by the operating environment to optimize sequential access.
- The OBS= and FIRSTOBS= system options can now be used with a WHERE expression, which is supported for the WHERE statement, the WHERE= data set option, and the WHERE clause in the SQL procedure.





## CHAPTER

## 2

# Base SAS Software: Procedures

---

|                                 |    |
|---------------------------------|----|
| <i>The CIMPORT Procedure</i>    | 9  |
| <i>The COPY Procedure</i>       | 9  |
| <i>The CPORT Procedure</i>      | 9  |
| <i>The DATASETS Procedure</i>   | 9  |
| <i>The EXPORT Procedure</i>     | 10 |
| <i>The FREQ Procedure</i>       | 10 |
| <i>The IMPORT Procedure</i>     | 10 |
| <i>The PRINT Procedure</i>      | 10 |
| <i>The REPORT Procedure</i>     | 11 |
| <i>The SORT Procedure</i>       | 11 |
| <i>The SQL Procedure</i>        | 11 |
| <i>The UNIVARIATE Procedure</i> | 11 |

---

## The CIMPORT Procedure

The new DATECOPY option copies the date and time that a SAS file was created and the date and time a SAS file was last modified when importing the file from transport format.

---

## The COPY Procedure

The new DATECOPY option copies the date and time that a SAS file was created and the date and time a SAS file was last modified when creating a copy of the file.

---

## The CPORT Procedure

The new DATECOPY option copies the date and time that a SAS file was created and the date and time a SAS file was last modified when exporting the file to transport format.

---

## The DATASETS Procedure

- The IC CREATE statement supports a new referential action for foreign key integrity constraints, CASCADE, which enables the values in a primary key

integrity constraint to be updated and also updates matching values in the foreign key data file.

- IC CREATE also supports a new MSGTYPE= option, which enables you to control the format of integrity-constraint error messages.
- The AUDIT statement now supports the GENNUM= data set option.
- The new DATECOPY option on the COPY statement copies the date and time a SAS file was created as well as the date and time the file was last modified when a copy of the file is created.
- The new DTC= option on the MODIFY statement enables you to specify the date and time you want to show that a SAS file was created.

---

## The EXPORT Procedure

Depending on the operating environment, the EXPORT procedure now supports the following additional external data sources:

- Microsoft Access 2000 table
- Microsoft Excel 2000 spreadsheet.

---

## The FREQ Procedure

The TABLES statement supports the following new options:

- CONTENTS= specifies the text for the links in the HTML contents file to the crosstabulation tables produced by the TABLES statement.
- FORMAT= specifies a format for the following crosstabulation table cell values: frequency, expected frequency, and deviation. PROC FREQ also uses this format to display the total row and column frequencies for crosstabulation tables. You can use any standard SAS numeric format or a numeric format that was defined with the FORMAT procedure.
- OUTCUM includes the cumulative frequency and the cumulative percent for one-way tables in the output data set when you specify the OUT= option. The variable CUM\_FREQ contains the cumulative frequency for each level of the analysis variable, and the variable CUM\_PCT contains the cumulative percent for each level.

---

## The IMPORT Procedure

Depending on the operating environment, the IMPORT procedure now supports the following additional external data sources:

- Microsoft Access 2000 table
- Microsoft Excel 2000 spreadsheet.

---

## The PRINT Procedure

- The CONTENTS= option on the PROC PRINT statement now enables you to customize the HTML contents link to the output.

---

## The REPORT Procedure

- The CONTENTS= option on the PROC REPORT statement now enables you to customize the HTML contents link to the output.

---

## The SORT Procedure

- The new DATECOPY option copies the date and time that a SAS data set was created as well as the date and time that a SAS data set was last modified to the resulting sorted data set.
- The EQUALS option is now disabled when a very large number of observations are sorted.

---

## The SQL Procedure

- The constraint-specification argument of the CREATE TABLE and ALTER TABLE statements supports a new referential action for foreign key integrity constraints, CASCADE.
- The CREATE TABLE and ALTER TABLE constraint specifications also support two new options, MESSAGE= and MSGTYPE=, which enable you to customize integrity-constraint error messages.

---

## The UNIVARIATE Procedure

The PROBPLOT statement supports the following new options:

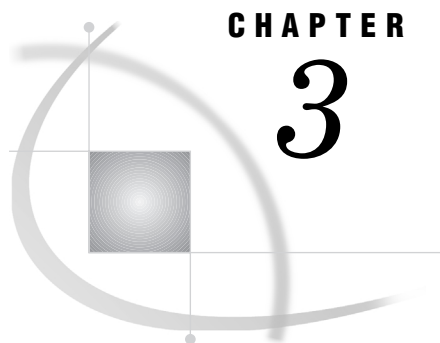
- CFRAMESIDE= specifies the color to fill the frame area for the row labels that are displayed along the left side of the comparative probability plot. This color also fills the frame area for the label of the corresponding class variable (if you associate a label with the variable).
- CFRAMETOP= specifies the color to fill the frame area for the column labels that are displayed across the top of the comparative probability plot. This color also fills the frame area for the label of the corresponding class variable (if you associate a label with the variable).
- INTERTILE= specifies the distance in horizontal percentage screen units between the framed areas, which are called *tiles*.
- NCOLS= specifies the number of columns in the comparative probability plot.
- NROWS= specifies the number of rows in the comparative probability plot.

The QQPLOT statement supports the following new options:

- CFRAMESIDE= specifies the color to fill the frame area for the row labels that are displayed along the left side of the comparative probability plot. This color also fills the frame area for the label of the corresponding class variable (if you associate a label with the variable).
- CFRAMETOP= specifies the color to fill the frame area for the column labels that are displayed across the top of the comparative probability plot. This color also

fills the frame area for the label of the corresponding class variable (if you associate a label with the variable).

- INTERTILE= specifies the distance in horizontal percentage screen units between the framed areas, which are called *tiles*.
- NCOLS= specifies the number of columns in the comparative probability plot.
- NROWS= specifies the number of rows in the comparative probability plot.



## CHAPTER

## 3

## SAS Software in the CMS Environment

---

*Statements* 13

*System Options* 13

---

### Statements

The FILENAME statement for the EMAIL device type has the following new e-mail options:

- The FROM= option enables you to customize the e-mail address that appears in the FROM line of the e-mail message. The default is the user ID and host name of the user who is running SAS.
- The ATTACH= option has the following new suboptions:
  - TYPE= enables you to specify a MIME content type for an attachment.
  - EXTENSION= enables you to specify an extension other than the CMS file type for an attachment.
  - NAME= enables you to specify a name other than the CMS filename for an attachment.

---

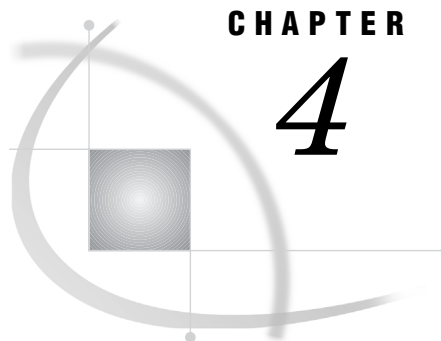
### System Options

The following SAS system options are new or enhanced:

- EMAILSYS= accepts a new value, SMTP, for Simple Mail Transport Protocol. You can now choose between a BSMTP (spool) e-mail interface and an SMTP (socket) e-mail interface.
- EMAILHOST= identifies the e-mail server to use.
- EMAILPORT= identifies the e-mail port number.
- HELPCASE controls how text is displayed in the help browser.
- LOADMEMSIZE= specifies how much memory SAS uses to load executables.







## CHAPTER

## 4

# SAS Software in the Microsoft Windows Environment

---

|                                                                       |    |
|-----------------------------------------------------------------------|----|
| <i>Graphical User Interface</i>                                       | 15 |
| <i>Application Integration</i>                                        | 15 |
| <i>SAS Language Elements</i>                                          | 15 |
| <i>Functions and CALL Routines</i>                                    | 15 |
| <i>Macros</i>                                                         | 16 |
| <i>Commands</i>                                                       | 16 |
| <i>System Options</i>                                                 | 16 |
| <i>Enhancements for Running SAS under Windows NT and Windows 2000</i> | 16 |

---

## Graphical User Interface

New GUI features include:

- ☐ The Program Editor can be suppressed when SAS starts.
  - ☐ Multiple files can be opened in the Enhanced Editor by using the Open dialog box.
  - ☐ You can open Help for a SAS procedure by pressing F1 in the Enhanced Editor when the cursor is placed within the procedure name.
  - ☐ The contents of all Enhanced Editor windows can be automatically saved.
  - ☐ The Results Viewer displays RTF output from the Output Delivery System.
- 

## Application Integration

- ☐ SCL multidimensional arrays can now be passed to and received by the OLE automation server.
- 

## SAS Language Elements

### Functions and CALL Routines

- ☐ The new SLEEP function suspends the execution of the DATA step for a specified amount of time.

---

## Macros

- The SASCC automatic macro variable now returns values for Windows 2000 operating environments.

---

## Commands

The following SAS commands have been added:

- FILEOPEN opens the Enhanced Editor Open dialog box.
- WNEXTEDIT moves the active window among all open Enhanced Editor windows.

The following SAS commands have been enhanced:

- WAUTOSAVE now accepts arguments to automatically save Enhanced Editor windows.
- WNEWTITLE now opens an Enhanced Editor window when the active window is an Enhanced Editor window.
- WSTATUSLN now displays the Enhanced Editor cursor position for the active Enhanced Editor window.

---

## System Options

The following SAS system options have been added:

- APPEND appends a pathname to the existing value of a specified system option.
- SMTP (Simple Mail Transport Protocol) is a new option for the EMAILSYS= system option.
- INSERT adds a pathname to the beginning of the value of a specified system option.
- LOADMEMSIZE specifies how much memory SAS uses to load executable programs.
- MAXMEMQUERY specifies the limit on the maximum amount of memory that is allocated for certain procedures.

The following SAS system option has been enhanced:

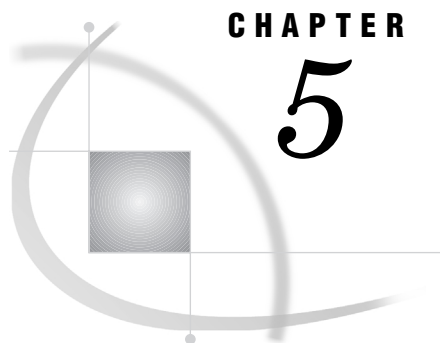
- AWSMENUMERGE is now set to NOAWSMENUMERGE in batch mode.

Note that specifying the NLOG system option does not suppress messages to the SAS log. The NOTERMINAL system option is necessary to suppress messages.

---

## Enhancements for Running SAS under Windows NT and Windows 2000

- SAS can now be started as a service under Windows NT and Windows 2000.



## CHAPTER

## 5

# SAS Software in the OpenVMS Environment

---

|                              |    |
|------------------------------|----|
| <i>Transport Files</i>       | 17 |
| <i>SAS Language Elements</i> | 17 |
| <i>Commands</i>              | 17 |
| <i>Data Set Options</i>      | 17 |
| <i>System Options</i>        | 17 |
| <i>SAS Session Manager</i>   | 18 |

---

## Transport Files

The default record attribute for SAS transport files has been changed to NONE.

---

## SAS Language Elements

---

### Commands

The following SAS commands are new or enhanced:

- Multiple instances of the X command no longer result in separate DECTERM windows.
  - The new TPU command invokes the TPU editor.
- 

### Data Set Options

You can use the new OUTREP= data set option to transport files between OpenVMS Alpha and OpenVMS VAX.

---

### System Options

The following SAS system option has been added:

- The EMAILSYS= system option specifies which e-mail interface to use for sending electronic mail from within the SAS System.

The following SAS system options have been enhanced:

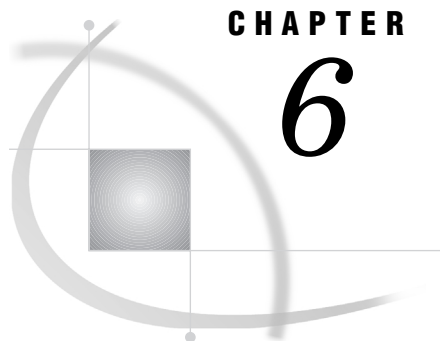
- The BLKSIZE= system option is no longer supported.
- The name of the CACHESIZE= system option has been changed from CACHESIZ=, and it now accepts values that are larger than 65024.

- The DBCSTYPE= system option has new valid values.
- The new default for the SORTPGM= system option is BEST.
- The UNIVERSALPRINT system option is the default for Universal Printing from OpenVMS.
- The name of the WORKCACHE= system option has been changed from WRKCACHE=, and its maximum values are different on OpenVMS VAX and OpenVMS Alpha.

---

## SAS Session Manager

The new SAS Session Manager enables you to control your SAS sessions.



## CHAPTER

## 6

## SAS Software in the OS/2 Environment

*Functions and CALL Routines* 19

*System Options* 19

### Functions and CALL Routines

- The new SLEEP function suspends the execution of the DATA step for a specified amount of time.

### System Options

The following SAS system options have been added:

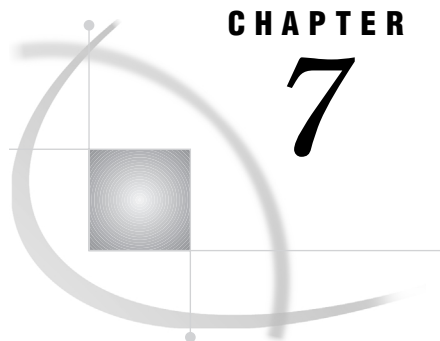
- APPEND appends a pathname to the existing value of a specified system option.
- SMTP (Simple Mail Transport Protocol) is a new option for the EMAILSYS= system option.
- INSERT adds a pathname to the beginning of the value of a specified system option.
- LOADMEMSIZE specifies how much memory SAS uses to load executable programs.
- MAXMEMQUERY specifies the limit on the maximum amount of memory that is allocated for certain procedures.

The following SAS system option has been enhanced:

- AWSMENUMERGE is now set to NOAWSMENUMERGE in batch mode.

Note that specifying the NLOG system option does not suppress messages to the SAS log. The NOTERMINAL system option is necessary to suppress messages.





## CHAPTER

## 7

# SAS Software in the OS/390 Environment

*Statements* 21

*System Options* 21

## Statements

The following statements have been enhanced:

- The FILENAME statement supports the EMAIL device type, which enables you to send electronic mail from within the SAS System.
- The FILE and INFILE statements support an MVS argument to indicate that a file is not an HFS file when an ambiguous physical filename is specified.

## System Options

The following SAS system options have been added:

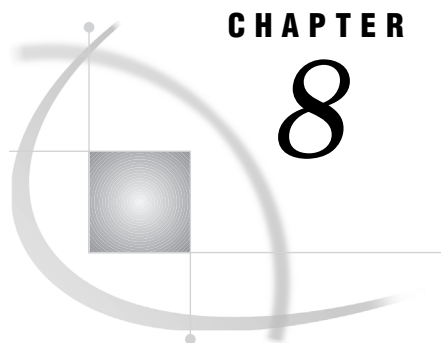
- CTRANSLOC= enables you to allocate the C Transient Library for SAS/CONNECT and SAS/SHARE software from within SAS.
- EMAILHOST= identifies the e-mail server.
- EMAILPORT= identifies the e-mail port number.
- EMAILSYS= specifies which e-mail interface to use for sending electronic mail from within the SAS system.
- HELPCASE controls how text is displayed in the help browser.

The following SAS system options have been enhanced:

- ASYNCHIO is now the default for both TSO and non-TSO sessions.
- AUTOEXEC= now has a default value of null.
- SEQENGINE= now has a default value of TAPE and no longer supports values in the form SASVnSEQ.







## CHAPTER

## 8

## SAS Software in UNIX Environments

---

*Call Routines* 23

*Commands* 23

*System Options* 23

---

### Call Routines

- The new CALL SLEEP routine suspends the execution of the DATA step for a specified amount of time.

---

### Commands

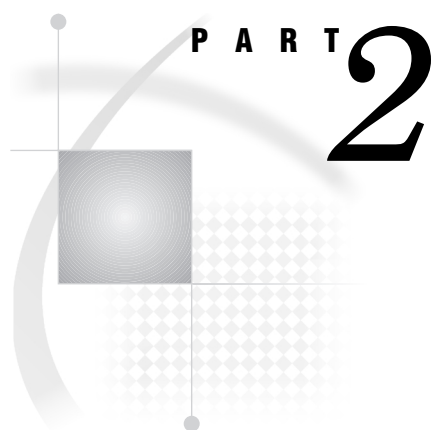
- The SAS command no longer assumes a **.sas** filename extension when you do not specify an extension on input filenames.

---

### System Options

- The new LOADMEMSIZE system option specifies how much memory SAS uses to load executable programs.
- The default settings for the MEMSIZE and SORTSIZE system options have been increased.

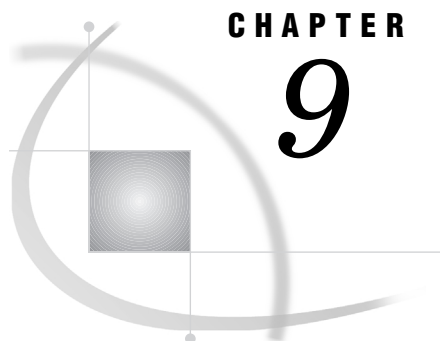




## Other SAS Software Products

|                   |                                             |    |
|-------------------|---------------------------------------------|----|
| <i>Chapter 9</i>  | <b>SAS/ACCESS Software</b>                  | 27 |
| <i>Chapter 10</i> | <b>SAS/AF Software</b>                      | 31 |
| <i>Chapter 11</i> | <b>SAS Component Language</b>               | 33 |
| <i>Chapter 12</i> | <b>SAS/CONNECT Software</b>                 | 35 |
| <i>Chapter 13</i> | <b>SAS/EIS Software</b>                     | 37 |
| <i>Chapter 14</i> | <b>SAS/ETS Software</b>                     | 39 |
| <i>Chapter 15</i> | <b>SAS/GIS Software</b>                     | 41 |
| <i>Chapter 16</i> | <b>SAS/GRAPH Software</b>                   | 43 |
| <i>Chapter 17</i> | <b>SAS/IML Software</b>                     | 45 |
| <i>Chapter 18</i> | <b>SAS/INSIGHT Software</b>                 | 47 |
| <i>Chapter 19</i> | <b>SAS Integration Technologies</b>         | 49 |
| <i>Chapter 20</i> | <b>SAS/IntrNet Software</b>                 | 51 |
| <i>Chapter 21</i> | <b>SAS OLAP Server Software</b>             | 53 |
| <i>Chapter 22</i> | <b>SAS/OR Software</b>                      | 55 |
| <i>Chapter 23</i> | <b>SAS/QC Software</b>                      | 57 |
| <i>Chapter 24</i> | <b>SAS/SHARE Software</b>                   | 59 |
| <i>Chapter 25</i> | <b>SAS/STAT Software</b>                    | 61 |
| <i>Chapter 26</i> | <b>SAS/Warehouse Administrator Software</b> | 67 |





## CHAPTER

## 9

## SAS/ACCESS Software

---

|                                                                    |    |
|--------------------------------------------------------------------|----|
| <i>All SAS/ACCESS Relational Interfaces</i>                        | 27 |
| <i>SAS/ACCESS Interface to DB2 under UNIX and PC Hosts</i>         | 27 |
| <i>SAS/ACCESS Interface to DB2 under VM (formerly SQL/DS)</i>      | 27 |
| <i>SAS/ACCESS Interface to Microsoft SQL Server</i>                | 27 |
| <i>SAS/ACCESS Interface to Oracle Rdb</i>                          | 28 |
| <i>SAS/ACCESS Interface to ODBC</i>                                | 28 |
| <i>SAS/ACCESS Interface to OLE DB</i>                              | 28 |
| <i>SAS/ACCESS Interface to ORACLE</i>                              | 28 |
| <i>SAS/ACCESS Interface to Teradata</i>                            | 28 |
| <i>SAS/ACCESS Interface to ADABAS (available on OS/390)</i>        | 29 |
| <i>SAS/ACCESS Interface to CA-DATACOM/DB (available on OS/390)</i> | 29 |
| <i>SAS/ACCESS Software for PC File Formats</i>                     | 29 |
| <i>SAS/ACCESS Interface to Baan</i>                                | 29 |
| <i>SAS/ACCESS Interface to R/3</i>                                 | 29 |

---

### All SAS/ACCESS Relational Interfaces

- The data set options DBCREATE\_TABLE\_OPTS and DBCOMMIT are now LIBNAME options as well.

---

### SAS/ACCESS Interface to DB2 under UNIX and PC Hosts

- Bulk loading, which improves performance when data is loaded into a DBMS table, is now supported.

---

### SAS/ACCESS Interface to DB2 under VM (formerly SQL/DS)

- The SAS/ACCESS LIBNAME statement is now supported.

---

### SAS/ACCESS Interface to Microsoft SQL Server

- This new SAS/ACCESS interface provides LIBNAME, SQL Procedure Pass-Through Facility, and DBLOAD access to SQL Server on UNIX platforms.

---

## SAS/ACCESS Interface to Oracle Rdb

- The SAS/ACCESS LIBNAME statement is now supported for Rdb version 7 and later.

---

## SAS/ACCESS Interface to ODBC

- A new LIBNAME and data set option, INSERT\_SQL, determines which method is used to insert rows into a data source.
- PROC DATASETS now returns all tables in the current schema, including system tables if they are present.

---

## SAS/ACCESS Interface to OLE DB

- Three new bulk loading options have been added:
  - BL\_KEEPIENTITY determines how the identity column is populated.
  - BL\_KEEPNULLS indicates how null values are handled during bulk loading.
  - BL\_OPTIONS passes options that affect how data is loaded and processed.
- A new LIBNAME option, QUALIFY\_ROWS, uniquely qualifies all member values in a result set.
- A new LIBNAME and data set option, INSERT\_SQL, determines which method is used to insert rows into a data source.
- You can now specify a maximum of 5 axis dimensions for OLAP. In Version 8, this interface supported a maximum of 2 axes.

---

## SAS/ACCESS Interface to ORACLE

- Bulk loading, which improves performance when data is loaded into a DBMS table, is now supported on OS/390 (and on all other platforms except CMS).
- A new data set option, BL\_PRESERVE\_BLANKS, enables you to specify how blank values are inserted into columns that are defined as NOT NULL when the bulk loading facility is used.

---

## SAS/ACCESS Interface to Teradata

- The Fastload bulk loading facility, which improves performance when data is loaded into an empty Teradata table, is now supported.
- COMMIT statements are now required for PROC SQL Pass-Through Data Definition Language (DDL) statements and to complete transactions. If you do not add the necessary COMMIT statements, your changes are not saved to the database.

---

## SAS/ACCESS Interface to ADABAS (available on OS/390)

- The interface view engine now supports:
  - the ADABAS multifetch feature. When reading large amounts of data sequentially, this can significantly reduce the number of system commands needed to complete ADABAS calls.
  - ADABAS Version 6 new features, including 2-byte database identifiers (DBIDs), 2-byte file numbers, and 4-byte internal sequence numbers (ISNs).
  - periodic groups with up to 191 occurrences.
- Five new data set options enable you to maintain ADABAS buffer lengths: ADBFMTL, ADBRECL, ADBSCHL, ADBVALL, and ADBISNL.

---

## SAS/ACCESS Interface to CA-DATACOM/DB (available on OS/390)

- The interface view engine now supports 5-byte database identifiers.
- A new data set option, DDBCOMIT=, enables you to specify how often to commit updates when database logging is in effect.
- The interface view engine no longer recognizes special user-defined URT names (US`tttnnn` and UW`tttnnn`) as a method for loading the User Requirements Table.

---

## SAS/ACCESS Software for PC File Formats

- SAS/ACCESS Software for PC File Formats enables you to read and load Excel 4 and 5 files. Under Windows in Release 8.1, you use the Import/Export Facility to read or load these Excel files, as well as Excel 7 (Excel 95), Excel 97, and Excel 2000 files. The Import/Export Facility also enables you to read or load Microsoft Access files, including Access 97 and Access 2000 files. These same files can be used in the IMPORT and EXPORT procedures as well.

---

## SAS/ACCESS Interface to Baan

- The SAS/ACCESS Interface to Baan enables you to read Baan data into SAS. Help is available from the interface windows. In Release 8.1, the online Help contains additional index entries.

---

## SAS/ACCESS Interface to R/3

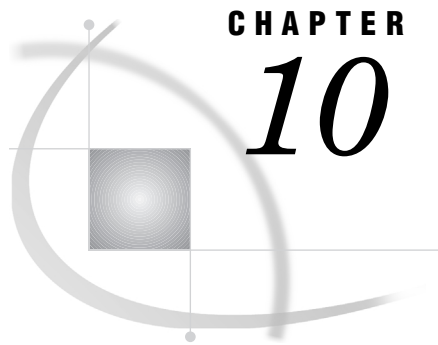
Four new SAS macros and three new SAS macro variables are now supported:

- The %CALLRFC macro enables you to call a specified R/3 function.
- The SR3DLM macro variable enables you to specify the delimiter for output files that are created when you use the %CALLRFC macro.
- The RFC\_LOGON\_INFO macro variable enables you to specify the logon information and connection parameters that are used in the %CALLRFC macro.



- The CALLRFC\_EXE macro variable enables you to specify the location of the CALLRFC executable.
- The %R3CONNB macro enables you to use a profile to log on to the R/3 System for batch processing.
- The %R3CONNC macro enables you to log on to the R/3 System when performing batch processing.
- The %R3CONNE macro enables you to log off of the R/3 System when performing batch processing.

In addition, Release 8.1 provides online Help for the Wizard to Load Metadata in the SAS/ACCESS Interface to R/3.



## CHAPTER

## 10

**SAS/AF Software**

---

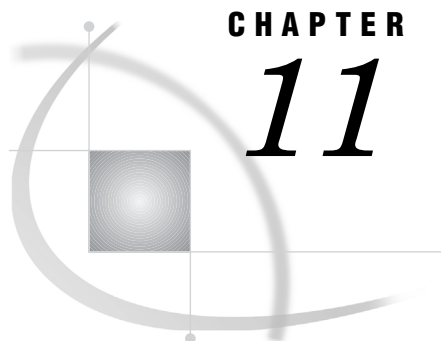
*New Classes* 31

---

**New Classes**

- The Catalog Entry Viewer class provides a control for viewing catalog entries within a frame.
- The Column Properties class can be used in conjunction with the SAS Data Set class and the SCL List Model class to provide information about table columns.
- The Data Model class is an abstract class that you can subclass in order to implement your own data model.
- The Data Set Column Properties class can be used in conjunction with the SAS Data Set class to provide information about table columns.
- The External File Viewer class provides a control from which you can view external system files.
- The Form Viewer class provides a control that can help you interact with a single row of data from a particular data model.
- The Image Viewer class provides a control that displays a bitmapped image.
- The Progress Bar class provides horizontal or vertical progress-bar controls on a frame. Progress bars keep users informed as to the progress of a specific process.
- The SAS Data Set Model class provides a model that can be attached to specific controls to create objects that provide functionality that was previously available through data tables and data forms.
- The SCL Exception class is a utility component that helps implement the throw-catch exception-handling mechanism of the SAS Component Language (SCL).
- The SCL List Model class enables you to create SCL lists.
- The SCL Throwable class is a utility component that helps implement the throw-catch exception-handling mechanism of the SAS Component Language (SCL). You can use an SCL Exception child class to make error recovery a natural part of your application development efforts.
- The Table Viewer class provides a control that enables you to display, print, and interact with tabular data.





## SAS Component Language

---

*New Functions*    **33**

*Enhanced Functions*    **33**

---

### New Functions

The following functions are new:

- The ICDESCRIBE function obtains the attributes for all existing integrity constraints within a SAS table.
- The REDIMPORT function enables you to mark an array as implicitly growable.

---

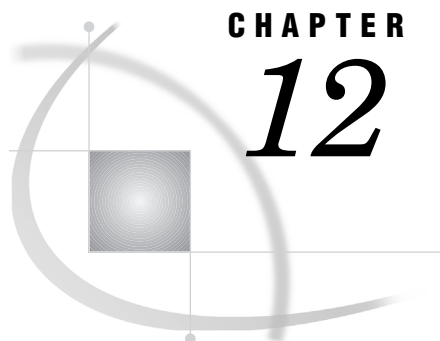
### Enhanced Functions

The following functions have been enhanced:

- The ATTRC function has a new attribute, **DATAREP**, which enables you to determine whether a file is a native file or a foreign file.
- The ICCREATE function supports a new referential action, **CASCADE**, which updates all matching foreign key values whenever a primary key value is modified. Also, with the new *message* and *message-type* arguments, you can now specify your own messages to be displayed when a data value fails to comply with a constraint.
- The FILEDIALOG function supports a new dialog type, **SAVEASNOAPPEND**, which lists the files to which a user can write but cannot append.
- The MESSAGEBOX function supports two new sets of command buttons:

|             |                             |
|-------------|-----------------------------|
| <b>YYNC</b> | Yes, Yes to all, No, Cancel |
| <b>RAC</b>  | Replace, Append, Cancel     |





## SAS/CONNECT Software

---

*Firewall Configuration* 35

*System Options* 35

---

### Firewall Configuration

SAS/CONNECT software now permits TCP/IP connections between clients outside a firewall and spawners that run on hosts inside the firewall.

---

### System Options

□ AUTOSIGNON=

Note that in order to activate the auto-sign-on feature of the RSUBMIT statement, the AUTOSIGNON= system option must be explicitly set to YES.

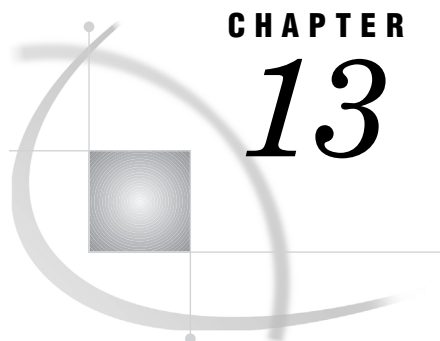
□ CTRANSLOC=

The new CTRANSLOC= system option can be used on OS/390 hosts for connections with the TCP/IP access method to allocate the SAS/C Transient Library for SAS execution.

□ SASCMD=

For OS/390 hosts that use MP CONNECT by means of the XMS access method, the new SASCMD= system option is used for specifying options and values that are passed to a dynamically created DMR session.





## CHAPTER

## 13

**SAS/EIS Software**

---

*General Functionality* 37

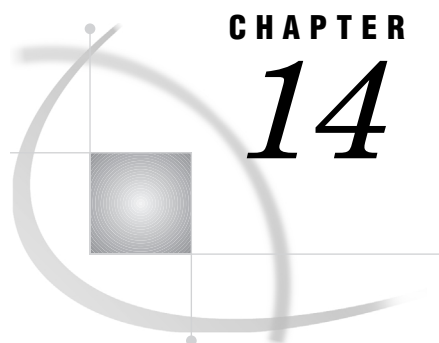
---

**General Functionality**

- The NUNIQUE statistic, provided by SAS/MDDB Server software, is now supported by SAS/EIS software. The SAS/EIS multidimensional data model (SASHELP.MDDB.EMDDB\_M.CLASS), SAS/EIS multidimensional reports, and the metabase have been updated to support NUNIQUE columns. A new interface enables you to define NUNIQUE column registrations. You can also define NUNIQUE columns at run time, and you can use the Save and Save as features.
- Class values from an MDDB can now be displayed using a different format than the one that was used to create the MDDB. By default, class values are stored as formatted character values on the MDDB. The formatted class values are displayed in all reports that are produced from the MDDB. In some cases, users want to store an unformatted value on the MDDB, and to apply the formats when the reports are displayed.
- Performance options have been added to the Incremental MDDB Update object. The options for setting block size for data swapping (PKTSIZE) and for setting memory size (VMEMSIZE), provided by SAS/MDDB Server software, are now available in the Incremental MDDB Update object.
- The METABASE command enables you to access the Metabase interface from a SAS command line without invoking SAS/EIS software.
- Repositories are now write accessible from more than one repository manager. Prior to Release 8.1, a repository was write accessible only from the repository manager in which it was first registered.







## SAS/ETS Software

---

|                                   |           |
|-----------------------------------|-----------|
| <i>Investment Analysis System</i> | <b>39</b> |
| <i>SASECRSP Engine</i>            | <b>39</b> |
| <i>QLIM Procedure</i>             | <b>39</b> |
| <i>VARMAX Procedure</i>           | <b>40</b> |
| <i>X12 Procedure</i>              | <b>40</b> |
| <i>For Additional Information</i> | <b>40</b> |

---

### Investment Analysis System

The new Investment Analysis system provides time-value-of-money analysis for a variety of investments. Various analyses are provided for the value analysis of alternative investments: time value, periodic equivalent, internal rate of return, benefit-cost ratio, and breakeven analysis. These analyses can help answer numerous questions, including: Which option is more profitable or less costly? How much is legally deductible? Is this a reasonable price? Is it better to buy or rent? The current version offers interfaces for the following investment types: loans, savings, depreciations, bonds, and generic cashflows.

---

### SASECRSP Engine

A new SAS/ETS interface engine to the CRSPAccess database (SASECRSP) is now available as an experimental release.

---

### QLIM Procedure

The new QLIM (Qualitative and LIMited dependent variable Model) procedure is an experimental procedure that analyzes models where dependent variables take discrete values or where dependent variables are observed only over a limited range of values. This procedure includes logit, probit, tobit, Poisson regression, and general simultaneous equation models. The simultaneous equations model consists of discrete choice and limited endogenous variables as well as continuous endogenous variables. Univariate models use the maximum likelihood method, whereas simultaneous equations models are estimated using a two-stage estimation method.

---

## VARMAX Procedure

Often, economic or financial variables are not only contemporaneously correlated to each other, they are also correlated to each other's past values. The new VARMAX procedure can be used to model these types of time-related relationships and to estimate the associated Vector Autoregressive and Moving-Average with exogenous regressors (VARMAX) models. There are features to aid in order selection, model checking, residual analysis, and forecasting. Tests can be performed for stationarity, cointegration, and Granger-causality.

The VARMAX procedure provides for both Vector Autoregressive (VAR) and Bayesian Vector Autoregressive (BVAR) models. To cope with the problem of high dimensionality in the parameters of the VAR model, the VARMAX procedure provides both the Vector Error Correction Model (VECM) and Bayesian Vector Error Correction Model (BVECM). Bayesian models are used when prior information about the model parameters is available.

---

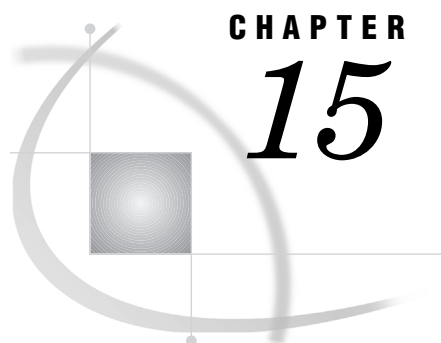
## X12 Procedure

The new X12 procedure, an adaptation of the U.S. Bureau of the Census X-12-ARIMA Seasonal Adjustment program, seasonally adjusts monthly or quarterly time series. The procedure makes additive or multiplicative adjustments and creates an output data set that contains the adjusted time series and intermediate calculations.

---

## For Additional Information

For additional information about these features, refer to *SAS/ETS Software: Changes and Enhancements, Release 8.1*.



## CHAPTER

## 15

## SAS/GIS Software

|                                   |    |
|-----------------------------------|----|
| <i>New Tutorial</i>               | 41 |
| <i>PROC GIS Statement</i>         | 41 |
| <i>Displayed Maps</i>             | 41 |
| <i>Data Administration Window</i> | 42 |

## New Tutorial

### *Getting Started with SAS/GIS Software*

is a step-by-step, high-level tutorial for SAS/GIS software. In this scenario-based tutorial, you learn to use SAS/GIS to import spatial data and create a map, to modify the map using the GIS procedure, to geocode addresses and add points to the map, and to link attribute data and spatial data. You also learn to work with map layers, actions, and selections.

You can access this new tutorial from the Web at <http://www.sas.com/software/tutorials8/gis/>.

## PROC GIS Statement

- The ON and OFF arguments on the PROC GIS LAYER statement have been replaced with the ONSCALE and OFFSCALE arguments. The arguments enable you to set a scale at which a layer is turned on or off. The syntax for the arguments now enables you to specify units for the scale and to turn the scale off after it has been specified.
- The PROC GIS MAP statement now includes the arguments SELECTABLE and UNSELECTABLE. These arguments enable you to specify one or more map layers as either selectable or unselectable.

## Displayed Maps

You can now remove chains from the spatial data of a displayed map. With a map displayed in edit mode and with one or more chains selected, select

Edit ► Remove Current Selection

to remove the currently selected chain or select

Edit ► Remove All Selections

to remove all selected chains.

---

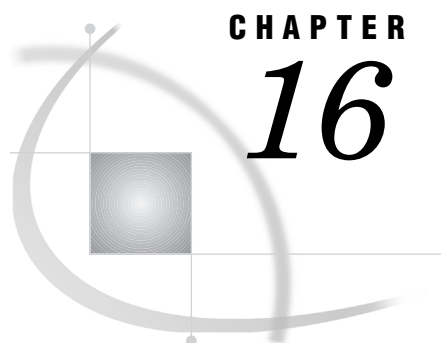
## Data Administration Window

The Data Administration window is no longer marked as “Experimental.” The menu selection to display the window has been changed from

Tools ► Data Administration

to

Edit ► Data Administration



## CHAPTER

## 16

## SAS/GRAPH Software

*Using Image Files in SAS/GRAPH* 43

*FONTLIST Command* 43

*All Procedures* 43

*GCHART Procedure* 43

*GPlot Procedure* 44

*GRADAR Procedure* 44

*SYMBOL Statement* 44

## Using Image Files in SAS/GRAPH

SAS/GRAPH software now enables you to display images as part of your graph. For example, you can use your company logo in the graph. You can place the image in the graph's background area, on the frame of graphs that support frames, or on the bars of two-dimensional bar charts. You can also display images at graph coordinate positions by using the Annotate facility or the DATA Step Graphics Interface (DSGI).

## FONTLIST Command

The new FONTLIST command displays a list of the software fonts that are available to your SAS session in your operating environment.

## All Procedures

The DESCRIPTION= option on all procedures now supports descriptions of up to 256 characters. Previously, descriptions were limited to 40 characters.

## GCHART Procedure

Several options on the HBAR and HBAR3D statements now support a new keyword, NONE, which suppresses the labels that are displayed for that option. For example, SUMLABEL=NONE suppresses the labels for the SUM statistic. Each of the following HBAR and HBAR3D options can specify the keyword NONE:

- FREQLABEL=

- ☐ CFREQLABEL=
- ☐ PERCENTLABEL=
- ☐ CPERCENTLABEL=
- ☐ MEANLABEL=
- ☐ SUMLABEL=

---

## **GPLOT Procedure**

Because the PLOT2 statement now supports the HTML= and HTML\_LEGEND= options, you can produce graphs with drill-down capabilities on the second plot line, which is plotted on the graph's right vertical axis. Previous SAS versions only supported drill-down capabilities on the first plot line that is generated from the PLOT statement.

---

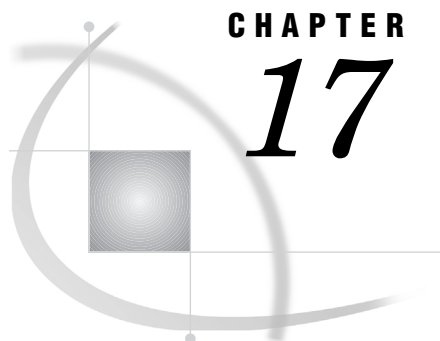
## **GRADAR Procedure**

The new GRADAR procedure creates radar charts that show the relative frequency of data measures in quality control data or market research data. Radar charts display statistics along spokes that radiate from the center of the chart. Multiple charts are often stacked on top of one another, and circular reference lines can be displayed to show the relative magnitude of the statistical values that are represented in each chart. Stacking charts around a center point reveals the interrelationships among data measures and facilitates data comparisons. For example, radar charts are useful for comparing the same measures across different test cases, or for evaluating the results of market research studies.

---

## **SYMBOL Statement**

The POINTLABEL= option now honors the format that is assigned to the variable whose values are used to label the plot points.



## SAS/IML Software

---

*Financial Functions* 45  
*Robust Regression* 45  
*Multivariate Time Series Analysis* 45

---

### Financial Functions

Functions that compute information about financial quantities have been added:

- The CONVEXIT function calculates the convexity of a non-contingent cash flow.
  - The DURATION function computes the modified duration of a non-contingent cash flow.
  - The FORWARD function computes a vector of forward rates.
  - The PV function computes the present value of cash flows based on the specified frequency and rates.
  - The RATES function returns a vector of interest rates converted from one base to another.
  - The SPOT function returns a vector of spot rates.
  - The YIELD function computes the yield-to-maturity of a cash-flow stream based on a specified frequency and value.
- 

### Robust Regression

New algorithms for robust regression have been added:

- The LTS call performs robust regression.
  - The MCD call finds the minimum covariance determinant estimator.
- 

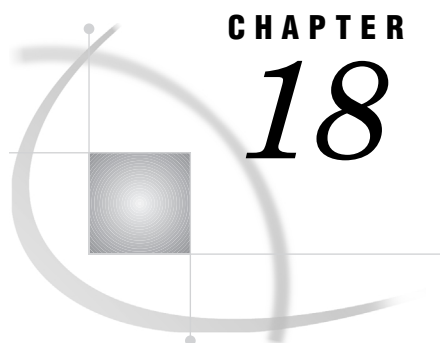
### Multivariate Time Series Analysis

New functions for creating and analyzing multivariate time series have been added:

- The VARMACOV call computes the theoretical cross-covariance matrices for a stationary VARMA( $p, q$ ) model.
- The VARMALIK call computes the log-likelihood function for a VARMA( $p, q$ ) model.
- The VARMASIM call generates a VARMA( $p, q$ ) series.
- The VNORMAL call generates multivariate normal random series.
- The VTSROOT call calculates the characteristic roots of a VARMA model from AR and MA characteristic functions.







## SAS/INSIGHT Software

---

*Graphical Enhancements* 47

*Distribution Analyses* 47

*Fit Analyses* 47

*Multivariate Analyses* 47

---

### Graphical Enhancements

SAS/INSIGHT software now includes surface plots, contour plots, response surfaces, comparison of means circles in the box plots, and color blending of up to five colors. Several methods for surface fitting are provided, including linear interpolation, thin-plate spline, kernel estimation, and using a parametric model.

---

### Distribution Analyses

The distribution window now includes the following statistics:

- ☐ basic confidence intervals of mean, standard deviation, and variance
- ☐ robust measures of scale
- ☐ tests for normality.

---

### Fit Analyses

When there are two explanatory variables in the model, a parametric response surface plot is created by default. You can also generate a nonparametric kernel or thin-plate smoothing spline response surface plot.

With more than two explanatory variables in the model, you can create a parametric profile of a response surface plot with the first two explanatory variables. The values of the remaining explanatory variables are set to their corresponding means in the plot. You can use sliders to change the values of the remaining explanatory variables.

---

### Multivariate Analyses

When performing pairwise correlations, SAS/INSIGHT software now drops an observation from all calculations if the observation has a missing value for any variable that is used in the analysis.

You can now display *biplots*, a joint display of two sets of variables in which the data points are displayed in a scatter plot of principal components.

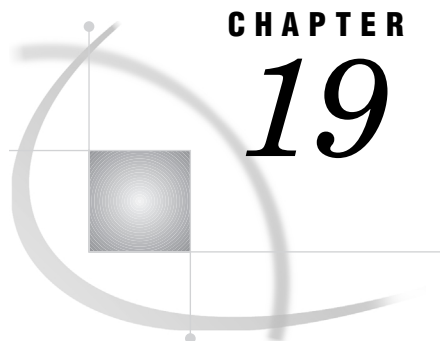
The following multivariate analyses have been added:

- ☐ canonical correlation analyses
- ☐ maximum redundancy analysis
- ☐ canonical discriminant analysis.

The following multivariate analysis has been enhanced:

- ☐ principal component analysis

You can now create tables that display statistical relationships between the Y variables and the rotated principal components.



## CHAPTER

## 19

## SAS Integration Technologies

---

*New Features* 49

*For Additional Information* 49

---

### New Features

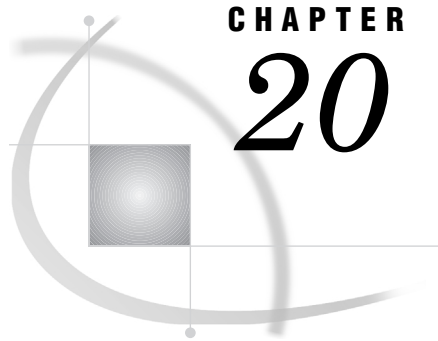
- Workspace pooling for Java clients improves efficiency by enabling multiple clients to reuse server workspaces. This feature is particularly useful for Web-enabled applications.
- The SAS Package Retriever application enables you to retrieve packages of information that have been published to available archives and message queues.
- Stored process paths can now be managed from the IT Administrator application.
- The message queue library of SAS CALL routines has now been expanded to enable you to configure and access Geneva message queues in SAS stored processes. (Experimental)
- The WebDAV transport method enables you to publish packages as collections of resources, as specified by the HTTP Extensions for Web Distributed Authoring and Versioning. (Experimental)

---

### For Additional Information

Additional information about the software is available on the SAS Integration Technologies Web site at <http://www.sas.com/rnd/itech/intro.html>.





## CHAPTER

## 20

## SAS/IntrNet Software

*New Features and Enhancements in Application Dispatcher* 51

*New Features and Enhancements for SAS Java Tools* 51

### New Features and Enhancements in Application Dispatcher

- New Application Server communication access method

With the new access method, the Application Server can now return results using the request connection. A new Broker configuration directive (FullDuplex) enables communication between the Broker and the Application Server through one socket. This feature eliminates the need for the fwproxy when a firewall is between the Broker and the Application Server. It also reduces the number of TCP/IP connections made by the Application Server.

- STATISTICS command

The new STATISTICS command enables you to capture request statistics and to store them in a SAS data set. You can then use the data set to analyze Application Server usage and performance.

- Automatic header generation

The Application Server now automatically generates a set of default HTTP headers for requests. The default headers include Content-type and Set-Cookie, if applicable.

- Encryption

All data that is sent between the Broker and the Application Server can now be encrypted. SAS proprietary fixed-encoding encryption is available to all SAS/IntrNet users. SAS/SECURE software adds support for RC2, RC4, DES, and other encryption standards. In addition, a new Broker configuration directive (Encrypt) enables encryption for a service.

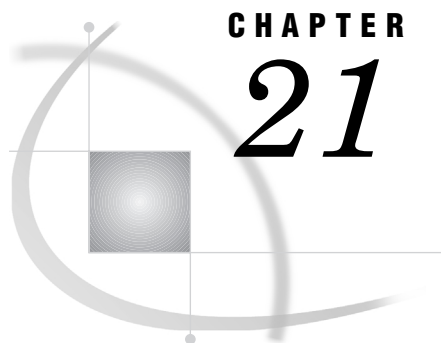
- Improved administration of multi-server services

Administrative programs such as PING, STATUS, and STOP are now being sent to all Application Servers that run the specified service (for example, pool, launch, or socket service).

### New Features and Enhancements for SAS Java Tools

- Both the SAS/CONNECT Driver for Java and the SAS/SHARE Driver for JDBC have been enhanced to support the encryption features of the Release 8.1 SAS servers. The encryption support that was available for earlier SAS servers is still provided.





## CHAPTER

## 21

## SAS OLAP Server Software

---

*Product Features* 53

---

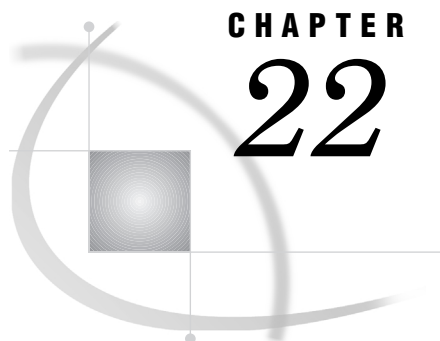
### Product Features

SAS OLAP Server software includes the following components:

- SAS/MDDB Server software, which enables users to create and manage MDDBs. MDDBs provide fast and easy data access to tools such as multidimensional viewers.
- Access Control, which enables administrators to set up and maintain group access to data and applications in OLAP environments.
- Metadata Registration Facilities, which enable users to define metadata for their OLAP cubes.
- HOLAP (Hybrid OLAP) Registration Facilities, which enable users to define metadata for data that is spread across different data sources.
- OLAP Model Coordination Facilities, which enable multiple concurrent applications to share a synchronized view of the data.







## CHAPTER

## 22

## SAS/OR Software

|                                   |    |
|-----------------------------------|----|
| <i>BOM Procedure</i>              | 55 |
| <i>CPM Procedure</i>              | 55 |
| <i>DTREE Procedure</i>            | 55 |
| <i>GANTT Procedure</i>            | 56 |
| <i>INTPOINT Procedure</i>         | 56 |
| <i>NETDRAW Procedure</i>          | 56 |
| <i>NETFLOW Procedure</i>          | 56 |
| <i>PM Procedure</i>               | 56 |
| <i>For Additional Information</i> | 56 |

## BOM Procedure

The BOM procedure is a new experimental procedure for performing bill-of-material processing. It composes a series of single-level bills of material into a multilevel tree-structured bill of material, determines the level of each part in the bill, and represents the multilevel bill of material structure in the form of an indented bill of material. PROC BOM can also output a summarized bill of material.

## CPM Procedure

- The MULTIPLEALTERNATES option has been added to the RESOURCE statement. It enables multiple alternate resources to be used simultaneously for a single resource.
- The CPM procedure now allows the use of alternate resources for resources that drive an activity's duration.
- Auxiliary resources can now be specified for any resource.

## DTREE Procedure

Two new options, NOPAGENUM and PAGENUM, enable you to specify whether or not to display the page number in the top right corner of each page of a multipage decision-tree diagram.

---

## GANTT Procedure

The NOPAGENUM option has been added to the CHART statement.

---

## INTPOINT Procedure

The INTPOINT procedure is a new experimental procedure for performing optimization using the Interior Point algorithm. Using PROC INTPOINT is essentially the same as using PROC NETFLOW with its Interior Point algorithm. In the future, PROC INTPOINT will exploit parallel processing.

---

## NETDRAW Procedure

- The PAGENUMBER option has been added to the ACTNET statement.
  - The ROTATE option has been added to the ACTNET statement for displaying a hierarchical tree in a top-down format.
- 

## NETFLOW Procedure

You can now see the Interior Point algorithm's progress to the optimum by specifying the PRINTLEVEL2 option. The NETFLOW procedure produces a table in the SAS log. A row of the table is generated during each iteration. Each row consists of values that are useful for gauging how well the optimization is converging to the optimum.

Several new options have been added that give you greater control over when the Interior Point algorithm should stop.

---

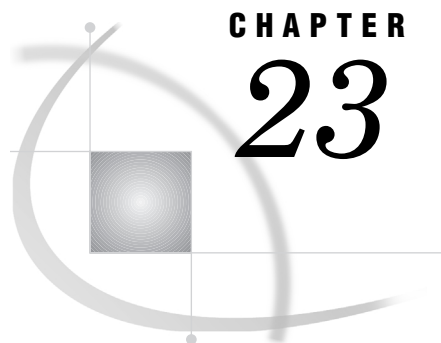
## PM Procedure

A new dialog window has been added to the Gantt view that provides activity, precedence, and resource information for each task. With this window you can also change the activity duration. You activate the Task Information dialog window from the schedule bar pop-up menu in the Gantt View.

---

## For Additional Information

For additional information, refer to *SAS/OR Software: Changes and Enhancements, Release 8.1*.



## SAS/QC Software

---

|                           |    |
|---------------------------|----|
| <i>SHEWHART Procedure</i> | 57 |
| <i>ADX Interface</i>      | 57 |
| <i>Split-plot Designs</i> | 57 |
| <i>Ridge Analysis</i>     | 57 |
| <i>n-Way Effect Plot</i>  | 57 |
| <i>Box Plot</i>           | 58 |

---

### SHEWHART Procedure

- The BOXSTYLE=POINTSSCHEMATIC option specifies a new box style that combines the POINTS and SCHEMATIC box styles.
- The MRRESTART option enables the moving range computation on the IRCHART to be restarted when a missing value is encountered.
- The TESTRESET= and TEST2RESET= options enable you to reset tests for special causes.

---

### ADX Interface

---

#### Split-plot Designs

The ADX interface can now be used to analyze split-plot data. You first need to create the split-plot design outside ADX and then import the design data into ADX. The analysis is done using PROC MIXED.

---

#### Ridge Analysis

Ridge analysis, which is useful for determining the direction in which further experimentation should be conducted, has been added.

---

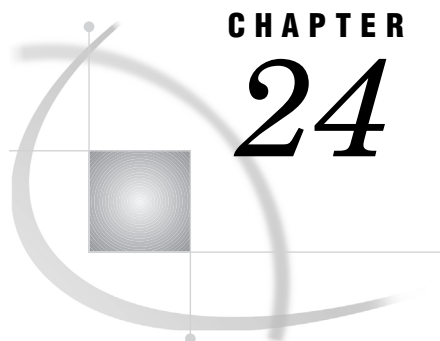
#### n-Way Effect Plot

The new n-way effect plot enables you to look in detail at potential 3- and 4-way interactions.

---

## Box Plot

Box plots have been added to display the differences in the response distribution between different levels of factors.



## CHAPTER

## 24

**SAS/SHARE Software**

---

*Cross-Platform Catalog Access* 59

*Overlapping I/O* 59

*SAS/SHARE Data Provider* 59

---

**Cross-Platform Catalog Access**

SAS/SHARE software now enables you to access catalogs that reside on a server that has a different architecture from the one on which your SAS program is running. For example, Windows NT clients can now access catalogs on an HP-UX server.

---

**Overlapping I/O**

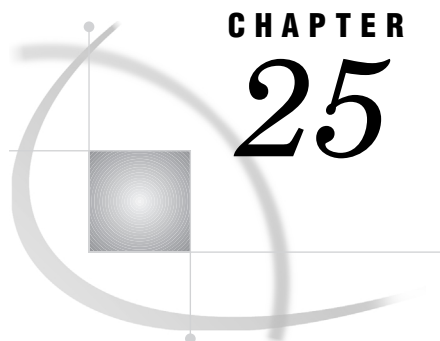
The performance of sequential read operations has been enhanced for applications that use the remote engine through the use of overlapping I/O.

---

**SAS/SHARE Data Provider**

The new SAS/SHARE data provider enables you to use the Microsoft OLE DB and Active Data Objects (ADO) interfaces to read and modify data sets on a SAS/SHARE server.





## SAS/STAT Software

---

|                                   |    |
|-----------------------------------|----|
| <i>CATMOD Procedure</i>           | 61 |
| <i>FACTOR Procedure</i>           | 61 |
| <i>FREQ Procedure</i>             | 62 |
| <i>GAM Procedure</i>              | 62 |
| <i>LOESS Procedure</i>            | 62 |
| <i>LOGISTIC Procedure</i>         | 63 |
| <i>MI Procedure</i>               | 63 |
| <i>MIANALYZE Procedure</i>        | 63 |
| <i>MIXED Procedure</i>            | 64 |
| <i>MODECLUS Procedure</i>         | 64 |
| <i>MULTTEST Procedure</i>         | 64 |
| <i>NLMIXED Procedure</i>          | 64 |
| <i>PHREG Procedure</i>            | 64 |
| <i>SURVEYMEANS Procedure</i>      | 65 |
| <i>For Additional Information</i> | 65 |
| <i>References</i>                 | 65 |

---

### CATMOD Procedure

The CATMOD procedure now supports the iterative proportional fitting (IPF) algorithm for fitting hierarchical log-linear models with no independent variables and no population variables. The advantage of the IPF algorithm is that you can obtain the log likelihood,  $G^2$ , and the predicted cell counts without performing expensive parameter estimation and covariance computations. To request IPF fitting, you specify the ML=IPF option. Several options for controlling the convergence criterion are available, as well as options for controlling how degrees of freedom for  $G^2$  are computed.

The new MISSING= option specifies whether a missing cell is treated as a sampling or structural zero. The new ZEROES= option specifies whether a non-missing cell with zero weight is treated as a sampling or structural zero.

---

### FACTOR Procedure

The generalized Crawford-Ferguson family of orthogonal and oblique rotations (Jennrich 1973) is now available through the ROTATE= option, which performs these new rotations. The family includes the direct oblimin (Harman 1976) and the Crawford-Ferguson family (Crawford and Ferguson 1970) as special cases.

The ROTATE= option performs these new rotations and many other specific rotations such as biquartimax, factor parsimax, (direct) quartimin, biquartimin, and covarimin.



You can also control the convergence properties of the rotation by using the `RCONVERGE=` and the `RITER=` options. The `RCONVERGE=` option defines the convergence criterion, and the `RITER=` option sets the maximum number of cycles for factor rotations.

When `METHOD=ML` (maximum likelihood), the new `SE` option requests standard error estimates for the unrotated factor loadings, the rotated factor loadings, the factor correlations, and the factor structure loadings. The standard error estimates are available for the generalized Crawford-Ferguson family of rotations but not for the promax and Harris-Kaiser rotations. The new `CI` option requests the confidence intervals for the factor loadings. You can control the coverage percentage of the confidence intervals with the `ALPHA=` option.

---

## **FREQ Procedure**

The new `FORMAT=` option specifies a format for the following crosstabulation table cell values: frequency, expected frequency, and deviation. `PROC FREQ` also uses this format to display the total row and column frequencies for crosstabulation tables.

The new `OUTCUM` keyword includes the cumulative frequency and the cumulative percentage for one-way tables in the output data set when you specify the `OUT=` option.

---

## **GAM Procedure**

The new `GAM` procedure, which is experimental in Release 8.1, fits generalized additive models. Generalized additive models were proposed by Hastie and Tibshirani (1990) as a way of extending the additive model to a broader range of distributional families. The additive model estimates an additive approximation to the multivariate regression function.

`GAM` models enable the mean of the dependent variable to depend on an additive predictor through a nonlinear link function. Many widely used statistical models belong to this class: they include additive models for Gaussian data, nonparametric logistic models for binary data, and nonparametric log-linear models for Poisson data. Besides giving the user the flexibility of nonparametric regression, generalized additive models also have the advantage of easy interpretability, which comes from modeling the regression surface as a sum of smooth terms.

---

## **LOESS Procedure**

The `LOESS` procedure now does automatic smoothing parameter selection by default. You can use the `SELECT=` option to specify the criterion among `AICC`, `AICC1`, or `GCV`. In that case, `PROC LOESS` uses a smoothing parameter value that yields a local minimum of the specified criterion. In addition, if you specify the `SMOOTH=value-list` option, then `PROC LOESS` selects the largest value in this list that yields the global minimum of the specified criterion. `PROC LOESS` now also supports higher-order blending methods, which you specify with the `INTERP=` option.

---

## LOGISTIC Procedure

The new ALPHA= option in the PROC LOGISTIC statement enables you to specify a default level of significance for computing confidence intervals for the MODEL, EXACT, CONTRAST, and OUTPUT statements.

The new EXACT statement enables you to perform exact conditional logistic regression for binary (dichotomous) response variables. Inference on the parameters of the specified effects (possibly including the intercept) is performed by conditioning on the sufficient statistics of all the other model parameters.

The analysis provides two tests for the null hypothesis that the parameters for the effects specified are zero: the exact probability test and the exact conditional scores test. You obtain the test statistics, the exact  $p$ -value, and a mid  $p$ -value, which adjusts for the discreteness of the distribution.

In addition, you can produce, for each parameter, a point estimate, a confidence interval, and a  $p$ -value for a two-sided test that the parameter is zero. You can also estimate the odds ratio. A joint test for multiple effects can be requested, and you can also output the derived distributions and summary statistics.

---

## MI Procedure

Multiple imputation is a strategy for dealing with data sets that contain missing values. You replace each missing value with a set of plausible values that represent the uncertainty about the right value to impute. You create multiply imputed data sets, analyze them with standard analyses, and then combine the results. You produce valid statistical inferences that properly reflect the uncertainty due to the missing values.

The new MI procedure, which is experimental in Release 8.1, creates multiply imputed data sets for incomplete  $p$ -dimensional multivariate data. It offers three methods for creating the imputed data sets: the regression method, the propensity score method, and the Markov Chain Monte Carlo (MCMC) method. The procedure creates an output data set that contains  $m$  imputed versions of the original data. In each version, the missing values are replaced with imputed values. For the MCMC method, you can specify whether you want a single chain for all  $m$  imputations or a separate chain for each imputation. You can also specify the initial estimates for the MCMC method.

After analyzing your imputed data with standard procedures, you use the MIANALYZE procedure to combine the results.

---

## MIANALYZE Procedure

The new MIANALYZE procedure, which is experimental in Release 8.1, combines the results of the analyses of  $m$  imputations and generates valid statistical inferences. The procedure uses as input either a specially structured SAS data set that contains the parameter estimates and the associated covariance matrix from each imputed data set, or a pair of specially structured SAS data sets that contain the parameter estimates and the covariance matrices of the parameter estimates, respectively.

---

## MIXED Procedure

PROC MIXED now prints AIC and BIC in smaller-is-better forms in the "Fit Statistics" table. A finite-sample corrected version of AIC (AICC) is also included. When you specify METHOD=ML, these criteria now incorporate the effective number of fixed-effects parameters (the rank of the  $X$  matrix) in addition to the number of estimated covariance parameters. Refer to Burnham and Anderson (1998) for additional details.

---

## MODECLUS Procedure

The MODECLUS procedure now includes a default smoothing parameter.

---

## MULTTEST Procedure

Two new  $p$ -value adjustment methods are available: Fisher combination and Hommel. You can obtain these adjustments by specifying the FISHER\_C and HOMMEL option, respectively, in the PROC MULTTEST statement. The FISHER\_C option requests adjusted  $p$ -values using closed tests, based on the idea of Fisher's combination test. Hommel's (1988) method is a closed testing procedure that is based on Simes' (1986) test.

The new WEIGHT= option for the STRATA statement specifies the type of strata weighting to use when computing the Freeman-Tukey and  $t$ -tests for the mean. Valid values for the WEIGHT= option in the STRATA statement are SAMPLESIZE, HARMONIC, and EQUAL. SAMPLESIZE is the default method. It requests weights that are proportional to the within-stratum sample sizes. HARMONIC sets up weights that are equal to the harmonic mean of the non-missing within-stratum CLASS sizes; it is similar to a Type II analysis in PROC GLM. EQUAL specifies equal weights; it is similar to a Type III analysis in PROC GLM.

---

## NLMIXED Procedure

The gamma and negative binomial distributions are now available in the MODEL statement. They are specified as  $gamma(a,b)$  and  $negbin(n,p)$ .

Only the smaller-is-better forms of AIC and BIC are now printed in the "Fit Statistics" table. A finite-sample corrected version of AIC (AICC) is also included.

---

## PHREG Procedure

The new COVSANDWICH option in the PROC PHREG statement produces the Lin and Wei (1989) robust sandwich estimate of the covariance matrix. The specification COVSANDWICH(AGGREGATE) enables you to sum the score residuals within each ID pattern in the computation of this covariance estimate.

The cumulative hazard (intensity) function is also available when there are time-dependent explanatory variables.

---

## SURVEYMEANS Procedure

The new DOMAIN statement requests analysis for subpopulations, or domains, in addition to analysis for the entire study population.

---

## For Additional Information

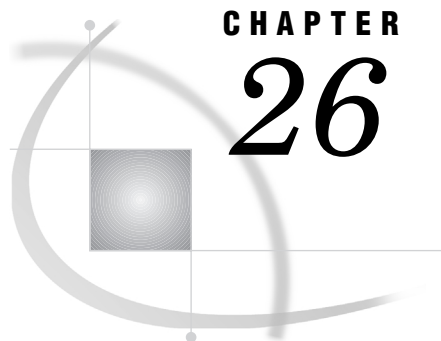
For additional information, refer to *SAS/STAT Software: Changes and Enhancements, Release 8.1*. Syntax changes for the production procedures are also included in the SAS Help system.

---

## References

- Burnham, K.P. and Anderson, D.R. (1998), *Model Selection and Inference: A Practical Information-Theoretic Approach*, New York: Springer-Verlag.
- Crawford, C.B. and Ferguson, G.A. (1970), "A General Rotation Criterion and Its Use in Orthogonal Rotation," *Psychometrika*, 35, 321-332.
- Harman, H.H. (1976), *Modern Factor Analysis*, Third Edition, Chicago: University of Chicago Press.
- Hastie, T.J. and Tibshirani, R.J. (1990), *Generalized Additive Models*, Monographs on Statistics and Applied Probability, 43, London: Chapman and Hall.
- Hommel, G. (1988), "A Comparison of Two Modified Bonferroni Procedures," *Biometrika*, 75, 383-386.
- Jennrich, R.I. (1973), "Standard Errors for Obliquely Rotated Factor Loadings," *Psychometrika*, 38, 593-604.
- Lin, D.Y. and Wei, L.J. (1989), "The Robust Inference for the Proportional Hazards Model," *Journal of the American Statistical Association*, 84, 1074-1078.
- Simes, R.J. (1986), "An Improved Bonferroni Procedure for Multiple Tests of Significance," *Biometrika*, 73, 751-754.





## CHAPTER

## 26

## SAS/Warehouse Administrator Software

|                                                       |           |
|-------------------------------------------------------|-----------|
| <i>New Tutorial</i>                                   | <b>67</b> |
| <i>Faster Loading of DBMS Tables</i>                  | <b>67</b> |
| <i>User Interface Enhancements</i>                    | <b>67</b> |
| <i>Metadata API Enhancements</i>                      | <b>68</b> |
| <i>Experimental Software: LDAP Security Interface</i> | <b>69</b> |

### New Tutorial

A new *Getting Started with SAS/Warehouse Administrator Software* tutorial is now available. This tutorial shows you how to add a new Environment to your Warehouse Administrator desktop and guides you through the process of creating a sample data warehouse.

To create the data and view the tutorial, first access SAS/Warehouse Administrator by selecting the following menu path from any SAS window:

Solutions ► Development and Programming ► Warehouse Administrator

. From the SAS/Warehouse Administrator desktop, select the **Getting Started** icon and follow the directions.

### Faster Loading of DBMS Tables

SAS/Warehouse Administrator can now generate code that supports bulk loading of DBMS tables. On the Options tab of the DBMS Table Properties window, you can specify options to the PROC APPEND statement, such as the SAS/ACCESS LIBNAME data set option BULKLOAD=.

### User Interface Enhancements

- The Columns tab on the Properties window for all data stores now enables you to
  - select multiple columns for removal.
  - add an ordinal position attribute to a metadata column, to control which column appears first, second, third, and so on in the target data store.
  - sort column metadata by attribute, such as **Name** or **Type**, by clicking on the attribute label (You can also select a column and move it by clicking on the up/down arrows.)

- use the pop-up menu on the **OLAP Roles** organization chart to
  - create all possible crossings in SAS/Warehouse Administrator; the resulting table is similar to the table produced by PROC SUMMARY.
  - create all dimensional crossings and subcrossings.
- The functionality of the Columns tab has also been enhanced in the following ways:
  - When you import column metadata, you can easily skip multiple columns that have duplicate names, and you can cancel the importation of duplicate column names.
  - When you import column metadata from a source, the column order that is specified in the source is preserved. (Metadata columns that are imported from ODDs are no longer automatically sorted in alphabetic order in the target data store.)
  - When you select individual columns for import, the columns will be imported in the order that they are added to the Selected Columns list.
- You have more ways to organize data stores in the SAS/Warehouse Administrator Explorer. You can now
  - add Subjects and OLAP Groups to Data Groups.
  - add ODD Groups within other ODD Groups.
- In the SAS/Warehouse Administrator Explorer, you can now copy and paste OLAP Groups, Tables, and MDDBs. In the Explorer and the Process Editor, you can cut and paste these items.

---

## Metadata API Enhancements

The topics in this section supersede the documentation for the corresponding metadata types in *SAS/Warehouse Administrator Metadata API Reference, Release 2.0*.

- The TABLE OPTIONS property of the WHDBMSST type has a new sublist: the APPEND sublist. The APPEND sublist contains any SAS/ACCESS LIBNAME data set options that were used to create or load the table, such as BULKLOAD=**yes**.
- Load-process options for warehouse tables, such as GENERATION LEVEL and DROP INDEXES, are now surfaced through the WHPRCLDR type and all of its subtypes. For example, you can write a SAS/Warehouse Administrator add-in that reads the load options that are specified in a table's load process and uses these options to load the corresponding table. The affected metadata types are
  - WHPRCLDR
  - WHLDRDAT
  - WHLDRDLT
  - WHLDREXT
  - WHLDRIMF
  - WHLDRINF
  - WHLDRLDT
  - WHLDRMDB
  - WHLDRODD
  - WHLDRODT
  - WHLDRSUM
  - WHLDOTBL

- WHLDOMDD
- WHLDOPRX
- You can use the new WHOST property to determine the operating system and version of the SAS System that are associated with a particular host. For example, you can write a SAS/Warehouse Administrator add-in that first reads the host metadata that is associated with a given data store, then uses these values to generate code that is appropriate for the operating system and the version of SAS.
- You can now write OLAP objects through the Metadata API. The following metadata types have been updated:
  - WHMDDSTR
  - WHOLPSTC
  - WHGRPOLP
  - WHOLPTBL
  - WHOLPMDD
  - WHCOLOLP
  - WHOLPDIM
  - WHOLPHIR
  - WHOLPCRS
  - WHOLPCUB.
- Metadata for columns that are selected via point and click in the Expression Builder (to be used in either a WHERE clause or a row selector) is now surfaced through the WHSUBSET and WHROWSEL types. For example, you can write a SAS/Warehouse Administrator add-in that reads the column metadata that is associated with a WHERE clause or a row selector and uses this metadata to generate the appropriate code.
- You can now update the EXTENDED ATTRIBUTES property and other properties in the WHCOLTIM type. For example, you can use an add-tool to add data mining attributes to a \_LOADTM column, export the metadata for the table to Enterprise Miner software, and analyze the \_LOADTM column in Enterprise Miner.
- The RUN COMMAND property in the WHJOB type has been changed from a list type to a character type to make it easier to return this property for a Job object.
- The documentation for the DBMS LIBNAME and OPTIONS properties in the WHLIBRY type has been corrected.

---

## Experimental Software: LDAP Security Interface

The new experimental LDAP security interface enables you to define a security hierarchy for the data stores in a data warehouse. This hierarchy can be used to control access to warehouse data stores in much the same way that an operating system uses Access Control Lists (ACLs) to control access to files.





# Index

## A

access methods  
     Application Server (SAS/IntrNet) 51  
     performance 4  
 ADABAS 29  
 ADX interface (SAS/QC) 57  
 aliases (ODS) 3  
 ALTER TABLE statement, SQL procedure 11  
 APPEND system option  
     OS/2 environment 19  
     Windows environment 16  
 Application Dispatcher (SAS/IntrNet) 51  
 Application Server access method (SAS/IntrNet) 51  
 arrays, multidimensional (Windows) 15  
 ASYNCHIO system option (OS/390) 21  
 ATTRC function (SCL) 33  
 AUDIT statement, DATASETS procedure 10  
 AUTOEXEC= system option (OS/390) 21  
 AUTOSIGNON= system option (SAS/CONNECT) 35  
 AWSMENUMERGE system option  
     OS/2 environment 19  
     Windows environment 16

## B

Baan 29  
 base SAS software  
     features and language elements 3  
     procedures 9  
 Bayesian Vector Autoregressive model (SAS/ETS) 40  
 Bayesian Vector Error Correction Model (SAS/ETS) 40  
 bill-of-material processing (SAS/OR) 55  
 biplots (SAS/INSIGHT) 48  
 BLKSIZE= system option (OpenVMS) 17  
 BOM procedure (SAS/OR) 55  
 box plots (SAS/QC) 58  
 BUFSIZE= data set option 4  
 BUFSIZE= system option 4  
 bulk loading  
     DB2 under UNIX and PC hosts 27  
     OLE DB 28  
     ORACLE 28  
     SAS/Warehouse Administrator software 67  
     Teradata 28

## C

CACHESIZE= system option (OpenVMS) 17  
 CALL routines 6  
     message queue library 49  
     OS/2 environment 19  
     UNIX environments 23  
     Windows environment 15  
 CALL SLEEP routine 6  
     UNIX environments 23  
 CALL STREAMINIT routine 6  
 CATMOD procedure (SAS/STAT) 61  
 CIMPORT procedure 9  
 classes (SAS/AF) 31  
 CMS environment 13  
     EMAIL device type 13  
     statements 13  
     system options 13  
 COLLAPSE ALL command (ODS) 4  
 commands  
     My Favorite Folders window 5  
     OpenVMS environment 17  
     SAS/GRAPH software 43  
     SAS/IntrNet software 51  
     UNIX environments 23  
     Windows environment 16  
 CONVEXIT function (SAS/IML) 45  
 COPY procedure 9  
 COPY statement, DATASETS procedure 10  
 COPYITEM command 5  
 CPM procedure (SAS/OR) 55  
 CPORT procedure 9  
 CREATE TABLE statement, SQL procedure 11  
 CRSPAccess database 39  
 CTRANSLOC= system option  
     OS/390 environment 21  
     SAS/CONNECT software 35

## D

Data Administration window (SAS/GIS) 41  
 Data Definition Language (DDL) 28  
 data set options 6  
 data sets, holding in memory 4  
 DATASETS procedure 9  
     AUDIT statement 10  
     COPY statement 10  
     IC CREATE statement 9  
     MODIFY statement 10

DB2  
     under UNIX and PC hosts 27  
     under VM 27  
 DBCSTYPE= system option (OpenVMS) 18  
 DBMS tables, loading (SAS/Warehouse Administrator) 67  
 DDL (Data Definition Language) 28  
 DELETEITEM command 5  
 DESCRIPTION= option (SAS/GRAPH) 43  
 distribution analyses (SAS/INSIGHT) 47  
 drill-down capability (SAS/GRAPH) 44  
 DTDATW format 6  
 DTREE procedure (SAS/OR) 55  
 DURATION function (SAS/IML) 45

## E

EMAIL device type (CMS) 13  
 EMAILHOST= system option 7  
     OS/390 environment 21  
 EMAILPORT= system option 7  
     OS/390 environment 21  
 EMAILSYS= system option  
     OpenVMS environment 17  
     OS/2 environment 19  
     OS/390 environment 21  
     Windows environment 16  
 encryption (SAS/IntrNet) 51  
 engines  
     SASECRSP (SAS/ETS) 39  
     XML 4  
 error messages, for integrity constraints 5  
 EXPAND ALL command (ODS) 4  
 EXPORT procedure 10

## F

FACTOR procedure (SAS/STAT) 61  
 File Shortcut Assignment window 6  
 FILE statement (OS/390) 21  
 FILEDIALOG function (SCL) 33  
 FILENAME statement  
     CMS environment 13  
     OS/390 environment 21  
 FILEOPEN command (Windows) 16  
 filerefs 6  
 financial functions (SAS/IML) 45  
 firewall configuration (SAS/CONNECT) 35

FIRSTOBS= data set option 5  
 FIRSTOBS= system option 5  
 fit analyses (SAS/INSIGHT) 47  
 FONTLIST command (SAS/GRAPH) 43  
 formats 6  
 FORWARD function (SAS/IML) 45  
 FREQ procedure (SAS/STAT) 62  
 FREQ procedure, TABLES statement 10  
 functions 6  
   financial functions (SAS/IML) 45  
   multivariate time series (SAS/IML) 45  
   OS/2 environment 19  
   SCL (SAS Component Language) 33  
   Windows environment 15

## G

GAM procedure (SAS/STAT) 62  
 GANTT procedure (SAS/OR) 56  
 GCHART procedure (SAS/GRAPH) 43  
 generalized additive models (SAS/STAT) 62  
 global statements 7  
 GPLOT procedure (SAS/GRAPH) 44  
 GRADAR procedure (SAS/GRAPH) 44  
 GUI (Windows) 15

## H

HELPCASE system option (OS/390) 21

## I

IC CREATE statement, DATASETS procedure 9  
 ICCREATE function (SCL) 33  
 ICDESCRIBE function (SCL) 33  
 image files (SAS/GRAPH) 43  
 Import/Export Wizard 6  
 IMPORT procedure 10  
 INFILE statement (OS/390) 21  
 INSERT system option  
   OS/2 environment 19  
   Windows environment 16  
 Integration Technologies 49  
 integrity constraints 5  
 Interior Point algorithm (SAS/OR) 56  
 INTPOINT procedure (SAS/OR) 56  
 Investment Analysis application (SAS/ETS) 39  
 iterative proportional fitting (SAS/STAT) 61

## J

Java clients 49  
 Java tools (SAS/IntrNet) 51

## L

LDAP security interface (SAS/Warehouse Administrator) 69  
 LOADMEMSIZE system option  
   OS/2 environment 19  
   UNIX environments 23

Windows environment 16  
 LOCK statement 5  
 locking SAS files 5  
 LOESS procedure (SAS/STAT) 62  
 LOGISTIC procedure (SAS/STAT) 63  
 LTS call (SAS/IML) 45

## M

macros  
   R/3 29  
   Windows environment 16  
 maps (SAS/GIS) 41  
 MAXMEMQUERY system option  
   OS/2 environment 19  
   Windows environment 16  
 MCD call (SAS/IML) 45  
 MEMSIZE system option (UNIX) 23  
 MESSAGE= option, SQL procedure 5  
 message queue library (CALL routines) 49  
 MESSAGEBOX function (SCL) 33  
 Metadata API (SAS/Warehouse Administrator) 68  
 MI procedure (SAS/STAT) 63  
 MIANALYZE procedure (SAS/STAT) 63  
 Microsoft SQL Server 27  
 MIXED procedure (SAS/STAT) 64  
 MODECLUS procedure (SAS/STAT) 64  
 MODIFY statement, DATASETS procedure 10  
 MOVEITEM command 5  
 Moving-Average with exogenous regressors  
   model (SAS/ETS) 40  
 multidimensional arrays (Windows) 15  
 multiple imputation (SAS/STAT) 63  
 multivariate analyses (SAS/INSIGHT) 47  
 multivariate time series analysis (SAS/IML) 45  
 MULTTEST procedure (SAS/STAT) 64  
 My Favorite Folders window 5

## N

n-way effect plots (SAS/QC) 57  
 NETDRAW procedure (SAS/OR) 56  
 NETFLOW procedure (SAS/OR) 56  
 NLMIXED procedure (SAS/STAT) 64  
 NOLOG system option  
   OS/2 environment 19  
   Windows environment 16  
 NOTERMINAL system option  
   OS/2 environment 19  
   Windows environment 16

## O

OBS= data set option 5  
 OBS= system option 5  
 ODBC 28  
 ODS (Output Delivery System) 3  
 ODS HTML statement 4  
 ODS PRINTER statement 3  
 ODS PROCTITLE statement 3  
 ODS RTF statement 3  
 OLE DB 28

OpenVMS environment 17  
   commands 17  
   data set options 17  
   SAS Session Manager 18  
   system options 17  
   transport files 17  
 ORACLE 28  
 Oracle Rdb 28  
 OS/2 environment 19  
   CALL routines 19  
   functions 19  
   system options 19  
 OS/390 environment 21  
   ADABAS on 29  
   statements 21  
   system options 21  
 Output Delivery System (ODS) 3  
 OUTREP= data set option (OpenVMS) 17

## P

Package Retriever 49  
 PC File Formats 29  
 performance improvement  
   access methods 4  
   holding data sets in memory 4  
 PHREG procedure (SAS/STAT) 64  
 PM procedure (SAS/OR) 56  
 previewing SAS windows 5  
 PRINT procedure 10  
 printing  
   SAS windows 5  
   Universal Printing 5  
 PROBLOT statement, UNIVARIATE procedure 11  
 PROC GIS statement 41  
 procedures 9  
   SAS/ETS software 39  
   SAS/GRAPH software 43  
   SAS/OR software 55  
   SAS/QC software 57  
   SAS/STAT software 61  
 PV function (SAS/IML) 45

## Q

QLIM procedure (SAS/ETS) 39  
 QQPLOT statement, UNIVARIATE procedure 11  
 Qualitative and Limited dependent variable model  
   (SAS/ETS) 39

## R

R/3 29  
 radar charts (SAS/GRAPH) 44  
 RAND function 6  
 random number generation 6  
 RATES function (SAS/IML) 45  
 Rdb 28  
 REDIMOPT function (SCL) 33  
 RENAMEITEM command 5  
 REPORT procedure 11

Results window (ODS) 4  
 ridge analysis (SAS/QC) 57  
 robust regression (SAS/IML) 45

## S

SAS/ACCESS software 27  
   ADABAS 29  
   Baan 29  
   DB2 under UNIX and PC hosts 27  
   DB2 under VM 27  
   ODBC 28  
   OLE DB 28  
   ORACLE 28  
   PC File Formats 29  
   R/3 29  
   Rdb 28  
   relational interfaces 27  
   SQL Server 27  
   Teradata 28  
 SAS/AF software, classes 31  
 SAS Component Language, functions 33  
 SAS/CONNECT software 35  
   Driver for Java 51  
   firewall configuration 35  
   system options 35  
 SAS DATA step, suspending 6  
 SAS/EIS software 37  
 SAS/ETS software 39  
   Investment Analysis application 39  
   QLIM procedure 39  
   SASECRSP engine 39  
   VARMAX procedure 40  
   X12 procedure 40  
 SAS files, locking 5  
 SAS/GIS software 41  
   Data Administration window 41  
   displayed maps 41  
   PROC GIS statement 41  
 SAS/GRAPH software 43  
   DESCRIPTION= option 43  
   drill-down capability 44  
   FONTLIST command 43  
   GCHART procedure 43  
   GPLOT procedure 44  
   GRADAR procedure 44  
   image files in 43  
   SYMBOL statement 44  
 SAS I/O  
   access methods performance 4  
   holding data sets in memory 4  
   integrity constraints 5  
   locking SAS files 5  
   WHERE expressions 5  
   XML documents 4  
 SAS/IML software 45  
   financial functions 45  
   multivariate time series analysis 45  
   robust regression 45  
 SAS/INSIGHT software 47  
   distribution analyses 47  
   fit analyses 47  
   graphical enhancements 47  
   multivariate analyses 47  
 SAS Integration Technologies 49  
 SAS/IntrNet software 51  
   Application Dispatcher 51  
   encryption 51  
   Java tools 51  
 SAS/MDDDB Server software 53  
 SAS OLAP Server software 53  
 SAS/OR software 55  
   BOM procedure 55  
   CPM procedure 55  
   DTREE procedure 55  
   GANTT procedure 56  
   INTPOINT procedure 56  
   NETDRAW procedure 56  
   NETFLOW procedure 56  
   PM procedure 56  
 SAS/QC software 57  
   ADX interface 57  
   box plots 58  
   n-way effect plots 57  
   ridge analysis 57  
   SHEWHART procedure 57  
   split-plot designs 57  
 SAS Session Manager (OpenVMS) 18  
 SAS/SHARE software 59  
   cross-platform catalog access 59  
   data provider 59  
   Driver for Java 51  
   overlapping I/O 59  
 SAS/STAT software 61  
   CATMOD procedure 61  
   FACTOR procedure 61  
   FREQ procedure 62  
   GAM procedure 62  
   LOESS procedure 62  
   LOGISTIC procedure 63  
   MI procedure 63  
   MIANALYZE procedure 63  
   MIXED procedure 64  
   MODECLUS procedure 64  
   MULTTEST procedure 64  
   NLMIXED procedure 64  
   PHREG procedure 64  
   SURVEYMEANS procedure 65  
 SAS/Warehouse Administrator software 67  
   LDAP security interface 69  
   loading DBMS tables 67  
   Metadata API 68  
   user interface 67  
 SAS windows, printing or previewing 5  
 SASCC macro (Windows) 16  
 SASCMD= system option (SAS/CONNECT) 35  
 SASECRSP engine (SAS/ETS) 39  
 SASFILE statement 4  
 SCL functions 33  
 SEQENGINE= system option (OS/390) 21  
 sequential access, performance of 4  
 SHEWHART procedure (SAS/QC) 57  
 Simple Mail Transport Protocol (SMTP) 7  
 SLEEP function 6  
   OS/2 environment 19  
   Windows environment 15  
 SMTP (Simple Mail Transport Protocol) 7  
 SORT procedure 11  
 SORTPGM= system option (OpenVMS) 18  
 SORTSIZE system option (UNIX) 23  
 SOURCE statement, TEMPLATE procedure 4  
 split-plot designs (SAS/QC) 57

SPOT function (SAS/IML) 45  
 SQL/DS 27  
 SQL procedure 11  
   ALTER TABLE statement 11  
   CREATE TABLE statement 11  
   MESSAGE= option 5  
 SQL Procedure Pass-Through Facility 28  
 SQL Server 27  
 statements 7  
   CMS environment 13  
   ODS 3  
   OS/390 environment 21  
 STATISTICS command (SAS/IntrNet) 51  
 stored process paths 49  
 SURVEYMEANS procedure (SAS/STAT) 65  
 suspending SAS DATA step 6  
 SYMBOL statement (SAS/GRAPH) 44  
 system options 7  
   CMS environment 13  
   OS/2 environment 19  
   OS/390 environment 21  
   SAS/CONNECT software 35  
   UNIX environments 23  
   Windows environment 16

## T

TABLES statement, FREQ procedure 10  
 TCP/IP connections (SAS/CONNECT) 35  
 TEMPLATE procedure, SOURCE statement 4  
 Teradata 28  
 time series, seasonal adjustments (SAS/ETS) 40  
 TPU command (OpenVMS) 17  
 transport files (OpenVMS) 17

## U

UNIVARIATE procedure 11  
   PROBPLOT statement 11  
   QQPLOT statement 11  
 Universal Printing 5  
 UNIVERSALPRINT system option 5  
   OpenVMS environment 18  
 UNIX environments 23  
   CALL routines 23  
   commands 23  
   system options 23

## V

VARMACOV call (SAS/IML) 45  
 VARMALIK call (SAS/IML) 45  
 VARMASIM call (SAS/IML) 45  
 VARMAX procedure (SAS/ETS) 40  
 Vector AutoRegressive model (SAS/ETS) 40  
 Vector Error Correction Model (SAS/ETS) 40  
 VNORMAL call (SAS/IML) 45  
 VTSROOT call (SAS/IML) 45

## W

WAUTOSAVE command (Windows) 16

WebDAV transport method 49

WHERE expressions 5

WHERE statement 7

windowing environment

File Shortcut Assignment window 6

Import/Export Wizard 6

My Favorite Folders window commands 5

Windows 2000 environment 16

Windows environment 15

application integration 15

CALL routines 15

commands 16

functions 15

GUI 15

SCL multidimensional arrays 15

system options 16

Windows NT environment 16

WNEWTITLE command (Windows) 16

WNEXTEDIT command (Windows) 16

WORKCACHE= system option (OpenVMS) 18

WSTATUSLN command (Windows) 16

## X

X command (OpenVMS) 17

X12 procedure (SAS/ETS) 40

XML documents 4

XML engine 4

## Y

YIELD function (SAS/IML) 45

# Your Turn

---

If you have comments or suggestions about *What's New in SAS® Software for Release 8.1*, please send them to us on photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute  
Publications Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** yourturn@sas.com

For suggestions about the software, please return the photocopy to

SAS Institute  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** yourturn@sas.com



*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS® Publications Is Easy to Reach**

**Visit our SAS Publications Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user groups conferences**
- **trade show sites and dates**
- **authoring opportunities**
- **custom textbooks**

***Order books with ease at our secured Web page!***

**Explore all the services that Publications has to offer!**

### **Your Listserv Subscription Brings the News to You Automatically**

Do you want to be among the first to learn about the latest books and services available from SAS Publications? Subscribe to our listserv **newdocnews-l** and automatically receive the following once each month: a description of the new titles, the applicable environments or operating systems, and the applicable SAS release(s). To subscribe:

- 1.** Send an e-mail message to **listserv@vm.sas.com**
- 2.** Leave the "Subject" line blank
- 3.** Use the following text for your message:

**subscribe newdocnews-l *your-first-name your-last-name***

For example: subscribe newdocnews-l John Doe

**Please note:** newdocnews-l ←—— that's the letter "l" not the number "1".

For customers outside the U.S., contact your local SAS office for listserv information.



## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Publications Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## Read All about It in Authorline®!

Our User Publishing newsletter, *Authorline*, features author interviews, conference news, and informational updates and highlights from our User Publishing Program. Published quarterly, *Authorline* is available free of charge. To subscribe, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## See Observations®, Our Online Technical Journal

Feature articles from *Observations®: The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on any book orders placed during the course. Each course has a list of recommended books to choose from, and the books are displayed for you to see. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.



# SAS OnlineDoc® Version 8

Just a click away

## SAS Institute is proud to announce a new way to access SAS® software documentation!

One of the challenges you face in today's fast-paced work environment is managing large amounts of information. What you need is an electronic source with a powerful search engine to help you find the SAS reference material you use most. SAS OnlineDoc, delivered on CD-ROM, provides you with the SAS System reference documentation that you need when you need it!

### Overview

Using SAS OnlineDoc, Version 8, all SAS users can have quick access to SAS reference documentation: descriptions of SAS language elements — functions, statements, procedures, options, and so on — as well as product interfaces and displays, and software concepts. SAS OnlineDoc is designed for multiple users to access easily on an intranet. Individual users can access SAS OnlineDoc by installing it to disk or running it from a CD drive.

### Requirements

- 335 MB of disk space
- Microsoft Internet Explorer 4.01 or higher, or Netscape Navigator 4.08 for Windows and UNIX platforms (Go to [www.sas.com/pubs](http://www.sas.com/pubs) to see requirements for other platforms.)
- for intranet installation, a browser that supports the CGI standard

### Distribution Information

One copy of SAS OnlineDoc is shipped FREE with Version 8 software. You can copy and distribute SAS OnlineDoc in the same manner that you copy and distribute your SAS software media as outlined in the license agreement for this product.

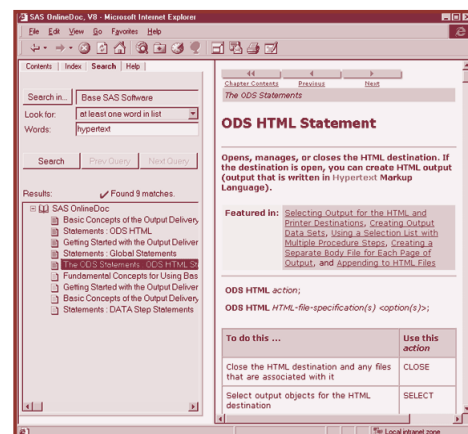
### Printing

If you would like to print selected chapters from SAS OnlineDoc, Version 8, you can purchase a version of the CD that contains PDF files. Like the free version of SAS OnlineDoc that you receive with your Version 8 software, the version that contains PDF files can be installed on an internal web or it can be accessed directly from a local CD drive on a PC. If you install it on an internal web, then multiple users will be able to print chapters of the documentation.

### Features

- Search - full-text search facility
- Master Index - comprehensive index as well as individual book indexes
- Help - includes information about this CD and the license agreement
- Quick Tour - a tutorial on using the CD

- Get the Latest News - subscribe to a listserv to get monthly updates about publications
- What's New - contains a link to the changes and enhancements for Version 8
- Hardcopy Documentation Update - complete list of hardcopy books that apply to Version 8
- Table of Contents - expands and contracts for Java-enabled browsers
- Non-Java Table of Contents - for browsers without Java support



"This product gives us faster access to information for the entire corporation. It has set a new standard for software documentation at GTE!"

Steve Phillipy of Decision Consultants, Inc. works with GTE Enterprise Information Technology Service in Computer Support Services.

# SAS OnlineDoc® Version 8

“SAS OnlineDoc is as good or better than the best online documentation products I’ve seen. It’s easy to use, complete, accurate, intuitive, and well organized. It’s definitely a first-class product.”

Larry Lutz, SAS® Programmer, Systems Applications, Inc., Vienna, Virginia, USA

The following books are on the SAS OnlineDoc, Version 8 CD-ROM:

## Base SAS

SAS® Language Reference: Dictionary  
SAS® Procedures Guide  
SAS® Macro Language: Reference  
SAS® SQL Query Window User’s Guide  
Moving and Accessing SAS® Files across Operating Environments  
The Complete Guide to the SAS® Output Delivery System  
Host Companions for these environments:  
CMS  
OS/390®  
OS/2®  
OpenVMS  
Microsoft Windows  
UNIX

## SAS/ACCESS

SAS/ACCESS® Software for PC File Formats: Reference  
SAS/ACCESS® Software for Relational Databases: Reference (chapters include: CA-OpenIngres, DB2® under UNIX and PC Hosts and DB2® under OS/390®, Informix, ODBC, OLE DB, ORACLE®, Oracle Rdb™, SYBASE®, and Teradata)

SAS/ACCESS® Interface to SYSTEM 2000® Data Management Software  
SAS/ACCESS® Interface to CA-IDMS  
SAS/ACCESS® Interface to CA-DATACOM/DB  
SAS/ACCESS® Interface to IMS-DL/I  
SAS/ACCESS® Interface To ADABAS

## SAS/AF

SAS/AF® Software: Class Dictionary  
SAS/AF® Software Procedure Guide  
SAS® Component Language: Reference

## SAS/ASSIST

Getting Started with the SAS® System  
Using SAS/ASSIST® Software  
Doing More with SAS/ASSIST® Software

## SAS/CONNECT and SAS/SHARE

SAS/CONNECT® User’s Guide  
SAS/SHARE® User’s Guide  
Communications Access Methods for SAS/SHARE® and SAS/CONNECT® Software

## SAS/EIS

SAS/EIS® Software: Administrator’s Guide

## SAS/ETS

SAS/ETS® User’s Guide

## SAS/FSP

SAS/FSP® Software Procedures Guide

## SAS/GIS

Working with Spatial Data Using SAS/GIS® Software

## SAS/GRAPH

SAS/GRAPH® Software: Reference

## SAS/IML

SAS/IML® User’s Guide

## SAS/INSIGHT

SAS/INSIGHT® User’s Guide

## SAS/MDDB

SAS/MDDB® Server Administrator’s Guide

## SAS® ODBC Driver

SAS® ODBC Driver: User’s Guide and Programmer’s Reference

## SAS/OR

SAS/OR® User’s Guide: Mathematical Programming  
SAS/OR® Software: Project Management  
SAS/OR® Software: The netWorks Application  
SAS/OR® Software: The QSIM Application

## SAS/QC

SAS/QC® User’s Guide,

## SAS/SPECTRAVIEW

SAS/SPECTRAVIEW® Software User’s Guide

## SAS/STAT

SAS/STAT® User’s Guide  
The Analyst Application, First Edition

## Software, Operating Systems, and Releases Addressed

Base SAS®, SAS/ACCESS®, SAS/AF®, SAS/ASSIST®, SAS/CONNECT®, SAS/EIS®, SAS/ETS®, SAS/GIS®, SAS/GRAPH®, SAS/IML®, SAS/INSIGHT®, SAS/MDDB®, SAS/OR®, SAS/QC®, SAS/SHARE®, SAS/SPECTRAVIEW®, SAS/STAT®, SAS® ODBC software, and host companions.

## SAS OnlineDoc, Version 8 with PDF Files

Win95, Win98, WinNT, AIX/6000®, Compaq Tru64  
UNIX, HP-UX, Solaris, Intel ABI, IRIX, OpenVMS Alpha,  
OpenVMS VAX, OS/2®

Version 8

ISBN: 1-58025-591-4

Order No. P57717

## SAS OnlineDoc, Version 8 with PDF Files - EBCDIC

CMS, MVS, OS/390®

Version 8

ISBN: 1-58025-596-5

Order No. P57735

For more information: [www.sas.com/pubs](http://www.sas.com/pubs)



## **SAS OnlineTutor™** **for Version 8: SAS® Programming**

Save money, save time, savor your  
breakfast while you learn the SAS® System.

According to industry experts, technology-based  
training systems are hot. Now, the world leader in information  
delivery introduces you to an exciting new training experience.

SAS OnlineTutor combines the instructional quality you expect from  
SAS Institute with the benefits of web-based training.

[www.sas.com/tutor](http://www.sas.com/tutor)







**SAS OnlineTutor™ for Version 8** brings training to your desktop, requiring only a standard Web browser! SAS OnlineTutor is a library of 35 lessons that teach you how to use the Version 8 SAS environment and how to write programming code to access, manage, analyze, and present data. Whether you're a beginner or an experienced SAS programmer, SAS OnlineTutor has lessons for you, ranging from basic concepts to features such as producing drill-down graphs and enhanced tables with HTML output.

### Introduction to SAS Programming

- Working in the SAS Environment
- Basic Concepts
- Editing and Debugging SAS Programs
- Setting Up Your SAS Session

### Producing Basic Reports

- Creating List Reports
- Creating Tabular Reports
- Creating Plots
- Creating Bar and Pie Charts
- Creating Drill-Down Graphs in HTML
- Performing Queries Using SQL

### Creating SAS Data Sets

- Reading Raw Data
- Reading and Concatenating SAS Data Sets
- Merging SAS Data Sets
- Accessing Other Vendors' DBMS Data

### Reading Various Types of Raw Data

- Understanding DATA Step Processing
- Reading Raw Data in Fixed Fields
- Reading Free-Format Data
- Reading Date and Time Values
- Creating a Single Observation from Multiple Records
- Creating Multiple Observations from a Single Record
- Reading Hierarchical Files
- Reading Variable-Length Records

### Creating and Modifying Variables

- Creating Variables
- Transforming Data with SAS Functions
- Formatting Variable Values
- Improving Program Efficiency with Macro Variables

### DATA Step Processing

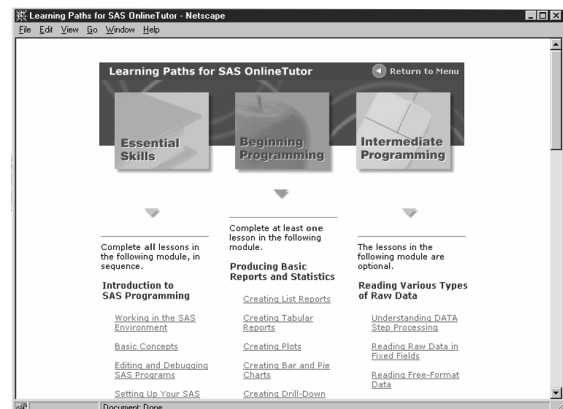
- Debugging and Testing DATA Steps
- Generating Data with DO Loops
- Processing Variables with Arrays

### Producing Descriptive Statistics

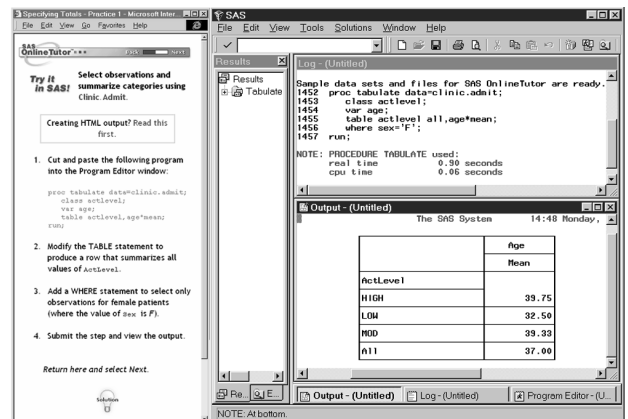
- Computing Statistics for Numeric Variables
- Computing Frequency Distributions

### Enhancing and Storing Output

- Enhancing Reports
- Enhancing and Exporting Charts and Plots
- Producing HTML Output
- Enhancing HTML Tabular Reports



Suggested learning paths



Practice in the actual SAS environment

Includes a Q&A page and a link to the experts at SAS Institute who can answer your questions about the lessons.



### System Recommendations

- Microsoft Internet Explorer 4.0 or higher or Netscape Navigator 4.0 or higher
- Java for full-text search functionality
- You must have Version 8 of the SAS System to complete the practice exercises

### Pricing

- Personal 30-day Web License - \$199 USD per user
- [www.sas.com/tutor](http://www.sas.com/tutor) to get started

### CD-ROM Annual License

To implement SAS OnlineTutor department-wide, contact your SAS Training Services Account Representative at 919.677.8000, Ext. 7321.

# **SAS/ETS<sup>®</sup> Software: Changes and Enhancements, Release 8.1**

---

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ETS® Software: Changes and Enhancements, Release 8.1*, Cary, NC: SAS Institute Inc., 2000

**SAS/ETS® Software: Changes and Enhancements, Release 8.1**

Copyright © 2000 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-657-0

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, May 2000

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. indicates USA registration.

IBM® and all other International Business Machines Corporation product or service names are registered trademarks or trademarks of International Business Machines Corporation in the USA and other countries.

Oracle® and all other Oracle Corporation product or service names are registered trademarks or trademarks of Oracle Corporation in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

|                                                    |                |
|----------------------------------------------------|----------------|
| <b>Part 1. General Information</b>                 | <b>1</b>       |
| Chapter 1. New Financial Functions . . . . .       | 3              |
| Chapter 2. The SASECRSP Interface Engine . . . . . | 11             |
| Chapter 3. The SASEFAME Interface Engine . . . . . | 21             |
| <br><b>Part 2. Procedure Reference</b>             | <br><b>39</b>  |
| Chapter 4. The VARMAX Procedure . . . . .          | 41             |
| Chapter 5. The X12 Procedure . . . . .             | 161            |
| <br><b>Part 3. Investment Analysis</b>             | <br><b>185</b> |
| Chapter 6. Overview . . . . .                      | 187            |
| Chapter 7. Portfolios . . . . .                    | 193            |
| Chapter 8. Investments . . . . .                   | 201            |
| Chapter 9. Computations . . . . .                  | 241            |
| Chapter 10. Analyses . . . . .                     | 253            |
| Chapter 11. Details . . . . .                      | 269            |
| <br><b>Subject Index</b>                           | <br><b>283</b> |
| <br><b>Syntax Index</b>                            | <br><b>285</b> |





# Part 1

## General Information

### Table of Contents

---

|                                                    |    |
|----------------------------------------------------|----|
| Chapter 1. New Financial Functions . . . . .       | 3  |
| Chapter 2. The SASECRSP Interface Engine . . . . . | 11 |
| Chapter 3. The SASEFAME Interface Engine . . . . . | 21 |

*Part 1. General Information*

# Chapter 1

## New Financial Functions

### Chapter Table of Contents

---

|                                     |   |
|-------------------------------------|---|
| <b>TIMEVALUE FUNCTION</b> . . . . . | 5 |
| Overview . . . . .                  | 5 |
| Syntax . . . . .                    | 5 |
| Details . . . . .                   | 5 |
| Examples . . . . .                  | 5 |
| <b>SAVINGS FUNCTION</b> . . . . .   | 6 |
| Overview . . . . .                  | 6 |
| Syntax . . . . .                    | 6 |
| Details . . . . .                   | 7 |
| Examples . . . . .                  | 7 |
| <b>EFFRATE FUNCTION</b> . . . . .   | 7 |
| Overview . . . . .                  | 7 |
| Syntax . . . . .                    | 8 |
| Details . . . . .                   | 8 |
| Examples . . . . .                  | 8 |
| <b>NOMRATE FUNCTION</b> . . . . .   | 8 |
| Overview . . . . .                  | 8 |
| Syntax . . . . .                    | 8 |
| Details . . . . .                   | 9 |
| Examples . . . . .                  | 9 |

*Part 1. General Information*

# Chapter 1

## New Financial Functions

---

### TIMEVALUE Function

---

#### Overview

The TIMEVALUE function returns the equivalent of a reference amount at a base date using variable interest rates. TIMEVALUE computes the time-value equivalent of a date-amount pair at a specified date.

---

#### Syntax

**TIMEVALUE**( *BaseDate*, *ReferenceDate*, *ReferenceAmount*, *Interval*, *Date-1*, *Rate-1* [ , *Date-2*, *Rate-2*, ... ] )

#### Description

|                        |                                                                                                                                      |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <i>BaseDate</i>        | is a SAS date. The returned value is the time value of <i>ReferenceAmount</i> at <i>BaseDate</i> .                                   |
| <i>ReferenceDate</i>   | is a SAS date. This is the date of <i>ReferenceAmount</i> .                                                                          |
| <i>ReferenceAmount</i> | is a numeric. This is the amount at <i>ReferenceDate</i> .                                                                           |
| <i>Interval</i>        | is a SAS interval. This is the compounding interval.                                                                                 |
| <i>Date-i</i>          | is a SAS date. Each date is paired with a rate. The date <i>Date-i</i> is the time that <i>Rate-i</i> takes effect.                  |
| <i>Rate-i</i>          | is a numeric percentage. Each rate is paired with a date. The rate <i>Rate-i</i> is the interest rate that starts on <i>Date-i</i> . |

---

#### Details

- The list of date-rate pairs does not need to be sorted by date.
- When more than one date-rate pair is entered, SAS sorts dates chronologically and ignores duplicates (keeping the last occurrence of the date).
- Simple interest is applied for partial periods.
- There must be a valid date-rate pair whose date is at or prior to both the *ReferenceDate* and *BaseDate*.

---

#### Examples

You can express the accumulated value of an investment of \$1,000 at a nominal interest rate of 10% compounded monthly for one year as

## Part 1. General Information

```
amount_base1 = TIMEVALUE ("01jan2000"d, "01jan2001"d, 1000,  
                           "MONTH", "01jan2000"d, 10);
```

If the interest rate jumps to 20% halfway through the year, the resulting calculation would be

```
amount_base2 = TIMEVALUE ("01jan2000"d, "01jan2001"d, 1000,  
                           "MONTH",  
                           "01jan2000"d, 10, "01jul2000"d, 20);
```

Recall that the date-rate pairs do not need to be sorted by date. This flexibility allows amount\_base2 and amount\_base3 to assume the same value.

```
amount_base3 = TIMEVALUE ("01jan2000"d, "01jan2001"d, 1000,  
                           "MONTH",  
                           "01jul2000"d, 20, "01jan2000"d, 10);
```

---

# SAVINGS Function

---

## Overview

The SAVINGS function returns the balance of a uniform, periodic savings using variable interest rates. SAVINGS computes the accumulated balance of a savings account. It assumes uniform deposits and uniform compounding of interest.

---

## Syntax

**SAVINGS**( *BaseDate*, *DepositDate*, *DepositAmount*, *DepositNumber*, *DepositInterval*, *Interval*, *Date-1*, *Rate-1* [ , *Date-2*, *Rate-2*, ... ] )

## Description

|                        |                                                                                                                                      |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <i>BaseDate</i>        | is a SAS date. The returned value is the balance of the savings at <i>BaseDate</i> .                                                 |
| <i>DepositDate</i>     | is a SAS date. This is the date of the first deposit.                                                                                |
| <i>DepositAmount</i>   | is a numeric. All deposits are assumed constant. This is the value of each deposit.                                                  |
| <i>DepositNumber</i>   | is a positive integer. This is the number of deposits.                                                                               |
| <i>DepositInterval</i> | is a SAS interval. This is the frequency at which you make deposits.                                                                 |
| <i>Interval</i>        | is a SAS interval. This is the compounding interval.                                                                                 |
| <i>Date-i</i>          | is a SAS date. Each date is paired with a rate. The date <i>Date-i</i> is the time that <i>Rate-i</i> takes effect.                  |
| <i>Rate-i</i>          | is a numeric percentage. Each rate is paired with a date. The rate <i>Rate-i</i> is the interest rate that starts on <i>Date-i</i> . |

---

## Details

- The list of date-rate pairs does not need to be given in chronological order.
- When more than one date-rate pair is entered, SAS sorts dates chronologically and ignores duplicates (keeping the last occurrence of the date).
- Simple interest is applied for partial periods.
- There must be a valid date-rate pair whose date is at or prior to both the *DepositDate* and *BaseDate*.

---

## Examples

Suppose you deposit \$300 monthly for two years into an account that compounds quarterly at an annual rate of 4%. The balance of the account after five years can be expressed as

```
amount_base1 = SAVINGS ("01jan2005"d, "01jan2000"d, 300, 24,
                        "MONTH", "QUARTER", "01jan2000"d, 4.00);
```

Suppose the interest rate increases by a quarter-point each year. Then the balance of the account could be expressed as

```
amount_base2 = SAVINGS ("01jan2005"d, "01jan2000"d, 300, 24,
                        "MONTH", "QUARTER", "01jan2000"d, 4.00,
                        "01jan2001"d, 4.25, "01jan2002"d, 4.50,
                        "01jan2003"d, 4.75, "01jan2004"d, 5.00);
```

If you want to know the balance after one year of deposits, the following statement sets `amount_base3` to the desired balance.

```
amount_base3 = SAVINGS ("01jan2001"d, "01jan2000"d, 300, 24,
                        "MONTH", "QUARTER", "01jan2000"d, 4);
```

Recall that SAS ignores deposits after the base date, so the deposits after the *ReferenceDate* do not affect the returned value.

---

## EFFRATE Function

---

### Overview

The EFFRATE function returns the effective annual interest rate. EFFRATE computes the effective annual interest rate corresponding to a nominal annual interest rate.



---

## Syntax

**EFFRATE**( *Interval*, *Rate* )

### Description

*Interval* is a SAS interval. This is how often *Rate* compounds.

*Rate* is a numeric. This is a nominal annual interest rate (expressed as a percentage) that is compounded each *Interval*.

---

## Details

Consider a nominal interest *Rate* and a compounding *Interval*. Suppose *m* *Intervals* occur in a year. The value returned by **EFFRATE**(*Interval*, *Rate*) equals

$$\left(1 + \frac{Rate}{m}\right)^m - 1$$

---

## Examples

Suppose a nominal rate is 10%. The corresponding effective rate when interest is compounded monthly can be expressed as

```
effective_rate1 = EFFRATE ("MONTH", 10);
```

Again, suppose a nominal rate is 10%. The corresponding effective rate when interest is compounded quarterly can be expressed as

```
effective_rate2 = EFFRATE ("QUARTER", 10);
```

---

## NOMRATE Function

---

### Overview

The **NOMRATE** function returns the nominal annual interest rate. **NOMRATE** computes the nominal annual interest rate corresponding to an effective annual interest rate.

---

## Syntax

**NOMRATE**( *Interval*, *Rate* )

### Description

*Interval* is a SAS interval. This is how often the returned value is compounded.

*Rate* is a numeric. This is an effective annual interest rate (expressed as a percentage) that is compounded each *Interval*.

---

## Details

Consider an effective interest *Rate* and a compounding *Interval*. Suppose *m Intervals* occur in a year. The value returned by `NOMRATE(Interval, Rate)` equals

$$m \left( (1 + \text{Rate})^{\frac{1}{m}} - 1 \right)$$

---

## Examples

Suppose an effective rate is 10% when compounding monthly. The corresponding nominal rate can be expressed as

```
effective_rate1 = NOMRATE ("MONTH", 10);
```

Suppose an effective rate is 10% when compounding quarterly. The corresponding nominal rate can be expressed as

```
effective_rate2 = NOMRATE ("QUARTER", 10);
```

*Part 1. General Information*

## Chapter 2

# The SASECRSP Interface Engine

### Chapter Table of Contents

---

|                                                                   |    |
|-------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                         | 13 |
| <b>GETTING STARTED</b> . . . . .                                  | 13 |
| Structure of a SAS Data Set Containing Time Series Data . . . . . | 13 |
| Reading CRSP Data Files . . . . .                                 | 13 |
| Using the SAS DATA Step . . . . .                                 | 13 |
| Using SAS Procedures . . . . .                                    | 14 |
| <b>SYNTAX</b> . . . . .                                           | 14 |
| The LIBNAME <i>libref</i> SASECRSP Statement . . . . .            | 14 |
| <b>DETAILS</b> . . . . .                                          | 15 |
| The SAS Output Data Set . . . . .                                 | 15 |
| Available Data Sets . . . . .                                     | 16 |
| <b>REFERENCES</b> . . . . .                                       | 20 |



## Chapter 2

# The SASECRSP Interface Engine

---

### Overview

The SASECRSP interface engine enables SAS users to access and process time series data residing in CRSPAccess data files and provides a seamless interface between CRSP and SAS data processing.

The SASECRSP engine uses the LIBNAME statement to enable you to specify which time series you would like to read from the CRSPAccess data files, and how you would like to perform selection on the CRSP set you choose to access. You choose the daily CRSP data by specifying SETID=10, or the monthly CRSP data by specifying SETID=20. You can select which securities you wish to access by specifying PERMNO=*number*, and specify your range of dates for the selected time series by using RANGE=*'begdt-enddt'*, or specify an input SAS data set named *setname* as input for issues with the INSET=*'setname'* option. The SAS Data step can then be used to perform further subsetting and to store the resulting time series into a SAS data set. You can perform more analysis if desired either in the same SAS session or in another session at a later time.

---

### Getting Started

---

#### Structure of a SAS Data Set Containing Time Series Data

SAS requires time series data to be in a specific form that is recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as a time ID variable. The SASECRSP engine provides a time ID variable called DATE.

---

#### Reading CRSP Data Files

The SASECRSP engine supports reading time series, events, and portfolios from CRSPAccess data files. Only the series specified by the CRSP *setid* is written to the SAS data set.

---

#### Using the SAS DATA Step

If desired, you can store the selected series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. Once the data is stored in a SAS data set you can use it as you would any other SAS data set.

---

## Using SAS Procedures

You can print the output SAS data set by using the PRINT procedure and report information concerning the contents of your data set by using the CONTENTS procedure. You can create a view of the CRSPAccess data base by using the SQL procedure to create your view using the SASECRSP engine in your *libref*, along with the USING clause.

---

## Syntax

The SASECRSP engine uses standard engine syntax. Options used by SASECRSP are summarized in the table below.

| Description                                                                                                                                                                                  | Statement                      | Option  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|---------|
| specify a CRSPAccess set id# where 10 designates daily and 20 monthly, which limits the frequency selection of time series that are included in the SAS data set. This is a required option. | LIBNAME <i>libref</i> SASECRSP | SETID=  |
| specify the PERMNO of a security to be kept in the SAS data set.                                                                                                                             | LIBNAME <i>libref</i> SASECRSP | PERMNO= |
| specify the range of CRSP data to access. <i>begdt</i> and <i>enddt</i> specify the range of CRSPAccess data in YYYYMMDD format.                                                             | LIBNAME <i>libref</i> SASECRSP | RANGE=  |
| uses a SAS data set named <i>setname</i> as input for issues.                                                                                                                                | LIBNAME <i>libref</i> SASECRSP | INSET=  |

---

## The LIBNAME *libref* SASECRSP Statement

**LIBNAME** *libref* **SASECRSP** '*physical name*' *options*;

The following options can be used in the LIBNAME *libref* SASECRSP statement:

### **SETID=***crsp\_setidnumber*

specifies the CRSP set ID number where 10 is the daily data set ID and 20 is the monthly data set ID number. Two possible values for *crsp\_setidnumber* are 10 or 20. SETID is a required option for the SASECRSP engine. For more details about the CRSPAccess *crsp\_setidnumber*, refer to “Using Multiple Products” in the *CRSPAccess Database Format Installation Guide*. For a more complete discussion of SASECRSP set names, key fields, and date fields, see Table 2.1 on page 16. As an example, to access monthly data, you would use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'  
SETID=20;
```

**PERMNO=***crsp\_permnumber*

By default, the SASECRSP engine reads all PERMNOs in the CRSPAccess database that you name in your SASECRSP *libref*. You can limit the time series read from the CRSP database by specifying the PERMNO= option on your LIBNAME statement. From a performance standpoint, the PERMNO= option does random access and reads only the data for the PERMNOs listed. There is no limit to the number of *crsp\_permnumber* options that you can use. As an example, to access monthly data for Microsoft Corporation and for International Business Machine Corporation,

```
LIBNAME myLib sasecrsp 'physical-name'  
SETID=20,  
PERMNO=10107,  
PERMNO=12490;
```

**RANGE=***'crsp\_begdt-crsp\_enddt'*

To limit the time range of data read from the CRSPAccess database, specify the RANGE= option in your SASECRSP *libref*, where *crsp\_begdt* is the beginning date in YYYYMMDD format and *crsp\_enddt* is the ending date of the range in YYYYMMDD format. From a performance standpoint, the engine reads all the data for a company and then restricts the data range before passing it back.

As an example, to access daily data for Microsoft Corporation and for International Business Machine Corporation for the first quarter of 2000, use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'  
SETID=20,  
PERMNO=10107,  
PERMNO=12490,  
RANGE='19990101-19990331';
```

**INSET=***'setname[,keyfieldname][,keyfieldtype][,date1field][,date2field]'*

When you specify a SAS data set named *setname* as input for issues, the SASECRSP engine assumes a default PERMNO field containing selected CRSP PERMNOs is present in the data set. If optional parameters are included, the first one is the key name, and the second is the key type. The third optional parameter is the beginning date or event date, and the fourth is the ending date in YYYYMMDD format.

---

## Details

---

### The SAS Output Data Set

You can use the SAS DATA step to write the selected CRSP data to a SAS data set. This enables the user to easily analyze the data using SAS. The name of the output data set is specified by you on the DATA statement. This causes the engine



supervisor to create a SAS data set using the specified name in either the SAS WORK library, or if specified, the USER library. For more about naming your SAS data set, see the section “Characteristics of SAS Data Libraries” in *SAS Language Reference: Dictionary*.

The contents of the SAS data set include the DATE of each observation, the series names of each series read from the CRSPAccess database, event variables, and the label or description of each series/event.

## Available Data Sets

Table 2.1 shows the available data sets provided by the SASECRSP interface. Missing values are represented as ‘.’ in the SAS data set. You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. You can use PROC SQL along with the SASECRSP engine to create a view of your SAS data set.

**Table 2.1.** Data Sets Available

| Dataset                                              | Fields    | Label                                      | Type      |
|------------------------------------------------------|-----------|--------------------------------------------|-----------|
| STKHEAD<br>Header Identification<br>and Summary Data | PERMNO    | PERMNO                                     | Numeric   |
|                                                      | PERMCO    | PERMCO                                     | Numeric   |
|                                                      | COMPNO    | Nasdaq Company Number                      | Numeric   |
|                                                      | ISSUNO    | Nasdaq Issue Number                        | Numeric   |
|                                                      | HEXCD     | Exchange Code Header                       | Numeric   |
|                                                      | HSHRCD    | Share Code Header                          | Numeric   |
|                                                      | HSICCD    | Standard Industrial<br>Classification Code | Numeric   |
|                                                      | BEGDT     | Begin of Stock Data                        | Numeric   |
|                                                      | ENDDT     | End of Stock Data                          | Numeric   |
|                                                      | DLSTCD    | Delisting Code Header                      | Numeric   |
|                                                      | HCUSIP    | CUSIP Header                               | Character |
|                                                      | HTICK     | Ticker Symbol Header                       | Character |
|                                                      | HCOMNAM   | Company Name Header                        | Character |
| NAMES<br>Name History Array                          | PERMNO    | PERMNO                                     | Numeric   |
|                                                      | NAMEDT    | Names Date                                 | Numeric   |
|                                                      | NAMEENDDT | Names Ending Date                          | Numeric   |
|                                                      | SHRCD     | Share Code                                 | Numeric   |
|                                                      | EXCHCD    | Exchange Code                              | Numeric   |
|                                                      | SICCD     | Standard Industrial<br>Classification Code | Numeric   |
|                                                      | NCUSIP    | CUSIP                                      | Numeric   |
|                                                      | TICKER    | Ticker Symbol                              | Character |
|                                                      | COMNAM    | Company Name                               | Character |
|                                                      | SHRCLS    | Share Class                                | Numeric   |
| DISTS<br>Distribution Event<br>Array                 | PERMNO    | PERMNO                                     | Numeric   |
|                                                      | DISTCD    | Distribution Code                          | Numeric   |
|                                                      | DIVAMT    | Dividend Cash Amount                       | Numeric   |
|                                                      | FACPR     | Factor to Adjust Price                     | Numeric   |
|                                                      | FACSHR    | Factor to Adjust Share                     | Numeric   |

Table 2.1. (continued)

| Dataset                                           | Fields    | Label                                      | Type    |
|---------------------------------------------------|-----------|--------------------------------------------|---------|
|                                                   | DCLRDT    | Distribution Declaration Date              | Numeric |
|                                                   | EXDT      | Ex-Distribution Date                       | Numeric |
|                                                   | RCRDDT    | Record Date                                | Numeric |
|                                                   | PAYDT     | Payment Date                               | Numeric |
|                                                   | ACPERM    | Acquiring PERMNO                           | Numeric |
|                                                   | ACCOMP    | Acquiring PERMCO                           | Numeric |
| SHARES<br>Shares Outstanding<br>Observation Array | PERMNO    | PERMNO                                     | Numeric |
|                                                   | SHROUT    | Shares Outstanding                         | Numeric |
|                                                   | SHRSDT    | Shares Outstanding<br>Observation Date     | Numeric |
|                                                   | SHRENDT   | Shares Outstanding<br>Observation End Date | Numeric |
|                                                   | SHRFLG    | Shares Outstanding<br>Observation Flag     | Numeric |
| DELIST<br>Delisting<br>History Array              | PERMNO    | PERMNO                                     | Numeric |
|                                                   | DLSTDT    | Delisting Date                             | Numeric |
|                                                   | DLSTCD    | Delisting Code                             | Numeric |
|                                                   | NWPERM    | New PERMNO                                 | Numeric |
|                                                   | NWCOMP    | New PERMCO                                 | Numeric |
|                                                   | NEXTDT    | Delisting<br>Next Price Date               | Numeric |
|                                                   | DLAMT     | Delisting Amount                           | Numeric |
|                                                   | DLRETX    | Delisting Return<br>Without Dividends      | Numeric |
|                                                   | DLPRC     | Delisting Price                            | Numeric |
|                                                   | DLPDT     | Delisting Amount Date                      | Numeric |
|                                                   | DLRET     | Delisting Return                           | Numeric |
| NASDIN<br>Nasdaq Information<br>Array             | PERMNO    | PERMNO                                     | Numeric |
|                                                   | TRTSCD    | Nasdaq Traits Code                         | Numeric |
|                                                   | TRTSDT    | Nasdaq Traits Date                         | Numeric |
|                                                   | TRTSENDDT | Nasdaq Traits End Date                     | Numeric |
|                                                   | NMSIND    | Nasdaq National Market<br>Indicator        | Numeric |
|                                                   | MMCNT     | Market Maker Count                         | Numeric |
|                                                   | NSDINX    | Nasd Index Code                            | Numeric |
| PRC<br>Price or Bid/Ask<br>Average Time Series    | PERMNO    | PERMNO                                     | Numeric |
|                                                   | CALDT     | Calendar Trading Date                      | Numeric |
|                                                   | PRC       | Price or Bid/Ask Aver                      | Numeric |
| RET<br>Returns Time Series                        | PERMNO    | PERMNO                                     | Numeric |
|                                                   | CALDT     | Calendar Trading Date                      | Numeric |
|                                                   | RET       | Returns                                    | Numeric |
| ASKHI<br>Ask or High<br>Time Series               | PERMNO    | PERMNO                                     | Numeric |
|                                                   | CALDT     | Calendar Trading Date                      | Numeric |
|                                                   | ASKHI     | Ask or High                                | Numeric |

Part 1. General Information

**Table 2.1.** (continued)

| Dataset               | Fields   | Label                  | Type    |
|-----------------------|----------|------------------------|---------|
| BIDLO                 | PERMNO   | PERMNO                 | Numeric |
| Bid or Low            | CALDT    | Calendar Trading Date  | Numeric |
| Time Series           | BIDLO    | Bid or Low             | Numeric |
| BID                   | PERMNO   | PERMNO                 | Numeric |
| Bid Time Series       | CALDT    | Calendar Trading Date  | Numeric |
|                       | BID      | Bid                    | Numeric |
| ASK                   | PERMNO   | PERMNO                 | Numeric |
| Ask Time Series       | CALDT    | Calendar Trading Date  | Numeric |
|                       | ASK      | Ask                    | Numeric |
| RETX                  | PERMNO   | PERMNO                 | Numeric |
| Returns without       | CALDT    | Calendar Trading Date  | Numeric |
| Dividends Time Series | RETX     | Returns w/o Dividends  | Numeric |
| SPREAD                | PERMNO   | PERMNO                 | Numeric |
| Spread Between Bid    | CALDT    | Calendar Trading Date  | Numeric |
| and Ask Time Series   | SPREAD   | Spread Between Bid Ask | Numeric |
| ALTPRC                | PERMNO   | PERMNO                 | Numeric |
| Price Alternate       | CALDT    | Calendar Trading Date  | Numeric |
| Time Series           | ALTPRC   | Price Alternate        | Numeric |
| VOL                   | PERMNO   | PERMNO                 | Numeric |
| Volume Time Series    | CALDT    | Calendar Trading Date  | Numeric |
|                       | VOL      | Volume                 | Numeric |
| NUMTRD                | PERMNO   | PERMNO                 | Numeric |
| Number of Trades      | CALDT    | Calendar Trading Date  | Numeric |
| Time Series           | NUMTRD   | Number of Trades       | Numeric |
| ALTPRCDT              | PERMNO   | PERMNO                 | Numeric |
| Alternate Price Date  | CALDT    | Calendar Trading Date  | Numeric |
| Time Series           | ALTPRCDT | Alternate Price Date   | Numeric |
| PORT1                 | PERMNO   | PERMNO                 | Numeric |
| Portfolio Data for    | CALDT    | Calendar Trading Date  | Numeric |
| Portfolio Type 1      | PORT1    | Portfolio Assignment   | Numeric |
|                       |          | for Portfolio Type 1   | Numeric |
|                       | STAT1    | Portfolio Statistic    | Numeric |
|                       |          | for Portfolio Type 1   | Numeric |
| PORT2                 | PERMNO   | PERMNO                 | Numeric |
| Portfolio Data for    | CALDT    | Calendar Trading Date  | Numeric |
| Portfolio Type 2      | PORT2    | Portfolio Assignment   | Numeric |
|                       |          | for Portfolio Type 2   | Numeric |
|                       | STAT2    | Portfolio Statistic    | Numeric |
|                       |          | for Portfolio Type 2   | Numeric |
| PORT3                 | PERMNO   | PERMNO                 | Numeric |
| Portfolio Data for    | CALDT    | Calendar Trading Date  | Numeric |
| Portfolio Type 3      | PORT3    | Portfolio Assignment   | Numeric |
|                       |          | for Portfolio Type 3   | Numeric |
|                       | STAT3    | Portfolio Statistic    | Numeric |
|                       |          | for Portfolio Type 3   | Numeric |

**Table 2.1.** (continued)

| Dataset                                         | Fields                                | Label                                                                                                                          | Type                                                |
|-------------------------------------------------|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|
| PORT4<br>Portfolio Data for<br>Portfolio Type 4 | PERMNO<br>CALDT<br>PORT4<br><br>STAT4 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 4<br>Portfolio Statistic<br>for Portfolio Type 4 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |
| PORT5<br>Portfolio Data for<br>Portfolio Type 5 | PERMNO<br>CALDT<br>PORT5<br><br>STAT5 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 5<br>Portfolio Statistic<br>for Portfolio Type 5 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |
| PORT6<br>Portfolio Data for<br>Portfolio Type 6 | PERMNO<br>CALDT<br>PORT6<br><br>STAT6 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 6<br>Portfolio Statistic<br>for Portfolio Type 6 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |
| PORT7<br>Portfolio Data for<br>Portfolio Type 7 | PERMNO<br>CALDT<br>PORT7<br><br>STAT7 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 7<br>Portfolio Statistic<br>for Portfolio Type 7 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |
| PORT8<br>Portfolio Data for<br>Portfolio Type 8 | PERMNO<br>CALDT<br>PORT8<br><br>STAT8 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 8<br>Portfolio Statistic<br>for Portfolio Type 8 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |
| PORT9<br>Portfolio Data for<br>Portfolio Type 9 | PERMNO<br>CALDT<br>PORT9<br><br>STAT9 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 9<br>Portfolio Statistic<br>for Portfolio Type 9 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |

---

## References

- Center for Research in Security Prices (2000), *CRSPAccess Database Format Data Description Guide*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/stock\\_ind\\_data\\_descriptions.pdf](http://www.crsp.uchicago.edu/file_guides/stock_ind_data_descriptions.pdf)].
- Center for Research in Security Prices (2000), *CRSPAccess Database Format Programmer's Guide*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/stock\\_ind\\_programming.pdf](http://www.crsp.uchicago.edu/file_guides/stock_ind_programming.pdf)].
- Center for Research in Security Prices (2000), *CRSPAccess Database Format Release Notes*, Chicago: The University of Chicago Graduate School of Business, [[www.crsp.uchicago.edu/file\\_guides/ca\\_release\\_notes.pdf](http://www.crsp.uchicago.edu/file_guides/ca_release_notes.pdf)].
- Center for Research in Security Prices (2000), *CRSPAccess Database Format Utilities Guide*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/stock\\_ind\\_utilities.pdf](http://www.crsp.uchicago.edu/file_guides/stock_ind_utilities.pdf)].
- Center for Research in Security Prices (2000), *CRSP SFA Database Format Guide*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/stock\\_ind\\_sfa.pdf](http://www.crsp.uchicago.edu/file_guides/stock_ind_sfa.pdf)].

# Chapter 3

## The SASEFAME Interface Engine

### Chapter Table of Contents

---

|                                                                            |    |
|----------------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                                  | 23 |
| <b>GETTING STARTED</b> . . . . .                                           | 23 |
| Structure of a SAS Data Set Containing Time Series Data . . . . .          | 23 |
| Reading and Converting FAME Database Time Series . . . . .                 | 23 |
| Using the SAS DATA Step . . . . .                                          | 23 |
| Using SAS Procedures . . . . .                                             | 24 |
| <b>SYNTAX</b> . . . . .                                                    | 24 |
| The LIBNAME <i>libref</i> SASEFAME Statement . . . . .                     | 24 |
| <b>DETAILS</b> . . . . .                                                   | 25 |
| The SAS Output Data Set . . . . .                                          | 25 |
| Mapping FAME Frequencies to SAS Time Intervals . . . . .                   | 26 |
| <b>EXAMPLES</b> . . . . .                                                  | 28 |
| Example 3.1 Converting an Entire FAME Database . . . . .                   | 28 |
| Example 3.2 Reading Time Series from the FAME Database . . . . .           | 29 |
| Example 3.3 Writing Time Series to the SAS Data Set . . . . .              | 30 |
| Example 3.4 Limiting the Time Range of Data . . . . .                      | 31 |
| Example 3.5 Creating a View Using the SQL Procedure and SASEFAME . . . . . | 32 |
| <b>REFERENCES</b> . . . . .                                                | 38 |



## Chapter 3

# The SASEFAME Interface Engine

---

### Overview

The SASEFAME interface engine enables SAS users to access and process time series data residing in a FAME database, and provides a seamless interface between FAME and SAS data processing.

The SASEFAME engine uses the LIBNAME statement to enable you to specify which time series you would like to read from the FAME database, and how you would like to convert the selected time series to the same time scale. The SAS DATA step can then be used to perform further subsetting and to store the resulting time series into a SAS data set. You can perform more analysis if desired either in the same SAS session or in another session at a later time.

---

### Getting Started

---

#### Structure of a SAS Data Set Containing Time Series Data

SAS requires time series data to be in a specific form recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as a time ID variable. The SASEFAME engine does provide a time ID variable by the name of DATE.

---

#### Reading and Converting FAME Database Time Series

The SASEFAME engine supports reading and converting time series stored in FAME databases. The SASEFAME engine uses the FAME WORK database to temporarily store the converted time series. Only the series specified by the FAME wildcard are written to the FAME WORK database.

---

#### Using the SAS DATA Step

If desired, you can store the converted series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. Once your data is stored in a SAS data set you can use it as you would any other SAS data set.



---

## Using SAS Procedures

You can print the output SAS data set by using the PRINT procedure and report information concerning the contents of your data set by using the CONTENTS procedure, as in Example 3.1. You can create a view of the FAME data base by using the SQL procedure to create your view using the SASEFAME engine in your *libref*, along with the using clause. See Example 3.5.

---

## Syntax

The SASEFAME engine uses standard engine syntax. Options used by SASEFAME are summarized in the table below.

| Description                                                                                                                                                            | Statement                      | Option    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|-----------|
| specifies the FAME frequency and the FAME technique.                                                                                                                   | LIBNAME <i>libref</i> SASEFAME | CONVERT=  |
| specifies a FAME wildcard to match data object series names within the FAME database, which limits the selection of time series that are included in the SAS data set. | LIBNAME <i>libref</i> SASEFAME | WILDCARD= |

---

## The LIBNAME *libref* SASEFAME Statement

**LIBNAME** *libref* SASEFAME 'physical name' options;

The following options can be used in the LIBNAME *libref* SASEFAME statement:

**CONVERT=**(*FREQ=fame\_frequency TECH= fame\_technique*)

specifies the FAME frequency and the FAME technique just as you would in the FAME CONVERT function. There are four possible values for *fame\_technique*: CONSTANT (default), CUBIC, DISCRETE, or LINEAR. All FAME frequencies except PPY and YPP are supported by the SASEFAME engine. For a more complete discussion of FAME frequencies and SAS time intervals see the section “Mapping FAME Frequencies to SAS Time Intervals” on page 26. For all possible *fame\_frequency* values, refer to “Understanding Frequencies” in the *User’s Guide to FAME*. As an example,

```
LIBNAME libref sasefame 'physical-name'
      CONVERT=(TECH=CONSTANT FREQ=TWICEMONTHLY);
```

**WILDCARD="fame\_wildcard"**

By default, the SASEFAME engine reads all time series in the FAME database that you name in your SASEFAME libref. You can limit the time series read from the FAME database by specifying the WILDCARD= option on your LIBNAME statement. The *fame\_wildcard* is a quoted string containing the FAME wildcard you wish to use. The wildcard is used against the data object names (time series only) in the FAME database that resides in the library you are in the process of assigning. For more information about wildcarding, see “Specifying Wildcards” in the *User’s Guide to FAME*.

For example, to read all time series in the TEST library being accessed by the SASEFAME engine, you would specify

```
LIBNAME test sasefame 'physical name of test data base'  
WILDCARD="?";
```

To read series with names such as A\_DATA, B\_DATA, C\_DATA, you could specify

```
LIBNAME test sasefame 'physical name of test data base'  
WILDCARD="^_DATA";
```

When you use the WILD= option, you are limiting the number of time series that are read and converted to the desired frequency. This option can help you save resources when processing large databases or when processing a large number of observations, such as daily, hourly, or minutely frequencies. Since the SASEFAME engine uses the FAME WORK database to store the converted time series, using wildcards is recommended to prevent your WORK space from getting too large.

---

## Details

---

### The SAS Output Data Set

You can use the SAS DATA step to write the FAME converted series to a SAS data set. This allows the user the ability to easily analyze the data using SAS. You can specify the name of the output data set on the DATA statement. This causes the engine supervisor to create a SAS data set using the specified name in either the SAS WORK library, or if specified, the USER library. For more about naming your SAS data set see the section “Characteristics of SAS Data Libraries” in *SAS Language Reference: Dictionary*.

The contents of the SAS data set include the DATE of each observation, the name of each series read from the FAME database as specified by the WILDCARD option, and the label or FAME description of each series. Missing values are represented as '.' in the SAS data set. You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. You can use PROC SQL along with the SASEFAME engine to create a view of your SAS data set.

The DATE variable in the SAS data set contains the date of the observation. For FAME weekly intervals that end on a Friday, FAME reports the date on the Friday that ends the week whereas SAS reports the date on the Saturday that begins the week. A more detailed discussion of how to map FAME frequencies to SAS Time Intervals follows.

## Mapping FAME Frequencies to SAS Time Intervals

The following table summarizes the mapping of FAME frequencies to SAS time intervals. It is important to note that FAME frequencies often have a sample unit in parentheses following the keyword frequency. This sample unit is an end-of-interval unit. SAS dates are represented using begin-of-interval notation. For more on SAS time intervals, see “Date Intervals, Formats, and Functions” in *SAS/ETS User’s Guide*. For more on FAME frequencies, see the section “Understanding Frequencies” in the *User’s Guide to FAME*.

| FAME FREQUENCY        | SAS TIME INTERVAL |
|-----------------------|-------------------|
| WEEKLY (SUNDAY)       | WEEK.2            |
| WEEKLY (MONDAY)       | WEEK.3            |
| WEEKLY (TUESDAY)      | WEEK.4            |
| WEEKLY (WEDNESDAY)    | WEEK.5            |
| WEEKLY (THURSDAY)     | WEEK.6            |
| WEEKLY (FRIDAY)       | WEEK.7            |
| WEEKLY (SATURDAY)     | WEEK.1            |
| BIWEEKLY (ASUNDAY)    | WEEK2.2           |
| BIWEEKLY (AMONDAY)    | WEEK2.3           |
| BIWEEKLY (ATUESDAY)   | WEEK2.4           |
| BIWEEKLY (AWEDNESDAY) | WEEK2.5           |
| BIWEEKLY (ATHURSDAY)  | WEEK2.6           |
| BIWEEKLY (AFRIDAY)    | WEEK2.7           |
| BIWEEKLY (ASATURDAY)  | WEEK2.1           |
| BIWEEKLY (BSUNDAY)    | WEEK2.9           |
| BIWEEKLY (BMONDAY)    | WEEK2.10          |
| BIWEEKLY (BTUESDAY)   | WEEK2.11          |
| BIWEEKLY (BWEDNESDAY) | WEEK2.12          |
| BIWEEKLY (BTHURSDAY)  | WEEK2.13          |
| BIWEEKLY (BFRIDAY)    | WEEK2.14          |
| BIWEEKLY (BSATURDAY)  | WEEK2.8           |
| BIMONTHLY (NOVEMBER)  | MONTH2.2          |
| BIMONTHLY             | MONTH2.1          |
| QUARTERLY (OCTOBER)   | QTR.2             |
| QUARTERLY (NOVEMBER)  | QTR.3             |
| QUARTERLY             | QTR.1             |

| FAME FREQUENCY         | SAS TIME INTERVAL |
|------------------------|-------------------|
| ANNUAL (JANUARY)       | YEAR.2            |
| ANNUAL (FEBRUARY)      | YEAR.3            |
| ANNUAL (MARCH)         | YEAR.4            |
| ANNUAL (APRIL)         | YEAR.5            |
| ANNUAL (MAY)           | YEAR.6            |
| ANNUAL (JUNE)          | YEAR.7            |
| ANNUAL (JULY)          | YEAR.8            |
| ANNUAL (AUGUST)        | YEAR.9            |
| ANNUAL (SEPTEMBER)     | YEAR.10           |
| ANNUAL (OCTOBER)       | YEAR.11           |
| ANNUAL (NOVEMBER)      | YEAR.12           |
| ANNUAL                 | YEAR.1            |
| SEMIANNUAL (JULY)      | SEMIYEAR.2        |
| SEMIANNUAL (AUGUST)    | SEMIYEAR.3        |
| SEMIANNUAL (SEPTEMBER) | SEMIYEAR.4        |
| SEMIANNUAL (OCTOBER)   | SEMIYEAR.5        |
| SEMIANNUAL (NOVEMBER)  | SEMIYEAR.6        |
| SEMIANNUAL             | SEMIYEAR.1        |
| YPP                    | not supported     |
| PPY                    | not supported     |
| SECONDLY               | SECOND            |
| MINUTELY               | MINUTE            |
| HOURLY                 | HOUR              |
| DAILY                  | DAY               |
| BUSINESS               | WEEKDAY           |
| TENDAY                 | TENDAY            |
| TWICEMONTHLY           | SEMIMONTH         |
| MONTHLY                | MONTH             |

---

## Examples

---

---

### Example 3.1. Converting an Entire FAME Database

---

To enable conversion of all Time Series no wildcard is specified, so the default “?” wildcard is used. Always consider both the number of time series and the number of observations generated by the conversion process. The converted series are stored in the FAME WORK database during the SAS DATA step. You may further limit your resulting SAS data set by using KEEP, DROP, or WHERE statements inside your data step.

The following statements convert a FAME database and print out its contents:

```
libname famedir sasefame '.'
              convert=(freq=annual technique=constant);

libname mydir '/mine/data/europe/sas/oeedir';

data mydir.a; /* add data set to mydir */
    set famedir.oeed1;
    /* do nothing special */
run;

proc print data=mydir.a; run;
```

In the above example, the FAME database is called `oeed1.db` and it resides in the `famedir` directory. The DATA statement names the SAS output data set 'a' which will reside in `mydir`. All time series in the FAME `oeed1.db` data base will be converted to an annual frequency and stored in the `mydir.a` SAS data set. The PROC PRINT statement creates a listing of all of the observations in the `mydir.a` SAS data set.

---

**Example 3.2. Reading Time Series from the FAME Database**

Use the FAME wildcard option to limit the number of series converted. For example, suppose you want to read only series starting with “WSPCA”. You could use the following code:

```
libname lib1 sasefame '/mine/data/econ_fame/sampdir'
              wildcard="wspca?"
              convert=(technique=constant freq=twicemonthly );

libname lib2 '/mine/data/econ_sas/sampdir';

data lib2.twild(label='Annual Series from the FAMEECON.db');
  set lib1.subecon;
  /* keep only */
  keep date wspca;
  run;

proc contents data=lib2.twild; run;

proc print data=lib2.twild; run;
```

The wildcard=“wspca?” option limits reading only those series whose names begin with WSPCA. The SAS KEEP statement further restricts the SAS data set to include only the series named WSPCA and the DATE variable. The time interval used for the conversion is TWICEMONTHLY.

---

### Example 3.3. Writing Time Series to the SAS Data Set

You can use the KEEP or DROP statement to include or exclude certain series names from the SAS data set.

```
libname famedir sasefame '.'
      convert=(freq=annual technique=constant);

libname mydir '/mine/data/europe/sas/oecdir';

data mydir.a; /* add data set to mydir */
  set famedir.oecd1;
  drop ita_dird--jpn_herd tur_dird--usa_herd;
run;

proc print data=mydir.a; run;
```

You can rename your SAS variables by using the RENAME statement.

```
option validvarname=any;

libname famedir sasefame '.'
      convert=(freq=annual technique=constant);

libname mydir '/mine/data/europe/sas/oecdir';

data mydir.a; /* add data set to mydir */
  set famedir.oecd1;
  /* keep and rename */
  keep date ita_dird--jpn_herd tur_dird--usa_herd;
  rename ita_dird='Italy.dirdes'n
         jpn_dird='Japan.dirdes'n
         tur_dird='Turkey.dirdes'n
         usa_dird='UnitedStates.dirdes'n ;
run;

proc print data=mydir.a; run;
```

---

### Example 3.4. Limiting the Time Range of Data

You may also limit the time range of the data in the SAS data set by using the WHERE statement in the data step to process the time ID variable DATE only when it falls in the range you are interested in.

```
libname famedir SASEFAME '.'
      convert=(freq=annual technique=constant);

libname mydir '/mine/data/europe/sas/oecdir';

data mydir.a; /* add data set to mydir */
  set famedir.oecd1;
  /* where only */
  where date between '01jan88'd and '31dec90'd;
run;

proc print data=mydir.a; run;
```

All data for 1988, 1989, and 1990 are included in the SAS data set. See the *SAS Language: Reference, Version 7* for more information on KEEP, DROP, RENAME and WHERE statements.



### Example 3.5. Creating a View Using the SQL Procedure and SASEFAME

This example creates a view using the SQL procedure's from and using clauses. See *SQL Procedure Guide, Version 7* for details on SQL views.

```

title1 'famesql5: PROC SQL Dual Embedded Libraries w/ FAME option';
options validvarname=any;

/* Dual Embedded Library Allocations (With FAME Option) */
/*****

/* OECD1 Database */
/*****/

title2 'OECD1: Dual Embedded Library Allocations
        with FAME Option';
proc sql;
  create view fameview as
  select date, 'fin.herd'n
  from lib1.oecd1
  using libname lib1 sasefame '/economic/databases/testdat'
        convert=(tech=constant freq=annual),
        libname temp '/usr/local/scratch/mine'
  quit;

title2 'OECD1: Print of View from Embedded Library
        with FAME Option';
proc print data=fameview;
run;

```

**Output 3.5.1.** Printout of the FAME View of OECD Data

```

PROC SQL Dual Embedded Libraries w/ FAME option
OECD1: Print of View from Embedded Library with Option

```

| Obs | DATE | FIN.HERD |
|-----|------|----------|
| 1   | 1985 | 1097.0   |
| 2   | 1986 | 1234.0   |
| 3   | 1987 | 1401.3   |
| 4   | 1988 | 1602.0   |
| 5   | 1989 | 1725.5   |
| 6   | 1990 | 1839.0   |
| 7   | 1991 | .        |

```

/* SUBECON Database */
/*****/

title2 'SUBECON: Dual Embedded Library Allocations
        with FAME Option';
proc sql;
  create view fameview as
  select date, gaa

```

```
from lib1.subecon
using libname lib1 sasefame '/economic/databases/testdat'
                    convert=(tech=constant freq=annual),
                    libname temp '/usr/local/scratch/mine'
quit;

title2 'SUBECON: Print of View from Embedded Library
        with FAME Option';
proc print data=fameview;
run;
```

Part 1. General Information

Output 3.5.2. Printout of the FAME View of DRI Basic Economic Data

| PROC SQL Dual Embedded Libraries w/ FAME option<br>SUBECON: Print of View from Embedded Library with Option |      |           |
|-------------------------------------------------------------------------------------------------------------|------|-----------|
| Obs                                                                                                         | DATE | GAA       |
| 1                                                                                                           | 1946 | .         |
| 2                                                                                                           | 1947 | .         |
| 3                                                                                                           | 1948 | 23174.00  |
| 4                                                                                                           | 1949 | 19003.00  |
| 5                                                                                                           | 1950 | 24960.00  |
| 6                                                                                                           | 1951 | 21906.00  |
| 7                                                                                                           | 1952 | 20246.00  |
| 8                                                                                                           | 1953 | 20912.00  |
| 9                                                                                                           | 1954 | 21056.00  |
| 10                                                                                                          | 1955 | 27168.00  |
| 11                                                                                                          | 1956 | 27638.00  |
| 12                                                                                                          | 1957 | 26723.00  |
| 13                                                                                                          | 1958 | 22929.00  |
| 14                                                                                                          | 1959 | 29729.00  |
| 15                                                                                                          | 1960 | 28444.00  |
| 16                                                                                                          | 1961 | 28226.00  |
| 17                                                                                                          | 1962 | 32396.00  |
| 18                                                                                                          | 1963 | 34932.00  |
| 19                                                                                                          | 1964 | 40024.00  |
| 20                                                                                                          | 1965 | 47941.00  |
| 21                                                                                                          | 1966 | 51429.00  |
| 22                                                                                                          | 1967 | 49164.00  |
| 23                                                                                                          | 1968 | 51208.00  |
| 24                                                                                                          | 1969 | 49371.00  |
| 25                                                                                                          | 1970 | 44034.00  |
| 26                                                                                                          | 1971 | 52352.00  |
| 27                                                                                                          | 1972 | 62644.00  |
| 28                                                                                                          | 1973 | 81645.00  |
| 29                                                                                                          | 1974 | 91028.00  |
| 30                                                                                                          | 1975 | 89494.00  |
| 31                                                                                                          | 1976 | 109492.00 |
| 32                                                                                                          | 1977 | 130260.00 |
| 33                                                                                                          | 1978 | 154357.00 |
| 34                                                                                                          | 1979 | 173428.00 |
| 35                                                                                                          | 1980 | 156096.00 |
| 36                                                                                                          | 1981 | 147765.00 |
| 37                                                                                                          | 1982 | 113216.00 |
| 38                                                                                                          | 1983 | 133495.00 |
| 39                                                                                                          | 1984 | 146448.00 |
| 40                                                                                                          | 1985 | 128521.99 |
| 41                                                                                                          | 1986 | 111337.99 |
| 42                                                                                                          | 1987 | 160785.00 |
| 43                                                                                                          | 1988 | 210532.00 |
| 44                                                                                                          | 1989 | 201637.00 |
| 45                                                                                                          | 1990 | 218702.00 |
| 46                                                                                                          | 1991 | 210666.00 |
| 47                                                                                                          | 1992 | .         |
| 48                                                                                                          | 1993 | .         |

```

title2 'DB77: Dual Embedded Library Allocations
       with FAME Option';
proc sql;
  create view fameview as
  select date, ann, 'qandom.x'n
  from lib1.db77
  using libname lib1 sasefame '/economic/databases/testdat'
  convert=(tech=constant freq=annual),

```

```

libname temp '/usr/local/scratch/mine'
quit;

title2 'DB77: Print of View from Embedded Library
       with FAME Option';
proc print data=fameview;
run;

```

Output 3.5.3. Printout of the FAME View of DB77 Data

| PROC SQL Dual Embedded Libraries w/ FAME option<br>DB77: Print of View from Embedded Library with Option |      |     |          |  |
|----------------------------------------------------------------------------------------------------------|------|-----|----------|--|
| Obs                                                                                                      | DATE | ANN | QANDOM.X |  |
| 1                                                                                                        | 1959 | .   | 0.56147  |  |
| 2                                                                                                        | 1960 | .   | 0.51031  |  |
| 3                                                                                                        | 1961 | .   | .        |  |
| 4                                                                                                        | 1962 | .   | .        |  |
| 5                                                                                                        | 1963 | .   | .        |  |
| 6                                                                                                        | 1964 | .   | .        |  |
| 7                                                                                                        | 1965 | .   | .        |  |
| 8                                                                                                        | 1966 | .   | .        |  |
| 9                                                                                                        | 1967 | .   | .        |  |
| 10                                                                                                       | 1968 | .   | .        |  |
| 11                                                                                                       | 1969 | .   | .        |  |
| 12                                                                                                       | 1970 | .   | .        |  |
| 13                                                                                                       | 1971 | .   | .        |  |
| 14                                                                                                       | 1972 | .   | .        |  |
| 15                                                                                                       | 1973 | .   | .        |  |
| 16                                                                                                       | 1974 | .   | .        |  |
| 17                                                                                                       | 1975 | .   | .        |  |
| 18                                                                                                       | 1976 | .   | .        |  |
| 19                                                                                                       | 1977 | .   | .        |  |
| 20                                                                                                       | 1978 | .   | .        |  |
| 21                                                                                                       | 1979 | .   | .        |  |
| 22                                                                                                       | 1980 | 100 | .        |  |
| 23                                                                                                       | 1981 | 101 | .        |  |
| 24                                                                                                       | 1982 | 102 | .        |  |
| 25                                                                                                       | 1983 | 103 | .        |  |
| 26                                                                                                       | 1984 | 104 | .        |  |
| 27                                                                                                       | 1985 | 105 | .        |  |
| 28                                                                                                       | 1986 | 106 | .        |  |
| 29                                                                                                       | 1987 | 107 | .        |  |
| 30                                                                                                       | 1988 | 109 | .        |  |
| 31                                                                                                       | 1989 | 111 | .        |  |

```

/* DRIECON Database */
/*****

title2 'DRIECON: Dual Embedded Library Allocations
       with FAME Option';
proc sql;
  create view fameview as
  select date, husts
  from lib1.driecon
  using libname lib1 sasefame '/economic/databases/testdat'
         convert=(tech=constant freq=annual),
         libname temp '/usr/local/scratch/mine'
quit;

```

```
title2 'DRIECON: Print of View from Embedded Library  
with FAME Option';  
proc print data=fameview;  
run;
```

Note that the SAS option VALIDVARNAME=ANY was used at the top of this example due to special characters being present in the time series names. The output from this example shows how each FAME view is the output of the SASEFAME engine's processing. Note that different engine options could have been used in the USING LIBNAME clause if desired.

**Output 3.5.4.** Printout of the FAME View of DRI Basic Economic Data

PROC SQL Dual Embedded Libraries w/ FAME option  
 DRIECON: Print of View from Embedded Library with Option

| Obs | DATE | HUSTS   |
|-----|------|---------|
| 1   | 1947 | 1.26548 |
| 2   | 1948 | 1.33470 |
| 3   | 1949 | 1.43617 |
| 4   | 1950 | 1.90041 |
| 5   | 1951 | 1.43759 |
| 6   | 1952 | 1.44883 |
| 7   | 1953 | 1.40279 |
| 8   | 1954 | 1.53525 |
| 9   | 1955 | 1.61970 |
| 10  | 1956 | 1.32400 |
| 11  | 1957 | 1.17300 |
| 12  | 1958 | 1.31717 |
| 13  | 1959 | 1.53450 |
| 14  | 1960 | 1.25505 |
| 15  | 1961 | 1.31188 |
| 16  | 1962 | 1.45996 |
| 17  | 1963 | 1.58858 |
| 18  | 1964 | 1.53950 |
| 19  | 1965 | 1.46966 |
| 20  | 1966 | 1.16507 |
| 21  | 1967 | 1.28573 |
| 22  | 1968 | 1.50314 |
| 23  | 1969 | 1.48531 |
| 24  | 1970 | 1.43565 |
| 25  | 1971 | 2.03775 |
| 26  | 1972 | 2.36069 |
| 27  | 1973 | 2.04307 |
| 28  | 1974 | 1.32855 |
| 29  | 1975 | 1.16164 |
| 30  | 1976 | 1.53468 |
| 31  | 1977 | 1.96218 |
| 32  | 1978 | 2.00184 |
| 33  | 1979 | 1.71847 |
| 34  | 1980 | 1.29990 |
| 35  | 1981 | 1.09574 |
| 36  | 1982 | 1.05862 |
| 37  | 1983 | 1.70580 |
| 38  | 1984 | 1.76351 |
| 39  | 1985 | 1.74258 |
| 40  | 1986 | 1.81205 |
| 41  | 1987 | 1.62914 |
| 42  | 1988 | 1.48748 |
| 43  | 1989 | 1.38218 |
| 44  | 1990 | 1.20161 |
| 45  | 1991 | 1.00878 |
| 46  | 1992 | 1.20159 |
| 47  | 1993 | 1.29201 |
| 48  | 1994 | 1.44684 |
| 49  | 1995 | 1.35845 |
| 50  | 1996 | 1.48336 |

---

## References

- DRI/McGraw-Hill (1997), *DataLink*, Lexington, MA.
- DRI/McGraw-Hill Data Search and Retrieval for Windows (1996), *DRIPRO User's Guide*, Lexington, MA.
- FAME Information Services (1995), *User's Guide to FAME*, Ann Arbor, Michigan.
- FAME Information Services (1995), *Reference Guide to Seamless C HLI*, Ann Arbor, Michigan.
- FAME Information Services (1995), *Command Reference for Release 7.6, Vols. 1 and 2*, Ann Arbor, Michigan.
- Organization For Economic Cooperation and Development (1992), *Annual National Accounts: Volume I. Main Aggregates Content Documentation for Magnetic Tape Subscription*, Paris, France.
- Organization For Economic Cooperation and Development (1992), *Annual National Accounts: Volume II. Detailed Tables Technical Documentation for Magnetic Tape Subscription*, Paris, France.
- Organization For Economic Cooperation and Development (1992), *Main Economic Indicators Database Note*, Paris, France.
- Organization For Economic Cooperation and Development (1992), *Main Economic Indicators Inventory*, Paris, France.
- Organization For Economic Cooperation and Development (1992), *Main Economic Indicators OECD Statistics on Magnetic Tape Document*, Paris, France.
- Organization For Economic Cooperation and Development (1992), *OECD Statistical Information Research and Inquiry System Magnetic Tape Format Documentation*, Paris, France.
- Organization For Economic Cooperation and Development (1992), *Quarterly National Accounts Inventory of Series Codes*, Paris, France.
- Organization For Economic Cooperation and Development (1992), *Quarterly National Accounts Technical Documentation*, Paris, France.

# Part 2

## Procedure Reference

### Table of Contents

---

|                                           |     |
|-------------------------------------------|-----|
| Chapter 4. The VARMAX Procedure . . . . . | 41  |
| Chapter 5. The X12 Procedure . . . . .    | 161 |





# Chapter 4

## The VARMAX Procedure

### Chapter Table of Contents

---

|                                                        |     |
|--------------------------------------------------------|-----|
| <b>OVERVIEW</b>                                        | 43  |
| <b>GETTING STARTED</b>                                 | 45  |
| Vector Autoregressive Process                          | 45  |
| Bayesian Vector Autoregressive Process                 | 51  |
| Vector Error Correction Model                          | 51  |
| Bayesian Vector Error Correction Model                 | 56  |
| Vector Autoregressive Process with Exogenous Variables | 57  |
| Parameter Estimation and Testing on Restrictions       | 60  |
| Causality Testing                                      | 62  |
| <b>SYNTAX</b>                                          | 63  |
| Functional Summary                                     | 63  |
| PROC VARMAX Statement                                  | 65  |
| BY Statement                                           | 66  |
| CAUSAL Statement                                       | 66  |
| COINTEG Statement                                      | 67  |
| ID Statement                                           | 69  |
| MODEL Statement                                        | 69  |
| OUTPUT Statement                                       | 80  |
| RESTRICT Statement                                     | 81  |
| TEST Statement                                         | 82  |
| <b>DETAILS</b>                                         | 82  |
| Missing Values                                         | 82  |
| VARMAX Modeling                                        | 82  |
| Dynamic Simultaneous Equations Modeling                | 83  |
| State-Space Modeling                                   | 84  |
| Forecasting                                            | 87  |
| Tentative Order Selection                              | 96  |
| VAR Modeling                                           | 101 |
| Model Diagnostic Checks                                | 107 |
| Bayesian VAR Modeling                                  | 108 |
| Cointegration                                          | 110 |
| Common Trends                                          | 111 |
| Vector Error Correction Modeling                       | 113 |

## Part 2. Procedure Reference

|                                                      |     |
|------------------------------------------------------|-----|
| I(2) Model . . . . .                                 | 125 |
| OUT= Data Set . . . . .                              | 127 |
| OUTEST= Data Set . . . . .                           | 129 |
| OUTSTAT= Data Set . . . . .                          | 130 |
| Printed Output . . . . .                             | 132 |
| ODS Table Names . . . . .                            | 133 |
| <b>EXAMPLES</b> . . . . .                            | 137 |
| Example 4.1 Analysis of Real Output Series . . . . . | 137 |
| Example 4.2 Analysis of Real Output Series . . . . . | 145 |
| Example 4.3 Numerous Examples . . . . .              | 154 |
| <b>REFERENCES</b> . . . . .                          | 157 |

## Chapter 4

# The VARMAX Procedure

---

### Overview

Given a multivariate time series, the VARMAX procedure estimates the model parameters and generates forecasts associated with Vector AutoRegressive and Moving-Average processes with exogenous regressors (VARMAX) models. Often, economic or financial variables are not only contemporaneously correlated to each other, they are also correlated to each other's past values. The VARMAX procedure can be used to model these types of time relationships. In many economic and financial applications, the variables of interest (dependent, response, or endogenous variables) are influenced by variables external to the system under consideration (independent, input, predictor, regressor, or exogenous variables). The VARMAX procedure enables you to model both the dynamic relationship between the dependent variables and between the dependent and independent variables.

VARMAX models are defined in terms of the orders of the autoregressive or moving-average process (or both). When you use the VARMAX procedure, these orders can be specified by options or they can be automatically determined. Criteria for automatically determining these orders include

- Akaike Information Criterion (AIC)
- Corrected AIC (AICC)
- Hannan-Quinn (HQ) Criterion
- Final Prediction Error (FPE)
- Schwarz Bayesian Criterion (SBC), also known as Bayesian Information Criterion (BIC)

If you do not wish to use the automatic order selection, the VARMAX procedure provides these order identification aids: partial cross-correlations, Yule-Walker estimates, partial autoregressive coefficients, and partial canonical correlations.

For situations where the stationarity of the time series is in question, the VARMAX procedure provides tests to aid in determining the presence of unit roots and cointegration. These tests include

- Dickey-Fuller test
- Johansen cointegration test for nonstationary vector processes of integrated order one

## Part 2. Procedure Reference

- Stock-Watson common trends test for the possibility of cointegration among nonstationary vector processes of integrated order one
- Johansen cointegration test for nonstationary vector processes of integrated order two

For stationary vector times series (or nonstationary series made stationary by appropriate differencing), the VARMAX procedure provides for both Vector AutoRegressive (VAR) and Bayesian Vector AutoRegressive (BVAR) models. To cope with the problem of high dimensionality in the parameters of the VAR model, the VARMAX procedure provides both the Vector Error Correction Model (VECM) and Bayesian Vector Error Correction Model (BVECM). Bayesian models are used when prior information about the model parameters is available. The VARMAX procedure can allow exogenous (independent) variables with their distributed lags to influence endogenous (dependent) variables. The model parameter estimation methods are

- Least Squares (LS)
- Maximum Likelihood (ML)

The VARMAX procedure provides various hypothesis tests of long-run effects and adjustment coefficients using the likelihood ratio test based on Johansen cointegration analysis. The VARMAX procedure offers the likelihood ratio test of the weak exogeneity for each variable.

After fitting the model parameters, the VARMAX procedure provides for model checks and residual analysis using the following tests:

- Durbin-Watson (DW) statistics
- $F$  test for autoregressive conditional heteroscedastic (ARCH) disturbance
- $F$  test for AR disturbance
- Jarque-Bera normality test
- Portmanteau test

Forecasting is one of the main objectives of multivariate time series analysis. After successfully fitting the VAR, VARX, BVAR, VECM, and BVECM model parameters, the VARMAX procedure computes predicted values based on the parameter estimates and the past values of the vector time series.

The VARMAX procedure supports several modeling features, including

- seasonal deterministic terms
- subset models
- multiple regression with distributed lags

- dead-start model that does not have present values of the exogenous variables

The VARMAX procedure provides a Granger-Causality test to determine the Granger-causal relationships between two distinct groups of variables. It also provides

- infinite order AR representation
- impulse response function (or infinite order MA representation)
- decomposition of the predicted error covariances
- roots of the characteristic functions for both the AR and MA parts to evaluate the proximity of the roots to the unit circle

---

## Getting Started

This section outlines the use of the VARMAX procedure and gives five different examples of the kind of models supported.

---

### Vector Autoregressive Process

Let  $\mathbf{y}_t = (y_{1t}, \dots, y_{kt})'$ ,  $t = 0, 1, \dots$ , denote a  $k$ -dimensional time series vector of random variables of interest. The  $p$ th-order VAR process is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \dots + \Phi_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t$$

where the  $\boldsymbol{\epsilon}_t$  is a vector white noise process with  $\boldsymbol{\epsilon}_t = (\epsilon_{1t}, \dots, \epsilon_{kt})'$  such that  $E(\boldsymbol{\epsilon}_t) = 0$ ,  $E(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t') = \Sigma$ , and  $E(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_s') = 0$  for  $t \neq s$ .

Analyzing and modeling the series jointly enables you to understand the dynamic relationships over time among the series and to improve the accuracy of forecasts for individual series by utilizing the additional information available from the related series and their forecasts.

#### Example of Vector Autoregressive Model

Consider the first-order stationary vector autoregressive model

$$\mathbf{y}_t = \begin{pmatrix} 1.2 & -0.5 \\ 0.6 & 0.3 \end{pmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t, \quad \text{with } \Sigma = \begin{pmatrix} 1.0 & 0.5 \\ 0.5 & 1.25 \end{pmatrix}.$$

The following IML procedure statements check whether the specified vector time series model is stationary and simulate a bivariate vector time series from this model to provide test data for the VARMAX procedure:

```
proc iml;
  sig = {1.0 0.5, 0.5 1.25};
  phi = {1.2 -0.5, 0.6 0.3};
  /* to simulate the vector time series */
```

```

call varmasim(y,phi) sigma = sig n = 100 seed = 34657;
cn = {'y1' 'y2'};
create simul1 from y[colname=cn];
append from y;
quit;

```

The following statements plot the simulated vector time series  $y_t$  shown in Figure 4.1:

```

data simul1;
set simul1;
t+1;

proc gplot data=simul1;
symbol1 v = none i = join l = 1;
symbol2 v = none i = join l = 2;
plot y1 * t = 1
      y2 * t = 2 / overlay;
run;

```

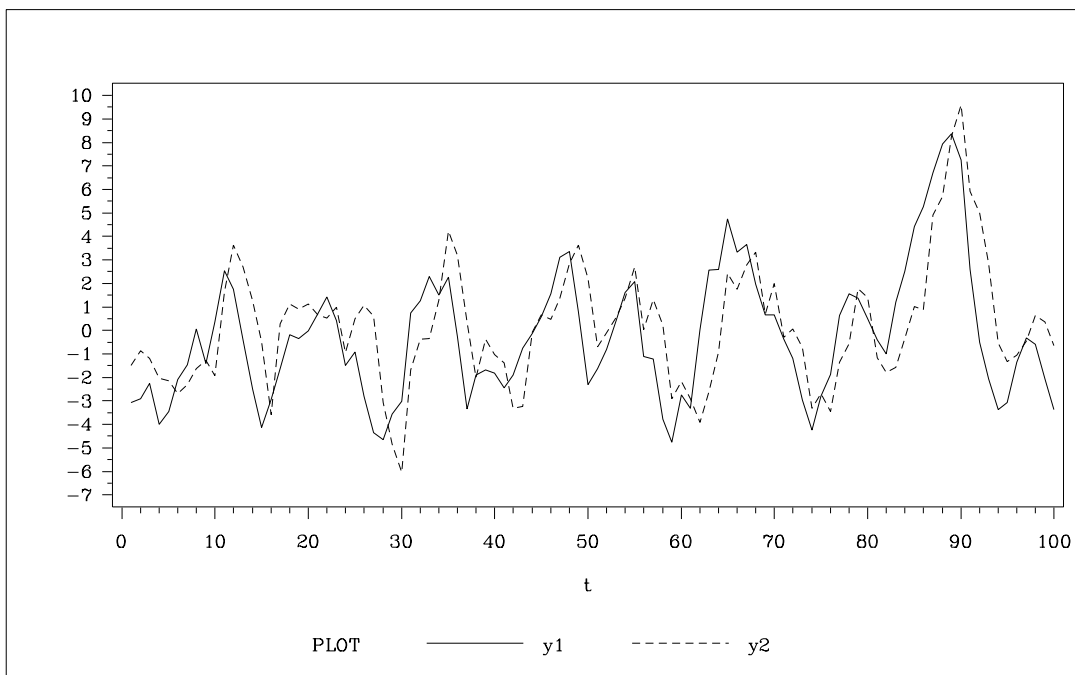


**Figure 4.1.** Plot of Generated Data Process

The following statements fit a VAR(1) model to the simulated data. You first specify the input data set in the PROC VARMAX statement. Then, you use the MODEL statement to read the time series  $y_1$  and  $y_2$ . To estimate a VAR model with mean zero, you specify the order of the autoregressive model with the P= option and the NOINT option. The MODEL statement fits the model to the data and prints parameter estimates and various diagnostic tests. The LAGMAX=3 option is used to print the output for the residual diagnostic checks. For the forecasts, you specify the OUTPUT statement. If you wish to forecast five steps ahead, you use the option LEAD=5.

The VARMAX procedure output is shown in Figure 4.2 through Figure 4.9.

```
proc varmax data=simul1;
  id date interval=year;
  model y1 y2 / p=1 noint lagmax=3;
  output lead=5;
run;
```

| The VARMAX Procedure |      |         |                            |         |                          |         |
|----------------------|------|---------|----------------------------|---------|--------------------------|---------|
|                      |      |         | Number of Observations     |         | 100                      |         |
|                      |      |         | Number of Pairwise Missing |         | 0                        |         |
| Variable             | Type | NoMissN | Mean                       | StdDev  | Min                      | Max     |
| y1                   | DEP  | 100     | -0.21653                   | 2.78210 | -4.75826                 | 8.37032 |
| y2                   | DEP  | 100     | 0.16905                    | 2.58184 | -6.04718                 | 9.58487 |
| The VARMAX Procedure |      |         |                            |         |                          |         |
|                      |      |         | Type of Model              |         | VAR(1)                   |         |
|                      |      |         | Estimation Method          |         | Least Squares Estimation |         |

**Figure 4.2.** Descriptive Statistics and Model Type

The VARMAX procedure first displays descriptive statistics and the type of the fitted model for the simulated data, as shown in Figure 4.2.

| The VARMAX Procedure      |           |          |           |          |         |          |
|---------------------------|-----------|----------|-----------|----------|---------|----------|
| AR Coefficient Estimates  |           |          |           |          |         |          |
|                           | Lag       | Variable | y1        | y2       |         |          |
|                           | 1         | y1       | 1.15977   | -0.51058 |         |          |
|                           |           | y2       | 0.54634   | 0.38499  |         |          |
| Model Parameter Estimates |           |          |           |          |         |          |
| Equation                  | Parameter | Estimate | Std Error | T Ratio  | Prob> T | Variable |
| y1(t)                     | AR1_1_1   | 1.15977  | 0.05508   | 21.06    | 0.0001  | y1(t-1)  |
|                           | AR1_1_2   | -0.51058 | 0.05898   | -8.66    | 0.0001  | y2(t-1)  |
| y2(t)                     | AR1_2_1   | 0.54634  | 0.05779   | 9.45     | 0.0001  | y1(t-1)  |
|                           | AR1_2_2   | 0.38499  | 0.06188   | 6.22     | 0.0001  | y2(t-1)  |

**Figure 4.3.** Parameter Estimates

Figure 4.3 shows the AR coefficient matrix in terms of lag 1, the parameter estimates, and their significances that indicate how well the model fits the data. The first column gives the left-hand-side variable of the equation; the second column, the parameter name  $ARl\_i\_j$ , which indicates the  $(i, j)$ th element of the lag  $l$  autoregressive coefficient; the last column, the regressor corresponding to its parameter.



The fitted VAR(1) model with estimated standard errors in parentheses is given as

$$\mathbf{y}_t = \begin{pmatrix} 1.160 & -0.511 \\ (0.055) & (0.059) \\ 0.546 & 0.385 \\ (0.058) & (0.062) \end{pmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t.$$

Clearly, all parameter estimates in the coefficient matrix  $\Phi_1$  are significant.

The model can also be written as two univariate regression equations.

$$\begin{aligned} y_{1t} &= 1.160 y_{1,t-1} - 0.511 y_{2,t-1} + \epsilon_{1t} \\ y_{2t} &= 0.546 y_{1,t-1} + 0.385 y_{2,t-1} + \epsilon_{2t} \end{aligned}$$

| The VARMAX Procedure                   |          |         |
|----------------------------------------|----------|---------|
| Covariance Matrix for the Innovation   |          |         |
| Variable                               | y1       | y2      |
| y1                                     | 1.28875  | 0.39751 |
| y2                                     | 0.39751  | 1.41839 |
| Information Criteria                   |          |         |
| AICC(Corrected AIC)                    | 0.554443 |         |
| HQC(Hannan-Quinn Criterion)            | 0.595201 |         |
| AIC(Akaike Information Criterion)      | 0.552777 |         |
| SBC(Schwarz Bayesian Criterion)        | 0.65763  |         |
| FPEC(Final Prediction Error Criterion) | 1.738092 |         |

**Figure 4.4.** Innovation Covariance Estimates and Information Criteria

The table in Figure 4.4 shows the innovation covariance matrix estimates and the various information criteria results. The smaller value of information criteria fits the data better when it is compared to other models.

| The VARMAX Procedure                |          |          |          |  |
|-------------------------------------|----------|----------|----------|--|
| Residual Cross-Correlation Matrices |          |          |          |  |
| Lag                                 | Variable | y1       | y2       |  |
| 0                                   | y1       | 1.00000  | 0.28113  |  |
|                                     | y2       | 0.28113  | 1.00000  |  |
| 1                                   | y1       | 0.01309  | 0.02385  |  |
|                                     | y2       | -0.05569 | -0.07328 |  |
| 2                                   | y1       | 0.05277  | 0.06052  |  |
|                                     | y2       | 0.00847  | -0.04307 |  |
| 3                                   | y1       | -0.00163 | 0.06800  |  |
|                                     | y2       | -0.01644 | 0.05422  |  |

| Schematic Representation of<br>Residual Cross Correlations |    |    |    |    |
|------------------------------------------------------------|----|----|----|----|
| Variable/<br>Lag                                           | 0  | 1  | 2  | 3  |
| y1                                                         | ++ | .. | .. | .. |
| y2                                                         | ++ | .. | .. | .. |

|                            |
|----------------------------|
| + is > 2*std error, - is < |
| -2*std error, . is between |

Figure 4.5. Multivariate Diagnostic Checks

| The VARMAX Procedure                                |                |    |                |  |
|-----------------------------------------------------|----------------|----|----------------|--|
| Portmanteau Test for Residual<br>Cross Correlations |                |    |                |  |
| To<br>Lag                                           | Chi-<br>Square | DF | Prob><br>ChiSq |  |
| 2                                                   | 1.84           | 4  | 0.7659         |  |
| 3                                                   | 2.57           | 8  | 0.9582         |  |

Figure 4.6. Multivariate Diagnostic Checks Continued

Figure 4.5 and Figure 4.6 show tests for white noise residuals. The output shows that you cannot reject the hypothesis that the residuals are uncorrelated.

| The VARMAX Procedure               |          |        |         |        |
|------------------------------------|----------|--------|---------|--------|
| Univariate Model Diagnostic Checks |          |        |         |        |
| Variable                           | R-square | StdDev | F Value | Prob>F |
| y1                                 | 0.8369   | 1.1352 | 497.67  | <.0001 |
| y2                                 | 0.7978   | 1.1910 | 382.76  | <.0001 |

Figure 4.7. Univariate Diagnostic Checks

The VARMAX procedure provides diagnostic checks for the univariate form of the equations. The table in Figure 4.7 describes how well each univariate equation fits the data. From two univariate regression equations in Figure 4.3, the values of  $R^2$  in the second column are 0.84 and 0.80 for each equation. The standard deviations

in the third column are the square root of the diagonal elements of the covariance matrix from Figure 4.4. The  $F$  statistics are in the fourth column for hypotheses to test  $\phi_{11} = \phi_{12} = 0$  and  $\phi_{21} = \phi_{22} = 0$ , respectively, where  $\phi_{ij}$  is the  $(i, j)$ th element of the matrix  $\Phi_1$ . The last column shows the  $p$ -values of the  $F$  statistics. The results show that each univariate model is significant.

| The VARMAX Procedure               |         |                    |                |                  |         |        |         |        |
|------------------------------------|---------|--------------------|----------------|------------------|---------|--------|---------|--------|
| Univariate Model Diagnostic Checks |         |                    |                |                  |         |        |         |        |
| Variable                           | DW(1)   | Normality<br>ChiSq | Prob><br>ChiSq | ARCH1<br>F Value | Prob>F  |        |         |        |
| y1                                 | 1.97    | 3.32               | 0.1900         | 0.13             | 0.7199  |        |         |        |
| y2                                 | 2.14    | 5.46               | 0.0653         | 2.10             | 0.1503  |        |         |        |
|                                    |         |                    |                |                  |         |        |         |        |
| Univariate Model Diagnostic Checks |         |                    |                |                  |         |        |         |        |
| Variable                           | AR1     |                    | AR1-2          |                  | AR1-3   |        | AR1-4   |        |
|                                    | F Value | Prob>F             | F Value        | Prob>F           | F Value | Prob>F | F Value | Prob>F |
| y1                                 | 0.02    | 0.8980             | 0.14           | 0.8662           | 0.09    | 0.9629 | 0.82    | 0.5164 |
| y2                                 | 0.52    | 0.4709             | 0.41           | 0.6650           | 0.32    | 0.8136 | 0.32    | 0.8664 |

**Figure 4.8.** Univariate Diagnostic Checks Continued

The check for white noise residuals in terms of the univariate equation is shown in Figure 4.8. This output contains information that indicates whether the residuals are correlated and heteroscedastic. In the first table, the second column contains the Durbin-Watson test statistics; the third and fourth columns show the Jarque-Bera normality test statistics and their  $p$ -values; the last two columns show  $F$  statistics and their  $p$ -values for ARCH(1) disturbances. The second table includes  $F$  statistics and their  $p$ -values for AR(1) to AR(4) disturbances. The results indicate that the residuals are uncorrelated and homoscedastic.

| The VARMAX Procedure |     |      |          |                   |                          |         |
|----------------------|-----|------|----------|-------------------|--------------------------|---------|
| Forecasts            |     |      |          |                   |                          |         |
| Variable             | Obs | Time | Forecast | Standard<br>Error | 95% Confidence<br>Limits |         |
| y1                   | 101 | 2000 | -3.5921  | 1.1352            | -5.8171                  | -1.3671 |
|                      | 102 | 2001 | -3.0945  | 1.7091            | -6.4443                  | 0.2554  |
|                      | 103 | 2002 | -2.1743  | 2.1447            | -6.3779                  | 2.0292  |
|                      | 104 | 2003 | -1.1139  | 2.4317            | -5.8799                  | 3.6520  |
|                      | 105 | 2004 | -0.1434  | 2.5874            | -5.2146                  | 4.9278  |
| y2                   | 101 | 2000 | -2.0987  | 1.1910            | -4.4330                  | 0.2355  |
|                      | 102 | 2001 | -2.7705  | 1.4767            | -5.6647                  | 0.1237  |
|                      | 103 | 2002 | -2.7572  | 1.7421            | -6.1717                  | 0.6572  |
|                      | 104 | 2003 | -2.2494  | 2.0193            | -6.2071                  | 1.7082  |
|                      | 105 | 2004 | -1.4746  | 2.2517            | -5.8878                  | 2.9386  |

**Figure 4.9.** Forecasts

The table in Figure 4.9 gives forecasts and their prediction error covariances.

## Bayesian Vector Autoregressive Process

The Bayesian Vector AutoRegressive (BVAR) model is used to avoid problems of collinearity and over-parameterization that often occur with the use of VAR models. BVAR models do this by imposing priors on the AR parameters.

The following statements fit a BVAR(1) model to the simulated data. You specify the PRIOR= option with the hyper-parameters. The LAMBDA=0.9 and THETA=0.1 are hyper-parameters controlling the prior covariance. Part of the VARMAX procedure output is shown in Figure 4.10.

```
proc varmax data=simul1;
  model y1 y2 / p=1 noint
              prior=(lambda=0.9 theta=0.1);
run;
```

| The VARMAX Procedure                 |           |                              |           |         |         |          |
|--------------------------------------|-----------|------------------------------|-----------|---------|---------|----------|
| Type of Model                        |           |                              | BVAR(1)   |         |         |          |
| Estimation Method                    |           | Method of Moments Estimation |           |         |         |          |
| Prior LAMBDA                         |           | 0.9                          |           |         |         |          |
| Prior THETA                          |           | 0.1                          |           |         |         |          |
| Model Parameter Estimates            |           |                              |           |         |         |          |
| Equation                             | Parameter | Estimate                     | Std Error | T Ratio | Prob> T | Variable |
| y1(t)                                | AR1_1_1   | 1.05623                      | 0.05050   | 20.92   | 0.0001  | y1(t-1)  |
|                                      | AR1_1_2   | -0.34707                     | 0.04824   | -7.19   | 0.0001  | y2(t-1)  |
| y2(t)                                | AR1_2_1   | 0.40068                      | 0.04889   | 8.20    | 0.0001  | y1(t-1)  |
|                                      | AR1_2_2   | 0.48728                      | 0.05740   | 8.49    | 0.0001  | y2(t-1)  |
| Covariance Matrix for the Innovation |           |                              |           |         |         |          |
| Variable                             |           | y1                           | y2        |         |         |          |
| y1                                   |           | 1.35807                      | 0.44152   |         |         |          |
| y2                                   |           | 0.44152                      | 1.45070   |         |         |          |

**Figure 4.10.** Parameter Estimates for BVAR(1) Model

The output in Figure 4.10 shows that parameter estimates are slightly different from those in Figure 4.3. By choosing the appropriate priors, you may be able to get more accurate forecasts using a BVAR model rather than using an unconstrained VAR model.

## Vector Error Correction Model

A Vector Error Correction Model (VECM) can lead to a better understanding of the nature of any nonstationarity among the different component series and can also improve longer term forecasting over an unconstrained model.

The VECM( $p$ ) form is written as

$$\Delta \mathbf{y}_t = \boldsymbol{\delta} + \Pi \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where  $\Delta$  is the differencing operator, such that  $\Delta \mathbf{y}_t = \mathbf{y}_t - \mathbf{y}_{t-1}$ .

It has an equivalent VAR( $p$ ) representation as described in the preceding section.

$$\mathbf{y}_t = \boldsymbol{\delta} + (I_k + \Pi + \Phi_1^*) \mathbf{y}_{t-1} + \sum_{i=2}^{p-1} (\Phi_i^* - \Phi_{i-1}^*) \mathbf{y}_{t-i} - \Phi_{p-1}^* \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t.$$

### Example of Vector Error Correction Model

An example of the second-order nonstationary vector autoregressive model is

$$\mathbf{y}_t = \begin{pmatrix} -0.2 & 0.1 \\ 0.5 & 0.2 \end{pmatrix} \mathbf{y}_{t-1} + \begin{pmatrix} 0.8 & 0.7 \\ -0.4 & 0.6 \end{pmatrix} \mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$$

with

$$\Sigma = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix} \text{ and } \mathbf{y}_0 = 0.$$

This process can be given the following VECM(2) representation:

$$\Delta \mathbf{y}_t = \begin{pmatrix} -0.4 \\ 0.1 \end{pmatrix} (1, -2) \mathbf{y}_{t-1} - \begin{pmatrix} 0.8 & 0.7 \\ -0.4 & 0.6 \end{pmatrix} \Delta \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t.$$

The following PROC IML statements generate simulated data for the VECM(2) form specified above and plot the data as shown in Figure 4.11:

```
proc iml;
  sig = 100*i(2);
  phi = {-0.2 0.1, 0.5 0.2, 0.8 0.7, -0.4 0.6};
  call varmasim(y,phi) sigma = sig n = 100 initial = 0
               seed = 45876;
  cn = {'y1' 'y2'};
  create simul2 from y[colname=cn];
  append from y;
quit;

data simul2;
  set simul2;
  t+1;
```

```

proc gplot data=simul2;
  symbol1 v = none i = join l = 1;
  symbol2 v = none i = join l = 2;
  plot y1 * t = 1
       y2 * t = 2 / overlay;
run;

```

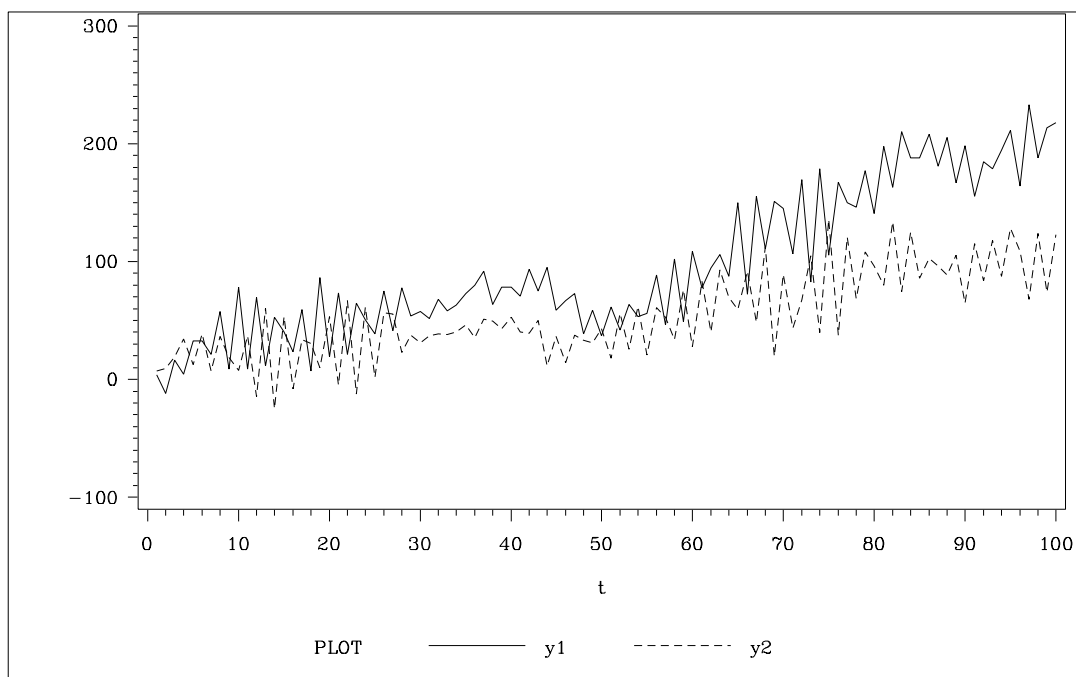


**Figure 4.11.** Plot of Generated Data Process

### Cointegration Testing

The following statements use the Johansen cointegration rank test. The COINTTEST=(JOHANSEN) option does the Johansen trace test and is equivalent to the COINTTEST or COINTTEST=(JOHANSEN=(TYPE=TRACE)) option.

```

proc varmax data=simul2;
  model y1 y2 / p=2 noint cointtest=(johansen);
run;

```

| The VARMAX Procedure    |                |            |       |                   |                |                    |
|-------------------------|----------------|------------|-------|-------------------|----------------|--------------------|
| Cointegration Rank Test |                |            |       |                   |                |                    |
| H_0:<br>Rank=r          | H_1:<br>Rank>r | Eigenvalue | Trace | Critical<br>Value | Drift<br>InECM | DriftIn<br>Process |
| 0                       | 0              | 0.5086     | 70.73 | 12.21             | NOINT          | Constant           |
| 1                       | 1              | 0.0111     | 1.09  | 4.14              |                |                    |

**Figure 4.12.** Cointegration Rank Test

Figure 4.12 shows the output for the Johansen cointegration rank test. The last two columns explain the drift in the model or process. Since the NOINT option is speci-

fied, the model is

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \Phi_1^* \Delta \mathbf{y}_{t-1} + \epsilon_t$$

but the process has a constant drift before differencing.

H\_0 is the null hypothesis and H\_1 is the alternative hypothesis. The first row tests  $r = 0$  against  $r > 0$ ; the second row tests  $r = 1$  against  $r > 1$ . The trace test statistics in the fourth column are computed by  $-T \sum_{i=r+1}^k \log(1 - \lambda_i)$  where  $T$  is the available number of observations and  $\lambda_i$  is the eigenvalue in the third column. By default, the critical values at 5% significance level are used for testing. You can compare the test statistics and critical values in each row; there is one cointegrated process.

The following statements fit a VECM(2) form to the simulated data. From the result in Figure 4.12, the time series are cointegrated with rank=1. You specify the ECM= option with the RANK=1. For normalizing the value of the cointegrated vector, you specify the normalized variable with the NORMALIZE= option. The VARMAX procedure output is shown in Figure 4.13 and Figure 4.14.

```
proc varmax data=simul2;
  model y1 y2 / p=2 noint print=(iarr)
              ecm=(rank=1 normalize=y1);
run;
```

| The VARMAX Procedure                                        |                              |
|-------------------------------------------------------------|------------------------------|
| Type of Model                                               | VECM(2)                      |
| Estimation Method                                           | Method of Moments Estimation |
| Cointegrated Rank                                           | 1                            |
| Long-Run Parameter<br>BETA Estimates<br>given RANK = 1      |                              |
| Variable                                                    | Dummy 1                      |
| y1                                                          | 1.00000                      |
| y2                                                          | -1.95575                     |
| Adjustment Coefficient<br>ALPHA Estimates<br>given RANK = 1 |                              |
| Variable                                                    | Dummy 1                      |
| y1                                                          | -0.46680                     |
| y2                                                          | 0.10667                      |

**Figure 4.13.** Parameter Estimates for VECM(2) Form

The ECM= option produces the estimates of the long-run parameter,  $\beta$ , and the adjustment coefficient,  $\alpha$ . In Figure 4.13, “Dummy 1” indicates the first column of the  $\alpha$  and  $\beta$  matrices. Since the cointegration rank is 1 in the bivariate system,  $\alpha$  and  $\beta$

are two-dimensional vectors. The estimated cointegrating vector is  $\hat{\beta} = (1, -1.96)'$ ; the long-run relationship between  $y_{1t}$  and  $y_{2t}$  is  $y_{1t} = 1.96y_{2t}$ . The first element of  $\hat{\beta}$  is 1 since  $y_1$  is specified as the normalized variable.

| The VARMAX Procedure              |           |          |           |         |         |           |
|-----------------------------------|-----------|----------|-----------|---------|---------|-----------|
| Parameter ALPHA * BETA' Estimates |           |          |           |         |         |           |
| Variable                          | y1        | y2       |           |         |         |           |
| y1                                | -0.46680  | 0.91295  |           |         |         |           |
| y2                                | 0.10667   | -0.20862 |           |         |         |           |
| AR Coefficient Estimates          |           |          |           |         |         |           |
| DIF_Lag                           | Variable  | y1       | y2        |         |         |           |
| 1                                 | y1        | -0.74332 | -0.74621  |         |         |           |
|                                   | y2        | 0.40493  | -0.57157  |         |         |           |
| Model Parameter Estimates         |           |          |           |         |         |           |
| Equation                          | Parameter | Estimate | Std Error | T Ratio | Prob> T | Variable  |
| D_y1(t)                           | AR1_1_1   | -0.46680 | 0.04786   | -9.75   | 0.0001  | y1(t-1)   |
|                                   | AR1_1_2   | 0.91295  | 0.09359   | 9.75    | 0.0001  | y2(t-1)   |
|                                   | AR2_1_1   | -0.74332 | 0.04526   | -16.42  | 0.0001  | D_y1(t-1) |
|                                   | AR2_1_2   | -0.74621 | 0.04769   | -15.65  | 0.0001  | D_y2(t-1) |
| D_y2(t)                           | AR1_2_1   | 0.10667  | 0.05146   | 2.07    | 0.0409  | y1(t-1)   |
|                                   | AR1_2_2   | -0.20862 | 0.10064   | -2.07   | 0.0409  | y2(t-1)   |
|                                   | AR2_2_1   | 0.40493  | 0.04867   | 8.32    | 0.0001  | D_y1(t-1) |
|                                   | AR2_2_2   | -0.57157 | 0.05128   | -11.15  | 0.0001  | D_y2(t-1) |

**Figure 4.14.** Parameter Estimates for VECM(2) Form Continued

Figure 4.14 shows parameter estimates in terms of one lagged coefficient,  $y_{t-1}$ , and one differenced lagged coefficient,  $\Delta y_{t-1}$ , and their significances. “ALPHA \* BETA” indicates the coefficient of  $y_{t-1}$  and is obtained by multiplying the “ALPHA” and “BETA” estimates in Figure 4.13. AR1\_ $i$ \_ $j$  corresponds to the elements in the matrix “ALPHA \* BETA’”; AR2\_ $i$ \_ $j$  corresponds to the elements in the matrix of AR coefficient estimates.

The fitted model is given as

$$\Delta \mathbf{y}_t = \begin{pmatrix} -0.467 & 0.913 \\ (0.048) & (0.094) \\ 0.107 & -0.209 \\ (0.051) & (0.100) \end{pmatrix} \mathbf{y}_{t-1} + \begin{pmatrix} -0.743 & -0.746 \\ (0.045) & (0.048) \\ 0.405 & -0.572 \\ (0.049) & (0.051) \end{pmatrix} \Delta \mathbf{y}_{t-1} + \epsilon_t.$$

A  $D_{-}$  prefixed to a variable name implies differencing.



| The VARMAX Procedure             |          |          |         |  |
|----------------------------------|----------|----------|---------|--|
| Infinite Order AR Representation |          |          |         |  |
| Lag                              | Variable | y1       | y2      |  |
| 1                                | y1       | -0.21013 | 0.16674 |  |
|                                  | y2       | 0.51160  | 0.21980 |  |
| 2                                | y1       | 0.74332  | 0.74621 |  |
|                                  | y2       | -0.40493 | 0.57157 |  |

**Figure 4.15.** Change VECM(2) Form to VAR(2) Model

The PRINT=(IARR) option in the previous SAS statements prints the reparamerized coefficient estimates. The VECM(2) form in Figure 4.15 can be rewritten as the following second-order vector autoregressive model:

$$\mathbf{y}_t = \begin{pmatrix} -0.210 & 0.167 \\ 0.512 & 0.220 \end{pmatrix} \mathbf{y}_{t-1} + \begin{pmatrix} 0.743 & 0.746 \\ -0.405 & 0.572 \end{pmatrix} \mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t.$$

## Bayesian Vector Error Correction Model

Bayesian inference on a cointegrated system begins by using the priors of  $\beta$  obtained from the VECM(p) form. Bayesian vector error correction models can improve forecast accuracy for cointegrated processes. The following statements fit a BVECM(2) form to the simulated data. You specify both the PRIOR= and ECM= options for the Bayesian vector error correction model. The VARMAX procedure output is shown in Figure 4.16.

```
proc varmax data=simul2;
  model y1 y2 / p=2 noint
               prior=(lambda=0.5 theta=0.2)
               ecm=(rank=1 normalize=y1);
run;
```

| The VARMAX Procedure              |                              |          |          |
|-----------------------------------|------------------------------|----------|----------|
| Type of Model                     |                              |          | BVECM(2) |
| Estimation Method                 | Method of Moments Estimation |          |          |
| Cointegrated Rank                 |                              |          | 1        |
| Prior LAMBDA                      |                              |          | 0.5      |
| Prior THETA                       |                              |          | 0.2      |
| Adjustment Coefficient            |                              |          |          |
| ALPHA Estimates                   |                              |          |          |
| given RANK = 1                    |                              |          |          |
| Variable                          |                              | Dummy 1  |          |
| y1                                |                              | -0.34392 |          |
| y2                                |                              | 0.16659  |          |
| Parameter ALPHA * BETA' Estimates |                              |          |          |
| Variable                          | y1                           | y2       |          |
| y1                                | -0.34392                     | 0.67262  |          |
| y2                                | 0.16659                      | -0.32581 |          |
| AR Coefficient Estimates          |                              |          |          |
| DIF_Lag                           | Variable                     | y1       | y2       |
| 1                                 | y1                           | -0.80070 | -0.59320 |
|                                   | y2                           | 0.33417  | -0.53480 |

**Figure 4.16.** Parameter Estimates for BVECM(2) Form

Figure 4.16 shows the model type fitted to the data, the estimates of the adjustment coefficient  $\alpha$ , the parameter estimates in terms of one lagged coefficient, and one differenced lagged coefficient.

## Vector Autoregressive Process with Exogenous Variables

A VAR process can be affected by other observable variables that are determined outside the system of interest. Such variables are called exogenous (independent) variables. Exogenous variables can be stochastic or nonstochastic. The process can also be affected by the lags of exogenous variables. A model used to describe this process is called a VARX( $p,s$ ) model.

The VARX( $p,s$ ) model is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^s \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t.$$

For example, a VARX(1,0) model is

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \Theta_0^* \mathbf{x}_t + \boldsymbol{\epsilon}_t$$

where  $\mathbf{y}_t = (y_{1t}, y_{2t}, y_{3t})'$  and  $\mathbf{x}_t = (x_{1t}, x_{2t})'$ .

The following statements fit the VARX(1,0) model to the given data:

```
data grunfeld;
  input year y1 y2 y3 x1 x2 x3;
  label y1='Gross Investment GE'
        y2='Capital Stock Lagged GE'
        y3='Value of Outstanding Shares GE Lagged'
        x1='Gross Investment W'
        x2='Capital Stock Lagged W'
        x3='Value of Outstanding Shares Lagged W';
datalines;
... data lines omitted ...
;

proc varmax data=grunfeld;
  model y1-y3 = x1 x2 / p=1;
run;
```

The VARMAX procedure output is shown in Figure 4.17 through Figure 4.19.

| The VARMAX Procedure |       |          |                                       |           |           |           |
|----------------------|-------|----------|---------------------------------------|-----------|-----------|-----------|
|                      |       |          | Number of Observations                |           | 20        |           |
|                      |       |          | Number of Pairwise Missing            |           | 0         |           |
| Variable             | Type  | NoMissN  | Mean                                  | StdDev    | Min       | Max       |
| y1                   | DEP   | 20       | 102.29000                             | 48.58450  | 33.10000  | 189.60000 |
| y2                   | DEP   | 20       | 1941                                  | 413.84329 | 1171      | 2803      |
| y3                   | DEP   | 20       | 400.16000                             | 250.61885 | 97.80000  | 888.90000 |
| x1                   | INDEP | 20       | 42.89150                              | 19.11019  | 12.93000  | 90.08000  |
| x2                   | INDEP | 20       | 670.91000                             | 222.39193 | 191.50000 | 1194      |
|                      |       | Variable | Label                                 |           |           |           |
|                      |       | y1       | Gross Investment GE                   |           |           |           |
|                      |       | y2       | Capital Stock Lagged GE               |           |           |           |
|                      |       | y3       | Value of Outstanding Shares GE Lagged |           |           |           |
|                      |       | x1       | Gross Investment W                    |           |           |           |
|                      |       | x2       | Capital Stock Lagged W                |           |           |           |

**Figure 4.17.** Descriptive Statistics for VARX(1,0) Model

Figure 4.17 shows the descriptive statistics for the endogenous (dependent) and exogenous (independent) variables with labels.

| The VARMAX Procedure       |                          |           |          |
|----------------------------|--------------------------|-----------|----------|
| Type of Model              | VARX(1,0)                |           |          |
| Estimation Method          | Least Squares Estimation |           |          |
| Constant Estimates         |                          |           |          |
|                            | Variable                 | Constant  |          |
|                            | y1                       | -12.01279 |          |
|                            | y2                       | 702.08673 |          |
|                            | y3                       | -22.42110 |          |
| XLag Coefficient Estimates |                          |           |          |
| Lag                        | Variable                 | x1        | x2       |
| 0                          | y1                       | 1.69281   | -0.00859 |
|                            | y2                       | -6.09850  | 2.57980  |
|                            | y3                       | -0.02317  | -0.01274 |

**Figure 4.18.** Parameter Estimates for VARX(1,0) Model

Figure 4.18 shows parameter estimates for the constant and the lag zero exogenous variables.

| The VARMAX Procedure      |           |           |           |         |         |          |
|---------------------------|-----------|-----------|-----------|---------|---------|----------|
| Model Parameter Estimates |           |           |           |         |         |          |
| Equation                  | Parameter | Estimate  | Std Error | T Ratio | Prob> T | Variable |
| y1(t)                     | CONST1    | -12.01279 | 27.47108  | -0.44   | 0.6691  |          |
|                           | XL0_1_1   | 1.69281   | 0.54395   | 3.11    | 0.0083  | x1(t)    |
|                           | XL0_1_2   | -0.00859  | 0.05361   | -0.16   | 0.8752  | x2(t)    |
|                           | AR1_1_1   | 0.23699   | 0.20668   | 1.15    | 0.2722  | y1(t-1)  |
|                           | AR1_1_2   | 0.00763   | 0.01627   | 0.47    | 0.6470  | y2(t-1)  |
| y2(t)                     | AR1_1_3   | 0.02941   | 0.04852   | 0.61    | 0.5548  | y3(t-1)  |
|                           | CONST2    | 702.08673 | 256.48046 | 2.74    | 0.0169  |          |
|                           | XL0_2_1   | -6.09850  | 5.07849   | -1.20   | 0.2512  | x1(t)    |
|                           | XL0_2_2   | 2.57980   | 0.50056   | 5.15    | 0.0002  | x2(t)    |
|                           | AR1_2_1   | -2.46656  | 1.92967   | -1.28   | 0.2235  | y1(t-1)  |
| y3(t)                     | AR1_2_2   | 0.16379   | 0.15193   | 1.08    | 0.3006  | y2(t-1)  |
|                           | AR1_2_3   | -0.84090  | 0.45304   | -1.86   | 0.0862  | y3(t-1)  |
|                           | CONST3    | -22.42110 | 10.31166  | -2.17   | 0.0487  |          |
|                           | XL0_3_1   | -0.02317  | 0.20418   | -0.11   | 0.9114  | x1(t)    |
|                           | XL0_3_2   | -0.01274  | 0.02012   | -0.63   | 0.5377  | x2(t)    |
|                           | AR1_3_1   | 0.95116   | 0.07758   | 12.26   | 0.0001  | y1(t-1)  |
|                           | AR1_3_2   | 0.00224   | 0.00611   | 0.37    | 0.7201  | y2(t-1)  |
|                           | AR1_3_3   | 0.93801   | 0.01821   | 51.50   | 0.0001  | y3(t-1)  |

**Figure 4.19.** Parameter Estimates for VARX(1,0) Model Continued

Figure 4.19 shows the parameter estimates and their significances.

The fitted model is given as

$$\begin{pmatrix} y_{1t} \\ y_{2t} \\ y_{3t} \end{pmatrix} = \begin{pmatrix} -12.013 \\ (27.471) \\ 702.086 \\ (256.480) \\ -22.421 \\ (10.312) \end{pmatrix} + \begin{pmatrix} 1.693 & -0.009 \\ (0.544) & (0.054) \\ -6.099 & 2.580 \\ (5.078) & (0.501) \\ -0.023 & -0.013 \\ (0.204) & (0.020) \end{pmatrix} \begin{pmatrix} x_{1t} \\ x_{2t} \end{pmatrix} + \begin{pmatrix} 0.237 & 0.008 & 0.029 \\ (0.207) & (0.016) & (0.049) \\ -2.467 & 0.164 & -0.841 \\ (1.930) & (0.152) & (0.453) \\ 0.951 & 0.002 & 0.938 \\ (0.078) & (0.006) & (0.018) \end{pmatrix} \begin{pmatrix} y_{1,t-1} \\ y_{2,t-1} \\ y_{3,t-1} \end{pmatrix} + \begin{pmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \\ \epsilon_{3,t} \end{pmatrix}.$$

---

## Parameter Estimation and Testing on Restrictions

In the previous example, the VARX(1,0) model is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \Theta_0^* \mathbf{x}_t + \Phi_1 \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

with

$$\Theta_0^* = \begin{pmatrix} \theta_{11}^* & \theta_{12}^* \\ \theta_{21}^* & \theta_{22}^* \\ \theta_{31}^* & \theta_{32}^* \end{pmatrix} \quad \Phi_1 = \begin{pmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{pmatrix}.$$

In Figure 4.19, you can see the coefficients XL\_0\_1\_2, AR\_1\_1\_2, and AR\_1\_3\_2 are insignificant. The following statements restrict the coefficients of  $\theta_{12}^* = \phi_{12} = \phi_{32} = 0$  for the VARX(1,0) model.

```
proc varmax data=grunfeld;
  model y1-y3 = x1 x2 / p=1;
  restrict XL(0,1,2)=0, AR(1,1,2)=0, AR(1,3,2)=0;
run;
```

| The VARMAX Procedure       |          |          |          |          |
|----------------------------|----------|----------|----------|----------|
| XLag Coefficient Estimates |          |          |          |          |
| Lag                        | Variable | x1       | x2       |          |
| 0                          | y1       | 1.67592  | 0        |          |
|                            | y2       | -6.30880 | 2.65308  |          |
|                            | y3       | -0.03576 | -0.00919 |          |
| AR Coefficient Estimates   |          |          |          |          |
| Lag                        | Variable | y1       | y2       | y3       |
| 1                          | y1       | 0.27671  | 0        | 0.01747  |
|                            | y2       | -2.16968 | 0.10945  | -0.93053 |
|                            | y3       | 0.96398  | 0        | 0.93412  |

**Figure 4.20.** Parameter Estimation on Restrictions

The output in Figure 4.20 shows that three parameters  $\theta_{12}^*$ ,  $\phi_{12}$ , and  $\phi_{32}$  are replaced by the restricted values.

| The VARMAX Procedure |                     |           |         |         |
|----------------------|---------------------|-----------|---------|---------|
| Restriction Results  |                     |           |         |         |
| Parameter            | Lagrange Multiplier | Std Error | T Ratio | Prob> T |
| XL0_1_2              | 1.74969             | 21.44026  | 0.08    | 0.9389  |
| AR1_1_2              | 30.36254            | 70.74347  | 0.43    | 0.6899  |
| AR1_3_2              | 55.42191            | 164.03075 | 0.34    | 0.7524  |

**Figure 4.21.** RESTRICT Statement Results

The output in Figure 4.21 shows the estimates of the Lagrangian parameters and their significances. You cannot reject the null hypotheses  $\theta_{12}^* = 0$ ,  $\phi_{12} = 0$ , and  $\phi_{32} = 0$  with the 0.05 significance level.

The TEST statement in the following example tests  $\phi_{31} = 0$  and  $\theta_{12}^* = \phi_{12} = \phi_{32} = 0$  for the VARX(1,0) model:

```
proc varmax data=grunfeld;
  model y1-y3 = x1 x2/ p=1;
  test AR(1,3,1)=0;
  test XL(0,1,2)=0, AR(1,1,2)=0, AR(1,3,2)=0;
run;
```

| The VARMAX Procedure |            |    |            |
|----------------------|------------|----|------------|
| Test Results         |            |    |            |
| Test                 | Chi-Square | DF | Prob>Chisq |
| 1                    | 150.31     | 1  | <.0001     |
| 2                    | 0.34       | 3  | 0.9522     |

**Figure 4.22.** TEST Statement Results

The output in Figure 4.22 shows that the first column in the output is the index corresponding to each TEST statement; you can reject the hypothesis test  $\phi_{31} = 0$  at the 0.05 significance level; you cannot reject the joint hypothesis test  $\theta_{12}^* = \phi_{12} = \phi_{32} = 0$  at the 0.05 significance level.

## Causality Testing

The following statements use the CAUSAL statement to compute the Granger-Causality test for the VAR(1) model:

```
proc varmax data=grunfeld;
  model y1-y3 = x1 x2 / p=1 noprint;
  causal group1=(y1 y3) group2=(x2);
  causal group1=(y1 y3) group2=(y2);
run;
```

| The VARMAX Procedure                                      |            |    |            |
|-----------------------------------------------------------|------------|----|------------|
| Granger Causality Wald Test                               |            |    |            |
| Test                                                      | Chi-Square | DF | Prob>Chisq |
| 1                                                         | 2.95       | 2  | 0.2291     |
| 2                                                         | 0.89       | 2  | 0.6420     |
| Test 1: Group 1 Variables: y1 y3<br>Group 2 Variables: x2 |            |    |            |
| Test 2: Group 1 Variables: y1 y3<br>Group 2 Variables: y2 |            |    |            |

**Figure 4.23.** CAUSAL Statement Results

The output in Figure 4.23 is associated with the CAUSAL statement. The first column in the output is the index corresponding to each CAUSAL statement. The output shows that you cannot reject Granger-noncausality from  $x_2$  to  $(y_1, y_3)$  using a 0.05 significance level; you cannot reject Granger-noncausality from  $y_2$  to  $(y_1, y_3)$ .

## Syntax

```

PROC VARMAX options ;
  BY variables ;
  CAUSAL group1 = (variables) group2 = (variables) ;
  COINTEG rank = number < options > ;
  ID variable interval = value < option > ;
  MODEL dependent variables < = regressors >
    < , dependent variables < = regressors > ... >
    < / options > ;
  OUTPUT < options > ;
  RESTRICT restrictions ;
  TEST restrictions ;

```

## Functional Summary

The statements and options used with the VARMAX procedure are summarized in the following table:

| Description                                                                                              | Statement | Option     |
|----------------------------------------------------------------------------------------------------------|-----------|------------|
| <b>Data Set Options</b>                                                                                  |           |            |
| specify the input data set                                                                               | VARMAX    | DATA=      |
| write parameter estimates to an output data set                                                          | VARMAX    | OUTEST=    |
| include covariances in the OUTEST= data set                                                              | VARMAX    | OUTCOV     |
| write the diagnostic checking tests for a model and the cointegration test results to an output data set | VARMAX    | OUTSTAT=   |
| write actuals, predictions, residuals, and confidence limits to an output data set                       | OUTPUT    | OUT=       |
| <b>BY Groups</b>                                                                                         |           |            |
| specify BY-group processing                                                                              | BY        |            |
| <b>ID Variable</b>                                                                                       |           |            |
| specify identifying variable                                                                             | ID        |            |
| specify the time interval between observations                                                           | ID        | INTERVAL=  |
| control the alignment of SAS Date values                                                                 | ID        | ALIGN=     |
| <b>Printing Control Options</b>                                                                          |           |            |
| specify how many lags to print results                                                                   | MODEL     | LAGMAX=    |
| suppress the printed output                                                                              | MODEL     | NOPRINT    |
| request all printing options                                                                             | MODEL     | PRINTALL   |
| request the printing format                                                                              | MODEL     | PRINTFORM= |
| <b>PRINT= Option</b>                                                                                     |           |            |
| print the correlation matrix of parameter estimates                                                      | MODEL     | CORRB      |



| Description                                                           | Statement | Option     |
|-----------------------------------------------------------------------|-----------|------------|
| print the cross-correlation matrices of independent variables         | MODEL     | CORRX      |
| print the cross-correlation of dependent variables                    | MODEL     | CORRY      |
| print the covariance matrices of prediction errors                    | MODEL     | COVPE      |
| print the cross-covariance matrices of the independent variables      | MODEL     | COVX       |
| print the cross-covariance matrices of the dependent variables        | MODEL     | COVY       |
| print the covariance matrix of parameter estimates                    | MODEL     | COVB       |
| print the decomposition of the prediction error covariance matrix     | MODEL     | DECOMPOSE  |
| print the infinite order AR representation                            | MODEL     | IARR       |
| print the impulse response function                                   | MODEL     | IMPULSE=   |
| print the impulse response function in the transfer function          | MODEL     | IMPULSX=   |
| print the partial autoregressive coefficient matrices                 | MODEL     | PARCOEF    |
| print the partial canonical correlation matrices                      | MODEL     | PCANCORR   |
| print the partial correlation matrices                                | MODEL     | PCORR      |
| print the eigenvalues of the companion matrix                         | MODEL     | ROOTS      |
| print the Yule-Walker estimates                                       | MODEL     | YW         |
| <b>Model Estimation and Order Selection Options</b>                   |           |            |
| center the dependent variables                                        | MODEL     | CENTER     |
| specify the degrees of differencing for the specified model variables | MODEL     | DIF=       |
| specify the degrees of differencing for all independent variables     | MODEL     | DIFX=      |
| specify the degrees of differencing for all dependent variables       | MODEL     | DIFY=      |
| specify the vector error correction model                             | MODEL     | ECM=       |
| specify the estimation method                                         | MODEL     | METHOD=    |
| select the tentative order                                            | MODEL     | MINIC=     |
| suppress the current values of independent variables                  | MODEL     | NOCURRENTX |
| suppress the intercept parameters                                     | MODEL     | NOINT      |
| specify the number of seasonal periods                                | MODEL     | NSEASON=   |
| specify the order of autoregressive polynomial                        | MODEL     | P=         |
| specify the Bayesian prior model                                      | MODEL     | PRIOR=     |
| center the seasonal dummies                                           | MODEL     | SCENTER    |
| specify the degree of time trend polynomial                           | MODEL     | TREND=     |
| specify the denominator for error covariance matrix estimates         | MODEL     | VARDEF=    |
| specify the lag order of independent variables                        | MODEL     | XLAG=      |

| Description                                                                | Statement        | Option                  |
|----------------------------------------------------------------------------|------------------|-------------------------|
| <b>Cointegration Related Options</b>                                       |                  |                         |
| print the results from the weak exogeneity test of the long-run parameters | COINTEG          | EXOGENEITY              |
| specify the restriction on the cointegrated coefficient matrix             | COINTEG          | H=                      |
| specify the restriction on the adjustment coefficient matrix               | COINTEG          | J=                      |
| specify the variable name whose cointegrating vectors are normalized       | COINTEG          | NORMALIZE=              |
| specify a cointegration rank                                               | COINTEG          | RANK=                   |
| print the Johansen cointegration rank test                                 | MODEL            | COINTTEST= (JOHANSEN= ) |
| print the Stock-Watson common trends test                                  | MODEL            | COINTTEST=(SW= )        |
| print the Dickey-Fuller unit root test                                     | MODEL            | DFTEST=                 |
| <b>Tests and Restrictions on Parameters</b>                                |                  |                         |
| test the Granger Causality                                                 | CAUSAL           | GROUP1= GROUP2=         |
| place and test restrictions on parameter estimates<br>test hypotheses      | RESTRICT<br>TEST |                         |
| <b>Output Control Options</b>                                              |                  |                         |
| specify the size of confidence limits for forecasting                      | OUTPUT           | ALPHA=                  |
| start forecasting before end of the input data                             | OUTPUT           | BACK=                   |
| specify how many periods to forecast                                       | OUTPUT           | LEAD=                   |
| suppress the printed forecasts                                             | OUTPUT           | NOPRINT                 |

## PROC VARMAX Statement

### PROC VARMAX *options* ;

The following options can be used in the PROC VARMAX statement:

#### **DATA=** *SAS-data-set*

specifies the input SAS data set. If the DATA= option is not specified, the PROC VARMAX statement uses the most recently created SAS data set.

#### **OUTEST=** *SAS-data-set*

writes the parameter estimates to the output data set.

**COVOUT  
OUTCOV**

writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

**OUTSTAT= SAS-data-set**

writes residual diagnostic results to an output data set. If the JOHANSEN= option is specified, the results of this option are also written to the output data set.

**Other Options**

In addition, any of the following MODEL statement options can be specified in the PROC VARMAX statement, which is equivalent to specifying the option for every MODEL statement: CENTER, DFTEST=, DIF=, DIFX=, DIFY=, LAG-MAX=, METHOD=, MINIC=, NOCURRENTX, NOINT, NOPRINT, NSEASON=, P=, PRINT=, PRINTALL, PRINTFORM=, SCENTER, TREND=, VARDEF=, and XLAG=.

---

**BY Statement**

**BY** *variables*;

A BY statement can be used with the PROC VARMAX statement to obtain separate analyses on observations in groups defined by the BY variables.

---

**CAUSAL Statement**

**CAUSAL GROUP1=(variables) GROUP2=(variables);**

A CAUSAL statement prints the Granger-Causality test by fitting the VAR( $p$ ) model. Any number of CAUSAL statements can be specified. The CAUSAL statement proceeds with the MODEL statement and uses the variables and the autoregressive order,  $p$ , specified in the MODEL statement. Variables in the GROUP1= and GROUP2= options should be defined in the MODEL statement. If  $p = 0$  is in the MODEL statement, the CAUSAL statement is not applicable.

See the “VAR Modeling” section on page 101 for details.

The following is an example of the CAUSAL statement. You specify the CAUSAL statement with the GROUP1= and GROUP2= options.

```
proc varmax data=one;
  model y1-y3 / p=1;
  causal group1=(y1 y3) group2=(y2);
  causal group1=(y2) group2=(y1 y3);
run;
```

## COINTEG Statement

**COINTEG** **RANK=** *number* **< H=** (*matrix*) **>** **< J=** (*matrix*) **>**  
**< EXOGENEITY** **>** **< NORMALIZE=** *variable* **>** ;

The COINTEG statement fits the vector error correction model to the data, tests the restrictions of the long-run parameters and the adjustment parameters, and tests for the weak exogeneity in the long-run parameters. The cointegrated system uses the maximum likelihood analysis proposed by Johansen and Juselius (1990) and Johansen (1995a, 1995b). Only one COINTEG statement is allowed.

The VECM( $p$ ) form is written as

$$\Delta \mathbf{y}_t = \alpha \beta' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \delta + \epsilon_t$$

where  $\alpha$  is a  $k \times r$  matrix called adjustment coefficient,  $r \leq k$ ;  $\beta$  is a  $k \times r$  matrix called the long-run coefficient;  $\Delta$  means  $\Delta \mathbf{y}_t = \mathbf{y}_t - \mathbf{y}_{t-1}$ .

The following statements are examples of the COINTEG statement and the two examples are equivalent. You specify the ECM= option for fitting the VECM( $p$ ) with the P= option in the MODEL statement or the COINTEG statement. For testing of the restrictions of either  $\alpha$  or  $\beta$  or both, you specify either J= or H= or both. You specify the EXOGENEITY option for tests of the weak exogeneity in the long-run parameters.

```
proc varmax data=one;
  model y1-y3 / p=2;
  cointeg rank=1 h=(1 0 0, -1 0 0, 0 1 0, 0 0 1)
    j=(1 0, 0 0, 0 1) exogeneity;
```

```
proc varmax data=one;
  model y1-y3 / p=2 ecm=(rank=1);
  cointeg rank=1 h=(1 0 0, -1 0 0, 0 1 0, 0 0 1)
    j=(1 0, 0 0, 0 1) exogeneity;
```

### EXOGENEITY

formulates the likelihood ratio tests for testing weak exogeneity in the long-run parameters. The hypothesis is that one variable is weakly exogenous for the others.

### H= (*matrix*)

specifies the restrictions  $H$  on the  $k \times r$  or  $(k + 1) \times r$  cointegrated coefficient matrix  $\beta$  such that  $\beta = H\phi$ , where  $H$  is known and  $\phi$  is unknown. The  $k \times m$  or  $(k + 1) \times m$  matrix  $H$  is specified using this option, where  $k$  is the number of endogenous variables, and  $m$  is  $r \leq m < k$  with RANK= $r$ . For example, consider that the system contains four variables and RANK=1 with  $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)'$ . The restriction matrix for the test of  $\beta_1 + \beta_2 = 0$  can be specified as

```
cointeg rank=1 h=(1 0 0, -1 0 0, 0 1 0, 0 0 1);
```

When the data have no deterministic trend, the constant term should be restricted by  $\alpha'_{\perp} \delta = 0$ . You can specify the restriction matrix for the previous test as follows:

```
cointeg rank=1  
h=(1 0 0 0, -1 0 0 0, 0 1 0 0, 0 0 1 0, 0 0 0 1);
```

When the cointegrated system contains three endogenous variables and RANK=2, you can specify the restriction matrix for the test of  $\beta_{1j} = -\beta_{2j}$  for  $j = 1, 2$  as follows:

```
cointeg rank=2 h=(1 0, -1 0, 0 1);
```

**J=** (*matrix*)

specifies the restrictions  $J$  on the  $k \times r$  adjustment matrix  $\alpha$  such that  $\alpha = J\psi$ , where  $J$  is known and  $\psi$  is unknown. The  $k \times m$  matrix  $J$  is specified using this option, where  $k$  is the number of endogenous variables, and  $m$  is  $r \leq m < k$  with RANK= $r$ . For example, when the system contains four variables and RANK=1, you can specify the restriction matrix for the test of  $\alpha_j = 0$  for  $j = 2, 3, 4$  as follows:

```
cointeg rank=1 j=(1, 0, 0, 0);
```

When the system contains three variables and RANK=2, you can specify the restriction matrix for the test of  $\alpha_{2j} = 0$  for  $j = 1, 2$  as follows:

```
cointeg rank=1 j=(1 0, 0 0, 0 1);
```

**NORMALIZE=** *variable*

specifies a single endogenous (dependent) variable name whose cointegrating vectors are normalized. If the variable name is different from that specified in the COINTTEST=(JOHANSEN=) or ECM= option in the MODEL statement, the variable name specified in the COINTEG statement is used. If the normalized variable is not specified, cointegrating vectors are not normalized.

**RANK=** *number*

specifies the cointegration rank of the cointegrated system. This option is required in the COINTEG statement. The rank of cointegration should be greater than zero and less than the number of endogenous (dependent) variables. If the value of the RANK= option in the COINTEG statement is different from that specified in the ECM= option, the rank specified in the COINTEG statement is used.

---

## ID Statement

**ID** *variable* **INTERVAL**= *value* < **ALIGN**= *value* > ;

The ID statement specifies a variable that identifies observations in the input data set. The variable specified in the ID statement is included in the OUT= data set. Note that the ID *variable* is usually a SAS date valued variable. The values of the ID variable are extrapolated for the forecast observations based on the values of the INTERVAL= option.

### **ALIGN**= *value*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING. The ALIGN= option is used to align the ID variable to the beginning, middle, or end of the time ID interval specified by the INTERVAL= option.

### **INTERVAL**= *value*

specifies the time interval between observations. This option is required in the ID statement. The INTERVAL=*value* is used in conjunction with the ID variable to check that the input data are in order and have no missing periods. The INTERVAL= option is also used to extrapolate the ID values past the end of the input data.

---

## MODEL Statement

**MODEL** *dependents* < = *regressors* >  
 <, *dependents* < = *regressors* > ... >  
 </options> ;

The MODEL statement specifies endogenous (dependent) variables and exogenous (independent) variables for the VARMAX model. The multivariate model can have the same or different independent variables corresponding to the dependent variables. As a special case, the VARMAX procedure allows you to analyze one dependent variable with independent variables. The one MODEL statement is required.

For example, the following statements are equivalent ways of specifying the multivariate model for the vector ( $y_1, y_2, y_3$ ):

```
model y1 y2 y3 </options>;
model y1-y3 </options>;
```

The following statements are equivalent ways of specifying the multivariate model for the vectors ( $y_1, y_2, y_3, y_4$ ) and ( $x_1, x_2, x_3, x_4, x_5$ ):

```
model y1 y2 y3 y4 = x1 x2 x3 x4 x5 </options>;
model y1 y2 y3 y4 = x1-x5 </options>;
model y1 = x1-x5, y2 = x1-x5, y3 y4 = x1-x5 </options>;
model y1-y4 = x1-x5 </options>;
```

When the multivariate model has different independent variables corresponding to the dependent variables, equations are separated by commas (,) and the model can be specified as illustrated by the following MODEL statement:

```
model y1 = x1-x3, y2 = x3-x5, y3 y4 = x1-x5 </options>;
```

The following options can be used in the MODEL statement after a forward slash (/):

### General Options

#### CENTER

centers endogenous (dependent) variables by subtracting their means. Note that centering is done after differencing when the DIF= or DIFY= option is specified. If there are exogenous (independent) variables, this option is not applicable.

#### DIF= (variable(number-list)<... variable(number-list)>)

specifies the degrees of differencing to be applied to the specified dependent or independent variables. The differencing can be the same for all variables, or it can vary among variables. For example, DIF=( $y_1(1,4)$   $y_3(1)$   $x_2(2)$ ) specifies that the  $y_1$  series is differenced at lag 1 and at lag 4, which is  $(y_{1t} - y_{1,t-1}) - (y_{1,t-4} - y_{1,t-5})$ ;  $y_3$  at lag 1, which is  $(y_{3t} - y_{3,t-1})$ ;  $x_2$  at lag 2, which is  $(x_{2t} - x_{2,t-2})$ .

#### DIFX= (number-list)

specifies the degrees of differencing to be applied to all exogenous (independent) variables. For example, DIFX=(1) specifies that the series are differenced once at lag 1; DIFX=(1,4) at lag 1 and at lag 4. If exogenous variables are specified in the DIF= option, this option is ignored.

#### DIFY= (number-list)

specifies the degrees of differencing to be applied to all endogenous (dependent) variables. For details, see the DIFX= option. If endogenous variables are specified in the DIF= option, this option is ignored.

#### METHOD= value

requests the type of estimates to be computed. The possible values of the METHOD= option are

- |    |                                         |
|----|-----------------------------------------|
| LS | specifies least-squares estimates.      |
| ML | specifies maximum likelihood estimates. |

If the PRIOR= or ECM= option or both is specified, the default is METHOD=ML; otherwise, the default is METHOD=ML.

**NOCURRENTX**

suppresses the current values  $x_t$  of exogenous (independent) variables. In general, the VARMAX( $p, q, s$ ) model is

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^s \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^q \Theta_i \boldsymbol{\epsilon}_{t-i}.$$

If this option is specified, it suppresses the current values  $\mathbf{x}_t$  and starts with  $\mathbf{x}_{t-1}$ .

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \sum_{i=1}^s \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^q \Theta_i \boldsymbol{\epsilon}_{t-i}.$$

**NOINT**

suppresses the intercept parameters  $\boldsymbol{\delta}$ .

**NSEASON= number**

specifies the number of seasonal periods. When the NSEASON=*number* option is specified, (*number*-1) seasonal dummies are added to the regressors. If the NOINT option is specified, the NSEASON= option is not applicable.

**SCENTER**

centers seasonal dummies specified by the NSEASON= option. The centered seasonal dummies are generated by  $c - (1/s)$ , where  $c$  is a seasonal dummy generated by the NSEASON= $s$  option.

**TREND= value**

specifies the degree of deterministic time trend included in the model. Valid values are as follows:

|        |                                                          |
|--------|----------------------------------------------------------|
| LINEAR | includes a linear time trend as a regressor.             |
| QUAD   | includes linear and quadratic time trends as regressors. |

The TREND=QUAD option is not applicable when the ECM= option is specified.

**VARDEF= value**

corrects for the degrees of freedom of the denominator. This option is used to calculate an error covariance matrix for the METHOD=LS. If the METHOD=ML is specified, the VARDEF=N is used. Valid values are as follows:

|    |                                                                                             |
|----|---------------------------------------------------------------------------------------------|
| DF | specifies that the number of nonmissing observation minus the number of regressors be used. |
| N  | specifies that the number of nonmissing observation be used.                                |

**Printing Control Options****LAGMAX= number**

specifies the lag to compute and display the results obtained by the PRINT=(CORRX CORRY COVX COVY IARR IMPULSE= IMPULSX= PARCOEF PCANCORR



PCORR) option. This option is also used to print cross-covariances and cross-correlations of residuals. The default is  $\text{LAGMAX}=\min(12, T-2)$ , where  $T$  is the number of nonmissing observations.

**NOPRINT**

suppresses all printed output.

**PRINTALL**

requests all printing control options. The options set by PRINTALL are DFTEST=, MINIC=, PRINTFORM=BOTH, and PRINT=( CORRB CORRX CORRY COVB COVPE COVX COVY DECOMPOSE IARR IMPULSE=(ALL) IMPULSX=(ALL) PARCOEF PCANCORR PCORR ROOTS YW ).

You can also specify this option as ALL.

**PRINTFORM=** *value*

requests the printing format of outputs of the PRINT= option and cross-covariances and cross-correlations of residuals. Valid values are as follows:

|            |                                                     |
|------------|-----------------------------------------------------|
| BOTH       | prints outputs in both MATRIX and UNIVARIATE forms. |
| MATRIX     | prints outputs in matrix form. This is the default. |
| UNIVARIATE | prints outputs by variables.                        |

**Printing Options**

**PRINT=(options)**

The following options can be used in the PRINT=( ) option. The options are listed within parentheses.

**CORRB**

prints the estimated correlations of the parameter estimates.

**CORRX**

prints the cross-correlation matrices of exogenous (independent) variables using the number of lags specified by the  $\text{LAGMAX}=\text{number}$ .

**CORRY**

prints the cross-correlation matrices of endogenous (dependent) variables using the number of lags specified by the  $\text{LAGMAX}=\text{number}$ .

**COVB**

prints the estimated covariances of the parameter estimates.

**COVPE**

**COVPE(number)**

prints the covariance matrices of *number*-ahead prediction errors for the  $\text{VARMAX}(p,q,s)$  model. If the DIF= or DIFY= option is specified, the covariance matrices of multistep-ahead prediction errors are computed based on the differenced data. This option is not applicable when the PRIOR= option is specified. See the “Forecasting” section on page 87 for details.

**COVX**

prints the cross-covariance matrices of exogenous (independent) variables using the number of lags specified by the `LAGMAX=number`.

**COVY**

prints the cross-covariance matrices of endogenous (dependent) variables using the number of lags specified by the `LAGMAX=number`.

**DECOMPOSE****DECOMPOSE(number)**

prints the decomposition of the prediction error covariances using the number of lags specified by *number* in parentheses for the `VARMAX(p,q,s)` model. It can be interpreted as the contribution of innovations in one variable to the mean squared error of the multistep-ahead forecast of another variable. The `DECOMPOSE` option also prints proportions of the forecast error variance.

If the `DIF=` or `DIFY=` option is specified, the covariance matrices of multistep-ahead prediction errors are computed based on the differenced data. This option is not applicable when the `PRIOR=` option is specified. See the “Forecasting” section on page 87 for details.

**IARR**

prints the infinite order AR representation of a VARMA process. The coefficient matrices print the number of lags specified by the `LAGMAX=number`. If the moving-average order is zero, this option prints the maximum order of the autoregressive characteristic function. If the `ECM=` option is specified, the reparameterized AR coefficient matrices are printed.

**IMPULSE****IMPULSE= (SIMPLE ACCUM ORTH STDERR ALL)**

prints the impulse response function using the number of lags specified by the `LAGMAX= number`. It investigates the response of one variable to an impulse in another variable in a system that involves a number of other variables as well. It is an infinite order MA representation of a VARMA process. See the “Forecasting” section on page 87 for details.

The following options can be used in the `IMPULSE=( )` option. The options are listed within parentheses.

|                     |                                                                                                                                                                                                                                        |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ACCUM</code>  | prints the accumulated impulse function.                                                                                                                                                                                               |
| <code>ALL</code>    | equivalent to specifying all of <code>SIMPLE</code> , <code>ACCUM</code> , <code>ORTH</code> , and <code>STDERR</code> .                                                                                                               |
| <code>ORTH</code>   | prints the orthogonalized impulse function.                                                                                                                                                                                            |
| <code>SIMPLE</code> | prints the impulse response function. This is the default.                                                                                                                                                                             |
| <code>STDERR</code> | prints the standard errors of the impulse response function, the accumulated impulse response function, or the orthogonalized impulse response function. If the exogenous variables are used to fit the model, this option is ignored. |

## IMPULSX

### IMPULSX= (SIMPLE ACCUM ALL)

prints the impulse response function related to exogenous (independent) variables using the number of lags specified by the `LAGMAX=number`. See the “Forecasting” section on page 87 for details.

The following options can be used in the `IMPULSX=( )` option. The options are listed within parentheses.

|        |                                                                                     |
|--------|-------------------------------------------------------------------------------------|
| ACCUM  | prints the accumulated impulse response matrices in the transfer function.          |
| ALL    | equivalent to specifying both SIMPLE and ACCUM.                                     |
| SIMPLE | prints the impulse response matrices in the transfer function. This is the default. |

## PARCOEF

prints the partial autoregression coefficient matrices,  $\Phi_{mm}$ . With a VAR process, this option is useful for the identification of the order since the  $\Phi_{mm}$  have the characteristic property that they equal zero for  $m > p$  under the hypothetical assumption of a  $\text{VAR}(p)$  model. These matrices print the number of lags specified by the `LAGMAX=number`. See the “Tentative Order Selection” section on page 96 for details.

## PCANCORR

prints the partial canonical correlations of the process at lag  $m$  and the test for testing  $\Phi_m=0$  for  $m > p$ . The lag  $m$  partial canonical correlations are the canonical correlations between  $\mathbf{y}_t$  and  $\mathbf{y}_{t-m}$ , after adjustment for the dependence of these variables on the intervening values  $\mathbf{y}_{t-1}, \dots, \mathbf{y}_{t-m+1}$ . See the “Tentative Order Selection” section on page 96 for details.

## PCORR

prints the partial correlation matrices using the number of lags specified by the `LAGMAX=number`. With a VAR process, this option is useful for a tentative order selection by the same property as the partial autoregression coefficient matrices, as described in the PARCOEF option. See the “Tentative Order Selection” section on page 96 for details.

## ROOTS

prints the eigenvalues of the  $kp \times kp$  companion matrix associated with the AR characteristic function  $\Phi(B)$ , where  $k$  is the number of endogenous (dependent) variables, and  $\Phi(B)$  is the finite order matrix polynomial in the back-shift operator  $B$ , such that  $B^i \mathbf{y}_t = \mathbf{y}_{t-i}$ . These eigenvalues indicate the stationary condition of the process since the stationary condition on the roots of  $|\Phi(B)| = 0$  in the  $\text{VAR}(p)$  model is equivalent to the condition in the corresponding  $\text{VAR}(1)$  representation that all eigenvalues of the companion matrix be less than one in absolute value. Similarly, you can use this option to check the invertibility of the MA process.

**YW**

prints Yule-Walker estimates of the preliminary autoregressive model for the endogenous (dependent) variables. The coefficient matrices are printed using the maximum order of the autoregressive process.

**Lag Specification Options**

**P=** *number*

**P=** (*number-list*)

specifies the order of the vector autoregressive process. Subset models of vector autoregressive orders can be specified as, for example, P=(1,3,4). P=3 is equivalent to P=(1,2,3). The default is P=0.

If P=0 and there are no exogenous (independent) variables, the AR polynomial order is automatically determined by minimizing an information criterion; if P=0 and the PRIOR= or ECM= option or both is specified the AR polynomial order is determined.

If the PRIOR= or ECM= option or both is specified, subset models of vector autoregressive orders are not allowed and the AR maximum order is used.

**XLAG=** *number*

**XLAG=** (*number-list*)

specifies the lags of exogenous (independent) variables. Subset models of distributed lags can be specified as, for example, XLAG=(2). The default is XLAG=0. To exclude the present values of exogenous (independent) variables from the model, the NOCURRENTX option must be used.

**Tentative Order Selection Options****MINIC**

**MINIC=** (**TYPE=***value* **P=***number* **Q=***number* **PERROR=***number*)

prints the information criterion for the appropriate AR and MA tentative order selection and for the diagnostic checks of the fitted model.

If the MINIC= option is not specified, all types of information criteria are printed for diagnostic checks of the fitted model.

The following options can be used in the MINIC=( ) option. The options are listed within parentheses.

**P=** *number*

**P=** (*p<sub>min</sub>:p<sub>max</sub>*)

specifies the range of AR orders. The default is P=(0:5).

**PERROR=** *number*

**PERROR=** (*p<sub>ε,min</sub>:p<sub>ε,max</sub>*)

specifies the range of AR orders for obtaining the error series. The default is PERROR=(*p<sub>max</sub> : p<sub>max</sub> + q<sub>max</sub>*).

**Q=** *number*

**Q=** (*q<sub>min</sub>:q<sub>max</sub>*)

specifies the range of MA orders. The default is Q=(0:5).

**TYPE=** *value*

specifies the criterion for the model order selection. Valid criteria are as follows:

|      |                                                                                        |
|------|----------------------------------------------------------------------------------------|
| AIC  | specifies the Akaike Information Criterion.                                            |
| AICC | specifies the Corrected Akaike Information Criterion. This is the default criterion.   |
| FPE  | specifies the Final Prediction Error criterion.                                        |
| HQC  | specifies the Hanna-Quinn Criterion.                                                   |
| SBC  | specifies the Schwarz Bayesian Criterion. You can also specify this value as TYPE=BIC. |

### Cointegration Related Options

#### COINTTEST

**COINTTEST= (JOHANSEN<(=options)> SW<(=options)> SIGLEVEL=number)**

The following options can be used in the COINTTEST=( ) option. The options are listed within parentheses.

#### JOHANSEN

**JOHANSEN= (TYPE=value IORDER=number NORMALIZE=variable)**

prints the cointegration rank test for multivariate time series based on Johansen method. This test is provided when the number of endogenous (dependent) variables is less than or equal to 11. See the “Vector Error Correction Modeling” section on page 113 for details.

The VAR( $p$ ) model can be written as the error correction model

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + A D_t + \epsilon_t$$

where  $\Pi$ ,  $\Phi_i^*$ , and  $A$  are coefficient parameters;  $D_t$  is a deterministic term such as a constant, a linear trend, or seasonal dummies.

The  $I(1)$  model is defined by one reduced rank condition. If the cointegration rank is  $r < k$ , then there exist  $k \times r$  matrices  $\alpha$  and  $\beta$  of rank  $r$  such that  $\Pi = \alpha\beta'$ .

The  $I(1)$  model is rewritten as the  $I(2)$  model

$$\Delta^2 \mathbf{y}_t = \Pi \mathbf{y}_{t-1} - \Psi \Delta \mathbf{y}_{t-1} + \sum_{i=1}^{p-2} \Psi_i \Delta^2 \mathbf{y}_{t-i} + A D_t + \epsilon_t$$

where  $\Psi = I_k - \sum_{i=1}^{p-1} \Phi_i^*$  and  $\Psi_i = - \sum_{j=i+1}^{p-1} \Phi_j^*$ .

The  $I(2)$  model is defined by two reduced rank conditions. One is that  $\Pi = \alpha\beta'$ , where  $\alpha$  and  $\beta$  are  $k \times r$  matrices of full rank  $r$ . The other is that  $\alpha'_\perp \Psi \beta_\perp = \xi \eta'$  where  $\xi$  and  $\eta$  are  $(k-r) \times s$  matrices with  $s \leq k-r$ . and  $\alpha'_\perp$  and  $\beta_\perp$  are  $k \times (k-r)$  matrices of full rank  $k-r$  such that  $\alpha'_\perp \alpha_\perp = 0$  and  $\beta'_\perp \beta_\perp = 0$ .

The following options can be used in the JOHANSEN=( ) option. The options are listed within parentheses.

IORDER= *number* specifies the integrated order.

IORDER=1 prints the cointegration rank test for an integrated order 1 and prints the long-run parameter,  $\beta$ , and the adjustment coefficient,  $\alpha$ . This is the default. If IORDER=1 is specified, the AR order should be greater than or equal to 1. When  $P=0$ ,  $P$  is temporarily set to 1.

IORDER=2 prints the cointegration rank test for integrated orders 1 and 2. If IORDER=2 is specified, the AR order should be greater than or equal to 2. If  $P=1$ , the IORDER=1 is used; if  $P=0$ ,  $P$  is temporarily set to 2.

NORMALIZE= *variable* specifies the endogenous (dependent) variable name whose cointegration vectors are to be normalized. If the normalized variable is different from that specified in the ECM= option or the COINTEG statement, the latter is used.

TYPE= *value* specifies the type of cointegration rank test to be printed. Valid values are as follows:

MAX prints the cointegration maximum eigenvalue test.

TRACE prints the cointegration trace test. This is the default.

If the NOINT option is not specified, the procedure prints two different cointegration rank tests in the presence of the unrestricted and restricted deterministic terms (constant or linear trend) models. If IORDER=2 is specified, the procedure automatically determines that TYPE=TRACE.

**SIGLEVEL=** *value*

sets the size of cointegration rank tests and common trends tests. The SIGLEVEL=*value* option must be one of 0.1, 0.05, or 0.01. The default is SIGLEVEL=0.05.

**SW**

**SW=** (TYPE=*value* LAG=*number*)

prints common trends tests for a multivariate time series based on the Stock-Watson method. This test is provided when the number of endogenous (dependent) variables is less than or equal to 6. See the “Common Trends” section on page 111 for details.

The following options can be used in the SW=( ) option. The options are listed within parentheses.

|                    |                                                                                                                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LAG= <i>number</i> | specifies the number of lags. The default is $LAG=\max(1,p)$ for TYPE=FILTDIF or TYPE=FILTRES, where $p$ is the AR maximum order specified by the P= option; $LAG=O(T^{1/4})$ for TYPE=KERNEL, where $T$ is the number of nonmissing observations. If LAG= exceeds the default, it is replaced by the default. |
| TYPE= <i>value</i> | specifies the type of common trends test to be printed. Valid values are as follows:                                                                                                                                                                                                                           |
| FILTDIF            | prints the common trends test based on the filtering method applied to the differenced series. This is the default.                                                                                                                                                                                            |
| FILTRES            | prints the common trends test based on the filtering method applied to the residual series.                                                                                                                                                                                                                    |
| KERNEL             | prints the common trends test based on the kernel method.                                                                                                                                                                                                                                                      |

#### DFTEST

**DFTEST= (DLAG=*number*)**

prints the Dickey-Fuller unit root test. The DLAG=*number* specifies the regular or seasonal unit root test. If the *number* is greater than one, seasonal Dickey-Fuller tests are performed. The *number* should be one of 1, 2, 4, or 12. The default is DLAG=1.

### Bayesian VAR Estimation Options

#### PRIOR

**PRIOR= (MEAN=(*vector*) LAMBDA=*value* THETA=*value* IVAR<=(*variables*)> NREP=*number* SEED=*number*)**

specifies the prior value of parameters for the BVAR( $p$ ) model. If the ECM= option is specified with the PRIOR option, the BVECM( $p$ ) form is fitted. The following options can be used in the PRIOR option. For the standard errors of the predictors, the bootstrap procedure is used. See the “Bayesian VAR Modeling” section on page 108 for details.

The following options can be used in the PRIOR=( ) option. The options are listed within parentheses.

#### IVAR

**IVAR= (*variables*)**

specifies an integrated BVAR( $p$ ) model. If you use the IVAR option without *variables*, it sets the overall prior mean of the first lag of each variable equal to one in its own equation and sets all other coefficients to zero. If *variables* are specified, it sets the prior mean of the first lag of the specified variables equal to one in its own equation and sets all other coefficients to zero. When the series  $\mathbf{y}_t = (y_1, y_2)'$  follows a bivariate BVAR(2) process, the IVAR or IVAR=( $y_1 \ y_2$ ) option is equivalent to specifying MEAN=( 1 0 0 0 0 1 0 0 ).

If the PRIOR=(MEAN= ) or ECM= option is specified, the IVAR= option is ignored.

**LAMBDA=** *value*

specifies the prior standard deviation of the AR coefficient parameter matrices. It should be a positive number. The default is LAMBDA=1. As the value of the LAMBDA= is larger, a BVAR( $p$ ) model is close to a VAR( $p$ ) model.

**MEAN=** (*vector*)

specifies the mean vector in the prior distribution for the AR coefficients. If the vector is not specified, the prior value is assumed to be a zero vector. See the “Bayesian VAR Modeling” section on page 108 for details.

You can specify the mean vector by order of the equation. Let  $B = (\delta, \Phi_1, \dots, \Phi_p)'$  be the parameter sets to be estimated. If  $\Phi = (\Phi_1, \dots, \Phi_p)'$ , then MEAN= ( vec( $\Phi$ ) ).

For example, in case of a bivariate ( $k = 2$ ) BVAR(2) model,

$$\text{MEAN} = (\phi_{1,11} \ \phi_{1,12} \ \phi_{2,11} \ \phi_{2,12} \ \phi_{1,21} \ \phi_{1,22} \ \phi_{2,21} \ \phi_{2,22})$$

where  $\phi_{l,ij}$  is the  $(i, j)$ th element of the matrix  $\Phi_l$ .

The deterministic terms are considered to shrink toward zero; you must omit prior means of deterministic terms such as a constant, seasonal dummies, or trends.

For a Bayesian error correction model, you specify a mean vector for only lagged AR coefficients,  $\Phi_i^*$ , in terms of regressors  $\Delta \mathbf{y}_{t-i}$ , for  $i = 1, \dots, (p-1)$  in the VECM( $p$ ) representation. The diffused prior variance of  $\alpha$  is used since  $\beta$  is replaced by  $\hat{\beta}$  estimated in a nonconstrained VECM( $p$ ) form.

$$\Delta \mathbf{y}_t = \alpha \mathbf{z}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + AD_t + \epsilon_t$$

where  $\mathbf{z}_t = \beta' \mathbf{y}_t$ .

For example, in case of a bivariate ( $k = 2$ ) BVECM(2) form,

$$\text{MEAN} = (\phi_{1,11}^* \ \phi_{1,12}^* \ \phi_{1,21}^* \ \phi_{1,22}^*)$$

where  $\phi_{1,ij}^*$  is the  $(i, j)$ th element of the matrix  $\Phi_1^*$ .

**NREP=** *number*

specifies the number of bootstrap replications. The default is NREP=100.

**SEED=** *number*

specifies seeds to generate uniform random numbers for resampling. By default, the system clock is used to generate the random seed.

**THETA=** *value*

specifies the prior standard deviation of the AR coefficient parameter matrices. The *value* is in the interval (0,1). The default is THETA=0.1. As the value of the THETA= is close to 1, a BVAR( $p$ ) model is close to a VAR( $p$ ) model.



### Vector Error Correction Model Options

**ECM=(RANK=number NORMALIZE=variable)**

specifies a vector error correction model.

The following options can be used in the ECM=( ) option. The options are listed within parentheses.

**NORMALIZE= variable**

specifies a single endogenous variable name whose cointegrating vectors are normalized. If the variable name is different from that specified in the COINTEG statement, the latter is used.

**RANK= number**

specifies the cointegration rank. This option is required in the ECM= option. The value of the RANK= option should be greater than zero and less than or equal to the number of endogenous (dependent) variables,  $k$ . If the rank is different from that specified in the COINTEG statement, the latter is used.

---

## OUTPUT Statement

**OUTPUT < options >;**

The OUTPUT statement generates and prints forecasts based on the model estimated in the previous MODEL statement and, optionally, creates an output SAS data set that contains these forecasts.

**ALPHA= value**

sets the forecast confidence limits. The ALPHA=value must be between 0 and 1. When you specify ALPHA= $\alpha$ , the upper and lower confidence limits define the  $1 - \alpha$  confidence interval. The default is ALPHA=0.05, which produces 95% confidence intervals.

**BACK= number**

specifies the number of observations before the end of the data at which the multistep-ahead forecasts are to begin. The BACK= option value must be less than or equal to the number of observations minus the number of lagged regressors in the model. The default is BACK=0, which means that the forecast starts at the end of the available data.

**LEAD= number**

specifies the number of multistep-ahead forecast values to compute. The default is LEAD=12.

**NOPRINT**

suppresses the printed forecast values of each endogenous (dependent) variable.

**OUT= SAS-data-set**

writes the forecast values to an output data set.

---

## RESTRICT Statement

**RESTRICT** *restriction ... restriction ;*

The RESTRICT statement restricts the specified parameters to the specified values. Only one RESTRICT statement is allowed.

The *restriction*'s form is *parameter = value* and each restriction is separated by commas. Parameters are referred by the following keywords:

- **CONST**(*i*), the intercept parameter of the current value *i*th time series  $y_{it}$
- **AR**(*l, i, j*), the autoregressive parameter of the previous lag *l* value of the *j*th endogenous (dependent) variable,  $y_{j,t-l}$ , to the *i*th endogenous variable at time *t*,  $y_{it}$
- **XL**(*l, i, j*), the exogenous parameter of the previous lag *l* value of the *j*th exogenous (independent) variable,  $x_{j,t-l}$ , to the *i*th endogenous variable at time *t*,  $y_{it}$
- **SDUMMY**(*i, j*), the *j*th seasonal dummy of the *i*th time series at time *t*,  $y_{it}$ , where  $j = 1, \dots, (nseason - 1)$

To use the RESTRICT statement, you need to know the form of the model. If you do not specify any order of the model, the RESTRICT statement is not applicable.

Restricted parameter estimates are computed by introducing a Lagrangian parameter for each restriction (Pringle and Raynor 1971). The Lagrangian parameter measures the sensitivity of the sum of square errors to the restriction. The estimates of these Lagrangian parameters and their significances are printed in the restriction results table.

The following are examples of the RESTRICT statement. The first example shows a bivariate ( $k=2$ ) VAR(2) model,

```
proc varmax data=one;
  model y1 y2 / p=2;
  restrict AR(1,1,2)=0, AR(2,1,2)=0.3;
run;
```

The following shows a bivariate ( $k=2$ ) VARX(1,1) model with three exogenous variables,

```
proc varmax data=two;
  model y1 = x1 x2, y2 = x2 x3 / p=1 xlag=1;
  restrict XL(0,1,1)=-1.2, XL(1,2,3)=0;
run;
```

---

## TEST Statement

**TEST** *restriction ... restriction ;*

The TEST statement performs the Wald test for the joint hypothesis specified in the statement. The *restriction*'s form is *parameter = value* and each restriction is separated by commas. The *restriction*'s form is referred to by the same rule in the RESTRICT statement. Any number of TEST statements can be specified.

To use the TEST statement, you need to know the form of the model. If you do not specify any order of the model, the TEST statement is not applicable.

See the “Granger-Causality Test” section on page 104 for the Wald test.

The following is an example of the TEST statement. In case of a bivariate ( $k=2$ ) VAR(2) model,

```
proc varmax data=one;
  model y1 y2 / p=2;
  test AR(1,1,2)=0, AR(2,1,2)=0;
run;
```

---

## Details

---

### Missing Values

The VARMAX procedure currently does not support missing values. The procedure uses the first contiguous group of observations with no missing values for any of the MODEL statement variables. Observations at the beginning of the data set with missing values for any MODEL statement variables are not used or included in the output data set. At the end of the data set, observations can have endogenous (dependent) variables with missing values and exogenous (independent) variables with nonmissing values.

---

### VARMAX Modeling

The vector autoregressive and moving-average model with exogenous variables is called the VARMAX( $p,q,s$ ) model. The form of the model can be written as

$$\mathbf{y}_t = \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^s \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^q \Theta_i \boldsymbol{\epsilon}_{t-i}$$

where the output variables of interest,  $\mathbf{y}_t = (y_{1t}, \dots, y_{kt})'$ , can be influenced by other input variables,  $\mathbf{x}_t = (x_{1t}, \dots, x_{rt})'$ , which are determined outside the system of interest. The variables  $\mathbf{y}_t$  are referred to as dependent, response, or endogenous variables, and the variables  $\mathbf{x}_t$  are referred to as independent, input, predictor, regressor, or exogenous variables. The unobserved noise variables,  $\boldsymbol{\epsilon}_t = (\epsilon_{1t}, \dots, \epsilon_{kt})'$ , are a vector white noise process.

The VARMAX( $p, q, s$ ) model can be written

$$\Phi(B)\mathbf{y}_t = \Theta^*(B)\mathbf{x}_t + \Theta(B)\epsilon_t$$

where

$$\begin{aligned}\Phi(B) &= I_k - \Phi_1 B - \dots - \Phi_p B^p \\ \Theta^*(B) &= \Theta_0^* + \Theta_1^* B + \dots + \Theta_s^* B^s \\ \Theta(B) &= I_k - \Theta_1 B - \dots - \Theta_q B^q\end{aligned}$$

are matrix polynomials in  $B$  in the backshift operator  $B$ , such that  $B^i \mathbf{y}_t = \mathbf{y}_{t-i}$ , the  $\Phi_i$  and  $\Theta_i$  are  $k \times k$  matrices, and the  $\Theta_i^*$  are  $k \times r$  matrices.

The following assumptions are made:

- $E(\epsilon_t) = 0$ ,  $E(\epsilon_t \epsilon_t') = \Sigma$ , which is positive-definite, and  $E(\epsilon_t \epsilon_s') = 0$  for  $t \neq s$ .
- For stationarity and invertibility of the VARMAX process, the roots of  $|\Phi(z)| = 0$  and  $|\Theta(z)| = 0$  are outside the unit circle.
- The exogenous (independent) variables,  $\mathbf{x}_t$ , are not correlated with residuals  $\epsilon_t$ ,  $E(\mathbf{x}_t \epsilon_t') = 0$ . The exogenous variables,  $\mathbf{x}_t$ , can be stochastic or nonstochastic. When the exogenous variables,  $\mathbf{x}_t$ , are stochastic and their future values are unknown, then forecasts of these future values are needed in the forecasting of the future values of the  $\mathbf{y}_t$ . On occasion, future values of the  $\mathbf{x}_t$  can be assumed to be known because they are deterministic variables. Note that the VARMAX procedure assumes that the exogenous variables,  $\mathbf{x}_t$ , are nonstochastic if future values are available in the input data set. Otherwise, the exogenous variables are assumed to be stochastic and their future values are forecasted by assuming that they follow the VARMA( $p, q$ ) model.

---

## Dynamic Simultaneous Equations Modeling

In the econometrics literature, the VARMAX( $p, q, s$ ) model could be written in the following slightly different form, which is referred to as a *dynamic simultaneous equations* model or a *dynamic structural equations* model.

Since  $E(\epsilon_t \epsilon_t') = \Sigma$  is assumed to be positive-definite, there exists a lower triangular matrix  $A_0$  with ones on the diagonals such that  $A_0 \Sigma A_0' = \Sigma^d$ , where  $\Sigma^d$  is a diagonal matrix with positive diagonal elements.

$$A_0 \mathbf{y}_t - \sum_{i=1}^p A_i \mathbf{y}_{t-i} = \sum_{i=0}^s C_i^* \mathbf{x}_{t-i} + C_0 \epsilon_t - \sum_{i=1}^q C_i \epsilon_{t-i}$$

where  $A_i = A_0 \Phi_i$ ,  $C_i^* = A_0 \Theta_i^*$ ,  $C_0 = A_0$ ,  $C_i = A_0 \Theta_i$ . and  $C_0 \epsilon_t$  has a diagonal covariance matrix  $\Sigma^d$ .

A dynamic simultaneous equations model involves a leading (lower triangular) coefficient matrix for  $\mathbf{y}_t$  at lag 0 or a leading coefficient matrix for  $\epsilon_t$  at lag 0. Such a representation of the VARMAX( $p, q, s$ ) model can be more useful in certain circumstances than the standard representation.

## State-Space Modeling

Another representation of the VARMAX( $p, q, s$ ) model is in the form of a state-variable or a state-space model, which consists of a state equation

$$\mathbf{z}_t = F\mathbf{z}_{t-1} + K\mathbf{x}_t + G\boldsymbol{\epsilon}_t$$

and an observation equation

$$\mathbf{y}_t = H\mathbf{z}_t$$

where

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{y}_t \\ \vdots \\ \mathbf{y}_{t-p+1} \\ \mathbf{x}_t \\ \vdots \\ \mathbf{x}_{t-s+1} \\ \boldsymbol{\epsilon}_t \\ \vdots \\ \boldsymbol{\epsilon}_{t-q+1} \end{bmatrix}, \quad K = \begin{bmatrix} \Theta_0^* \\ 0_{k \times r} \\ \vdots \\ 0_{k \times r} \\ I_r \\ 0_{r \times r} \\ \vdots \\ 0_{r \times r} \\ 0_{k \times r} \\ \vdots \\ 0_{k \times r} \end{bmatrix}, \quad G = \begin{bmatrix} I_k \\ 0_{k \times k} \\ \vdots \\ 0_{k \times k} \\ 0_{r \times k} \\ \vdots \\ 0_{r \times k} \\ I_{k \times k} \\ 0_{k \times k} \\ \vdots \\ 0_{k \times k} \end{bmatrix}$$

$$F = \begin{bmatrix} \Phi_1 & \cdots & \Phi_{p-1} & \Phi_p & \Theta_1^* & \cdots & \Theta_{s-1}^* & \Theta_s^* & -\Theta_1 & \cdots & -\Theta_{q-1} & -\Theta_q \\ I_k & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & I_k & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & I_r & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & I_r & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

and

$$H = [I_k, 0_{k \times k}, \dots, 0_{k \times k}, 0_{k \times r}, \dots, 0_{k \times r}, 0_{k \times k}, \dots, 0_{k \times k}].$$

On the other hand, it is assumed that  $\mathbf{x}_t$  follows a VARMA( $p, q$ ) model

$$\mathbf{x}_t = \sum_{i=1}^p A_i \mathbf{x}_{t-i} + \mathbf{a}_t - \sum_{i=1}^q C_i \mathbf{a}_{t-i}$$

or  $A(B)\mathbf{x}_t = C(B)\mathbf{a}_t$ , where  $A(B) = I_r - A_1B - \cdots - A_pB^p$  and  $C(B) = I_r - C_1B - \cdots - C_qB^q$  are matrix polynomials in  $B$ , and the  $A_i$  and  $C_i$  are  $r \times r$

matrices. Without loss of generality, the AR and MA orders can be taken to be the same as the VARMAX( $p, q, s$ ) model, and  $\mathbf{a}_t$  and  $\boldsymbol{\epsilon}_t$  are independent white noise processes.

Under suitable (such as stationarity) conditions,  $\mathbf{x}_t$  is represented by an infinite order moving-average process

$$\mathbf{x}_t = A(B)^{-1}C(B)\mathbf{a}_t = \Psi^x(B)\mathbf{a}_t = \sum_{j=0}^{\infty} \Psi_j^x \mathbf{a}_{t-j}$$

where  $\Psi^x(B) = A(B)^{-1}C(B) = \sum_{j=0}^{\infty} \Psi_j^x B^j$ .

The optimal (Minimum Mean Squared Error, MMSE)  $i$ -step-ahead forecast of  $\mathbf{x}_{t+i}$  is

$$\begin{aligned} \mathbf{x}_{t+i|t} &= \sum_{j=i}^{\infty} \Psi_j^x \mathbf{a}_{t+i-j} \\ \mathbf{x}_{t+i|t+1} &= \mathbf{x}_{t+i|t} + \Psi_{i-1}^x \mathbf{a}_{t+1}. \end{aligned}$$

For  $i > q$ ,

$$\mathbf{x}_{t+i|t} = \sum_{j=1}^p A_j \mathbf{x}_{t+i-j|t}.$$

The VARMAX( $p, q, s$ ) model has an absolutely convergent representation as

$$\begin{aligned} \mathbf{y}_t &= \Phi(B)^{-1}\Theta^*(B)\mathbf{x}_t + \Phi(B)^{-1}\Theta(B)\boldsymbol{\epsilon}_t \\ &= \Psi^*(B)\Psi^x(B)\mathbf{a}_t + \Phi(B)^{-1}\Theta(B)\boldsymbol{\epsilon}_t \\ &= V(B)\mathbf{a}_t + \Psi(B)\boldsymbol{\epsilon}_t \end{aligned}$$

or

$$\mathbf{y}_t = \sum_{j=0}^{\infty} V_j \mathbf{a}_{t-j} + \sum_{j=0}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t-j}$$

where  $\Psi(B) = \Phi(B)^{-1}\Theta(B) = \sum_{j=0}^{\infty} \Psi_j B^j$ ,  $\Psi^*(B) = \Phi(B)^{-1}\Theta^*(B)$ , and  $V(B) = \Psi^*(B)\Psi^x(B) = \sum_{j=0}^{\infty} V_j B^j$ .

The optimal (MMSE)  $i$ -step-ahead forecast of  $\mathbf{y}_{t+i}$  is

$$\begin{aligned} \mathbf{y}_{t+i|t} &= \sum_{j=i}^{\infty} V_j \mathbf{a}_{t+i-j} + \sum_{j=i}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+i-j} \\ \mathbf{y}_{t+i|t+1} &= \mathbf{y}_{t+i|t} + V_{i-1} \mathbf{a}_{t+1} + \Psi_{i-1} \boldsymbol{\epsilon}_{t+1} \end{aligned}$$

## Part 2. Procedure Reference

for  $i = 1, \dots, v$  with  $v = \max(p, q + 1)$ . For  $i > q$ ,

$$\begin{aligned}
 \mathbf{y}_{t+i|t} &= \sum_{j=1}^p \Phi_j \mathbf{y}_{t+i-j|t} + \sum_{j=0}^s \Theta_j^* \mathbf{x}_{t+i-j|t} \\
 &= \sum_{j=1}^p \Phi_j \mathbf{y}_{t+i-j|t} + \Theta_0^* \mathbf{x}_{t+i|t} + \sum_{j=1}^s \Theta_j^* \mathbf{x}_{t+i-j|t} \\
 &= \sum_{j=1}^p \Phi_j \mathbf{y}_{t+i-j|t} + \Theta_0^* \sum_{j=1}^p A_j \mathbf{x}_{t+i-j|t} + \sum_{j=1}^s \Theta_j^* \mathbf{x}_{t+i-j|t} \\
 &= \sum_{j=1}^p \Phi_j \mathbf{y}_{t+i-j|t} + \sum_{j=1}^u (\Theta_0^* A_j + \Theta_j^*) \mathbf{x}_{t+i-j|t}
 \end{aligned}$$

where  $u = \max(p, s)$ .

Define  $\Pi_j = \Theta_0^* A_j + \Theta_j^*$ . For  $i = v > q$  with  $v = \max(p, q + 1)$ , you obtain

$$\begin{aligned}
 \mathbf{y}_{t+v|t} &= \sum_{j=1}^p \Phi_j \mathbf{y}_{t+v-j|t} + \sum_{j=1}^u \Pi_j \mathbf{x}_{t+v-j|t} \quad \text{for } u \leq v \\
 \mathbf{y}_{t+v|t} &= \sum_{j=1}^p \Phi_j \mathbf{y}_{t+v-j|t} + \sum_{j=1}^r \Pi_j \mathbf{x}_{t+v-j|t} \quad \text{for } u > v
 \end{aligned}$$

From the preceding relations, a state equation is

$$\mathbf{z}_{t+1} = F \mathbf{z}_t + K \mathbf{x}_t^* + G \mathbf{e}_{t+1}$$

and an observation equation is

$$\mathbf{y}_t = H \mathbf{z}_t$$

where

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{y}_t \\ \mathbf{y}_{t+1|t} \\ \vdots \\ \mathbf{y}_{t+v-1|t} \\ \mathbf{x}_t \\ \mathbf{x}_{t+1|t} \\ \vdots \\ \mathbf{x}_{t+v-1|t} \end{bmatrix}, \quad \mathbf{x}_t^* = \begin{bmatrix} \mathbf{x}_{t+v-u} \\ \mathbf{x}_{t+v-u+1} \\ \vdots \\ \mathbf{x}_{t-1} \end{bmatrix}, \quad \mathbf{e}_{t+1} = \begin{bmatrix} \mathbf{a}_{t+1} \\ \boldsymbol{\epsilon}_{t+1} \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & I_k & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & I_k & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi_v & \Phi_{v-1} & \Phi_{v-2} & \cdots & \Phi_1 & \Pi_v & \Pi_{v-1} & \Pi_{v-2} & \cdots & \Pi_1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & I_r & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & I_r & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & A_v & A_{v-1} & A_{v-2} & \cdots & A_1 \end{bmatrix}$$

$$K = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Pi_u & \Pi_{u-1} & \cdots & \Pi_{v+1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \quad G = \begin{bmatrix} V_0 & I_k \\ V_1 & \Psi_1 \\ \vdots & \vdots \\ V_{v-1} & \Psi_{v-1} \\ I_r & 0_{r \times k} \\ \Psi_1^x & 0_{r \times k} \\ \vdots & \vdots \\ \Psi_{v-1}^x & 0_{r \times k} \end{bmatrix}$$

and

$$H = [I_k, 0_{k \times k}, \dots, 0_{k \times k}, 0_{k \times r}, \dots, 0_{k \times r}].$$

Note that the matrix  $K$  and the input vector  $\mathbf{x}_t^*$  are defined only when  $u > v$ .

---

## Forecasting

The optimal (MMSE)  $l$ -step-ahead forecast of  $\mathbf{y}_{t+l}$  is

$$\begin{aligned} \mathbf{y}_{t+l|t} &= \sum_{j=1}^p \Phi_j \mathbf{y}_{t+l-j|t} + \sum_{j=0}^s \Theta_j^* \mathbf{x}_{t+l-j|t} - \sum_{j=1}^q \Theta_j \boldsymbol{\epsilon}_{t+l-j}, \quad l \leq q \\ \mathbf{y}_{t+l|t} &= \sum_{j=1}^p \Phi_j \mathbf{y}_{t+l-j|t} + \sum_{j=0}^s \Theta_j^* \mathbf{x}_{t+l-j|t}, \quad l > q \end{aligned}$$

with  $\mathbf{y}_{t+l-j|t} = \mathbf{y}_{t+l-j}$  and  $\mathbf{x}_{t+l-j|t} = \mathbf{x}_{t+l-j}$  for  $l \leq j$ . For the forecasts  $\mathbf{x}_{t+l-j|t}$ , see the previous section.

### Impulse Response Function

The VARMAX( $p, q, s$ ) model has a convergent representation

$$\mathbf{y}_t = \Psi^*(B) \mathbf{x}_t + \Psi(B) \boldsymbol{\epsilon}_t$$

where  $\Psi^*(B) = \Phi(B)^{-1} \Theta^*(B) = \sum_{j=0}^{\infty} \Psi_j^* B^j$  and  $\Psi(B) = \Phi(B)^{-1} \Theta(B) = \sum_{j=0}^{\infty} \Psi_j B^j$ .

The elements of the matrices  $\Psi_j$  from the operator  $\Psi(B)$ , called the impulse response, can be interpreted as the impact that a shock in one variable has on another



variable. Let  $\psi_{j,in}$  be the *element* of the  $\Psi_j$ . The notation  $i$  is the index for the impulse variable, and  $n$  is the index for the response variable (impulse  $\rightarrow$  response). For instance,  $\psi_{j,11}$  is an impulse response to  $y_{1t} \rightarrow y_{1t}$ , and  $\psi_{j,12}$  is an impulse response to  $y_{1t} \rightarrow y_{2t}$ .

The accumulated impulse response function is the cumulative sum of the impulse response function,  $\Psi_l^a = \sum_{j=0}^l \Psi_j$ .

The MA representation with a standardized white noise innovation process offers a further possibility to interpret a VARMA( $p,q$ ) model. Since  $\Sigma$  is positive-definite, there is a lower triangular matrix  $P$  such that  $\Sigma = PP'$ . The alternate MA representation is written as

$$\mathbf{y}_t = \Psi^o(B)\mathbf{u}_t$$

where  $\Psi^o(B) = \sum_{j=0}^{\infty} \Psi_j^o B^j$ ,  $\Psi_j^o = \Psi_j P$ , and  $\mathbf{u}_t = P^{-1}\boldsymbol{\epsilon}_t$ .

The elements of the matrices  $\Psi_j^o$ , called the *orthogonal impulse response*, can be interpreted as the effects of the components of the standardized shock process  $\mathbf{u}_t$  on the process  $\mathbf{y}_t$  at the lag  $j$ .

The coefficient matrix  $\Psi_j^*$  from the transfer function operator  $\Psi^*(B)$  can be interpreted as the effects that changes in the exogenous variables  $\mathbf{x}_t$  have on the output variable  $\mathbf{y}_t$  at the lag  $j$ , and is called an impulse response matrix in the transfer function.

The accumulated impulse response in the transfer function is the cumulative sum of the impulse response in the transfer function,  $\Psi_l^{*a} = \sum_{j=0}^l \Psi_j^*$ .

The asymptotic distributions of the impulse functions can be seen in the “VAR Modeling” section on page 101.

The following statements provide the impulse response and the accumulated impulse response in the transfer function for a VARX(1,0) model. Parts of the VARMAX procedure output are shown in Figure 4.24 and Figure 4.25.

```
proc varmax data=grunfeld;
  model y1-y3 = x1 x2 / p=1 print=(impulsx=(all)) lagmax=15
                                printform=univariate;
run;
```

| The VARMAX Procedure                                          |      |          |            |
|---------------------------------------------------------------|------|----------|------------|
| Impulse Response Matrices in<br>Transfer Function by Variable |      |          |            |
| Variable                                                      | Lead | x1       | x2         |
| y1                                                            | 0    | 1.69281  | -0.00859   |
|                                                               | 1    | 0.35399  | 0.01727    |
|                                                               | 2    | 0.09090  | 0.00714    |
|                                                               | :    | :        | :          |
|                                                               | 14   | 0.03319  | 0.00024164 |
|                                                               | 15   | 0.03195  | 0.00023260 |
| y2                                                            | 0    | -6.09850 | 2.57980    |
|                                                               | 1    | -5.15484 | 0.45445    |
|                                                               | 2    | -3.04168 | 0.04391    |
|                                                               | :    | :        | :          |
|                                                               | 14   | -1.32641 | -0.00966   |
|                                                               | 15   | -1.27682 | -0.00930   |
| y3                                                            | 0    | -0.02317 | -0.01274   |
|                                                               | 1    | 1.57476  | -0.01435   |
|                                                               | 2    | 1.80231  | 0.00398    |
|                                                               | :    | :        | :          |
|                                                               | 14   | 1.16268  | 0.00846    |
|                                                               | 15   | 1.11921  | 0.00815    |

Figure 4.24. Impulse Response in Transfer Function (IMPULSX= option)

| The VARMAX Procedure                                                      |      |           |          |
|---------------------------------------------------------------------------|------|-----------|----------|
| Accumulated Impulse Response Matrices<br>in Transfer Function by Variable |      |           |          |
| Variable                                                                  | Lead | x1        | x2       |
| y1                                                                        | 0    | 1.69281   | -0.00859 |
|                                                                           | 1    | 2.04680   | 0.00868  |
|                                                                           | 2    | 2.13770   | 0.01582  |
|                                                                           | :    | :         | :        |
|                                                                           | 14   | 2.63183   | 0.02162  |
|                                                                           | 15   | 2.66378   | 0.02185  |
| y2                                                                        | 0    | -6.09850  | 2.57980  |
|                                                                           | 1    | -11.25334 | 3.03425  |
|                                                                           | 2    | -14.29502 | 3.07816  |
|                                                                           | :    | :         | :        |
|                                                                           | 14   | -34.35946 | 2.93139  |
|                                                                           | 15   | -35.63628 | 2.92210  |
| y3                                                                        | 0    | -0.02317  | -0.01274 |
|                                                                           | 1    | 1.55159   | -0.02709 |
|                                                                           | 2    | 3.35390   | -0.02311 |
|                                                                           | :    | :         | :        |
|                                                                           | 14   | 20.71402  | 0.10051  |
|                                                                           | 15   | 21.83323  | 0.10866  |

Figure 4.25. Accumulated Impulse Response in Transfer Function (IMPULSX= option)

The following statements provide the impulse response function, the accumulated impulse response function, and the orthogonalized impulse response function with their standard errors for a VAR(1) model. Parts of the VARMAX procedure output are shown in Figure 4.26 through Figure 4.28.

```
proc varmax data=simul1;
  model y1 y2 / p=1 noint lagmax=15 print=(impulse=(all))
  printform=univariate;
run;
```

| The VARMAX Procedure         |      |          |          |
|------------------------------|------|----------|----------|
| Impulse Response by Variable |      |          |          |
| Variable                     | Lead | y1       | y2       |
| y1                           | 1    | 1.15977  | -0.51058 |
|                              | STD  | 0.05508  | 0.05898  |
|                              | 2    | 1.06612  | -0.78872 |
|                              | STD  | 0.10450  | 0.10702  |
|                              | :    | :        | :        |
|                              | 14   | 0.08202  | 0.02870  |
|                              | STD  | 0.10579  | 0.09483  |
|                              | 15   | 0.11080  | -0.03083 |
|                              | STD  | 0.09277  | 0.08778  |
|                              |      |          |          |
| y2                           | 1    | 0.54634  | 0.38499  |
|                              | STD  | 0.05779  | 0.06188  |
|                              | 2    | 0.84396  | -0.13073 |
|                              | STD  | 0.08481  | 0.08556  |
|                              | :    | :        | :        |
|                              | 14   | -0.03071 | 0.12557  |
|                              | STD  | 0.10081  | 0.09391  |
|                              | 15   | 0.03299  | 0.06403  |
|                              | STD  | 0.09375  | 0.08487  |
|                              |      |          |          |

**Figure 4.26.** Impulse Response Function (IMPULSE= option)

Figure 4.26 is the part of output in a univariate format associated with the IMPULSE= option for the impulse response function. The keyword STD stands for the standard errors of the elements.

| The VARMAX Procedure                     |      |         |          |
|------------------------------------------|------|---------|----------|
| Accumulated Impulse Response by Variable |      |         |          |
| Variable                                 | Lead | y1      | y2       |
| y1                                       | 1    | 2.15977 | -0.51058 |
|                                          | STD  | 0.05508 | 0.05898  |
|                                          | 2    | 3.22589 | -1.29929 |
|                                          | STD  | 0.21684 | 0.22776  |
|                                          | :    | :       | :        |
|                                          | 14   | 3.11982 | -2.53992 |
|                                          | STD  | 1.90364 | 1.84193  |
|                                          | 15   | 3.23062 | -2.57074 |
|                                          | STD  | 1.57743 | 1.52719  |
|                                          | :    | :       | :        |
| y2                                       | 1    | 0.54634 | 1.38499  |
|                                          | STD  | 0.05779 | 0.06188  |
|                                          | 2    | 1.39030 | 1.25426  |
|                                          | STD  | 0.17614 | 0.18392  |
|                                          | :    | :       | :        |
|                                          | 14   | 2.71782 | -0.73442 |
|                                          | STD  | 2.57030 | 2.32369  |
|                                          | 15   | 2.75080 | -0.67040 |
|                                          | STD  | 2.08022 | 1.96462  |
|                                          | :    | :       | :        |

**Figure 4.27.** Accumulated Impulse Response Function (IMPULSE= option)

Figure 4.27 is the part of output in a univariate format associated with the IMPULSE= option for the accumulated impulse response function.

| The VARMAX Procedure                        |      |         |          |
|---------------------------------------------|------|---------|----------|
| Orthogonalized Impulse Response by Variable |      |         |          |
| Variable                                    | Lead | y1      | y2       |
| y1                                          | 0    | 1.13523 | 0        |
|                                             | STD  | 0.08068 | 0        |
|                                             | 1    | 1.13783 | -0.58120 |
|                                             | STD  | 0.10666 | 0.14110  |
|                                             | 2    | 0.93412 | -0.89782 |
|                                             | STD  | 0.13113 | 0.16776  |
|                                             | :    | :       | :        |
|                                             | 14   | 0.10316 | 0.03267  |
|                                             | STD  | 0.09791 | 0.10849  |
|                                             | 15   | 0.11499 | -0.03509 |
| y2                                          | 0    | 0.35016 | 1.13832  |
|                                             | STD  | 0.11676 | 0.08855  |
|                                             | 1    | 0.75503 | 0.43824  |
|                                             | STD  | 0.06949 | 0.10937  |
|                                             | 2    | 0.91231 | -0.14881 |
|                                             | STD  | 0.10553 | 0.13565  |
|                                             | :    | :       | :        |
|                                             | 14   | 0.00910 | 0.14294  |
|                                             | STD  | 0.09504 | 0.10739  |
|                                             | 15   | 0.05987 | 0.07288  |
|                                             | STD  | 0.08779 | 0.09695  |

**Figure 4.28.** Orthogonalized Impulse Response Function (IMPULSE= option)

Figure 4.28 is the part of output in a univariate format associated with the IMPULSE= option for the orthogonalized impulse response function.

### **Covariance Matrices of Prediction Errors without Exogenous (Independent) Variables**

Under the stationarity assumption, the optimal (MMSE)  $l$ -step-ahead forecast of  $\mathbf{y}_{t+l}$  has an infinite moving-average form,  $\mathbf{y}_{t+l|t} = \sum_{j=l}^{\infty} \Psi_j \epsilon_{t+l-j}$ . The prediction error of the optimal  $l$ -step-ahead forecast is  $\mathbf{e}_{t+l|t} = \mathbf{y}_{t+l} - \mathbf{y}_{t+l|t} = \sum_{j=0}^{l-1} \Psi_j \epsilon_{t+l-j}$ , with zero mean and covariance matrix

$$\Sigma(l) = \text{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} \Psi_j \Sigma \Psi_j' = \sum_{j=0}^{l-1} \Psi_j^o \Psi_j^{o'}$$

where  $\Psi_j^o = \Psi_j P$  with a lower triangular matrix  $P$  such that  $\Sigma = PP'$ . Under the assumption of normality of the  $\epsilon_t$ , the  $l$ -step-ahead prediction error  $\mathbf{e}_{t+l|t}$  is also normally distributed as multivariate  $N(0, \Sigma(l))$ . Hence, it follows that the diagonal elements  $\sigma_{ii}^2(l)$  of  $\Sigma(l)$  can be used, together with the point forecasts  $y_{i,t+l|t}$ , to construct  $l$ -step-ahead prediction interval forecasts of the future values of the component series,  $y_{i,t+l}$ .

The following statements use the COVPE option to compute the covariance matrices of the prediction errors for a VAR(1) model. The parts of the VARMAX procedure output are shown in Figure 4.29 and Figure 4.30.

```
proc varmax data=simul1;
    model y1 y2 / p=1 noint print=(covpe(15))
                printform=both;
run;
```

| The VARMAX Procedure                 |          |         |         |  |
|--------------------------------------|----------|---------|---------|--|
| Prediction Error Covariance Matrices |          |         |         |  |
| Lead                                 | Variable | y1      | y2      |  |
| 1                                    | y1       | 1.28875 | 0.39751 |  |
|                                      | y2       | 0.39751 | 1.41839 |  |
| 2                                    | y1       | 2.92119 | 1.00189 |  |
|                                      | y2       | 1.00189 | 2.18051 |  |
| 3                                    | y1       | 4.59984 | 1.98771 |  |
|                                      | y2       | 1.98771 | 3.03498 |  |
| :                                    | :        | :       | :       |  |
| 14                                   | y1       | 7.93640 | 4.89643 |  |
|                                      | y2       | 4.89643 | 6.84041 |  |
| 15                                   | y1       | 7.94811 | 4.90204 |  |
|                                      | y2       | 4.90204 | 6.86092 |  |

**Figure 4.29.** Covariances of Prediction Errors (COVPE option)

Figure 4.29 is the output in a matrix format associated with the COVPE option for the prediction error covariance matrices.

| The VARMAX Procedure                     |      |         |         |
|------------------------------------------|------|---------|---------|
| Prediction Error Covariances by Variable |      |         |         |
| Variable                                 | Lead | y1      | y2      |
| y1                                       | 1    | 1.28875 | 0.39751 |
|                                          | 2    | 2.92119 | 1.00189 |
|                                          | 3    | 4.59984 | 1.98771 |
|                                          | :    | :       | :       |
|                                          | 14   | 7.93640 | 4.89643 |
|                                          | 15   | 7.94811 | 4.90204 |
| y2                                       | 1    | 0.39751 | 1.41839 |
|                                          | 2    | 1.00189 | 2.18051 |
|                                          | 3    | 1.98771 | 3.03498 |
|                                          | :    | :       | :       |
|                                          | 14   | 4.89643 | 6.84041 |
|                                          | 15   | 4.90204 | 6.86092 |

**Figure 4.30.** Covariances of Prediction Errors Continued

Figure 4.30 is the output in a univariate format associated with the COVPE option for the prediction error covariances. This printing format more easily explains the forecast limit of each variable.

### ***Covariance Matrices of Prediction Errors in Presence of Exogenous (Independent) Variables***

Exogenous variables can be both stochastic and nonstochastic (deterministic) variables. Considering the forecasts in the VARMAX( $p, q, s$ ) model, there are two cases.

#### **When exogenous (independent) variables are stochastic (future values not specified)**

As defined in the “State-space Modeling” section,  $\mathbf{y}_{t+l|t}$  has the representation

$$\mathbf{y}_{t+l|t} = \sum_{j=l}^{\infty} V_j \mathbf{a}_{t+l-j} + \sum_{j=l}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

and hence

$$\mathbf{e}_{t+l|t} = \sum_{j=0}^{l-1} V_j \mathbf{a}_{t+l-j} + \sum_{j=0}^{l-1} \Psi_j \boldsymbol{\epsilon}_{t+l-j}.$$

Therefore, the covariance matrix of the  $l$ -step-ahead prediction error is given as

$$\Sigma(l) = \text{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} V_j \Sigma_a V_j' + \sum_{j=0}^{l-1} \Psi_j \Sigma_{\epsilon} \Psi_j'$$

where  $\Sigma_a$  is the covariance of the white noise series  $\mathbf{a}_t$ , where  $\mathbf{a}_t$  is the white noise series for the VARMA( $p, q$ ) model of exogenous (independent) variables, which is assumed not to be correlated with  $\boldsymbol{\epsilon}_t$  or its lags. See the “Forecasting” section on page 87 for details.

### When future exogenous (independent) variables are specified

The optimal forecast  $\mathbf{y}_{t+l|t}$  of  $\mathbf{y}_t$  conditioned on the past information and also on known future values  $\mathbf{x}_{t+1}, \dots, \mathbf{x}_{t+l}$  can be represented as

$$\mathbf{y}_{t+l|t} = \sum_{j=0}^{\infty} \Psi_j^* \mathbf{x}_{t+l-j} + \sum_{j=l}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

and the forecast error is

$$\mathbf{e}_{t+l|t} = \sum_{j=0}^{l-1} \Psi_j \boldsymbol{\epsilon}_{t+l-j}.$$

Thus, the covariance matrix of the  $l$ -step-ahead prediction error is given as

$$\Sigma(l) = \text{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} \Psi_j \Sigma_{\epsilon} \Psi_j'.$$

### Decomposition of Prediction Error Covariances

In the relation  $\Sigma(l) = \sum_{j=0}^{l-1} \Psi_j \Psi_j'$ , the diagonal elements can be interpreted as providing a decomposition of the  $l$ -step-ahead prediction error covariance  $\sigma_{ii}^2(l)$  for each component series  $y_{it}$  into contributions from the components of the standardized innovations  $\boldsymbol{\epsilon}_t$ .

If you denote the  $(i, n)$ th element of  $\Psi_j$  by  $\psi_{j,in}$ , the MSE of  $y_{i,t+h|t}$  is

$$\text{MSE}(y_{i,t+h|t}) = E(y_{i,t+h} - y_{i,t+h|t})^2 = \sum_{j=0}^{l-1} \sum_{n=1}^k \psi_{j,in}^2.$$

Note that  $\sum_{j=0}^{l-1} \psi_{j,in}^2$  is interpreted as the contribution of innovations in variable  $n$  to the prediction error covariance of the  $l$ -step-ahead forecast of variable  $i$ .

The proportion,  $\omega_{l,in}$ , of the  $l$ -step-ahead forecast error covariance of variable  $i$  accounting for the innovations in variable  $n$  is

$$\omega_{l,in} = \sum_{j=0}^{l-1} \psi_{j,in}^2 / \text{MSE}(y_{i,t+h|t}).$$

The following statements use the DECOMPOSE option to compute the decomposition of prediction error covariances and their proportions for a VAR(1) model:

```
proc varmax data=simul1;
    model y1 y2 / p=1 noint print=(decompose(15))
        printform=univariate;
run;
```

| The VARMAX Procedure                                       |      |         |         |
|------------------------------------------------------------|------|---------|---------|
| Proportions of Prediction Error<br>Covariances by Variable |      |         |         |
| Variable                                                   | Lead | y1      | y2      |
| y1                                                         | 1    | 1.00000 | 0       |
|                                                            | 2    | 0.88436 | 0.11564 |
|                                                            | 3    | 0.75132 | 0.24868 |
|                                                            | :    | :       | :       |
|                                                            | 14   | 0.55184 | 0.44816 |
|                                                            | 15   | 0.55237 | 0.44763 |
| y2                                                         | 1    | 0.08644 | 0.91356 |
|                                                            | 2    | 0.31767 | 0.68233 |
|                                                            | 3    | 0.50247 | 0.49753 |
|                                                            | :    | :       | :       |
|                                                            | 14   | 0.46611 | 0.53389 |
|                                                            | 15   | 0.46473 | 0.53527 |

**Figure 4.31.** Decomposition of Prediction Error Covariances (DECOMPOSE option)

The proportions of decomposition of prediction error covariances of two variables are given in Figure 4.31. The output explains that about 91% of the one-step-ahead prediction error covariances of the variable  $y_{2t}$  is accounted for by its own innovations and about 9% is accounted for by  $y_{1t}$  innovations. For the long-term forecasts, 53.5% and 46.5% of the error variance is accounted for by  $y_{2t}$  and  $y_{1t}$  innovations.

### Forecasting of the Centered Series

If the CENTER option is specified, the sample mean vector is added to the forecast.

### Forecasting of the Differenced Series

If endogenous (dependent) variables are differenced, the final forecasts and their prediction error covariances are produced by integrating those of the differenced series. However, if the PRIOR option is specified, the forecasts and their prediction error variances of the differenced series are produced.

Let  $\mathbf{z}_t$  be the original series with some zero values appended corresponding to the unobserved past observations. Let  $\Delta(B)$  be the  $k \times k$  matrix polynomial in the back-shift operator corresponding to the differencing specified by the MODEL statement. The off-diagonal elements of  $\Delta_i$  are zero and the diagonal elements can be different. Then  $\mathbf{y}_t = \Delta(B)\mathbf{z}_t$ .

This gives the relationship

$$\mathbf{z}_t = \Delta^{-1}(B)\mathbf{y}_t = \sum_{j=0}^{\infty} \Lambda_j \mathbf{y}_{t-j}$$

where  $\Delta^{-1}(B) = \sum_{j=0}^{\infty} \Lambda_j B^j$  and  $\Lambda_0 = I_k$ .

The  $l$ -step-ahead prediction of  $\mathbf{z}_{t+l}$  is

$$\mathbf{z}_{t+l|t} = \sum_{j=0}^{l-1} \Lambda_j \mathbf{y}_{t+l-j|t} + \sum_{j=l}^{\infty} \Lambda_j \mathbf{y}_{t+l-j}.$$



The  $l$ -step-ahead prediction error of  $\mathbf{z}_{t+l}$  is

$$\sum_{j=0}^{l-1} \Lambda_j (\mathbf{y}_{t+l-j} - \mathbf{y}_{t+l-j|t}) = \sum_{j=0}^{l-1} \left( \sum_{u=0}^j \Lambda_u \Psi_{j-u} \right) \boldsymbol{\epsilon}_{t+l-j}.$$

Letting  $\Sigma_{\mathbf{z}}(0) = 0$ , the covariance matrix of the  $l$ -step-ahead prediction error of  $\mathbf{z}_{t+l}$ ,  $\Sigma_{\mathbf{z}}(l)$ , is

$$\begin{aligned} \Sigma_{\mathbf{z}}(l) &= \sum_{j=0}^{l-1} \left( \sum_{u=0}^j \Lambda_u \Psi_{j-u} \right) \Sigma_{\epsilon} \left( \sum_{u=0}^j \Lambda_u \Psi_{j-u} \right)' \\ &= \Sigma_{\mathbf{z}}(l-1) + \left( \sum_{j=0}^{l-1} \Lambda_j \Psi_{l-1-j} \right) \Sigma_{\epsilon} \left( \sum_{j=0}^{l-1} \Lambda_j \Psi_{l-1-j} \right)'. \end{aligned}$$

If there are stochastic exogenous (independent) variables, the covariance matrix of the  $l$ -step-ahead prediction error of  $\mathbf{z}_{t+l}$ ,  $\Sigma_{\mathbf{z}}(l)$ , is

$$\begin{aligned} \Sigma_{\mathbf{z}}(l) &= \Sigma_{\mathbf{z}}(l-1) + \left( \sum_{j=0}^{l-1} \Lambda_j \Psi_{l-1-j} \right) \Sigma_{\epsilon} \left( \sum_{j=0}^{l-1} \Lambda_j \Psi_{l-1-j} \right)' \\ &\quad + \left( \sum_{j=0}^{l-1} \Lambda_j V_{l-1-j} \right) \Sigma_a \left( \sum_{j=0}^{l-1} \Lambda_j V_{l-1-j} \right)'. \end{aligned}$$

---

## Tentative Order Selection

### Sample Cross-Covariance and Cross-Correlation Matrices

Given a stationary multivariate time series  $\mathbf{y}_t$ , cross-covariance matrices are

$$\Gamma(l) = E[(\mathbf{y}_t - \boldsymbol{\mu}_t)(\mathbf{y}_t - \boldsymbol{\mu}_t)']$$

where  $\boldsymbol{\mu}_t = E(\mathbf{y}_t)$ , and cross-correlation matrices are

$$\rho(l) = D^{-1} \Gamma(l) D^{-1}$$

where  $D$  is a diagonal matrix with the standard deviations of the components of  $\mathbf{y}_t$  on the diagonal.

The sample cross-covariance matrix at lag  $l$ , denoted as  $C(l)$ , is computed as

$$\hat{\Gamma}(l) = C(l) = \frac{1}{T} \sum_{t=1}^{T-l} \tilde{\mathbf{y}}_t \tilde{\mathbf{y}}_{t+l}'$$

where  $\tilde{\mathbf{y}}_t$  is the centered data and  $T$  is the number of nonmissing observations. Thus,  $\hat{\Gamma}(l)$  has  $(i, j)$ th element  $\hat{\gamma}_{ij}(l) = c_{ij}(l)$ . The sample cross-correlation matrix at lag  $l$  is computed as

$$\hat{\rho}_{ij}(l) = c_{ij}(l) / [c_{ii}(0)c_{jj}(0)]^{1/2}, \quad i, j = 1, \dots, k.$$

The following statements use the CORRY option to compute the sample cross-correlation matrices and their summary indicator plots in terms of +, −, and · :

```
proc varmax data=simul1;
  model y1 y2 / p=1 noint lagmax=3 print=(corry)
              printform=univariate;
run;
```

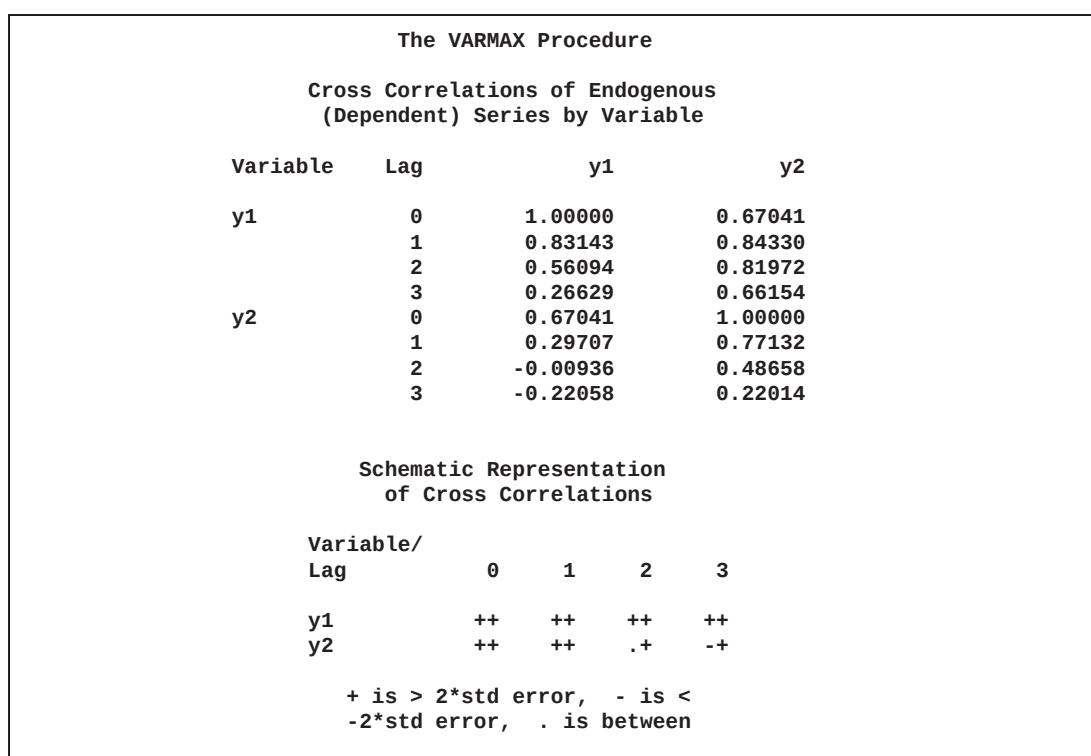


**Figure 4.32.** Cross-Correlations (CORRY option)

Figure 4.32 shows the sample cross-correlation matrices of  $y_{1t}$  and  $y_{2t}$ . As shown, the sample autocorrelation functions for each variable decay quickly, but are significant with respect to two standard errors.

### **Partial Autoregressive Matrices**

For each  $m = 1, 2, \dots$ , you can define a sequence of matrices  $\Phi_{mm}$ , which is called the partial autoregression matrices of lag  $m$ :

$$\Gamma(l) = \sum_{i=1}^m \Gamma(l-i) \Phi'_{im}, \quad l = 1, 2, \dots, m.$$

The sequence of the partial autoregression matrices  $\Phi_{mm}$  of order  $m$  has the characteristic property that if the process follows the  $AR(p)$ , then  $\Phi_{pp} = \Phi_p$  and  $\Phi_{mm} = 0$  for  $m > p$ . Hence, the matrices  $\Phi_{mm}$  have the cutoff property for a  $VAR(p)$  model, and so they can be useful in the identification of the order of a pure  $VAR$  model.

The following statements use the PARCOEF option to compute the partial autoregression matrices:

```
proc varmax data=simul1;
  model y1 y2 / p=1 noint lagmax=3 print=(parcoef);
run;
```

| The VARMAX Procedure            |          |          |          |
|---------------------------------|----------|----------|----------|
| Partial Autoregression Matrices |          |          |          |
| Lag                             | Variable | y1       | y2       |
| 1                               | y1       | 1.14844  | -0.50954 |
|                                 | y2       | 0.54985  | 0.37409  |
| 2                               | y1       | -0.00724 | 0.05138  |
|                                 | y2       | 0.02409  | 0.05909  |
| 3                               | y1       | -0.02578 | 0.03885  |
|                                 | y2       | -0.03720 | 0.10149  |

| Schematic Representation<br>of Partial Autoregression |    |    |    |
|-------------------------------------------------------|----|----|----|
| Variable/<br>Lag                                      | 1  | 2  | 3  |
| y1                                                    | +- | .. | .. |
| y2                                                    | ++ | .. | .. |

+ is > 2\*std error, - is <  
-2\*std error, . is between

**Figure 4.33.** Partial Autoregression Matrices (PARCOEF option)

Figure 4.33 shows that the model can be obtained by an  $AR$  order  $m = 1$  since partial autoregression matrices are insignificant after lag 1 with respect to two standard errors. The matrix for lag 1 is the same as the Yule-Walker autoregressive matrix.

### Partial Correlation Matrices

Define the forward autoregression

$$\mathbf{y}_t = \sum_{i=1}^{m-1} \Phi_{i,m-1} \mathbf{y}_{t-i} + \mathbf{u}_{m,t}$$

and the backward autoregression

$$\mathbf{y}_{t-m} = \sum_{i=1}^{m-1} \Phi_{i,m-1}^* \mathbf{y}_{t-m+i} + \mathbf{u}_{m,t-m}^*$$

The matrices  $P(m)$  defined by Ansley and Newbold (1979) are given by

$$P(m) = \Sigma_{m-1}^{*1/2} \Phi'_{mm} \Sigma_{m-1}^{-1/2}$$

where

$$\Sigma_{m-1} = \text{Cov}(\mathbf{u}_{m,t}) = \Gamma(0) - \sum_{i=1}^{m-1} \Gamma(-i) \Phi'_{i,m-1}$$

and

$$\Sigma_{m-1}^* = \text{Cov}(\mathbf{u}_{m,t-m}^*) = \Gamma(0) - \sum_{i=1}^{m-1} \Gamma(m-i) \Phi_{m-i,m-1}^{*'}.$$

$P(m)$  is called the partial cross-correlations matrices at lag  $m$  between the elements of  $\mathbf{y}_t$  and  $\mathbf{y}_{t-m}$ , given  $\mathbf{y}_{t-1}, \dots, \mathbf{y}_{t-m+1}$ . The matrices  $P(m)$  have the cutoff property for a VAR( $p$ ) model, and so they can be useful in the identification of the order of a pure VAR structure.

The following statements use the PCORR option to compute the partial cross-correlations matrices:

```
proc varmax data=simul1;
  model y1 y2 / p=1 noint lagmax=3 print=(pcorr)
               printform=univariate;
run;
```

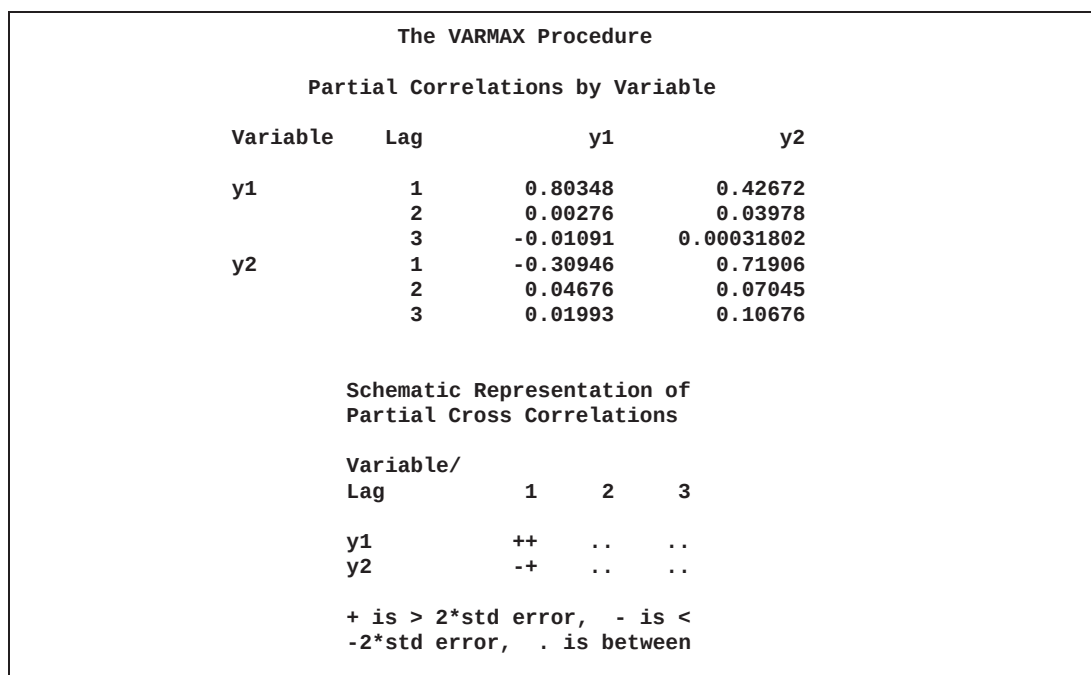


Figure 4.34. Partial Correlations (PCORR option)

The partial cross-correlation matrices in Figure 4.34 are insignificant after lag 1 with respect to two standard errors. This indicates that an AR order of  $m = 1$  can be an appropriate choice.

### Partial Canonical Correlation Matrices

The partial canonical correlations at lag  $m$  between the vectors  $\mathbf{y}_t$  and  $\mathbf{y}_{t-m}$ , given  $\mathbf{y}_{t-1}, \dots, \mathbf{y}_{t-m+1}$ , are  $1 \geq \rho_1(m) \geq \rho_2(m) \cdots \geq \rho_k(m)$ . The partial canonical correlations are the canonical correlations between the residual series  $\mathbf{u}_{m,t}$  and  $\mathbf{u}_{m,t-m}^*$ , where  $\mathbf{u}_{m,t}$  and  $\mathbf{u}_{m,t-m}^*$  are defined in the previous section. Thus, the squared partial canonical correlations  $\rho_i^2(m)$  are the eigenvalues of the matrix

$$\{\text{Cov}(\mathbf{u}_{m,t})\}^{-1} \text{E}(\mathbf{u}_{m,t} \mathbf{u}_{m,t-m}^{*'}) \{\text{Cov}(\mathbf{u}_{m,t-m}^*)\}^{-1} \text{E}(\mathbf{u}_{m,t-m}^* \mathbf{u}_{m,t}') = \Phi_{mm}^{*'} \Phi_{mm}'$$

It follows that the test statistic to test for  $\Phi_m = 0$  in the VAR model of order  $m > p$  is approximately

$$(T - m) \text{tr} \{\Phi_{mm}^{*'} \Phi_{mm}'\} \approx (T - m) \sum_{i=1}^k \rho_i^2(m)$$

and has an asymptotic chi-square distribution with  $k^2$  degrees of freedom for  $m > p$ .

The following statements use the PCANCORR option to compute the partial canonical correlations:

```
proc varmax data=simul1;
    model y1 y2 / p=1 noint lagmax=3 print=(pcancorr);
run;
```

| The VARMAX Procedure           |           |           |            |    |            |
|--------------------------------|-----------|-----------|------------|----|------------|
| Partial Canonical Correlations |           |           |            |    |            |
| Lag                            | PCanCorr1 | PCanCorr2 | Chi-Square | DF | Prob>ChiSq |
| 1                              | 0.91783   | 0.77335   | 142.61     | 4  | <.0001     |
| 2                              | 0.09171   | 0.01816   | 0.86       | 4  | 0.9307     |
| 3                              | 0.10861   | 0.01078   | 1.16       | 4  | 0.8854     |

**Figure 4.35.** Partial Canonical Correlations (PCANCORR option)

Figure 4.35 shows that the partial canonical correlations  $\rho_i(m)$  between  $\mathbf{y}_t$  and  $\mathbf{y}_{t-m}$  are  $\{0.918, 0.773\}$ ,  $\{0.092, 0.018\}$ , and  $\{0.109, 0.011\}$  for lags  $m = 1$  to 3. After lag  $m = 1$ , the partial canonical correlations are insignificant with respect to a 0.05 significance level, indicating that an AR order of  $m = 1$  can be an appropriate choice.

### The Minimum Information Criterion (MINIC) method

The MINimum Information Criterion (MINIC) method can tentatively identify the orders of a VARMA( $p, q$ ) process. Note that Spliid (1983), Koreisha and Pukkila (1989), and Quinn (1980) proposed this method. The first step of this method is to obtain estimates of the innovations series,  $\epsilon_t$ , from the VAR( $p_\epsilon$ ), where  $p_\epsilon$  is chosen

sufficiently large. The choice of the autoregressive order,  $p_\epsilon$ , is determined by use of a selection criterion. From the selected VAR( $p_\epsilon$ ) model, you obtain estimates of residual series

$$\tilde{\epsilon}_t = \mathbf{y}_t - \sum_{i=1}^{p_\epsilon} \hat{\Phi}_i^{p_\epsilon} \mathbf{y}_{t-i} - \hat{\delta}^{p_\epsilon}, \quad t = p_\epsilon + 1, \dots, T.$$

In the second step, you select the order  $(p, q)$  of the VARMA model for  $p$  in  $(p_{min} : p_{max})$  and  $q$  in  $(q_{min} : q_{max})$

$$\mathbf{y}_t = \delta + \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \epsilon_t - \sum_{i=1}^q \Theta_i \tilde{\epsilon}_{t-i}$$

which minimizes a selection criterion like SBC or HQ.

The following statements use the MINIC= option to compute a table containing the information criterion associated with various AR and MA orders:

```
proc varmax data=simul1;
  model y1 y2 / p=1 noint minic=(p=3 q=3);
run;
```

| The VARMAX Procedure          |           |           |           |           |
|-------------------------------|-----------|-----------|-----------|-----------|
| Minimum Information Criterion |           |           |           |           |
| Lag                           | MA 0      | MA 1      | MA 2      | MA 3      |
| AR 0                          | 3.2859653 | 3.0570091 | 2.7272377 | 2.3526368 |
| AR 1                          | 0.4862246 | 0.6507991 | 0.7120257 | 0.7859524 |
| AR 2                          | 0.5800236 | 0.7407785 | 0.802996  | 0.850487  |
| AR 3                          | 0.6696452 | 0.8131261 | 0.8395558 | 0.9094856 |

**Figure 4.36.** MINIC= option

Figure 4.36 shows the output associated with the MINIC= option. The criterion takes the smallest value at AR order 1.

## VAR Modeling

The  $p$ th-order VAR process is written as

$$\mathbf{y}_t - \boldsymbol{\mu} = \sum_{i=1}^p \Phi_i (\mathbf{y}_{t-i} - \boldsymbol{\mu}) + \epsilon_t \quad \text{or} \quad \Phi(B)(\mathbf{y}_t - \boldsymbol{\mu}) = \epsilon_t$$

Equivalently, it can be written as

$$\mathbf{y}_t = \delta + \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \epsilon_t \quad \text{or} \quad \Phi(B)\mathbf{y}_t = \delta + \epsilon_t$$

with  $\delta = (I_k - \sum_{i=1}^p \Phi_i)\mu$ .

### Stationarity

For stationarity of the VAR process, it must be expressible in the convergent causal infinite MA form as

$$\mathbf{y}_t = \mu + \sum_{j=0}^{\infty} \Psi_j \epsilon_{t-j}$$

where  $\Psi(B) = \Phi(B)^{-1} = \sum_{j=0}^{\infty} \Psi_j B^j$  with  $\sum_{j=0}^{\infty} \|\Psi_j\| < \infty$ , where  $\|A\|$  denotes a norm for the matrix  $A$  such as  $\|A\|^2 = \text{tr}\{A'A\}$ . The matrix  $\Psi_j$  can be recursively obtained from the relation  $\Phi(B)\Psi(B) = I$ , and is

$$\Psi_j = \Phi_1 \Psi_{j-1} + \Phi_2 \Psi_{j-2} + \cdots + \Phi_p \Psi_{j-p}$$

where  $\Psi_0 = I_k$  and  $\Psi_j = 0$  for  $j < 0$ .

The stationarity condition is satisfied if all roots of  $|\Phi(z)| = 0$  are outside of the unit circle. The stationarity condition is equivalent to the condition in the corresponding VAR(1) representation,  $\mathbf{Y}_t = \Phi \mathbf{Y}_{t-1} + \epsilon_t$ , that all eigenvalues of the  $kp \times kp$  companion matrix  $\Phi$  be less than one in absolute value, where  $\mathbf{Y}_t = (\mathbf{y}'_t, \dots, \mathbf{y}'_{t-p+1})'$ ,  $\epsilon_t = (\epsilon'_t, 0', \dots, 0')'$ , and

$$\Phi = \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_{p-1} & \Phi_p \\ I_k & 0 & \cdots & 0 & 0 \\ 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

If the stationarity condition is not satisfied, a nonstationary model (a differenced model or an error correction model) may be more appropriate.

The following statements estimate a VAR(1) model and use the ROOTS option to compute the characteristic polynomial roots:

```
proc varmax data=simul1;
    model y1 y2 / p=1 noint print=(roots);
run;
```

| The VARMAX Procedure                  |         |           |         |           |          |
|---------------------------------------|---------|-----------|---------|-----------|----------|
| Roots of AR Characteristic Polynomial |         |           |         |           |          |
| Index                                 | Real    | Imaginary | Modulus | ATAN(I/R) | Degree   |
| 1                                     | 0.77238 | 0.35899   | 0.8517  | 0.4351    | 24.9284  |
| 2                                     | 0.77238 | -0.35899  | 0.8517  | -0.4351   | -24.9284 |

Figure 4.37. Stationarity (ROOTS option)

Figure 4.37 shows the output associated with the ROOTS option, which indicates that the series is stationary since the modulus of the eigenvalue is less than one.

### Parameter Estimation

Consider the stationary VAR( $p$ ) model

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where  $\mathbf{y}_{-p+1}, \dots, \mathbf{y}_0$  are assumed to be available (for convenience of notation). This can be represented by the general form of the multivariate linear model,

$$Y = XB + E \quad \text{or} \quad \mathbf{y} = (X \otimes I_k)\boldsymbol{\beta} + \mathbf{e}$$

where

$$\begin{aligned} Y &= (\mathbf{y}_1, \dots, \mathbf{y}_T)' \\ B &= (\boldsymbol{\delta}, \Phi_1, \dots, \Phi_p)' \\ X &= (X_0, \dots, X_{T-1})' \\ X_t &= (1, \mathbf{y}_t', \dots, \mathbf{y}_{t-p+1}')' \\ E &= (\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_T)' \\ \mathbf{y} &= \text{vec}(Y') \\ \boldsymbol{\beta} &= \text{vec}(B') \\ \mathbf{e} &= \text{vec}(E') \end{aligned}$$

with  $\text{vec}$  denoting the column stacking operator.

The conditional least-squares estimator of  $\boldsymbol{\beta}$  is

$$\hat{\boldsymbol{\beta}} = ((X'X)^{-1}X' \otimes I_k)\mathbf{y}$$

and the estimate of  $\Sigma$  is

$$\hat{\Sigma} = (T - (kp + 1))^{-1} \sum_{t=1}^T \hat{\boldsymbol{\epsilon}}_t \hat{\boldsymbol{\epsilon}}_t'$$

where  $\hat{\boldsymbol{\epsilon}}_t$  is the residual vectors. Consistency and asymptotic normality of the LS estimator are that

$$\sqrt{T}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow{d} N(0, \Gamma_p^{-1} \otimes \Sigma)$$

where  $X'X/T$  converges in probability to  $\Gamma_p$  and  $\xrightarrow{d}$  denotes convergence in distribution.



The (conditional) maximum likelihood estimator in the VAR( $p$ ) model is equal to the (conditional) least-squares estimator on the assumption of normality of the error vectors.

### Asymptotic Distributions of Impulse Response Functions

As before,  $\text{vec}$  denotes the column stacking operator and  $\text{vech}$  is the corresponding operator that stacks the elements on and below the diagonal. The commutation matrix  $K_k$  defines as  $K_k \text{vec}(A) = \text{vec}(A')$ ; the duplication matrix  $D_k$ ,  $D_k \text{vech}(A) = \text{vec}(A)$ ; the elimination matrix  $L_k$ ,  $L_k \text{vec}(A) = \text{vech}(A)$ , for any  $k \times k$  matrix  $A$ .

The asymptotic distributions of the impulse response function is

$$\sqrt{T} \text{vec}(\hat{\Psi}_j - \Psi_j) \xrightarrow{d} N(0, G_j \Sigma_{\beta} G_j') \quad j = 1, 2, \dots$$

where  $\Sigma_{\beta} = \Gamma_p^{-1} \otimes \Sigma$  and

$$G_j = \frac{\partial \text{vec}(\Psi_j)}{\partial \beta'} = \sum_{i=0}^{j-1} \mathbf{J}(\Phi')^{j-1-i} \otimes \Psi_i$$

where  $\mathbf{J} = [I_k, 0, \dots, 0]$  is a  $k \times kp$  matrix and  $\Phi$  is a  $kp \times kp$  companion matrix.

The asymptotic distributions of the accumulated impulse response function is

$$\sqrt{T} \text{vec}(\hat{\Psi}_l^a - \Psi_l^a) \xrightarrow{d} N(0, F_l \Sigma_{\beta} F_l') \quad l = 1, 2, \dots$$

where  $F_l = \sum_{j=1}^l G_j$ .

The asymptotic distributions of the orthogonalized impulse response function is

$$\sqrt{T} \text{vec}(\hat{\Psi}_j^o - \Psi_j^o) \xrightarrow{d} N(0, C_j \Sigma_{\beta} C_j' + \bar{C}_j \Sigma_{\sigma} \bar{C}_j') \quad j = 0, 1, 2, \dots$$

where  $C_0 = 0$ ,  $C_j = (\Psi_0^{o'} \otimes I_k) G_j$ ,  $\bar{C}_j = (I_k \otimes \Psi_j) H$  and

$$H = \frac{\partial \text{vec}(\Psi_0^o)}{\partial \sigma'} = L_k' \{ L_k (I_{k^2} + K_k) (\Psi_0^o \otimes I_k) L_k' \}^{-1}$$

and  $\Sigma_{\sigma} = 2D_k^+ (\Sigma \otimes \Sigma) D_k^{+'}$  with  $D_k^+ = (D_k' D_k)^{-1} D_k'$  and  $\sigma = \text{vech}(\Sigma)$ .

### Granger-Causality Test

Let  $\mathbf{y}_t$  be arranged and partitioned in subgroups  $\mathbf{y}_{1t}$  and  $\mathbf{y}_{2t}$  with dimensions  $k_1$  and  $k_2$ , respectively ( $k = k_1 + k_2$ ); that is,  $\mathbf{y}_t = (\mathbf{y}_{1t}', \mathbf{y}_{2t}')'$  with the corresponding white noise process  $\epsilon_t = (\epsilon_{1t}', \epsilon_{2t}')'$ . Consider the VAR( $p$ ) model with partitioned coefficients  $\Phi_{ij}(B)$  for  $i, j = 1, 2$  as follows:

$$\Phi(B) \mathbf{y}_t = \begin{bmatrix} \Phi_{11}(B) & \Phi_{12}(B) \\ \Phi_{21}(B) & \Phi_{22}(B) \end{bmatrix} \begin{bmatrix} \mathbf{y}_{1t} \\ \mathbf{y}_{2t} \end{bmatrix} = \boldsymbol{\delta} + \begin{bmatrix} \epsilon_{1t} \\ \epsilon_{2t} \end{bmatrix}.$$

The variables  $\mathbf{y}_{1t}$  are said to cause  $\mathbf{y}_{2t}$ , but  $\mathbf{y}_{2t}$  do not cause  $\mathbf{y}_{1t}$  if  $\Phi_{12}(B) = 0$ . The implication of this model structure is that future values of the process  $\mathbf{y}_{1t}$  are influenced only by its own past and not by the past of  $\mathbf{y}_{2t}$ , where future values of  $\mathbf{y}_{2t}$  are influenced by the past of both  $\mathbf{y}_{1t}$  and  $\mathbf{y}_{2t}$ . If the future  $\mathbf{y}_{1t}$  are not influenced by the past values of  $\mathbf{y}_{2t}$ , then it can be better to model  $\mathbf{y}_{1t}$  separately from  $\mathbf{y}_{2t}$ .

Consider testing  $H_0 : C\beta = c$ , where  $C$  is a  $s \times (k^2p + k)$  matrix of rank  $s$  and  $c$  is a  $s$ -dimensional vector where  $s = k_1k_2p$ . Assuming that

$$\sqrt{T}(\hat{\beta} - \beta) \xrightarrow{d} N(0, \Gamma_p^{-1} \otimes \Sigma)$$

you get the Wald statistic

$$T(C\hat{\beta} - c)'[C(\Gamma_p^{-1} \otimes \Sigma)C']^{-1}(C\hat{\beta} - c) \xrightarrow{d} \chi^2(s).$$

For the Granger-Causality Test, the matrix  $C$  consists of zeros or ones and  $c$  is the zero vector.

### VARX Modeling

The Vector AutoRegressive model with eXogenous variables is called the VARX( $p, s$ ) model. The form of the VARX( $p, s$ ) model can be written as

$$\mathbf{y}_t = \delta + \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^s \Theta_i^* \mathbf{x}_{t-i} + \epsilon_t.$$

The parameter estimates can be obtained by representing the general form of the multivariate linear model,

$$Y = XB + E \text{ or } \mathbf{y} = (X \otimes I_k)\beta + \mathbf{e}$$

where

$$\begin{aligned} Y &= (\mathbf{y}_1, \dots, \mathbf{y}_T)' \\ B &= (\delta, \Phi_1, \dots, \Phi_p, \Theta_0^*, \dots, \Theta_s^*)' \\ X &= (X_0, \dots, X_{T-1})' \\ X_t &= (1, \mathbf{y}_t', \dots, \mathbf{y}_{t-p+1}', \mathbf{x}_{t+1}', \dots, \mathbf{x}_{t-s+1}')' \\ E &= (\epsilon_1, \dots, \epsilon_T)' \\ \mathbf{y} &= \text{vec}(Y') \\ \beta &= \text{vec}(B') \\ \mathbf{e} &= \text{vec}(E'). \end{aligned}$$

The conditional least-squares estimator of  $\beta$  can be obtained using the same method in a VAR( $p$ ) modeling. If the multivariate linear model has different independent variables corresponding to dependent variables, the SUR (Seemingly Unrelated Regression) method is used to improve the regression estimates.

The following example fits the ordinary regression model:

```
proc varmax data=one;
  model y1-y3 = x1-x5;
run;
```

This is equivalent to the REG procedure in the SAS/STAT<sup>®</sup> software.

```
proc reg data=one;
  model y1 = x1-x5;
  model y2 = x1-x5;
  model y3 = x1-x5;
run;
```

The following example fits the second-order lagged regression model:

```
proc varmax data=two;
  model y1 y2 = x / xlag=2;
run;
```

This is equivalent to the REG procedure in the SAS/STAT software.

```
data three;
  set two;
  xlag1 = lag1(x);
  xlag2 = lag2(x);
run;

proc reg data=three;
  model y1 = x xlag1 xlag2;
  model y2 = x xlag1 xlag2;
run;
```

The following example fits the ordinary regression model with different regressors:

```
proc varmax data=one;
  model y1 = x1-x3, y2 = x2 x3;
run;
```

This is equivalent to the following SYSLIN procedure statements.

```
proc syslin data=one vardef=nf sur;
  endogenous y1 y2;
  model y1 = x1-x3;
  model y2 = x2 x3;
run;
```

From the output in Figure 4.19, you can see that the parameters, XL0\_1\_2, XL0\_2\_1, XL0\_3\_1, and XL0\_3\_2, are not significant. The following example fits the VARX(1,0) model with different regressors:

```
proc varmax data=grunfeld;
  model y1 = x1, y2 = x2, y3 / p=1;
run;
```

| The VARMAX Procedure       |          |         |         |
|----------------------------|----------|---------|---------|
| XLag Coefficient Estimates |          |         |         |
| Lag                        | Variable | x1      | x2      |
| 0                          | y1       | 1.83231 | —       |
|                            | y2       | —       | 2.42110 |
|                            | y3       | —       | —       |

**Figure 4.38.** Parameter Estimates for VARX(1,0) Model

As you can see in Figure 4.38, the symbol ‘—’ in the elements of matrix correspond to endogenous variables that do not take exogenous variables.

## Model Diagnostic Checks

### Multivariate Model Diagnostic Checks

- Information Criterion

Various model selection criteria (normalized by  $T$ ) can be used to choose the appropriate model. The following list includes the Akaike Information Criterion (AIC), the corrected Akaike Information Criterion (AICC), the Final Prediction Error criterion (FPE), the Hannan-Quinn Criterion (HQC), and the Schwarz Bayesian Criterion (SBC), also referred to as BIC.

$$\begin{aligned}
 \text{AIC} &= \log(|\tilde{\Sigma}|) + 2r/T \\
 \text{AICC} &= \log(|\tilde{\Sigma}|) + 2r/(T - r/k) \\
 \text{FPE} &= \left(\frac{T + r/k}{T - r/k}\right)^k |\tilde{\Sigma}| \\
 \text{HQC} &= \log(|\tilde{\Sigma}|) + 2r \log(\log(T))/T \\
 \text{SBC} &= \log(|\tilde{\Sigma}|) + r \log(T)/T
 \end{aligned}$$

where  $r$  denotes the number of parameters estimated and  $\tilde{\Sigma}$  is the maximum likelihood estimate of  $\Sigma$ .

An example of the output was displayed in Figure 4.4.

- Portmanteau Statistic  $Q_s$

Let  $C_\epsilon(l)$  be the residual cross-covariance matrices and  $\hat{\rho}_\epsilon(l)$  be the residual cross-correlation matrices as

$$C_\epsilon(l) = T^{-1} \sum_{t=1}^{T-l} \epsilon_t \epsilon_{t+l}'$$

and

$$\hat{\rho}_\epsilon(l) = \hat{V}_\epsilon^{-1/2} C_\epsilon(l) \hat{V}_\epsilon^{-1/2} \quad \text{and} \quad \hat{\rho}_\epsilon(-l) = \hat{\rho}_\epsilon(l)'$$

where  $\hat{V}_\epsilon = \text{Diag}(\hat{\sigma}_{11}^2, \dots, \hat{\sigma}_{kk}^2)$  and  $\hat{\sigma}_{ii}^2$  are the diagonal elements of  $\hat{\Sigma}$ . The multivariate portmanteau test defined in Hosking (1980) is

$$Q_s = T^2 \sum_{l=1}^s (T-l)^{-1} \text{tr}\{\hat{\rho}_\epsilon(l) \Sigma^{-1} \hat{\rho}_\epsilon(-l) \Sigma^{-1}\}$$

The statistic  $Q_s$  has approximately the chi-square distribution with  $k^2(s-p)$  degrees of freedom. An example of the output was displayed in Figure 4.6.

### Univariate Model Diagnostic Checks

There are various ways to perform diagnostic checks for a univariate model. For details, see the chapter on the ARIMA or AUTOREG procedure. An example of the output was displayed in Figure 4.7 and Figure 4.8.

- Durbin-Watson (DW) statistics: The test statistics are computed from the residuals of the autoregressive model with order 1.
- $F$  tests for autoregressive conditional heteroscedastic (ARCH) disturbances: These test statistics are computed from the residuals of the ARCH(1) model.
- $F$  tests for AR disturbance: These test statistics are computed from the residuals of the univariate AR(1), AR(2), AR(3), and AR(4) models.
- Jarque-Bera normality test: This test is helpful in determining whether the model residuals represent a white noise process.

---

## Bayesian VAR Modeling

Consider the VAR( $p$ ) model

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \dots + \Phi_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t$$

or

$$\mathbf{y} = (X \otimes I_k) \boldsymbol{\beta} + \mathbf{e}.$$

When the parameter vector  $\boldsymbol{\beta}$  has a prior multivariate normal distribution with known mean  $\boldsymbol{\beta}^*$  and covariance matrix  $V_\beta$ , the prior density is written as

$$f(\boldsymbol{\beta}) = \left(\frac{1}{2\pi}\right)^{k^2 p/2} |V_\beta|^{-1/2} \exp\left[-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*) V_\beta^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}^*)\right].$$

The likelihood function for the Gaussian process becomes

$$\begin{aligned}\ell(\boldsymbol{\beta}|\mathbf{y}) &= \left(\frac{1}{2\pi}\right)^{kT/2} |I_T \otimes \Sigma|^{-1/2} \times \\ &\quad \exp\left[-\frac{1}{2}(\mathbf{y} - (X \otimes I_k)\boldsymbol{\beta})'(I_T \otimes \Sigma^{-1})(\mathbf{y} - (X \otimes I_k)\boldsymbol{\beta})\right].\end{aligned}$$

Therefore, the posterior density is derived as

$$f(\boldsymbol{\beta}|\mathbf{y}) \propto \exp\left[-\frac{1}{2}(\boldsymbol{\beta} - \bar{\boldsymbol{\beta}})' \bar{\Sigma}_{\boldsymbol{\beta}}^{-1}(\boldsymbol{\beta} - \bar{\boldsymbol{\beta}})\right]$$

where the posterior mean is

$$\bar{\boldsymbol{\beta}} = [V_{\boldsymbol{\beta}}^{-1} + (X'X \otimes \Sigma^{-1})]^{-1}[V_{\boldsymbol{\beta}}^{-1}\boldsymbol{\beta}^* + (X' \otimes \Sigma^{-1})\mathbf{y}]$$

and the posterior covariance matrix is

$$\bar{\Sigma}_{\boldsymbol{\beta}} = [V_{\boldsymbol{\beta}}^{-1} + (X'X \otimes \Sigma^{-1})]^{-1}.$$

In practice, the prior mean  $\boldsymbol{\beta}^*$  and the prior variance  $V_{\boldsymbol{\beta}}$  need to be specified. If all the parameters are considered to shrink toward zero, the null prior mean should be specified. According to Litterman (1986), the prior variance can be given by

$$v_{ij}(l) = \begin{cases} (\lambda/l)^2 & \text{if } i = j \\ (\lambda\theta\sigma_{ii}/l\sigma_{jj})^2 & \text{if } i \neq j \end{cases}$$

where  $v_{ij}(l)$  is the prior variance of the  $(i, j)$ th element of  $\Phi_l$ ,  $\lambda$  is the prior standard deviation of the diagonal elements of  $\Phi_l$ ,  $\theta$  is a constant in the interval  $(0, 1)$ , and  $\sigma_{ii}^2$  is the  $i$ th diagonal element of  $\Sigma$ . The deterministic terms have diffused prior variance. In practice, you replace the  $\sigma_{ii}^2$  by the diagonal element of the ML estimator of  $\Sigma$  in the nonconstrained model.

For example, for a bivariate BVAR(2) model,

$$\begin{aligned}y_{1t} &= 0 + \phi_{1,11}y_{1,t-1} + \phi_{1,12}y_{2,t-1} + \phi_{2,11}y_{1,t-2} + \phi_{2,12}y_{2,t-2} + \epsilon_{1t} \\ y_{2t} &= 0 + \phi_{1,21}y_{1,t-1} + \phi_{1,22}y_{2,t-1} + \phi_{2,21}y_{1,t-2} + \phi_{2,22}y_{2,t-2} + \epsilon_{2t}\end{aligned}$$

with the prior covariance matrix

$$V_{\boldsymbol{\beta}} = \text{Diag} \begin{pmatrix} \infty, \lambda^2, (\lambda\theta\sigma_1/\sigma_2)^2, (\lambda/2)^2, (\lambda\theta\sigma_1/2\sigma_2)^2, \\ \infty, (\lambda\theta\sigma_2/\sigma_1)^2, \lambda^2, (\lambda\theta\sigma_2/2\sigma_1)^2, (\lambda/2)^2 \end{pmatrix}.$$

For the Bayesian Estimation of integrated systems, the prior mean is set to the first lag of each variable equal to one in its own equation and all other coefficients at zero. For example, for a bivariate BVAR(2) model,

$$\begin{aligned}y_{1t} &= 0 + 1 y_{1,t-1} + 0 y_{2,t-1} + 0 y_{1,t-2} + 0 y_{2,t-2} + \epsilon_{1t} \\ y_{2t} &= 0 + 0 y_{1,t-1} + 1 y_{2,t-1} + 0 y_{1,t-2} + 0 y_{2,t-2} + \epsilon_{2t}\end{aligned}$$

### Forecasting of BVAR Modeling

The bootstrap procedure is used to estimate standard errors of the forecast (Litterman 1986). NREP=B simulations are performed. In each simulation the following steps are taken:

1. The procedure generates the available number of observations,  $T$ , and uniform random integers  $I_t$ , where  $t = 1, \dots, T$ .
2. A new observation,  $\tilde{\mathbf{y}}_t$ , is obtained as a sum of the forecast based on the estimates coefficients plus the vector of residuals from the  $I_t$ ; that is,

$$\tilde{\mathbf{y}}_t = \sum_{j=1}^p \hat{\Phi}_j \mathbf{y}_{t-j} + \hat{\epsilon}_{I_t}.$$

3. A new BVAR model is estimated by using the most recent observations, and a prediction value is made of the most recent observations.

The MSE measure of the  $l$ -step-ahead forecast is

$$MSE(l) = \frac{1}{B} \sum_{i=1}^B (\tilde{\mathbf{y}}_{t+l|t}^i - \bar{\bar{\mathbf{y}}}_t)^2$$

where  $\bar{\bar{\mathbf{y}}}_t = (1/B) \sum_{i=1}^B \tilde{\mathbf{y}}_t^i$ .

---

## Cointegration

This section briefly introduces the concepts of cointegration.

**Definition 1.** (Engle and Granger 1987): If a series  $y_t$  with no deterministic components can be represented by a stationary and invertible ARMA process after differencing  $d$  times, the series is integrated of order  $d$ , that is,  $y_t \sim I(d)$ .

**Definition 2.** (Engle and Granger 1987): If all elements of the vector  $\mathbf{y}_t$  are  $I(d)$  and there exists a cointegrating vector  $\beta \neq 0$  such that  $\beta' \mathbf{y}_t \sim I(d-b)$  for any  $b > 0$ , the vector process is said to be cointegrated  $CI(d, b)$ .

A simple example of a cointegrated process is the following bivariate system:

$$\begin{aligned} y_{1t} &= \gamma y_{2t} + \epsilon_{1t} \\ y_{2t} &= y_{2,t-1} + \epsilon_{2t} \end{aligned}$$

with  $\epsilon_{1t}$  and  $\epsilon_{2t}$  being uncorrelated white noise processes. In the second equation,  $y_{2t}$  is a random walk,  $\Delta y_{2t} = \epsilon_{2t}$ ,  $\Delta \equiv 1 - B$ . Differencing the first equation results in

$$\Delta y_{1t} = \gamma \Delta y_{2t} + \Delta \epsilon_{1t} = \gamma \epsilon_{2t} + \epsilon_{1t} - \epsilon_{1,t-1}.$$

Thus, both  $y_{1t}$  and  $y_{2t}$  are  $I(1)$  processes, but the linear combination  $y_{1t} - \gamma y_{2t}$  is stationary. Hence  $\mathbf{y}_t = (y_{1t}, y_{2t})'$  is cointegrated with a cointegrating vector  $\beta = (1, -\gamma)'$ .

In general, if the vector process  $\mathbf{y}_t$  has  $k$  components, then there can be more than one cointegrating vector  $\beta'$ . It is assumed that there are  $r$  linearly independent cointegrating vectors with  $r < k$ , which make the  $k \times r$  matrix  $\beta$ . The rank of matrix  $\beta$  is  $r$ , which is called the *cointegration rank* of  $\mathbf{y}_t$ .

---

## Common Trends

This section briefly discusses the implication of cointegration for the moving-average representation. Let  $\mathbf{y}_t$  be cointegrated  $CI(1, 1)$ , then  $\Delta \mathbf{y}_t$  has the Wold representation.

$$\Delta \mathbf{y}_t = \boldsymbol{\delta} + \Psi(B)\boldsymbol{\epsilon}_t, \quad \sum_{j=0}^{\infty} j|\Psi_j| < \infty$$

where  $\boldsymbol{\epsilon}_t$  is  $iid(0, \Omega)$  and  $\Psi(B) = \sum_{j=0}^{\infty} \Psi_j B^j$  with  $\Psi_0 = I_k$ .

Assume that  $\boldsymbol{\epsilon}_t = 0$  if  $t \leq 0$  and  $\mathbf{y}_0$  is a nonrandom initial value. Then the difference equation implies that

$$\mathbf{y}_t = \mathbf{y}_0 + \boldsymbol{\delta}t + \Psi(1) \sum_{i=0}^t \boldsymbol{\epsilon}_i + \Psi^*(B)\boldsymbol{\epsilon}_t$$

where  $\Psi^*(B) = (1 - B)^{-1}(\Psi(B) - \Psi(1))$  and  $\Psi^*(B)$  is absolutely summable.

Assume that the rank of  $\Psi(1)$  is  $m = k - r$ . When the process  $\mathbf{y}_t$  is cointegrated, there is a cointegrating  $k \times r$  matrix  $\beta$  such that  $\beta' \mathbf{y}_t$  is stationary.

Premultiplying  $\mathbf{y}_t$  by  $\beta'$  results in

$$\beta' \mathbf{y}_t = \beta' \mathbf{y}_0 + \beta' \Psi^*(B)\boldsymbol{\epsilon}_t$$

because  $\beta' \Psi(1) = 0$  and  $\beta' \boldsymbol{\delta} = 0$ .

Stock and Watson (1988) showed that the cointegrated process  $\mathbf{y}_t$  has a common trends representation derived from the moving-average representation. Since the rank of  $\Psi(1)$  is  $m = k - r$ , there is a  $k \times r$  matrix  $H_1$  with rank  $r$  such that  $\Psi(1)H_1 = 0$ . Let  $H_2$  be a  $k \times m$  matrix with rank  $m$  such that  $H_2' H_1 = 0$ , then  $A = C(1)H_2$  has rank  $m$ . The  $H = (H_1, H_2)$  has rank  $k$ . By construction of  $H$ ,

$$\Psi(1)H = [0, A] = AS_m$$

where  $S_m = (0_{m \times r}, I_m)$ . Since  $\beta' \Psi(1) = 0$  and  $\beta' \boldsymbol{\delta} = 0$ ,  $\boldsymbol{\delta}$  lies in the column space of  $\Psi(1)$  and can be written

$$\boldsymbol{\delta} = C(1)\tilde{\boldsymbol{\delta}}$$



where  $\tilde{\delta}$  is a  $k$ -dimensional vector. The common trends representation is written as

$$\begin{aligned} \mathbf{y}_t &= \mathbf{y}_0 + \Psi(1)[\tilde{\delta}t + \sum_{i=0}^t \boldsymbol{\epsilon}_i] + \Psi^*(B)\boldsymbol{\epsilon}_t \\ &= \mathbf{y}_0 + \Psi(1)H[H^{-1}\tilde{\delta}t + H^{-1}\sum_{i=0}^t \boldsymbol{\epsilon}_i] + \mathbf{a}_t \\ &= \mathbf{y}_0 + A\boldsymbol{\tau}_t + \mathbf{a}_t \end{aligned}$$

and

$$\boldsymbol{\tau}_t = \pi + \boldsymbol{\tau}_{t-1} + \mathbf{v}_t$$

where  $\mathbf{a}_t = \Psi^*(B)\boldsymbol{\epsilon}_t$ ,  $\pi = S_m H^{-1}\tilde{\delta}$ ,  $\boldsymbol{\tau}_t = S_m[H^{-1}\tilde{\delta}t + H^{-1}\sum_{i=0}^t \boldsymbol{\epsilon}_i]$ , and  $\mathbf{v}_t = S_m H^{-1}\boldsymbol{\epsilon}_t$ .

Stock and Watson showed that the common trends representation expresses  $\mathbf{y}_t$  as a linear combination of  $m$  random walks ( $\boldsymbol{\tau}_t$ ) with drift  $\pi$  plus  $I(0)$  components ( $\mathbf{a}_t$ ).

### Test for the Common Trends

Stock and Watson (1988) proposed statistics for common trends testing. The null hypothesis is that  $k$ -dimensional time series  $\mathbf{y}_t$  has  $m \leq k$  common stochastic trends, and the alternative is that it has  $s < m$  common trends. The test procedure of  $m$  vs  $s$  common stochastic trends is performed based on the first-order serial correlation matrix of  $\mathbf{y}_t$ . Let  $\beta_{\perp}$  be a  $k \times m$  matrix orthogonal to the cointegrating matrix such that  $\beta'_{\perp}\beta = 0$ , and  $\beta_{\perp}\beta'_{\perp} = I_m$ . Let  $\mathbf{z}_t = \beta'\mathbf{y}_t$  and  $\mathbf{w}_t = \beta'_{\perp}\mathbf{y}_t$ . Then

$$\mathbf{w}_t = \beta'_{\perp}\mathbf{y}_0 + \beta'_{\perp}\boldsymbol{\delta}t + \beta'_{\perp}\Psi(1)\sum_{i=0}^t \boldsymbol{\epsilon}_i + \beta'_{\perp}\Psi^*(B)\boldsymbol{\epsilon}_t.$$

Combining the expression of  $\mathbf{z}_t$  and  $\mathbf{w}_t$ ,

$$\begin{aligned} \begin{bmatrix} \mathbf{z}_t \\ \mathbf{w}_t \end{bmatrix} &= \begin{bmatrix} \beta'\mathbf{y}_0 \\ \beta'_{\perp}\mathbf{y}_0 \end{bmatrix} + \begin{bmatrix} 0 \\ \beta'_{\perp}\boldsymbol{\delta} \end{bmatrix} t + \begin{bmatrix} 0 \\ \beta'_{\perp}\Psi(1) \end{bmatrix} \sum_{i=1}^t \boldsymbol{\epsilon}_i \\ &\quad + \begin{bmatrix} \beta'\Psi^*(B) \\ \beta'_{\perp}\Psi^*(B) \end{bmatrix} \boldsymbol{\epsilon}_t. \end{aligned}$$

The Stock-Watson common trends test is performed based on the component  $\mathbf{w}_t$  by testing whether  $\beta'_{\perp}\Psi(1)$  has rank  $m$  against rank  $s$ .

The following statements test Stock-Watson common trends:

```
proc varmax data=simul2;
    model y1 y2 / p=2 cointtest=(sw);
run;
```

| The VARMAX Procedure                                               |                |            |          |                   |     |
|--------------------------------------------------------------------|----------------|------------|----------|-------------------|-----|
| Testing for Stock-Watson's Common Trends using Differencing Filter |                |            |          |                   |     |
| H_0:<br>Rank=m                                                     | H_1:<br>Rank=s | Eigenvalue | Filt     | Critical<br>Value | Lag |
| 1                                                                  | 0              | 1.0009     | 0.0906   | -14.10            | 2   |
| 2                                                                  | 0              | 0.9968     | -0.3237  | -8.80             |     |
|                                                                    | 1              | 0.6489     | -35.1092 | -23.00            |     |

**Figure 4.39.** Common Trends Test (COINTTEST=(SW) option)

In Figure 4.39, the first column is the null hypothesis that  $\mathbf{y}_t$  has  $m \leq k$  common trends; the second column, the alternative hypothesis that  $\mathbf{y}_t$  has  $s < m$  common trends; the fourth column, the test statistics using AR(2) filtering the data. The test statistics for testing for 2 versus 1 common trends are more negative (-35.1) than the critical value (-23.0). The test rejects the null hypothesis, which means that the series has a single common trend.

## Vector Error Correction Modeling

This section discusses the implication of cointegration for the autoregressive representation. Assume that the cointegrated series can be represented by a vector error correction model according to the Granger representation theorem (Engle and Granger 1987). Consider the vector autoregressive process with Gaussian errors defined by

$$\mathbf{y}_t = \sum_{i=1}^p \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

or

$$\Phi(B)\mathbf{y}_t = \boldsymbol{\epsilon}_t$$

where the initial values,  $\mathbf{y}_{-p+1}, \dots, \mathbf{y}_0$ , are fixed and  $\boldsymbol{\epsilon}_t \sim N(0, \Sigma)$ . Since the AR operator  $\Phi(B)$  can be re-expressed as  $\Phi(B) = \Phi^*(B)(1 - B) + \Phi(1)B$  where  $\Phi^*(B) = I_k - \sum_{i=1}^{p-1} \Phi_i^* B^i$  with  $\Phi_i^* = -\sum_{j=i+1}^p \Phi_j$ , the vector error correction model is

$$\Phi^*(B)(1 - B)\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

or

$$\Delta\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta\mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where  $\boldsymbol{\alpha}\boldsymbol{\beta}' = -\Phi(1) = -I_k + \Phi_1 + \Phi_2 + \dots + \Phi_p$ .

One motivation for the VECM( $p$ ) form is to consider the relation  $\beta' \mathbf{y}_t = \mathbf{c}$  as defining the underlying economic relations and assume that the agents react to the disequilibrium error  $\beta' \mathbf{y}_t - \mathbf{c}$  through the adjustment coefficient  $\alpha$  to restore equilibrium; that is, they satisfy the economic relations. The cointegrating vector,  $\beta$  is sometimes called the *long-run parameters*.

You can consider a vector error correction model with a deterministic term. The deterministic term  $D_t$  can contain a constant, a linear trend, seasonal dummy variables, or nonstochastic regressors.

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + A D_t + \epsilon_t$$

where  $\Pi = \alpha \beta'$ .

The alternative vector error correction representation considers the error correction term at lag  $t - p$  and is written as

$$\Delta \mathbf{y}_t = \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \Pi^* \mathbf{y}_{t-p} + A D_t + \epsilon_t$$

If the matrix  $\Pi$  has a full rank ( $r = k$ ), all components of  $\mathbf{y}_t$  are  $I(0)$ . On the other hand,  $\mathbf{y}_t$  are stationary in difference if  $\text{rank}(\Pi) = 0$ . When the rank of the matrix  $\Pi$  is  $r < k$ , there are  $k - r$  linear combinations that are nonstationary and  $r$  stationary cointegrating relations. Note that the linearly independent vector  $\mathbf{z}_t = \beta' \mathbf{y}_t$  is stationary and this transformation is not unique unless  $r = 1$ . There does not exist a unique cointegrating matrix  $\beta$  since the coefficient matrix  $\Pi$  can also be decomposed as

$$\Pi = \alpha M M^{-1} \beta' = \alpha^* \beta^{*'}$$

where  $M$  is an  $r \times r$  nonsingular matrix.

### Test for the Cointegration

The cointegration rank test determines the linearly independent columns of  $\Pi$ . Johansen (1988, 1995a) and Johansen and Juselius (1990) proposed the cointegration rank test using the reduced rank regression.

### Different Specifications of Deterministic Trends

When you construct the VECM( $p$ ) form from the VAR( $p$ ) model, the deterministic terms in the VECM( $p$ ) form can differ from those in the VAR( $p$ ) model. When there are deterministic cointegrated relationships among variables, deterministic terms in the VAR( $p$ ) model will not be present in the VECM( $p$ ) form. On the other hand, if there are stochastic cointegrated relationships, deterministic terms appear in the VECM( $p$ ) form via the error correction term or as an independent term in the VECM( $p$ ) form.

- **Case 1:** There is no separate drift in the VECM( $p$ ) form.

$$\Delta \mathbf{y}_t = \alpha \beta' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \epsilon_t$$

- **Case 2:** There is no separate drift in the VECM( $p$ ) form, but a constant enters only via the error correction term.

$$\Delta \mathbf{y}_t = \alpha (\beta', \beta_0) (\mathbf{y}'_{t-1}, 1)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \epsilon_t$$

- **Case 3:** There is a separate drift and no separate linear trend in the VECM( $p$ ) form.

$$\Delta \mathbf{y}_t = \alpha \beta' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \delta_0 + \epsilon_t$$

- **Case 4:** There is a separate drift and no separate linear trend in the VECM( $p$ ) form, but a linear trend enters only via the error correction term.

$$\Delta \mathbf{y}_t = \alpha (\beta', \beta_1) (\mathbf{y}'_{t-1}, t)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \delta_0 + \epsilon_t$$

- **Case 5:** There is a separate linear trend in the VECM( $p$ ) form.

$$\Delta \mathbf{y}_t = \alpha \beta' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \delta_0 + \delta_1 t + \epsilon_t$$

First, focus on cases 1, 3, and 5 to test the null hypothesis that there are at most  $r$  cointegrating vectors. Let

$$\begin{aligned} Z_{0t} &= \Delta \mathbf{y}_t \\ Z_{1t} &= \mathbf{y}_{t-1} \\ Z_{2t} &= [\Delta \mathbf{y}'_{t-1}, \dots, \Delta \mathbf{y}'_{t-p+1}, D_t]' \\ Z_0 &= [Z_{01}, \dots, Z_{0T}]' \\ Z_1 &= [Z_{11}, \dots, Z_{1T}]' \\ Z_2 &= [Z_{21}, \dots, Z_{2T}]' \end{aligned}$$

where  $D_t$  can be empty for Case 1; 1 for Case 3;  $(1, t)$  for Case 5. Let  $\Psi$  be the matrix of parameters consisting of  $\Phi_1^*, \dots, \Phi_{p-1}^*$  and  $A$ , where parameters  $A$  corresponds to regressors  $D_t$ . Then the VECM( $p$ ) form is rewritten in these variables as

$$Z_{0t} = \alpha \beta' Z_{1t} + \Psi Z_{2t} + \epsilon_t$$

The log-likelihood function is given by

$$\begin{aligned} \ell = & -\frac{kT}{2} \log 2\pi - \frac{T}{2} \log |\Sigma| \\ & - \frac{1}{2} \sum_{t=1}^T (Z_{0t} - \alpha\beta'Z_{1t} - \Psi Z_{2t})' \Sigma^{-1} (Z_{0t} - \alpha\beta'Z_{1t} - \Psi Z_{2t}) \end{aligned}$$

The residuals,  $R_{0t}$  and  $R_{1t}$ , are obtained by regressing  $Z_{0t}$  and  $Z_{1t}$  on  $Z_{2t}$ , respectively. The regression equation in residuals is

$$R_{0t} = \alpha\beta'R_{1t} + \hat{\epsilon}_t$$

The crossproducts matrices are computed

$$S_{ij} = \frac{1}{T} \sum_{t=1}^T R_{it}R'_{jt}, \quad i, j = 0, 1$$

Then the maximum likelihood estimator for  $\beta$  is obtained from the eigenvectors corresponding to the  $r$  largest eigenvalues of the following equation:

$$|\lambda S_{11} - S_{10}S_{00}^{-1}S_{01}| = 0$$

The eigenvalues of the preceding equation are squared canonical correlations between  $R_{0t}$  and  $R_{1t}$ , and the eigenvectors corresponding to the  $r$  largest eigenvalues are the  $r$  linear combinations of  $\mathbf{y}_{t-1}$ , which have the largest squared partial correlations with the stationary process  $\Delta\mathbf{y}_t$  after correcting for lags and deterministic terms. Such an analysis calls for a reduced rank regression of  $\Delta\mathbf{y}_t$  on  $\mathbf{y}_{t-1}$  corrected for  $(\Delta\mathbf{y}_{t-1}, \dots, \Delta\mathbf{y}_{t-p+1}, D_t)$ , as discussed by Anderson (1951). Johansen (1988) suggested two test statistics to test the null hypothesis that there are at most  $r$  cointegrating vectors

$$H_0 : \lambda_i = 0 \text{ for } i = r + 1, \dots, k$$

### Trace Test

$$\lambda_{trace} = -T \sum_{i=r+1}^k \log(1 - \lambda_i)$$

The asymptotic distribution of this statistic is given by

$$tr \left\{ \int_0^1 (dW) \bar{W}' \left( \int_0^1 \bar{W} \bar{W}' dr \right)^{-1} \int_0^1 \bar{W} (dW)' \right\}$$

where  $tr(A)$  is the trace of a matrix  $A$ ,  $W$  is the  $k - r$  dimensional Brownian motion, and  $\bar{W}$  is the Brownian motion itself, or the demeaned or detrended Brownian motion according to the different specifications of deterministic trends in the vector error correction model.

### Maximum Eigenvalue Test

$$\lambda_{max} = -T \log(1 - \lambda_{r+1})$$

The asymptotic distribution of this statistic is given by

$$\max\left\{\int_0^1 (dW) \tilde{W}' \left(\int_0^1 \tilde{W} \tilde{W}' dr\right)^{-1} \int_0^1 \tilde{W} (dW)'\right\}$$

where  $\max(A)$  is the maximum eigenvalue of a matrix  $A$ . Osterwald-Lenum (1992) provided the detailed tables of critical values of these statistics.

In case 2, consider that  $Z_{1t} = (\mathbf{y}'_t, 1)'$  and  $Z_{2t} = (\Delta \mathbf{y}'_{t-1}, \dots, \Delta \mathbf{y}'_{t-p+1})'$ . In case 4,  $Z_{1t} = (\mathbf{y}'_t, t)'$  and  $Z_{2t} = (\Delta \mathbf{y}'_{t-1}, \dots, \Delta \mathbf{y}'_{t-p+1}, 1)'$ .

The following statements use the JOHANSEN option to compute the Johansen cointegration rank test of integrated order 1:

```
proc varmax data=simul2;
  model y1 y2 / p=2 cointtest=(johansen=(normalize=y1));
run;
```

| The VARMAX Procedure                          |                |            |       |                   |                |                    |
|-----------------------------------------------|----------------|------------|-------|-------------------|----------------|--------------------|
| Cointegration Rank Test                       |                |            |       |                   |                |                    |
| H_0:<br>Rank=r                                | H_1:<br>Rank>r | Eigenvalue | Trace | Critical<br>Value | Drift<br>InECM | DriftIn<br>Process |
| 0                                             | 0              | 0.4644     | 61.75 | 15.34             | Constant       | Linear             |
| 1                                             | 1              | 0.0056     | 0.56  | 3.84              |                |                    |
| Cointegration Rank Test under the Restriction |                |            |       |                   |                |                    |
| H_0:<br>Rank=r                                | H_1:<br>Rank>r | Eigenvalue | Trace | Critical<br>Value | Drift<br>InECM | DriftIn<br>Process |
| 0                                             | 0              | 0.5209     | 76.38 | 19.99             | Constant       | Constant           |
| 1                                             | 1              | 0.0426     | 4.27  | 9.13              |                |                    |

**Figure 4.40.** Cointegration Rank Test (JOHANSEN option)

Suppose that the model has an intercept term. The first table in Figure 4.40 shows the trace statistics based on case 3; the second table, case 2. The output indicates that the series are cointegrated with rank 1.

| The VARMAX Procedure                   |                              |                |                    |    |                |
|----------------------------------------|------------------------------|----------------|--------------------|----|----------------|
| Test of the Restriction<br>when Rank=r |                              |                |                    |    |                |
|                                        | Hypo                         | Drift<br>InECM | DriftIn<br>Process |    |                |
|                                        | H_0                          | Constant       | Linear             |    |                |
|                                        | H_1                          | Constant       | Constant           |    |                |
| Test of the Restriction when Rank=r    |                              |                |                    |    |                |
| Rank                                   | Eigenvalue<br>On<br>Restrict | Eigenvalue     | Chi-<br>Square     | DF | Prob><br>ChiSq |
| 0                                      | 0.5209                       | 0.4644         | 14.63              | 2  | 0.0007         |
| 1                                      | 0.0426                       | 0.0056         | 3.71               | 1  | 0.0540         |

**Figure 4.41.** Cointegration Rank Test Continued

Figure 4.41 shows which result, case 3 (the hypothesis H\_0) and case 2 (the hypothesis H\_1), is appropriate. Since the cointegration rank is chosen to be 1 by the result in Figure 4.40, look at the last row corresponding to rank=1.

| The VARMAX Procedure                   |          |          |
|----------------------------------------|----------|----------|
| Long-Run Parameter BETA Estimates      |          |          |
| Variable                               | Dummy 1  | Dummy 2  |
| y1                                     | 1.00000  | 1.00000  |
| y2                                     | -2.04869 | -0.02854 |
| Adjustment Coefficient ALPHA Estimates |          |          |
| Variable                               | Dummy 1  | Dummy 2  |
| y1                                     | -0.46421 | -0.00502 |
| y2                                     | 0.17535  | -0.01275 |

**Figure 4.42.** Cointegration Rank Test Continued

Figure 4.42 shows the estimates of long-run parameter and adjustment coefficients based on case 3. Considering that the cointegration rank is 1, the long-run relationship of the series is

$$y_{1t} = 2.049 y_{2t}$$

| The VARMAX Procedure                                          |          |           |             |
|---------------------------------------------------------------|----------|-----------|-------------|
| Long-Run Coefficient BETA based<br>on the Restricted Trend    |          |           |             |
| Variable                                                      | Dummy 1  | Dummy 2   | Dummy 3     |
| y1                                                            | 1.00000  | 1.00000   | 1.00000     |
| y2                                                            | -2.04366 | -2.75773  | -0.73198    |
| 1                                                             | 6.75919  | 101.37051 | -42.50051   |
| Adjustment Coefficient ALPHA<br>based on the Restricted Trend |          |           |             |
| Variable                                                      | Dummy 1  | Dummy 2   | Dummy 3     |
| y1                                                            | -0.48015 | 0.01091   | 5.96583E-14 |
| y2                                                            | 0.12538  | 0.03722   | -2.834E-13  |

**Figure 4.43.** Cointegration Rank Test Continued

Figure 4.43 shows the estimates of long-run parameter and adjustment coefficients based on case 2. Considering that the cointegration rank is 1, the long-run relationship of the series is

$$y_{1t} = 2.044 y_{2t} - 6.760$$

#### **Estimation of Vector Error Correction Model**

Now consider cases 1, 3, and 5 for the vector error correction model. The preceding log-likelihood function is maximized for

$$\begin{aligned}
 \hat{\beta} &= S_{11}^{-1/2} [v_1, \dots, v_r] \\
 \hat{\alpha} &= S_{01} \hat{\beta} (\hat{\beta}' S_{11} \hat{\beta})^{-1} \\
 \hat{\Pi} &= \hat{\alpha} \hat{\beta}' \\
 \hat{\Psi} &= (Z_2' Z_2)^{-1} Z_2' (Z_0 - Z_1 \hat{\Pi}') \\
 \hat{\Sigma} &= (Z_0 - Z_2 \hat{\Psi}' - Z_1 \hat{\Pi}')' (Z_0 - Z_2 \hat{\Psi}' - Z_1 \hat{\Pi}') / T
 \end{aligned}$$

The estimators of the orthogonal complements of  $\alpha$  and  $\beta$  are

$$\hat{\beta}_{\perp} = S_{11} [v_{r+1}, \dots, v_k]$$

and

$$\hat{\alpha}_{\perp} = S_{00}^{-1} S_{01} [v_{r+1}, \dots, v_k]$$

The ML estimators have the following asymptotic properties:

$$\sqrt{T} \text{vec}([\hat{\Pi}, \hat{\Psi}] - [\Pi, \Psi]) \xrightarrow{d} N(0, \Sigma_{co})$$

where

$$\Sigma_{co} = \Sigma \otimes \left( \begin{bmatrix} \beta & 0 \\ 0 & I_k \end{bmatrix} \Omega^{-1} \begin{bmatrix} \beta' & 0 \\ 0 & I_k \end{bmatrix} \right)$$



and

$$\Omega = \text{plim} \frac{1}{T} \begin{bmatrix} \beta' Z_1' Z_1 \beta & \beta' Z_1' Z_2 \\ Z_2' Z_1 \beta & Z_2' Z_2 \end{bmatrix}$$

### Test for the Linear Restriction of $\beta$

Consider the example with the variables  $m_t$ , log real money,  $y_t$ , log real income,  $i_t^d$ , deposit interest rate, and  $i_t^b$ , bond interest rate. It seems a natural hypothesis that in the long-run relation, money and income have equal coefficients with opposite signs. This can be formulated as the hypothesis that the cointegrated relation contains only  $m_t$  and  $y_t$  through  $m_t - y_t$ . For the analysis, you can express these restrictions in the parameterization of  $H$  such that  $\beta = H\psi$ , where  $H$  is a known  $k \times s$  matrix and  $\psi$  is the  $s \times r$  ( $r \leq s < k$ ) parameter matrix to be estimated. For this example,  $H$  is given by

$$H = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**Restriction**  $H_0 : \beta = H\phi$

When the linear restriction  $\beta = H\phi$  is given, it implies that the same restrictions are imposed on all cointegrating vectors. You obtain the maximum likelihood estimator of  $\beta$  by reduced rank regression of  $\Delta \mathbf{y}_t$  on  $H \mathbf{y}_{t-1}$  corrected for  $(\Delta \mathbf{y}_{t-1}, \dots, \Delta \mathbf{y}_{t-p+1}, D_t)$ , solving the following equation

$$|\rho H' S_{11} H - H' S_{10} S_{00}^{-1} S_{01} H| = 0$$

for the eigenvalues  $1 > \rho_1 > \dots > \rho_s > 0$  and eigenvectors  $(v_1, \dots, v_s)$ ,  $S_{ij}$  are given in the preceding section. Then choose  $\hat{\phi} = (v_1, \dots, v_r)$  corresponding to the  $r$  largest eigenvalues, and the  $\hat{\beta}$  is  $H\hat{\phi}$ .

The test statistic for  $H_0 : \beta = H\phi$  is given by

$$T \sum_{i=1}^r \log\{(1 - \rho_i)/(1 - \lambda_i)\} \xrightarrow{d} \chi_{r(k-s)}^2$$

If the data have no deterministic trend, the constant term should be restricted by  $\alpha'_\perp \delta_0 = 0$  like Case 2. Then  $H$  is given by

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The following statements test that  $\beta_1 + 2\beta_2 = 0$ :

```
proc varmax data=simul2;
  model y1 y2 / p=2 ecm=(rank=1 normalize=y1);
  cointeg rank=1 h=(1, -2);
run;
```

| The VARMAX Procedure                                                  |                          |            |                |    |                |
|-----------------------------------------------------------------------|--------------------------|------------|----------------|----|----------------|
| Long-Run Coefficient<br>BETA with respect to<br>Hypothesis on BETA    |                          |            |                |    |                |
| Variable                                                              |                          | Dummy 1    |                |    |                |
| y1                                                                    |                          | 1.00000    |                |    |                |
| y2                                                                    |                          | -2.00000   |                |    |                |
| Adjustment Coefficient<br>ALPHA with respect to<br>Hypothesis on BETA |                          |            |                |    |                |
| Variable                                                              |                          | Dummy 1    |                |    |                |
| y1                                                                    |                          | -0.47404   |                |    |                |
| y2                                                                    |                          | 0.17534    |                |    |                |
| Test for Restricted Long-Run Coefficient BETA                         |                          |            |                |    |                |
| Index                                                                 | Eigenvalue<br>OnRestrict | Eigenvalue | Chi-<br>Square | DF | Prob><br>ChiSq |
| 1                                                                     | 0.4616                   | 0.4644     | 0.51           | 1  | 0.4738         |

**Figure 4.44.** Testing of Linear Restriction  $\beta$  (H= option)

Figure 4.44 shows the results of testing  $H_0 : \beta_1 + 2\beta_2 = 0$ . The input  $H$  matrix is  $H = (1, -2)'$ . The adjustment coefficient is reestimated under the restriction, and the test indicates that you cannot reject the null hypothesis.

### ***Test for the Weak Exogeneity and Restrictions of $\alpha$***

Consider a vector error correction model:

$$\Delta \mathbf{y}_t = \alpha \beta' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + A D_t + \epsilon_t$$

Divide the process  $\mathbf{y}_t$  into  $(\mathbf{y}'_{1t}, \mathbf{y}'_{2t})'$  with dimension  $k_1$  and  $k_2$  and the  $\Sigma$  into

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

Similarly, the parameters can be decomposed as follows:

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \quad \Phi_i^* = \begin{bmatrix} \Phi_{1i}^* \\ \Phi_{2i}^* \end{bmatrix} \quad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

## Part 2. Procedure Reference

Then the VECM(p) form can be rewritten using the decomposed parameters and processes:

$$\begin{bmatrix} \Delta \mathbf{y}_{1t} \\ \Delta \mathbf{y}_{2t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix} \boldsymbol{\beta}' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \begin{bmatrix} \Phi_{1i}^* \\ \Phi_{2i}^* \end{bmatrix} \Delta \mathbf{y}_{t-i} + \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} D_t + \begin{bmatrix} \boldsymbol{\epsilon}_{1t} \\ \boldsymbol{\epsilon}_{2t} \end{bmatrix}$$

The conditional model for  $\mathbf{y}_{1t}$  given  $\mathbf{y}_{2t}$  is

$$\begin{aligned} \Delta \mathbf{y}_{1t} &= \omega \Delta \mathbf{y}_{2t} + (\alpha_1 - \omega \alpha_2) \boldsymbol{\beta}' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} (\Phi_{1i}^* - \omega \Phi_{2i}^*) \Delta \mathbf{y}_{t-i} \\ &\quad + (A_1 - \omega A_2) D_t + \boldsymbol{\epsilon}_{1t} - \omega \boldsymbol{\epsilon}_{2t} \end{aligned}$$

and the marginal model of  $\mathbf{y}_{2t}$ ,

$$\Delta \mathbf{y}_{2t} = \alpha_2 \boldsymbol{\beta}' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_{2i}^* \Delta \mathbf{y}_{t-i} + A_2 D_t + \boldsymbol{\epsilon}_{2t}$$

where  $\omega = \Sigma_{12} \Sigma_{22}^{-1}$ .

The test of weak exogeneity of  $\mathbf{y}_{2t}$  for the parameters  $(\alpha_1, \boldsymbol{\beta})$  determines whether  $\alpha_2 = 0$ . Weak exogeneity means that there is no information about  $\boldsymbol{\beta}$  in the marginal model or that the variables  $\mathbf{y}_{2t}$  do not react to a disequilibrium.

**Restriction**  $H_0: \boldsymbol{\alpha} = J\psi$

Consider the null hypothesis  $H_0: \boldsymbol{\alpha} = J\psi$ , where  $J$  is a  $k \times m$  matrix with  $r \leq m < k$ .

From the previous residual regression equation

$$R_{0t} = \boldsymbol{\alpha} \boldsymbol{\beta}' R_{1t} + \hat{\boldsymbol{\epsilon}}_t = J\psi \boldsymbol{\beta}' R_{1t} + \hat{\boldsymbol{\epsilon}}_t$$

you can obtain

$$\begin{aligned} \bar{J}' R_{0t} &= \psi \boldsymbol{\beta}' R_{1t} + \bar{J}' \hat{\boldsymbol{\epsilon}}_t \\ J_{\perp}' R_{0t} &= J_{\perp}' \hat{\boldsymbol{\epsilon}}_t \end{aligned}$$

where  $\bar{J} = J(J'J)^{-1}$  and  $J_{\perp}$  is orthogonal to  $J$  such that  $J_{\perp}' J = 0$ .

Define

$$\Omega_{JJ_{\perp}} = \bar{J}' \Omega J_{\perp} \text{ and } \Omega_{J_{\perp} J_{\perp}} = J_{\perp}' \Omega J_{\perp}$$

and let  $\omega = \Omega_{JJ_{\perp}} \Omega_{J_{\perp} J_{\perp}}^{-1}$ . Then  $\bar{J}' R_{0t}$  can be written

$$\bar{J}' R_{0t} = \psi \boldsymbol{\beta}' R_{1t} + \omega J_{\perp}' R_{0t} + \bar{J}' \hat{\boldsymbol{\epsilon}}_t - \omega J_{\perp}' \hat{\boldsymbol{\epsilon}}_t$$

Using the marginal distribution of  $J'_{\perp} R_{0t}$  and the conditional distribution of  $\bar{J}' R_{0t}$ , the new residuals are computed as

$$\begin{aligned}\tilde{R}_{Jt} &= \bar{J}' R_{0t} - S_{JJ_{\perp}} S_{J_{\perp} J_{\perp}}^{-1} J'_{\perp} R_{0t} \\ \tilde{R}_{1t} &= R_{1t} - S_{1J_{\perp}} S_{J_{\perp} J_{\perp}}^{-1} J'_{\perp} R_{0t}\end{aligned}$$

where

$$S_{JJ_{\perp}} = \bar{J}' S_{00} J_{\perp}, \quad S_{J_{\perp} J_{\perp}} = J'_{\perp} S_{00} J_{\perp}, \quad \text{and} \quad S_{J_{\perp} 1} = J'_{\perp} S_{01}$$

In terms of  $\tilde{R}_{Jt}$  and  $\tilde{R}_{1t}$ , the MLE of  $\beta$  is computed by using the reduced rank regression. Let

$$S_{ij.J_{\perp}} = \frac{1}{T} \sum_{t=1}^T \tilde{R}_{it} \tilde{R}_{jt}', \quad \text{for } i, j = 1, J$$

Under the null hypothesis  $H_0 : \alpha = J\psi$ , the MLE  $\tilde{\beta}$  is computed by solving the equation

$$|\rho S_{11.J_{\perp}} - S_{1J.J_{\perp}} S_{JJ.J_{\perp}}^{-1} S_{J1.J_{\perp}}| = 0$$

Then  $\tilde{\beta} = (v_1, \dots, v_r)$ , where the eigenvectors correspond to the  $r$  largest eigenvalues. The likelihood ratio test for  $H_0 : \alpha = J\psi$  is

$$T \sum_{i=1}^r \log\{(1 - \rho_i)/(1 - \lambda_i)\} \xrightarrow{d} \chi^2_{r(k-m)}$$

The test of weak exogeneity of  $y_{2t}$  is the special case of the test  $\alpha = J\psi$ , considering  $J = (I_{k_1}, 0)'$ . Consider the previous example with four variables ( $m_t, y_t, i_t^b, i_t^d$ ). If  $r = 1$ , you formulate the weak exogeneity of  $(y_t, i_t^b, i_t^d)$  for  $m_t$  as  $J = [1, 0, 0, 0]'$  and the weak exogeneity of  $i_t^d$  for  $(m_t, y_t, i_t^b)$  as  $J = [I_3, 0]'$ .

The following statements test the weak exogeneity of other variables:

```
proc varmax data=simul2;
  model y1 y2 / p=2 ecm=(rank=1 normalize=y1);
  cointeg rank=1 exogeneity;
run;
```

| The VARMAX Procedure                             |            |    |            |  |
|--------------------------------------------------|------------|----|------------|--|
| Tests of Weak Exogeneity<br>of Each of Variables |            |    |            |  |
| Variable                                         | Chi-Square | DF | Prob>ChiSq |  |
| y1                                               | 53.46      | 1  | <.0001     |  |
| y2                                               | 8.76       | 1  | 0.0031     |  |

Figure 4.45. Testing of Weak Exogeneity (EXOGENEITY option)

Figure 4.45 shows that each variable is not the weak exogeneity of other variable.

### Forecasting of the VECM

Consider the cointegrated moving-average representation of the differenced process of  $\mathbf{y}_t$

$$\Delta \mathbf{y}_t = \boldsymbol{\delta} + \Psi(B)\boldsymbol{\epsilon}_t$$

Assume that  $\mathbf{y}_0 = 0$ . The linear process  $\mathbf{y}_t$  can be written

$$\mathbf{y}_t = \boldsymbol{\delta}t + \sum_{i=1}^t \sum_{j=0}^{t-i} \Psi_j \boldsymbol{\epsilon}_i$$

Therefore, for any  $l > 0$ ,

$$\mathbf{y}_{t+l} = \boldsymbol{\delta}(t+l) + \sum_{i=1}^t \sum_{j=0}^{t+l-i} \Psi_j \boldsymbol{\epsilon}_i + \sum_{i=1}^l \sum_{j=0}^{l-i} \Psi_j \boldsymbol{\epsilon}_{t+i}$$

The  $l$ -step-ahead forecast is derived from the preceding equation:

$$\mathbf{y}_{t+l|t} = \boldsymbol{\delta}(t+l) + \sum_{i=1}^t \sum_{j=0}^{t+l-i} \Psi_j \boldsymbol{\epsilon}_i$$

Note that

$$\lim_{l \rightarrow \infty} \boldsymbol{\beta}' \mathbf{y}_{t+l|t} = 0$$

since  $\lim_{l \rightarrow \infty} \sum_{j=0}^{t+l-i} \Psi_j = \Psi(1)$  and  $\boldsymbol{\beta}' \Psi(1) = 0$ . The long-run forecast of the cointegrated system shows that the cointegrated relationship holds, though there might exist some deviations from the equilibrium status in the short-run. The covariance matrix of the predict error  $\mathbf{e}_{t+l|t} = \mathbf{y}_{t+l} - \mathbf{y}_{t+l|t}$  is

$$\Sigma(l) = \sum_{i=1}^l [(\sum_{j=0}^{l-i} \Psi_j) \Sigma (\sum_{j=0}^{l-i} \Psi_j')]$$

When the linear process is represented as a VECM( $p$ ) model, you can obtain

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \sum_{j=1}^{p-1} \Phi_j^* \Delta \mathbf{y}_{t-j} + \boldsymbol{\delta} + \boldsymbol{\epsilon}_t$$

The transition equation is defined as

$$\mathbf{z}_t = F \mathbf{z}_{t-1} + \mathbf{e}_t$$

where  $\mathbf{z}_t = (\mathbf{y}'_{t-1}, \Delta \mathbf{y}'_t, \Delta \mathbf{y}'_{t-1}, \dots, \Delta \mathbf{y}'_{t-p+2})'$  is a state vector and the transition matrix is

$$F = \begin{bmatrix} I_k & I_k & 0 & \cdots & 0 \\ \Pi & (\Pi + \Phi_1^*) & \Phi_2^* & \cdots & \Phi_{p-1}^* \\ 0 & I_k & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

where  $\mathbf{0}$  is a  $k$ -dimensional zero vector. The observation equation can be written

$$\mathbf{y}_t = \delta t + H \mathbf{z}_t$$

where  $H = [I_k, I_k, 0, \dots, 0]$ .

The  $l$ -step-ahead forecast is computed as

$$\mathbf{y}_{t+l|t} = \delta(t+l) + H F^l \mathbf{z}_t$$

---

## I(2) Model

The VAR( $p$ ) model can be written as the error correction form.

$$\Delta \mathbf{y}_t = \alpha \beta' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + A D_t + \epsilon_t$$

Let  $\Phi^* = I_k - \sum_{i=1}^{p-1} \Phi_i^*$ . If  $\alpha$  and  $\beta$  have full rank  $r$ , and if

$$\text{rank}(\alpha'_{\perp} \Phi^* \beta_{\perp}) = k - r$$

then  $\mathbf{y}_t$  is an  $I(1)$  process. If the condition  $\text{rank}(\alpha'_{\perp} \Phi^* \beta_{\perp}) = k - r$  fails and  $\alpha'_{\perp} \Phi^* \beta_{\perp}$  has reduced rank  $\alpha'_{\perp} \Phi^* \beta_{\perp} = \xi \eta'$  where  $\xi$  and  $\eta$  are  $(k-r) \times s$  matrices with  $s \leq k-r$ ,  $\alpha_{\perp}$  and  $\beta_{\perp}$  are defined as  $k \times (k-r)$  matrices of full rank such that  $\alpha'_{\perp} \alpha_{\perp} = 0$  and  $\beta'_{\perp} \beta_{\perp} = 0$ . If  $\xi$  and  $\eta$  have full rank  $s$ , then the process  $\mathbf{y}_t$  is  $I(2)$ , which has the implication of  $I(2)$  model for the moving-average representation.

$$\mathbf{y}_t = B_0 + B_1 t + C_2 \sum_{j=1}^t \sum_{i=1}^j \epsilon_i + C_1 \sum_{i=1}^t \epsilon_i + C_0(B) \epsilon_t$$

The matrices  $C_1$ ,  $C_2$ , and  $C_0(B)$  are determined by the cointegration properties of the process, and  $B_0$  and  $B_1$  are determined by the initial values. For details, see Johansen (1995a).

The implication of the  $I(2)$  model for the autoregressive representation is given by

$$\Delta^2 \mathbf{y}_t = \Pi \mathbf{y}_{t-1} - \Phi^* \Delta \mathbf{y}_{t-1} + \sum_{i=1}^{p-2} \Psi_i \Delta^2 \mathbf{y}_{t-i} + A D_t + \epsilon_t$$

where  $\Psi_i = -\sum_{j=i+1}^{p-1} \Phi_i^*$  and  $\Phi^* = I_k - \sum_{i=1}^{p-1} \Phi_i^*$ .

**Test for  $I(2)$**

The  $I(2)$  cointegrated model is given by the following parameter restrictions:

$$H_{r,s} : \Pi = \alpha\beta' \text{ and } \alpha'_{\perp} \Phi^* \beta_{\perp} = \xi\eta'$$

where  $\xi$  and  $\eta$  are  $(k-r) \times s$  matrices with  $0 \leq s \leq k-r$ . Let  $H_r^0$  represent the  $I(1)$  model where  $\alpha$  and  $\beta$  have full rank  $r$ , let  $H_{r,s}^0$  represent the  $I(2)$  model where  $\xi$  and  $\eta$  have full rank  $s$ , and let  $H_{r,s}$  represent the  $I(2)$  model where  $\xi$  and  $\eta$  have rank  $\leq s$ . The following table shows the relation between the  $I(1)$  models and the  $I(2)$  models.

**Table 4.1.** Relation between the  $I(1)$  and  $I(2)$  Models

|                      |          | $I(2)$    |          |           |         |           | $I(1)$      |           |             |             |
|----------------------|----------|-----------|----------|-----------|---------|-----------|-------------|-----------|-------------|-------------|
| $r \backslash k-r-s$ | $k$      | $k-1$     | $\dots$  | $1$       |         |           |             |           |             |             |
| 0                    | $H_{00}$ | $\subset$ | $H_{01}$ | $\subset$ | $\dots$ | $\subset$ | $H_{0,k-1}$ | $\subset$ | $H_{0k}$    | $=$         |
| 1                    |          |           | $H_{10}$ | $\subset$ | $\dots$ | $\subset$ | $H_{1,k-2}$ | $\subset$ | $H_{1,k-1}$ | $=$         |
| $\vdots$             |          |           |          |           |         |           | $\vdots$    | $\vdots$  | $\vdots$    | $\vdots$    |
| $k-1$                |          |           |          |           |         |           | $H_{k-1,0}$ | $\subset$ | $H_{k-1,1}$ | $=$         |
|                      |          |           |          |           |         |           |             |           |             | $H_{k-1}^0$ |

Johansen (1995a) proposed the two-step procedure to analyze the  $I(2)$  model. In the first step, the values of  $(r, \alpha, \beta)$  are estimated using the reduced rank regression analysis, performing the regression analysis  $\Delta^2 \mathbf{y}_t$ ,  $\Delta \mathbf{y}_{t-1}$ , and  $\mathbf{y}_{t-1}$  on  $\Delta^2 \mathbf{y}_{t-1}, \dots, \Delta^2 \mathbf{y}_{t-p+2}, D_t$ . This gives residuals  $R_{0t}$ ,  $R_{1t}$ , and  $R_{2t}$  and residual product moment matrices

$$M_{ij} = \frac{1}{T} \sum_{t=1}^T R_{it} R'_{jt} \text{ for } i, j = 0, 1, 2$$

Perform the reduced rank regression analysis  $\Delta^2 \mathbf{y}_t$  on  $\mathbf{y}_{t-1}$  corrected for  $\Delta \mathbf{y}_{t-1}$ ,  $\Delta^2 \mathbf{y}_{t-1}, \dots, \Delta^2 \mathbf{y}_{t-p+2}, D_t$  and solve the eigenvalue problem of the equation

$$|\lambda M_{22.1} - M_{20.1} M_{00.1}^{-1} M_{02.1}| = 0$$

where  $M_{ij.1} = M_{ij} - M_{i1} M_{11}^{-1} M_{1j}$  for  $i, j = 0, 2$ .

In the second step, if  $(r, \alpha, \beta)$  are known, the values of  $(s, \xi, \eta)$  are determined using the reduced rank regression analysis, regressing  $\hat{\alpha}'_{\perp} \Delta^2 \mathbf{y}_t$  on  $\hat{\beta}'_{\perp} \Delta \mathbf{y}_{t-1}$  corrected for  $\Delta^2 \mathbf{y}_{t-1}, \dots, \Delta^2 \mathbf{y}_{t-p+2}, D_t$  and  $\hat{\beta}' \Delta \mathbf{y}_{t-1}$ .

The reduced rank regression analysis reduces to the solution of an eigenvalue problem for the equation

$$|\rho M_{\beta_{\perp} \beta_{\perp} \beta} - M_{\beta_{\perp} \alpha_{\perp} \beta} M_{\alpha_{\perp} \alpha_{\perp} \beta}^{-1} M_{\alpha_{\perp} \beta_{\perp} \beta}| = 0$$

where

$$\begin{aligned} M_{\beta_{\perp} \beta_{\perp} \beta} &= \beta'_{\perp} (M_{11} - M_{11} \beta (\beta' M_{11} \beta)^{-1} \beta' M_{11}) \beta_{\perp} \\ M'_{\beta_{\perp} \alpha_{\perp} \beta} &= M_{\alpha_{\perp} \beta_{\perp} \beta} = \bar{\alpha}'_{\perp} (M_{01} - M_{01} \beta (\beta' M_{11} \beta)^{-1} \beta' M_{11}) \beta_{\perp} \\ M_{\alpha_{\perp} \alpha_{\perp} \beta} &= \bar{\alpha}'_{\perp} (M_{00} - M_{01} \beta (\beta' M_{11} \beta)^{-1} \beta' M_{10}) \bar{\alpha}_{\perp} \end{aligned}$$

where  $\bar{\alpha} = \alpha(\alpha'\alpha)^{-1}$ .

The solution gives eigenvalues  $1 > \rho_1 > \dots > \rho_s > 0$  and eigenvectors  $(v_1, \dots, v_s)$ . Then, the ML estimators are

$$\begin{aligned}\hat{\eta} &= (v_1, \dots, v_s) \\ \hat{\xi} &= M_{\alpha_{\perp}\beta_{\perp}}\beta\hat{\eta}\end{aligned}$$

The likelihood ratio test for the reduced rank model  $H_{r,s}$  with rank  $\leq s$  in the model  $H_{r,k-r} = H_r^0$  is given by

$$Q_{r,s} = -T \sum_{i=s+1}^{k-r} \log(1 - \rho_i), \quad s = 0, \dots, k - r - 1$$

The following statements are to test the rank test for the cointegrated order 2:

```
proc varmax data=simul2;
  model y1 y2 / p=2 cointtest=(johansen=(iorder=2));
run;
```

| The VARMAX Procedure                      |           |           |          |                   |
|-------------------------------------------|-----------|-----------|----------|-------------------|
| Trace of Cointegration Rank Test for I(2) |           |           |          |                   |
| r\k-r-s                                   | 2         | 1         | Trace_I1 | Critical Value_I1 |
| 0                                         | 720.40735 | 308.69199 | 61.75    | 15.34             |
| 1                                         |           | 211.84512 | 0.56     | 3.84              |
| Critical Value_I2                         | 15.34000  | 3.84000   |          |                   |

**Figure 4.46.** Cointegrated I(2) Test (IORDER= option)

The last two columns in Figure 4.46 explain the cointegration rank test with integrated order 1. The results indicate that there is the cointegrated relationship with the cointegration rank 1 with respect to a 0.05 significance level. Now, look at the row in the case of  $r = 1$ . Compare the value to the critical value for the cointegrated order 2. There is no evidence that the series are integrated order 2 with a 0.05 significance level.

## OUT= Data Set

The OUT= data set contains the forecast values produced by the OUTPUT statement. The following output variables can be created:

- the BY variables
- the ID variable



- the MODEL statement endogenous (dependent) variables. These variables contain the actual values from the input data set.
- $FOR_i$ , numeric variables containing the forecasts. The  $FOR_i$  variables contain the forecasts for the  $i$ th endogenous variable in the MODEL statement list. Forecasts are 1-step-ahead predictions until the end of the data or until the observation specified by the BACK= option.
- $RES_i$ , numeric variables containing the residual for the forecast of the  $i$ th endogenous variable in the MODEL statement list. For forecast observations, the actual values are missing and the  $RES_i$  variables contain missing values.
- $STD_i$ , numeric variables containing the standard deviation for the forecast of the  $i$ th endogenous variable in the MODEL statement list. The values of the  $STD_i$  variables can be used to construct univariate confidence limits for the corresponding forecasts.
- $LCI_i$ , numeric variables containing the lower confidence limits for the corresponding forecasts of the  $i$ th endogenous variable in the MODEL statement list.
- $UCI_i$ , numeric variables containing the upper confidence limits for the corresponding forecasts of the  $i$ th endogenous variable in the MODEL statement list.

**Table 4.2.** OUT= Data Set

| Obs | ID variable | y1       | FOR1     | RES1     | STD1          | LCI1     | UCI1     |
|-----|-------------|----------|----------|----------|---------------|----------|----------|
| 1   | date        | $y_{11}$ | $f_{11}$ | $r_{11}$ | $\sigma_{11}$ | $l_{11}$ | $u_{11}$ |
| 2   | date        | $y_{12}$ | $f_{12}$ | $r_{12}$ | $\sigma_{11}$ | $l_{12}$ | $u_{12}$ |
| ⋮   |             |          |          |          |               |          |          |

| Obs | y2       | FOR2     | RES2     | STD2          | LCI2     | UCI2     |
|-----|----------|----------|----------|---------------|----------|----------|
| 1   | $y_{21}$ | $f_{21}$ | $r_{21}$ | $\sigma_{22}$ | $l_{21}$ | $u_{21}$ |
| 2   | $y_{22}$ | $f_{22}$ | $r_{22}$ | $\sigma_{22}$ | $l_{22}$ | $u_{22}$ |
| ⋮   |          |          |          |               |          |          |

The OUT= data set contains the values shown in Table 4.2 for a bivariate case.

Consider the following example:

```
proc varmax data=simul1 noprint;
  id date interval=year;
  model y1 y2 / p=1 noint;
  output out=out lead=5;
run;
proc print data=out(firstobs=98);
run;
```

| Obs | date     | y1       | FOR1     | RES1     | STD1     | LCI1     | UCI1     |
|-----|----------|----------|----------|----------|----------|----------|----------|
| 98  | 1997     | -0.58433 | -0.13500 | -0.44934 | 1.13523  | -2.36001 | 2.09002  |
| 99  | 1998     | -2.07170 | -1.00649 | -1.06522 | 1.13523  | -3.23150 | 1.21853  |
| 100 | 1999     | -3.38342 | -2.58612 | -0.79730 | 1.13523  | -4.81113 | -0.36111 |
| 101 | 2000     | .        | -3.59212 | .        | 1.13523  | -5.81713 | -1.36711 |
| 102 | 2001     | .        | -3.09448 | .        | 1.70915  | -6.44435 | 0.25539  |
| 103 | 2002     | .        | -2.17433 | .        | 2.14472  | -6.37792 | 2.02925  |
| 104 | 2003     | .        | -1.11395 | .        | 2.43166  | -5.87992 | 3.65203  |
| 105 | 2004     | .        | -0.14342 | .        | 2.58740  | -5.21463 | 4.92779  |
| Obs | y2       | FOR2     | RES2     | STD2     | LCI2     | UCI2     |          |
| 98  | 0.64397  | -0.34932 | 0.99329  | 1.19096  | -2.68357 | 1.98492  |          |
| 99  | 0.35925  | -0.07132 | 0.43057  | 1.19096  | -2.40557 | 2.26292  |          |
| 100 | -0.64999 | -0.99354 | 0.34355  | 1.19096  | -3.32779 | 1.34070  |          |
| 101 | .        | -2.09873 | .        | 1.19096  | -4.43298 | 0.23551  |          |
| 102 | .        | -2.77050 | .        | 1.47666  | -5.66469 | 0.12369  |          |
| 103 | .        | -2.75724 | .        | 1.74212  | -6.17173 | 0.65725  |          |
| 104 | .        | -2.24943 | .        | 2.01925  | -6.20709 | 1.70823  |          |
| 105 | .        | -1.47460 | .        | 2.25169  | -5.88782 | 2.93863  |          |

Figure 4.47. OUT= Data Set

The output in Figure 4.47 shows part of the results of the OUT= data set.

## OUTEST= Data Set

The OUTEST= data set contains estimation results of the fitted model. The following output variables can be created:

- the BY variables
- NAME, a character variable containing the name of endogenous (dependent) variables or the name of the parameters for the covariance of the matrix of the parameter estimates if the OUTCOV option is specified
- TYPE, a character variable containing the value EST for parameter estimates, the value STD for standard error of parameter estimates, and the value COV for the covariance of the matrix of the parameter estimates if the OUTCOV option is specified
- CONST, a numeric variable containing the estimates of constant parameters and their standard errors
- SD<sub>*i*</sub>, a numeric variable containing the estimates of seasonal dummy parameters and their standard errors, where  $i = 1, \dots, (nseason - 1)$
- XL<sub>*l*</sub><sub>*i*</sub>, numeric variables containing the estimates of exogenous parameters and their standard errors, where  $l$  is the lag  $l$ th coefficient matrix and  $i = 1, \dots, r$ ,  $r$  is the number of exogenous variables
- AR<sub>*l*</sub><sub>*i*</sub>, numeric variables containing the estimates of autoregressive parameters and their standard errors, where  $l$  is the lag  $l$ th coefficient matrix and  $i = 1, \dots, k$ ,  $k$  is the number of endogenous variables
- MA<sub>*l*</sub><sub>*i*</sub>, numeric variables containing the estimates of moving-average parameters and their standard errors, where  $l$  is the lag  $l$ th coefficient matrix and  $i = 1, \dots, k$ ,  $k$  is the number of endogenous variables

**Table 4.3.** OUTEST= Data Set

| Obs | NAME | TYPE | CONST          | AR1_1             | AR1_2             | AR2_1             | AR2_2             |
|-----|------|------|----------------|-------------------|-------------------|-------------------|-------------------|
| 1   | y1   | EST  | $\delta_1$     | $\phi_{1,11}$     | $\phi_{1,12}$     | $\phi_{2,11}$     | $\phi_{2,12}$     |
| 2   |      | STD  | $se(\delta_1)$ | $se(\phi_{1,11})$ | $se(\phi_{1,12})$ | $se(\phi_{2,11})$ | $se(\phi_{2,12})$ |
| 3   | y2   | EST  | $\delta_2$     | $\phi_{1,21}$     | $\phi_{1,22}$     | $\phi_{2,21}$     | $\phi_{2,22}$     |
| 4   |      | STD  | $se(\delta_2)$ | $se(\phi_{1,21})$ | $se(\phi_{1,22})$ | $se(\phi_{2,21})$ | $se(\phi_{2,22})$ |

The OUTEST= data set contains the values shown Table 4.3 for a bivariate case.

Consider the following example:

```
proc varmax data=simul2 outest=est;
    model y1 y2 / p=2 noint noprint
                ecm=(rank=1 normalize=y1);
run;
proc print data=est;
run;
```

| Obs | NAME | TYPE | AR1_1    | AR1_2    | AR2_1    | AR2_2    |
|-----|------|------|----------|----------|----------|----------|
| 1   | y1   | EST  | -0.46680 | 0.91295  | -0.74332 | -0.74621 |
| 2   |      | STD  | 0.04786  | 0.09359  | 0.04526  | 0.04769  |
| 3   | y2   | EST  | 0.10667  | -0.20862 | 0.40493  | -0.57157 |
| 4   |      | STD  | 0.05146  | 0.10064  | 0.04867  | 0.05128  |

**Figure 4.48.** OUTEST= Data Set

The output in Figure 4.48 shows the part of results of the OUTEST= data set.

---

## OUTSTAT= Data Set

The OUTSTAT= data set contains estimation results of the fitted model. The following output variables can be created. The sub-index  $i$  is  $1, \dots, k$ ,  $k$  is the number of endogenous variables.

- the BY variables
- NAME, a character variable containing the name of endogenous (dependent) variables
- SIGMA\_ $i$ , numeric variables containing the estimate covariance of the innovation covariance matrix
- AICC, a numeric variable containing the corrected Akaike's information criteria value
- RSquare, a numeric variable containing  $R^2$
- FValue, a numeric variable containing the  $F$  statistics
- PValue, a numeric variable containing  $p$ -value for the  $F$  statistics
- EValueI2\_ $i$ , numeric variables containing eigenvalues for the cointegration rank test of integrated order 2
- EValueI1, a numeric variable containing eigenvalues for the cointegration rank test of integrated order 1
- BETA\_ $i$ , numeric variables containing  $\beta$  long-run effect parameter estimates

- ALPHA\_ $i$ , numeric variables containing  $\alpha$  adjustment parameter estimates
- ETA\_ $i$ , numeric variables containing  $\eta$  parameter estimates in integrated order 2
- XI\_ $i$ , numeric variables containing  $\xi$  parameter estimates in integrated order 2

**Table 4.4.** OUTSTAT= Data Set

| Obs | NAME | SIGMA_1       | SIGMA_2       | AICC   | RSquare | FValue | PValue   |
|-----|------|---------------|---------------|--------|---------|--------|----------|
| 1   | y1   | $\sigma_{11}$ | $\sigma_{12}$ | $aicc$ | $R_1^2$ | $F_1$  | $prob_1$ |
| 2   | y2   | $\sigma_{21}$ | $\sigma_{22}$ | .      | $R_2^2$ | $F_2$  | $prob_2$ |

| Obs | EValueI2_1 | EValueI2_2 | EValueI1 | BETA_1       | BETA_2       |
|-----|------------|------------|----------|--------------|--------------|
| 1   | $e_{11}$   | $e_{12}$   | $e_1$    | $\beta_{11}$ | $\beta_{12}$ |
| 2   | $e_{21}$   | .          | $e_2$    | $\beta_{21}$ | $\beta_{21}$ |

| Obs | ALPHA_1       | ALPHA_2       | ETA_1       | ETA_2       | XI_1       | XI_2       |
|-----|---------------|---------------|-------------|-------------|------------|------------|
| 1   | $\alpha_{11}$ | $\alpha_{12}$ | $\eta_{11}$ | $\eta_{12}$ | $\xi_{11}$ | $\xi_{12}$ |
| 2   | $\alpha_{21}$ | $\alpha_{22}$ | $\eta_{21}$ | $\eta_{22}$ | $\xi_{21}$ | $\xi_{22}$ |

The OUTSTAT= data set contains the values shown Table 4.4 for a bivariate case.

Consider the following example:

```
proc varmax data=simul2 outstat=stat;
  model y1 y2 / p=2 noint
              cointtest=(johansen=(iorder=2))
              ecm=(rank=1 normalize=y1) noprint;
run;
proc print data=stat;
run;
```

| Obs | NAME | SIGMA_1  | SIGMA_2   | AICC    | RSquare | FValue | PValue |
|-----|------|----------|-----------|---------|---------|--------|--------|
| 1   | y1   | 94.75575 | 4.52684   | 9.37221 | 0.9391  | 482.78 | 0.0000 |
| 2   | y2   | 4.52684  | 109.57038 | .       | 0.9409  | 498.42 | 0.0000 |

| Obs | EValue<br>I2_1 | EValue<br>I2_2 | EValue<br>I1 | BETA_1   | BETA_2   | ALPHA_1  | ALPHA_2 |
|-----|----------------|----------------|--------------|----------|----------|----------|---------|
| 1   | 0.9849         | 0.9508         | 0.5086       | 1.00000  | 1.00000  | -0.46680 | 0.00794 |
| 2   | 0.8145         | .              | 0.0111       | -1.95575 | -1.33622 | 0.10667  | 0.03353 |

| Obs | ETA_1    | ETA_2   | XI_1      | XI_2      |
|-----|----------|---------|-----------|-----------|
| 1   | -0.01231 | 0.02703 | 54.16057  | -52.31444 |
| 2   | 0.01555  | 0.02309 | -79.42404 | -18.33083 |

**Figure 4.49.** OUTSTAT= Data Set

The output in Figure 4.49 shows the part of results of the OUTSTAT= data set.

---

## Printed Output

The default printed output produced by the VARMAX procedure is described in the following list:

- descriptive statistics, which include the number of observations used, the names of the variables, their means and standard deviations (STD), their minimums and maximums, the differencing operations used, and the labels of the variables
- a type of model to fit the data and an estimate method
- the estimates of the constant vector (or seasonal constant matrix), the trend vector, the coefficients matrices of the distributed lags, the AR coefficients matrices, and the MA coefficients matrices
- a table of parameter estimates showing the following for each parameter: the variable name for the left-hand side of equations, the parameter name, the parameter estimate, the approximate standard error,  $t$  value, the approximate probability ( $Pr > |t|$ ), and the variable name for the right-hand side of equations in terms of each parameter
- the innovation covariance matrix
- the Information criteria
- the cross-covariance and cross-correlation matrices of the residuals and tables of test statistics for the hypothesis that the residuals of the model are white noise:
  - Durbin-Watson (DW) statistics
  - $F$  test for autoregressive conditional heteroscedastic (ARCH) disturbances
  - $F$  test for AR disturbance
  - Jarque-Bera normality test
  - Portmanteau test

## ODS Table Names

The VARMAX procedure assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table:

**Table 4.5.** ODS Tables Produced in the VARMAX Procedure

| ODS Table Name                                   | Description                                                  | Option                           |
|--------------------------------------------------|--------------------------------------------------------------|----------------------------------|
| <b>ODS Tables Created by the MODEL Statement</b> |                                                              |                                  |
| AccumImpulse                                     | Accumulated Impulse Response Matrices                        | IMPULSE=(ACCUM)<br>IMPULSE=(ALL) |
| AccumImpulsX                                     | Accumulated Transfer Function Matrices                       | IMPULSX=(ACCUM)<br>IMPULSX=(ALL) |
| Alpha                                            | $\alpha$ Coefficients                                        | JOHANSEN=                        |
| AlphaInECM                                       | $\alpha$ Coefficients                                        | ECM=                             |
| AlphaOnDrift                                     | $\alpha$ Coefficients on Restriction of a Deterministic Term | JOHANSEN=                        |
| AlphaBetaInECM                                   | $\pi = \alpha\beta'$ Coefficients                            | ECM=                             |
| ARCoef                                           | AR Coefficients                                              | P=                               |
| ARRoots                                          | Roots of AR Characteristic Polynomial                        | ROOTS                            |
| Beta                                             | $\beta$ Coefficients                                         | JOHANSEN=                        |
| BetaInECM                                        | $\beta$ Coefficients                                         | ECM=                             |
| BetaOnDrift                                      | $\beta$ Coefficients on Restriction of a Deterministic Term  | JOHANSEN=                        |
| Constant                                         | Constant Estimates                                           | default                          |
| CorrB                                            | Correlations of Parameter Estimates                          | CORRB                            |
| CorrResiduals                                    | Cross-Correlations of Residuals                              | default                          |
| CorrResidualsGraph                               | Schematic Representation of Residual Cross-Correlations      | default                          |
| CorrGraph                                        | Schematic Representation of Sample Cross-Correlations        | CORRX CORRY                      |
| CorrXLags                                        | Cross-Correlation Matrices of Independent Series             | CORRX                            |
| CorrYLags                                        | Cross-Correlation Matrices of Dependent Series               | CORRY                            |
| CovB                                             | Covariance of Parameter Estimates                            | COVB                             |
| CovInnov                                         | Covariance Matrix for the Innovation                         | default                          |
| CovPredError                                     | Covariance Matrices of the Prediction Error                  | COVPE                            |
| CovResiduals                                     | Cross-Covariance Matrices of Residuals                       | default                          |
| CovXLags                                         | Cross-Covariance Matrices of Independent Series              | COVX                             |
| CovYLags                                         | Cross-Covariance Matrices of Dependent Series                | COVY                             |

**Table 4.5.** (ODS Tables Continued)

| ODS Table Name      | Description                                                                                 | Option                                       |
|---------------------|---------------------------------------------------------------------------------------------|----------------------------------------------|
| DecompCovPredError  | Decomposition of the Prediction Error Covariance                                            | DECOMPOSE                                    |
| DFTest              | Dickey-Fuller Tests                                                                         | DFTEST                                       |
| EigenvalueI2        | Eigenvalues in Integrated Order 2                                                           | JOHANSEN=(IORDER=2)                          |
| Eta                 | $\eta$ Coefficients                                                                         | JOHANSEN=(IORDER=2)                          |
| DriftHypo           | Hypothesis of Different Deterministic Terms in Cointegration Rank Test                      | JOHANSEN=                                    |
| DriftHypoTest       | Test Hypothesis of Different Deterministic Terms in Cointegration Rank Test                 | JOHANSEN=                                    |
| InfiniteARRepresent | Infinite Order AR Representation                                                            | IARR                                         |
| InfoCriterion       | Information criterion                                                                       | default                                      |
| LinearTrend         | Linear Trend Estimates                                                                      | TREND=                                       |
| MaxTest             | Cointegration Rank Test Using the Maximum Eigenvalue                                        | JOHANSEN=(TYPE=MAX)                          |
| MaxTestOnDrift      | Cointegration Rank Test Using the Maximum Eigenvalue on Restriction of a Deterministic Term | JOHANSEN=(TYPE=MAX)                          |
| ModelType           | Type of Model                                                                               | default                                      |
| NObs                | Number of Observations                                                                      | default                                      |
| OrthoImpulse        | Orthogonalized Impulse Response Matrices                                                    | IMPULSE=(ORTH)<br>IMPULSE=(ALL)              |
| ParameterEstimates  | Parameter Estimates Table                                                                   | default                                      |
| PartialAR           | Partial Autoregression Matrices                                                             | PARCOEF                                      |
| PartialARGraph      | Schematic Representation of Partial Autoregression                                          | PARCOEF                                      |
| PartialCanCorr      | Partial Canonical Correlation Analysis                                                      | PCANCORR                                     |
| PartialCorr         | Partial Cross-Correlation Matrices                                                          | PCORR                                        |
| PartialCorrGraph    | Schematic Representation of Partial Cross-Correlations                                      | PCORR                                        |
| PortmanteauTest     | Chi-Square Test Table for Residual Cross-Correlations                                       | default                                      |
| ProportionDecomp    | Proportions of Prediction Error Covariance Decomposition                                    | DECOMPOSE                                    |
| RankTestI2          | Cointegration Rank Test in Integrated Order 2                                               | JOHANSEN=(IORDER=2)                          |
| QuadTrend           | Quadratic Trend Estimates                                                                   | TREND=QUAD                                   |
| SConstant           | Seasonal Constant Estimates                                                                 | NSEASON=                                     |
| SimpleImpulse       | Impulse Response Matrices                                                                   | IMPULSE<br>IMPULSE=(SIMPLE)<br>IMPULSE=(ALL) |
| SimpleImpulsX       | Impulse Response Matrices in Transfer Function                                              | IMPULSX<br>IMPULSX=(SIMPLE)<br>IMPULSX=(ALL) |

**Table 4.5.** (ODS Tables Continued)

| ODS Table Name                                      | Description                                                                    | Option       |
|-----------------------------------------------------|--------------------------------------------------------------------------------|--------------|
| Summary                                             | Simple Summary Statistics                                                      | default      |
| SWTest                                              | Common Trends Test                                                             | SW SW=       |
| TentativeOrders                                     | Tentative Order Selection                                                      | MINIC MINIC= |
| TraceTest                                           | Cointegration Rank Test Using the Trace                                        | JOHANSEN=    |
|                                                     |                                                                                | (TYPE=TRACE) |
| TraceTestOnDrift                                    | Cointegration Rank Test Using the Trace on Restriction of a Deterministic Term | JOHANSEN=    |
|                                                     |                                                                                | (TYPE=TRACE) |
| UnivarDiagnostAR                                    | Check the AR Disturbance for the residuals                                     | default      |
| UnivarDiagnostCheck                                 | Univariate Model Diagnostic Checks                                             | default      |
| UnivarDiagnostTest                                  | Check the ARCH Disturbance and Normality for the residuals                     | default      |
| Xi                                                  | $\xi$ Coefficient Matrix                                                       | JOHANSEN=    |
|                                                     |                                                                                | (IORDER=2)   |
| XLagCoef                                            | Dependent Coefficients                                                         | XLAG=        |
| YWEstimates                                         | Yule-Walker Estimates                                                          | YW           |
| *ByVariable                                         | Prints by Variable                                                             | PRINTFORM=   |
| <b>ODS Tables Created by the COINTEG Statement</b>  |                                                                                |              |
| AlphaInECM                                          | $\alpha$ Coefficients                                                          | default      |
| AlphaBetaInECM                                      | $\pi = \alpha\beta'$ Coefficients                                              | default      |
| BetaInECM                                           | $\beta$ Coefficients                                                           | default      |
| AlphaOnTest                                         | $\alpha$ Coefficients under Restriction                                        | H= J=        |
| BetaOnTest                                          | $\beta$ Coefficients under Restriction                                         | H= J=        |
| RestrictMatrix                                      | Restriction Matrix for $\alpha$ or $\beta$                                     | H= J=        |
| RestrictTest                                        | Hypothesis Testing of $\alpha$ or $\beta$                                      | H= J=        |
| WeakExogeneity                                      | Testing Weak Exogeneity of each Dependent Variable with respect to BETA        | EXOGENEITY   |
| <b>ODS Tables Created by the CAUSAL Statement</b>   |                                                                                |              |
| Causality                                           | Granger-Causality Test                                                         | default      |
| <b>ODS Tables Created by the RESTRICT Statement</b> |                                                                                |              |
| Restrict                                            | Restriction table                                                              | default      |
| <b>ODS Tables Created by the TEST Statement</b>     |                                                                                |              |
| Test                                                | Wald test                                                                      | default      |
| <b>ODS Tables Created by the OUTPUT Statement</b>   |                                                                                |              |
| Forecasts                                           | Forecasts Table                                                                | PRINT        |



## *Part 2. Procedure Reference*

Note that the symbol \* corresponds to AccumImpulse, AccumImpulsX, CorrResiduals, CorrXLags, CorrYLags, CovResiduals, CovXLags, CovYLags, OrthoImpulse, PartialCorr, PredictMSE, DecompCovPredError, ProportionDecomp, SimpleImpulse, and SimpleImpulsX.

---

## Examples

---

### Example 4.1. Analysis of Real Output Series

---

Consider the following four-dimensional system of U.S. economic variables. Quarterly data for the years 1954 to 1987 are used (Lütkepohl 1993, Table E.3.).

```

symbol1 v=none height=1 c=black;
symbol2 v=none height=1 c=black;

title 'Analysis of US Economic Variables';
data us_money;
    date=intnx( 'qtr', '01jan54'd, _n_-1 );
    format date yyq. ;
    input y1 y2 y3 y4 @@;
    y1=log(y1);
    y2=log(y2);
    label y1='log(real money stock M1)'
          y2='log(GNP in bil. of 1982 dollars)'
          y3='Discount rate on 91-day T-bills'
          y4='Yield on 20-year Treasury bonds';
    datalines;
    ... data lines omitted ...
;

legend1 across=1 frame label=none;

proc gplot data=us_money;
    symbol1 i = join l = 1;
    symbol2 i = join l = 2;
    axis2 label = (a=-90 r=90 " ");
    plot y1 * date = 1 y2 * date = 2 /
        overlay vaxis=axis2 legend=legend1;
run;

proc gplot data=us_money;
    symbol1 i = join l = 1;
    symbol2 i = join l = 2;
    axis2 label = (a=-90 r=90 " ");
    plot y3 * date = 1 y4 * date = 2 /
        overlay vaxis=axis2 legend=legend1;
run;

proc varmax data=us_money;
    id date interval=qtr;
    model y1-y4 / p=2 lagmax=6 dftest
        print=(iarr)
        cointtest=(johansen=(iorder=2))
        ecm=(rank=1 normalize=y1);
    cointeg rank=1 normalize=y1 exogeneity;
run;
quit;
```

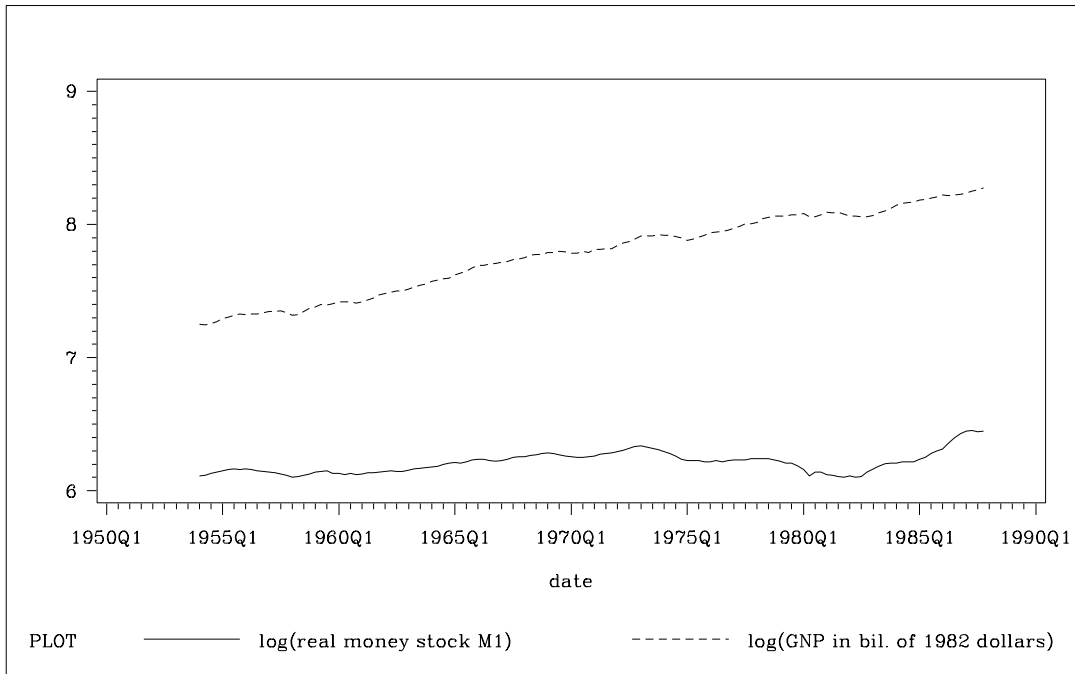


**Figure 4.50.** Plot of Data

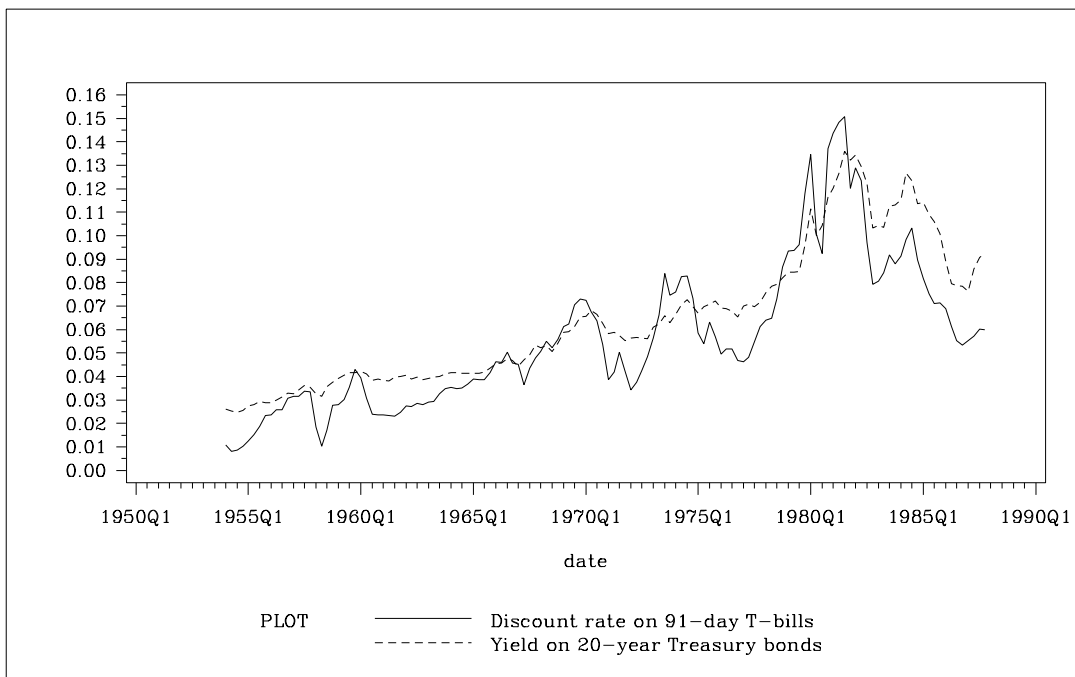


**Figure 4.51.** Plot of Data Continued

The VARMAX procedure output is shown in Output 4.1.1 through Output 4.1.10.

Output 4.1.1. Descriptive Statistics

| The VARMAX Procedure       |      |          |                                  |         |         |         |
|----------------------------|------|----------|----------------------------------|---------|---------|---------|
| Number of Observations     |      |          | 136                              |         |         |         |
| Number of Pairwise Missing |      |          | 0                                |         |         |         |
| Variable                   | Type | NoMissN  | Mean                             | StdDev  | Min     | Max     |
| y1                         | DEP  | 136      | 6.21295                          | 0.07924 | 6.10278 | 6.45331 |
| y2                         | DEP  | 136      | 7.77890                          | 0.30110 | 7.24508 | 8.27461 |
| y3                         | DEP  | 136      | 0.05608                          | 0.03109 | 0.00813 | 0.15087 |
| y4                         | DEP  | 136      | 0.06458                          | 0.02927 | 0.02490 | 0.13600 |
|                            |      | Variable | Label                            |         |         |         |
|                            |      | y1       | log(real money stock M1)         |         |         |         |
|                            |      | y2       | log(GNP in bil. of 1982 dollars) |         |         |         |
|                            |      | y3       | Discount rate on 91-day T-bills  |         |         |         |
|                            |      | y4       | Yield on 20-year Treasury bonds  |         |         |         |

Output 4.1.2. Unit Root Tests

| The VARMAX Procedure          |             |          |          |       |          |
|-------------------------------|-------------|----------|----------|-------|----------|
| Dickey-Fuller Unit Root Tests |             |          |          |       |          |
| Variable                      | Type        | Rho      | Prob<Rho | Tau   | Prob<Tau |
| y1                            | Zero Mean   | 0.051968 | 0.6934   | 1.14  | 0.9343   |
|                               | Single Mean | -2.96706 | 0.6572   | -0.76 | 0.8260   |
|                               | Trend       | -5.90836 | 0.7454   | -1.34 | 0.8725   |
| y2                            | Zero Mean   | 0.132569 | 0.7124   | 5.14  | 0.9999   |
|                               | Single Mean | -0.43009 | 0.9309   | -0.79 | 0.8176   |
|                               | Trend       | -9.20862 | 0.4787   | -2.16 | 0.5063   |
| y3                            | Zero Mean   | -1.28445 | 0.4255   | -0.69 | 0.4182   |
|                               | Single Mean | -8.85767 | 0.1700   | -2.27 | 0.1842   |
|                               | Trend       | -18.9729 | 0.0742   | -2.86 | 0.1803   |
| y4                            | Zero Mean   | 0.403766 | 0.7803   | 0.45  | 0.8100   |
|                               | Single Mean | -2.79092 | 0.6790   | -1.29 | 0.6328   |
|                               | Trend       | -12.1185 | 0.2923   | -2.33 | 0.4170   |

Output 4.1.3. Cointegration Rank Test

| The VARMAX Procedure                            |           |             |             |             |               |
|-------------------------------------------------|-----------|-------------|-------------|-------------|---------------|
| Eigenvalues of Cointegration Rank Test for I(2) |           |             |             |             |               |
| r\s                                             | 0         | 1           | 2           | 3           | Eigenvalue_I1 |
| 0                                               | 0.71927   | 0.54812     | 0.41092     | 0.24142     | 0.231643      |
| 1                                               | 0.62213   | 0.36989     | 0.18448     |             | 0.125740      |
| 2                                               | 0.31902   | 0.15225     |             |             | 0.019455      |
| 3                                               | 0.24857   |             |             |             | 0.000111      |
| Long-Run Parameter BETA Estimates               |           |             |             |             |               |
| Variable                                        | Dummy 1   | Dummy 2     | Dummy 3     | Dummy 4     |               |
| y1                                              | 1.00000   | 1.00000     | 1.00000     | 1.00000     |               |
| y2                                              | -0.46458  | -0.63174    | -0.69996    | -0.16140    |               |
| y3                                              | 14.51619  | -1.29864    | 1.37007     | -0.61806    |               |
| y4                                              | -9.35520  | 7.53672     | 2.47901     | 1.43731     |               |
| Adjustment Coefficient ALPHA Estimates          |           |             |             |             |               |
| Variable                                        | Dummy 1   | Dummy 2     | Dummy 3     | Dummy 4     |               |
| y1                                              | -0.01396  | 0.01396     | -0.01119    | 0.00007569  |               |
| y2                                              | -0.02811  | -0.02739    | -0.00031751 | 0.00076370  |               |
| y3                                              | -0.00215  | -0.04967    | -0.00183    | -0.00072255 |               |
| y4                                              | 0.00510   | -0.02514    | -0.00220    | 0.00015940  |               |
| Parameter ETA Estimates                         |           |             |             |             |               |
| Variable                                        | Dummy 1   | Dummy 2     | Dummy 3     | Dummy 4     |               |
| y1                                              | 52.74907  | 41.74502    | -20.80403   | 55.77415    |               |
| y2                                              | -49.10609 | -9.40081    | 98.87199    | 22.56416    |               |
| y3                                              | 68.29674  | -144.83173  | -27.35953   | 15.51142    |               |
| y4                                              | 121.25932 | 271.80496   | 85.85156    | -130.11599  |               |
| Parameter XI Estimates                          |           |             |             |             |               |
| Variable                                        | Dummy 1   | Dummy 2     | Dummy 3     | Dummy 4     |               |
| y1                                              | -0.00842  | -0.00052033 | -0.00208    | -0.00250    |               |
| y2                                              | 0.00141   | 0.00213     | -0.00736    | -0.00057797 |               |
| y3                                              | -0.00445  | 0.00541     | -0.00150    | 0.00310     |               |
| y4                                              | -0.00211  | -0.00063722 | -0.00130    | 0.00197     |               |

Output 4.1.4. Parameter Estimates

| The VARMAX Procedure                                        |                              |
|-------------------------------------------------------------|------------------------------|
| Type of Model                                               | VECM(2)                      |
| Estimation Method                                           | Method of Moments Estimation |
| Cointegrated Rank                                           | 1                            |
| Long-Run Parameter<br>BETA Estimates<br>given RANK = 1      |                              |
| Variable                                                    | Dummy 1                      |
| y1                                                          | 1.00000                      |
| y2                                                          | -0.46458                     |
| y3                                                          | 14.51619                     |
| y4                                                          | -9.35520                     |
| Adjustment Coefficient<br>ALPHA Estimates<br>given RANK = 1 |                              |
| Variable                                                    | Dummy 1                      |
| y1                                                          | -0.01396                     |
| y2                                                          | -0.02811                     |
| y3                                                          | -0.00215                     |
| y4                                                          | 0.00510                      |

Output 4.1.5. Parameter Estimates Continued

| The VARMAX Procedure              |          |          |            |          |          |
|-----------------------------------|----------|----------|------------|----------|----------|
| Constant Estimates                |          |          |            |          |          |
| Variable                          |          | Constant |            |          |          |
| y1                                |          | 0.04076  |            |          |          |
| y2                                |          | 0.08595  |            |          |          |
| y3                                |          | 0.00518  |            |          |          |
| y4                                |          | -0.01438 |            |          |          |
| Parameter ALPHA * BETA' Estimates |          |          |            |          |          |
| Variable                          |          | y1       | y2         | y3       | y4       |
| y1                                |          | -0.01396 | 0.00648    | -0.20263 | 0.13059  |
| y2                                |          | -0.02811 | 0.01306    | -0.40799 | 0.26294  |
| y3                                |          | -0.00215 | 0.00099887 | -0.03121 | 0.02011  |
| y4                                |          | 0.00510  | -0.00237   | 0.07407  | -0.04774 |
| AR Coefficient Estimates          |          |          |            |          |          |
| DIF_Lag                           | Variable | y1       | y2         | y3       | y4       |
| 1                                 | y1       | 0.34603  | 0.09131    | -0.35351 | -0.96895 |
|                                   | y2       | 0.09936  | 0.03791    | 0.23900  | 0.28661  |
|                                   | y3       | 0.18118  | 0.07859    | 0.02234  | 0.40508  |
|                                   | y4       | 0.03222  | 0.04961    | -0.03292 | 0.18568  |

Output 4.1.6. Parameter Estimates Continued

| The VARMAX Procedure      |           |            |           |         |         |           |
|---------------------------|-----------|------------|-----------|---------|---------|-----------|
| Model Parameter Estimates |           |            |           |         |         |           |
| Equation                  | Parameter | Estimate   | Std Error | T Ratio | Prob> T | Variable  |
| D_y1(t)                   | CONST1    | 0.04076    | 0.01418   | 2.87    | 0.0048  |           |
|                           | AR1_1_1   | -0.01396   | 0.00495   | -2.82   | 0.0056  | y1(t-1)   |
|                           | AR1_1_2   | 0.00648    | 0.00230   | 2.82    | 0.0056  | y2(t-1)   |
|                           | AR1_1_3   | -0.20263   | 0.07191   | -2.82   | 0.0056  | y3(t-1)   |
|                           | AR1_1_4   | 0.13059    | 0.04634   | 2.82    | 0.0056  | y4(t-1)   |
|                           | AR2_1_1   | 0.34603    | 0.06414   | 5.39    | 0.0001  | D_y1(t-1) |
|                           | AR2_1_2   | 0.09131    | 0.07334   | 1.25    | 0.2154  | D_y2(t-1) |
|                           | AR2_1_3   | -0.35351   | 0.11024   | -3.21   | 0.0017  | D_y3(t-1) |
| D_y2(t)                   | AR2_1_4   | -0.96895   | 0.20737   | -4.67   | 0.0001  | D_y4(t-1) |
|                           | CONST2    | 0.08595    | 0.01679   | 5.12    | 0.0001  |           |
|                           | AR1_2_1   | -0.02811   | 0.00586   | -4.79   | 0.0001  | y1(t-1)   |
|                           | AR1_2_2   | 0.01306    | 0.00272   | 4.79    | 0.0001  | y2(t-1)   |
|                           | AR1_2_3   | -0.40799   | 0.08514   | -4.79   | 0.0001  | y3(t-1)   |
|                           | AR1_2_4   | 0.26294    | 0.05487   | 4.79    | 0.0001  | y4(t-1)   |
|                           | AR2_2_1   | 0.09936    | 0.07594   | 1.31    | 0.1932  | D_y1(t-1) |
|                           | AR2_2_2   | 0.03791    | 0.08683   | 0.44    | 0.6632  | D_y2(t-1) |
| D_y3(t)                   | AR2_2_3   | 0.23900    | 0.13052   | 1.83    | 0.0695  | D_y3(t-1) |
|                           | AR2_2_4   | 0.28661    | 0.24552   | 1.17    | 0.2453  | D_y4(t-1) |
|                           | CONST3    | 0.00518    | 0.01608   | 0.32    | 0.7476  |           |
|                           | AR1_3_1   | -0.00215   | 0.00562   | -0.38   | 0.7024  | y1(t-1)   |
|                           | AR1_3_2   | 0.00099887 | 0.00261   | 0.38    | 0.7024  | y2(t-1)   |
|                           | AR1_3_3   | -0.03121   | 0.08151   | -0.38   | 0.7024  | y3(t-1)   |
|                           | AR1_3_4   | 0.02011    | 0.05253   | 0.38    | 0.7024  | y4(t-1)   |
|                           | AR2_3_1   | 0.18118    | 0.07271   | 2.49    | 0.0140  | D_y1(t-1) |
| D_y4(t)                   | AR2_3_2   | 0.07859    | 0.08313   | 0.95    | 0.3463  | D_y2(t-1) |
|                           | AR2_3_3   | 0.02234    | 0.12496   | 0.18    | 0.8584  | D_y3(t-1) |
|                           | AR2_3_4   | 0.40508    | 0.23506   | 1.72    | 0.0873  | D_y4(t-1) |
|                           | CONST4    | -0.01438   | 0.00803   | -1.79   | 0.0758  |           |
|                           | AR1_4_1   | 0.00510    | 0.00281   | 1.82    | 0.0713  | y1(t-1)   |
|                           | AR1_4_2   | -0.00237   | 0.00130   | -1.82   | 0.0713  | y2(t-1)   |
|                           | AR1_4_3   | 0.07407    | 0.04072   | 1.82    | 0.0713  | y3(t-1)   |
|                           | AR1_4_4   | -0.04774   | 0.02624   | -1.82   | 0.0713  | y4(t-1)   |
|                           | AR2_4_1   | 0.03222    | 0.03632   | 0.89    | 0.3768  | D_y1(t-1) |
|                           | AR2_4_2   | 0.04961    | 0.04153   | 1.19    | 0.2345  | D_y2(t-1) |
|                           | AR2_4_3   | -0.03292   | 0.06243   | -0.53   | 0.5990  | D_y3(t-1) |
|                           | AR2_4_4   | 0.18568    | 0.11744   | 1.58    | 0.1164  | D_y4(t-1) |

## Output 4.1.7. Diagnostic Checks

| The VARMAX Procedure                 |             |            |             |             |
|--------------------------------------|-------------|------------|-------------|-------------|
| Covariance Matrix for the Innovation |             |            |             |             |
| Variable                             | y1          | y2         | y3          | y4          |
| y1                                   | 0.00005072  | 0.00001486 | -0.00000803 | -0.00000326 |
| y2                                   | 0.00001486  | 0.00007110 | 0.00001636  | 0.00001008  |
| y3                                   | -0.00000803 | 0.00001636 | 0.00006517  | 0.00002296  |
| y4                                   | -0.00000326 | 0.00001008 | 0.00002296  | 0.00001627  |

| Information Criteria                   |          |
|----------------------------------------|----------|
| AICC(Corrected AIC)                    | -40.6284 |
| HQC(Hannan-Quinn Criterion)            | -40.4343 |
| AIC(Akaike Information Criterion)      | -40.6452 |
| SBC(Schwarz Bayesian Criterion)        | -40.1262 |
| FPEC(Final Prediction Error Criterion) | 2.23E-18 |

| Schematic Representation of Residual Cross Correlations |      |      |      |      |      |      |      |
|---------------------------------------------------------|------|------|------|------|------|------|------|
| Variable/<br>Lag                                        | 0    | 1    | 2    | 3    | 4    | 5    | 6    |
| y1                                                      | ++.. | .... | ++.. | .... | +... | ..-- | .... |
| y2                                                      | ++++ | .... | .... | .... | .... | .... | .... |
| y3                                                      | .+++ | .... | +..- | ..++ | -... | .... | .... |
| y4                                                      | .+++ | .... | .... | ..+  | .... | .... | .... |

+ is > 2\*std error, - is < -2\*std error, . is between

| Portmanteau Test for Residual<br>Cross Correlations |                |    |                |
|-----------------------------------------------------|----------------|----|----------------|
| To<br>Lag                                           | Chi-<br>Square | DF | Prob><br>ChiSq |
| 3                                                   | 53.90          | 16 | <.0001         |
| 4                                                   | 74.03          | 32 | <.0001         |
| 5                                                   | 103.08         | 48 | <.0001         |
| 6                                                   | 116.94         | 64 | <.0001         |



**Output 4.1.8.** Diagnostic Checks Continued

| The VARMAX Procedure               |          |                    |                |                  |         |        |         |        |
|------------------------------------|----------|--------------------|----------------|------------------|---------|--------|---------|--------|
| Univariate Model Diagnostic Checks |          |                    |                |                  |         |        |         |        |
| Variable                           | R-square | StdDev             | F Value        | Prob>F           |         |        |         |        |
| y1                                 | 0.6754   | 0.0071             | 32.51          | <.0001           |         |        |         |        |
| y2                                 | 0.3070   | 0.0084             | 6.92           | <.0001           |         |        |         |        |
| y3                                 | 0.1328   | 0.0081             | 2.39           | 0.0196           |         |        |         |        |
| y4                                 | 0.0831   | 0.0040             | 1.42           | 0.1963           |         |        |         |        |
| Univariate Model Diagnostic Checks |          |                    |                |                  |         |        |         |        |
| Variable                           | DW(1)    | Normality<br>ChiSq | Prob><br>ChiSq | ARCH1<br>F Value | Prob>F  |        |         |        |
| y1                                 | 2.13     | 7.19               | 0.0275         | 1.62             | 0.2053  |        |         |        |
| y2                                 | 2.04     | 1.20               | 0.5483         | 1.23             | 0.2697  |        |         |        |
| y3                                 | 1.87     | 253.76             | <.0001         | 1.78             | 0.1847  |        |         |        |
| y4                                 | 1.98     | 105.21             | <.0001         | 21.01            | <.0001  |        |         |        |
| Univariate Model Diagnostic Checks |          |                    |                |                  |         |        |         |        |
| Variable                           | AR1      |                    | AR1-2          |                  | AR1-3   |        | AR1-4   |        |
|                                    | F Value  | Prob>F             | F Value        | Prob>F           | F Value | Prob>F | F Value | Prob>F |
| y1                                 | 0.68     | 0.4126             | 2.98           | 0.0542           | 2.01    | 0.1154 | 2.48    | 0.0473 |
| y2                                 | 0.05     | 0.8185             | 0.12           | 0.8842           | 0.41    | 0.7453 | 0.30    | 0.8762 |
| y3                                 | 0.56     | 0.4547             | 2.86           | 0.0610           | 4.83    | 0.0032 | 3.71    | 0.0069 |
| y4                                 | 0.01     | 0.9340             | 0.16           | 0.8559           | 1.21    | 0.3103 | 0.95    | 0.4358 |

**Output 4.1.9.** Infinite Order AR Representation

| The VARMAX Procedure             |          |          |          |          |          |
|----------------------------------|----------|----------|----------|----------|----------|
| Infinite Order AR Representation |          |          |          |          |          |
| Lag                              | Variable | y1       | y2       | y3       | y4       |
| 1                                | y1       | 1.33208  | 0.09780  | -0.55614 | -0.83836 |
|                                  | y2       | 0.07125  | 1.05096  | -0.16899 | 0.54955  |
|                                  | y3       | 0.17903  | 0.07959  | 0.99113  | 0.42520  |
|                                  | y4       | 0.03732  | 0.04724  | 0.04116  | 1.13795  |
| 2                                | y1       | -0.34603 | -0.09131 | 0.35351  | 0.96895  |
|                                  | y2       | -0.09936 | -0.03791 | -0.23900 | -0.28661 |
|                                  | y3       | -0.18118 | -0.07859 | -0.02234 | -0.40508 |
|                                  | y4       | -0.03222 | -0.04961 | 0.03292  | -0.18568 |

**Output 4.1.10.** Weak Exogeneity Test

| The VARMAX Procedure                             |            |    |            |  |
|--------------------------------------------------|------------|----|------------|--|
| Tests of Weak Exogeneity<br>of Each of Variables |            |    |            |  |
| Variable                                         | Chi-Square | DF | Prob>ChiSq |  |
| y1                                               | 6.55       | 1  | 0.0105     |  |
| y2                                               | 12.54      | 1  | 0.0004     |  |
| y3                                               | 0.09       | 1  | 0.7695     |  |
| y4                                               | 1.81       | 1  | 0.1786     |  |

## Example 4.2. Analysis of Real Output Series

This example considers a three-dimensional VAR(2) model. The model contains the logarithms of quarterly seasonally adjusted West German fixed investment, disposable income, and consumption expenditures. The data used are in Lütkepohl (1993, Table E.1).

```

data west;
  date = intnx( 'qtr', '01jan60'd, _n_-1 );
  format date yyq. ;
  input y1 y2 y3 @@;
  y1 = log(y1);
  y2 = log(y2);
  y3 = log(y3);
  label y1 = 'logarithm of investment'
        y2 = 'logarithm of income'
        y3 = 'logarithm of consumption';

  datalines;
  ... data lines omitted ...
;

data use;
  set west;
  where date < '01jan79'd;
  keep date y1 y2 y3;

proc varmax data=use;
  id date interval=qtr align=E;
  model y1-y3 / p=2 dify=(1)
           print=(decompose(6) impulse=(stderr))
           printform=both lagmax=3;
  causal group1=(y1) group2=(y2 y3);
  output lead=5;
run;
quit;

```

The VARMAX procedure output is shown in Output 4.2.1 through Output 4.2.7.

**Output 4.2.1.** Descriptive Statistics

| The VARMAX Procedure |                          |                                           |         |         |          |         |            |
|----------------------|--------------------------|-------------------------------------------|---------|---------|----------|---------|------------|
|                      |                          | Number of Observations                    |         | 75      |          |         |            |
|                      |                          | Number of Pairwise Missing                |         | 0       |          |         |            |
|                      |                          | Observation(s) eliminated by differencing |         | 1       |          |         |            |
| Variable             | Type                     | NoMissN                                   | Mean    | StdDev  | Min      | Max     | Difference |
| y1                   | DEP                      | 75                                        | 0.01811 | 0.04680 | -0.14018 | 0.19358 | 1          |
| y2                   | DEP                      | 75                                        | 0.02071 | 0.01208 | -0.02888 | 0.05023 | 1          |
| y3                   | DEP                      | 75                                        | 0.01987 | 0.01040 | -0.01300 | 0.04483 | 1          |
| Variable Label       |                          |                                           |         |         |          |         |            |
| y1                   | logarithm of investment  |                                           |         |         |          |         |            |
| y2                   | logarithm of income      |                                           |         |         |          |         |            |
| y3                   | logarithm of consumption |                                           |         |         |          |         |            |

## Output 4.2.2. Parameter Estimates

| The VARMAX Procedure      |           |          |                          |          |         |          |
|---------------------------|-----------|----------|--------------------------|----------|---------|----------|
| Type of Model             |           |          | VAR(2)                   |          |         |          |
| Estimation Method         |           |          | Least Squares Estimation |          |         |          |
| Constant Estimates        |           |          |                          |          |         |          |
| Variable                  |           | Constant |                          |          |         |          |
| y1                        |           | -0.01672 |                          |          |         |          |
| y2                        |           | 0.01577  |                          |          |         |          |
| y3                        |           | 0.01293  |                          |          |         |          |
| AR Coefficient Estimates  |           |          |                          |          |         |          |
| Lag                       | Variable  | y1       | y2                       | y3       |         |          |
| 1                         | y1        | -0.31963 | 0.14599                  | 0.96122  |         |          |
|                           | y2        | 0.04393  | -0.15273                 | 0.28850  |         |          |
|                           | y3        | -0.00242 | 0.22481                  | -0.26397 |         |          |
| 2                         | y1        | -0.16055 | 0.11460                  | 0.93439  |         |          |
|                           | y2        | 0.05003  | 0.01917                  | -0.01020 |         |          |
|                           | y3        | 0.03388  | 0.35491                  | -0.02223 |         |          |
| Model Parameter Estimates |           |          |                          |          |         |          |
| Equation                  | Parameter | Estimate | Std Error                | T Ratio  | Prob> T | Variable |
| y1(t)                     | CONST1    | -0.01672 | 0.01723                  | -0.97    | 0.3352  |          |
|                           | AR1_1_1   | -0.31963 | 0.12546                  | -2.55    | 0.0132  | y1(t-1)  |
|                           | AR1_1_2   | 0.14599  | 0.54567                  | 0.27     | 0.7899  | y2(t-1)  |
|                           | AR1_1_3   | 0.96122  | 0.66431                  | 1.45     | 0.1526  | y3(t-1)  |
|                           | AR2_1_1   | -0.16055 | 0.12491                  | -1.29    | 0.2032  | y1(t-2)  |
|                           | AR2_1_2   | 0.11460  | 0.53457                  | 0.21     | 0.8309  | y2(t-2)  |
|                           | AR2_1_3   | 0.93439  | 0.66510                  | 1.40     | 0.1647  | y3(t-2)  |
| y2(t)                     | CONST2    | 0.01577  | 0.00437                  | 3.60     | 0.0006  |          |
|                           | AR1_2_1   | 0.04393  | 0.03186                  | 1.38     | 0.1726  | y1(t-1)  |
|                           | AR1_2_2   | -0.15273 | 0.13857                  | -1.10    | 0.2744  | y2(t-1)  |
|                           | AR1_2_3   | 0.28850  | 0.16870                  | 1.71     | 0.0919  | y3(t-1)  |
|                           | AR2_2_1   | 0.05003  | 0.03172                  | 1.58     | 0.1195  | y1(t-2)  |
|                           | AR2_2_2   | 0.01917  | 0.13575                  | 0.14     | 0.8882  | y2(t-2)  |
|                           | AR2_2_3   | -0.01020 | 0.16890                  | -0.06    | 0.9520  | y3(t-2)  |
| y3(t)                     | CONST3    | 0.01293  | 0.00353                  | 3.67     | 0.0005  |          |
|                           | AR1_3_1   | -0.00242 | 0.02568                  | -0.09    | 0.9251  | y1(t-1)  |
|                           | AR1_3_2   | 0.22481  | 0.11168                  | 2.01     | 0.0482  | y2(t-1)  |
|                           | AR1_3_3   | -0.26397 | 0.13596                  | -1.94    | 0.0565  | y3(t-1)  |
|                           | AR2_3_1   | 0.03388  | 0.02556                  | 1.33     | 0.1896  | y1(t-2)  |
|                           | AR2_3_2   | 0.35491  | 0.10941                  | 3.24     | 0.0019  | y2(t-2)  |
|                           | AR2_3_3   | -0.02223 | 0.13612                  | -0.16    | 0.8708  | y3(t-2)  |

## Output 4.2.3. Diagnostic Checks

| The VARMAX Procedure                                       |                |            |                |          |
|------------------------------------------------------------|----------------|------------|----------------|----------|
| Covariance Matrix for the Innovation                       |                |            |                |          |
| Variable                                                   | y1             | y2         | y3             |          |
| y1                                                         | 0.00213        | 0.00007162 | 0.00012324     |          |
| y2                                                         | 0.00007162     | 0.00013734 | 0.00006146     |          |
| y3                                                         | 0.00012324     | 0.00006146 | 0.00008920     |          |
| Information Criteria                                       |                |            |                |          |
| AICC(Corrected AIC)                                        |                |            | -24.4884       |          |
| HQC(Hannan-Quinn Criterion)                                |                |            | -24.2869       |          |
| AIC(Akaike Information Criterion)                          |                |            | -24.5494       |          |
| SBC(Schwarz Bayesian Criterion)                            |                |            | -23.8905       |          |
| FPEC(Final Prediction Error Criterion)                     |                |            | 2.18E-11       |          |
| Residual Cross-Correlation Matrices                        |                |            |                |          |
| Lag                                                        | Variable       | y1         | y2             | y3       |
| 0                                                          | y1             | 1.00000    | 0.13242        | 0.28275  |
|                                                            | y2             | 0.13242    | 1.00000        | 0.55526  |
|                                                            | y3             | 0.28275    | 0.55526        | 1.00000  |
| 1                                                          | y1             | 0.01461    | -0.00666       | -0.02394 |
|                                                            | y2             | -0.01125   | -0.00167       | -0.04515 |
|                                                            | y3             | -0.00993   | -0.06780       | -0.09593 |
| 2                                                          | y1             | 0.07253    | -0.00226       | -0.01621 |
|                                                            | y2             | -0.08096   | -0.01066       | -0.02047 |
|                                                            | y3             | -0.02660   | -0.01392       | -0.02263 |
| 3                                                          | y1             | 0.09915    | 0.04484        | 0.05243  |
|                                                            | y2             | -0.00289   | 0.14059        | 0.25984  |
|                                                            | y3             | -0.03364   | 0.05374        | 0.05644  |
| Schematic Representation of<br>Residual Cross Correlations |                |            |                |          |
| Variable/<br>Lag                                           | 0              | 1          | 2              | 3        |
| y1                                                         | ++             | ...        | ...            | ...      |
| y2                                                         | ..+            | ...        | ...            | ..+      |
| y3                                                         | +++            | ...        | ...            | ...      |
| + is > 2*std error, - is <<br>-2*std error, . is between   |                |            |                |          |
| Portmanteau Test for Residual<br>Cross Correlations        |                |            |                |          |
| To<br>Lag                                                  | Chi-<br>Square | DF         | Prob><br>ChiSq |          |
| 3                                                          | 9.69           | 9          | 0.3766         |          |

Output 4.2.4. Diagnostic Checks Continued

| The VARMAX Procedure               |          |                    |                |                  |         |        |         |        |
|------------------------------------|----------|--------------------|----------------|------------------|---------|--------|---------|--------|
| Univariate Model Diagnostic Checks |          |                    |                |                  |         |        |         |        |
| Variable                           | R-square | StdDev             | F Value        | Prob>F           |         |        |         |        |
| y1                                 | 0.1286   | 0.0461             | 1.62           | 0.1547           |         |        |         |        |
| y2                                 | 0.1142   | 0.0117             | 1.42           | 0.2210           |         |        |         |        |
| y3                                 | 0.2513   | 0.0094             | 3.69           | 0.0032           |         |        |         |        |
| Univariate Model Diagnostic Checks |          |                    |                |                  |         |        |         |        |
| Variable                           | DW(1)    | Normality<br>ChiSq | Prob><br>ChiSq | ARCH1<br>F Value | Prob>F  |        |         |        |
| y1                                 | 1.96     | 10.22              | 0.0060         | 12.39            | 0.0008  |        |         |        |
| y2                                 | 1.98     | 11.98              | 0.0025         | 0.38             | 0.5386  |        |         |        |
| y3                                 | 2.15     | 34.25              | <.0001         | 0.10             | 0.7480  |        |         |        |
| Univariate Model Diagnostic Checks |          |                    |                |                  |         |        |         |        |
| Variable                           | AR1      |                    | AR1-2          |                  | AR1-3   |        | AR1-4   |        |
|                                    | F Value  | Prob>F             | F Value        | Prob>F           | F Value | Prob>F | F Value | Prob>F |
| y1                                 | 0.01     | 0.9029             | 0.19           | 0.8291           | 0.39    | 0.7624 | 1.39    | 0.2481 |
| y2                                 | 0.00     | 0.9883             | 0.00           | 0.9961           | 0.46    | 0.7097 | 0.34    | 0.8486 |
| y3                                 | 0.68     | 0.4129             | 0.38           | 0.6861           | 0.30    | 0.8245 | 0.21    | 0.9320 |

Output 4.2.5. Impulse Response Function

| The VARMAX Procedure         |  |      |  |             |  |          |  |          |  |
|------------------------------|--|------|--|-------------|--|----------|--|----------|--|
| Impulse Response by Variable |  |      |  |             |  |          |  |          |  |
| Variable                     |  | Lead |  | y1          |  | y2       |  | y3       |  |
| y1                           |  | 1    |  | -0.31963    |  | 0.14599  |  | 0.96122  |  |
|                              |  | STD  |  | 0.12546     |  | 0.54567  |  | 0.66431  |  |
|                              |  | 2    |  | -0.05430    |  | 0.26174  |  | 0.41555  |  |
|                              |  | STD  |  | 0.12919     |  | 0.54728  |  | 0.66311  |  |
|                              |  | 3    |  | 0.11904     |  | 0.35283  |  | -0.40789 |  |
| y2                           |  | STD  |  | 0.08362     |  | 0.38489  |  | 0.47867  |  |
|                              |  | 1    |  | 0.04393     |  | -0.15273 |  | 0.28850  |  |
|                              |  | STD  |  | 0.03186     |  | 0.13857  |  | 0.16870  |  |
|                              |  | 2    |  | 0.02858     |  | 0.11377  |  | -0.08820 |  |
|                              |  | STD  |  | 0.03184     |  | 0.13425  |  | 0.16250  |  |
| y3                           |  | 3    |  | -0.00884    |  | 0.07147  |  | 0.11977  |  |
|                              |  | STD  |  | 0.01583     |  | 0.07914  |  | 0.09462  |  |
|                              |  | 1    |  | -0.00242    |  | 0.22481  |  | -0.26397 |  |
|                              |  | STD  |  | 0.02568     |  | 0.11168  |  | 0.13596  |  |
|                              |  | 2    |  | 0.04517     |  | 0.26088  |  | 0.10998  |  |
|                              |  | STD  |  | 0.02563     |  | 0.10820  |  | 0.13101  |  |
|                              |  | 3    |  | -0.00055044 |  | -0.09818 |  | 0.09096  |  |
|                              |  | STD  |  | 0.01646     |  | 0.07823  |  | 0.10280  |  |

**Output 4.2.6.** Proportions of Prediction Error Covariance Decomposition

| The VARMAX Procedure                                    |      |         |         |         |  |
|---------------------------------------------------------|------|---------|---------|---------|--|
| Proportions of Prediction Error Covariances by Variable |      |         |         |         |  |
| Variable                                                | Lead | y1      | y2      | y3      |  |
| y1                                                      | 1    | 1.00000 | 0       | 0       |  |
|                                                         | 2    | 0.95996 | 0.01751 | 0.02253 |  |
|                                                         | 3    | 0.94565 | 0.02802 | 0.02633 |  |
|                                                         | 4    | 0.94079 | 0.02936 | 0.02985 |  |
|                                                         | 5    | 0.93846 | 0.03018 | 0.03136 |  |
|                                                         | 6    | 0.93831 | 0.03025 | 0.03145 |  |
| y2                                                      | 1    | 0.01754 | 0.98246 | 0       |  |
|                                                         | 2    | 0.06025 | 0.90747 | 0.03228 |  |
|                                                         | 3    | 0.06959 | 0.89576 | 0.03465 |  |
|                                                         | 4    | 0.06831 | 0.89232 | 0.03937 |  |
|                                                         | 5    | 0.06850 | 0.89212 | 0.03938 |  |
|                                                         | 6    | 0.06924 | 0.89141 | 0.03935 |  |
| y3                                                      | 1    | 0.07995 | 0.27292 | 0.64713 |  |
|                                                         | 2    | 0.07725 | 0.27385 | 0.64890 |  |
|                                                         | 3    | 0.12973 | 0.33364 | 0.53663 |  |
|                                                         | 4    | 0.12870 | 0.33499 | 0.53631 |  |
|                                                         | 5    | 0.12859 | 0.33924 | 0.53217 |  |
|                                                         | 6    | 0.12852 | 0.33963 | 0.53185 |  |

**Output 4.2.7.** Forecasts

| The VARMAX Procedure |     |        |           |                |                       |        |
|----------------------|-----|--------|-----------|----------------|-----------------------|--------|
| Variable             | Obs | Time   | Forecasts |                |                       |        |
|                      |     |        | Forecast  | Standard Error | 95% Confidence Limits |        |
| y1                   | 77  | 1979:1 | 6.5403    | 0.0461         | 6.4498                | 6.6307 |
|                      | 78  | 1979:2 | 6.5511    | 0.0583         | 6.4369                | 6.6652 |
|                      | 79  | 1979:3 | 6.5722    | 0.0688         | 6.4373                | 6.7071 |
|                      | 80  | 1979:4 | 6.5845    | 0.0802         | 6.4273                | 6.7417 |
|                      | 81  | 1980:1 | 6.6019    | 0.0912         | 6.4232                | 6.7806 |
| y2                   | 77  | 1979:1 | 7.6847    | 0.0117         | 7.6618                | 7.7077 |
|                      | 78  | 1979:2 | 7.7051    | 0.0169         | 7.6719                | 7.7382 |
|                      | 79  | 1979:3 | 7.7221    | 0.0216         | 7.6798                | 7.7643 |
|                      | 80  | 1979:4 | 7.7427    | 0.0262         | 7.6914                | 7.7939 |
|                      | 81  | 1980:1 | 7.7624    | 0.0301         | 7.7035                | 7.8213 |
| y3                   | 77  | 1979:1 | 7.5402    | 0.00944        | 7.5217                | 7.5587 |
|                      | 78  | 1979:2 | 7.5549    | 0.0128         | 7.5298                | 7.5800 |
|                      | 79  | 1979:3 | 7.5747    | 0.0181         | 7.5393                | 7.6101 |
|                      | 80  | 1979:4 | 7.5934    | 0.0221         | 7.5502                | 7.6367 |
|                      | 81  | 1980:1 | 7.6123    | 0.0258         | 7.5618                | 7.6629 |

**Output 4.2.8.** Granger Causality Tests

| The VARMAX Procedure          |            |    |            |
|-------------------------------|------------|----|------------|
| Granger Causality Wald Test   |            |    |            |
| Test                          | Chi-Square | DF | Prob>Chisq |
| 1                             | 6.37       | 4  | 0.1734     |
| Test 1: Group 1 Variables: y1 |            |    |            |
| Group 2 Variables: y2 y3      |            |    |            |

```

proc varmax data=use;
  id date interval=qtr align=E;
  model y2 y3 = y1 / p=2 dify(1) difx(1) xlag=1
                    lagmax=3;
run;

```

The VARMAX procedure output is shown in Output 4.2.9 through Output 4.2.12.

**Output 4.2.9.** Parameter Estimates

| The VARMAX Procedure       |                          |          |          |
|----------------------------|--------------------------|----------|----------|
| Type of Model              | VARX(2,1)                |          |          |
| Estimation Method          | Least Squares Estimation |          |          |
| Constant Estimates         |                          |          |          |
| Variable                   | Constant                 |          |          |
| y2                         | 0.01542                  |          |          |
| y3                         | 0.01319                  |          |          |
| XLag Coefficient Estimates |                          |          |          |
| Lag                        | Variable                 | y1       |          |
| 0                          | y2                       | 0.02520  |          |
|                            | y3                       | 0.05130  |          |
| 1                          | y2                       | 0.03870  |          |
|                            | y3                       | 0.00363  |          |
| AR Coefficient Estimates   |                          |          |          |
| Lag                        | Variable                 | y2       | y3       |
| 1                          | y2                       | -0.12258 | 0.25811  |
|                            | y3                       | 0.24367  | -0.31809 |
| 2                          | y2                       | 0.01651  | 0.03498  |
|                            | y3                       | 0.34921  | -0.01664 |



Output 4.2.10. Parameter Estimates Continued

| The VARMAX Procedure      |           |          |           |         |         |          |
|---------------------------|-----------|----------|-----------|---------|---------|----------|
| Model Parameter Estimates |           |          |           |         |         |          |
| Equation                  | Parameter | Estimate | Std Error | T Ratio | Prob> T | Variable |
| y2(t)                     | CONST1    | 0.01542  | 0.00443   | 3.48    | 0.0009  |          |
|                           | XL0_1     | 0.02520  | 0.03130   | 0.81    | 0.4237  | y1(t)    |
|                           | XL1_1     | 0.03870  | 0.03252   | 1.19    | 0.2383  | y1(t-1)  |
|                           | AR1_1_1   | -0.12258 | 0.13903   | -0.88   | 0.3811  | y2(t-1)  |
|                           | AR1_1_2   | 0.25811  | 0.17370   | 1.49    | 0.1421  | y3(t-1)  |
|                           | AR2_1_1   | 0.01651  | 0.13766   | 0.12    | 0.9049  | y2(t-2)  |
| y3(t)                     | AR2_1_2   | 0.03498  | 0.16783   | 0.21    | 0.8356  | y3(t-2)  |
|                           | CONST2    | 0.01319  | 0.00346   | 3.81    | 0.0003  |          |
|                           | XL0_2     | 0.05130  | 0.02441   | 2.10    | 0.0394  | y1(t)    |
|                           | XL1_2     | 0.00363  | 0.02536   | 0.14    | 0.8868  | y1(t-1)  |
|                           | AR1_2_1   | 0.24367  | 0.10842   | 2.25    | 0.0280  | y2(t-1)  |
|                           | AR1_2_2   | -0.31809 | 0.13546   | -2.35   | 0.0219  | y3(t-1)  |
|                           | AR2_2_1   | 0.34921  | 0.10736   | 3.25    | 0.0018  | y2(t-2)  |
|                           | AR2_2_2   | -0.01664 | 0.13088   | -0.13   | 0.8992  | y3(t-2)  |

## Output 4.2.11. Diagnostic Checks

| The VARMAX Procedure                                       |                |            |                |     |
|------------------------------------------------------------|----------------|------------|----------------|-----|
| Covariance Matrix for the Innovation                       |                |            |                |     |
| Variable                                                   | y2             | y3         |                |     |
| y2                                                         | 0.00014113     | 0.00006214 |                |     |
| y3                                                         | 0.00006214     | 0.00008583 |                |     |
| Information Criteria                                       |                |            |                |     |
| AICC(Corrected AIC)                                        |                |            | -18.3902       |     |
| HQC(Hannan-Quinn Criterion)                                |                |            | -18.2558       |     |
| AIC(Akaike Information Criterion)                          |                |            | -18.4309       |     |
| SBC(Schwarz Bayesian Criterion)                            |                |            | -17.9916       |     |
| FPEC(Final Prediction Error Criterion)                     |                |            | 9.91E-9        |     |
| Residual Cross-Correlation Matrices                        |                |            |                |     |
| Lag                                                        | Variable       | y2         | y3             |     |
| 0                                                          | y2             | 1.00000    | 0.56462        |     |
|                                                            | y3             | 0.56462    | 1.00000        |     |
| 1                                                          | y2             | -0.02312   | -0.05927       |     |
|                                                            | y3             | -0.07056   | -0.09145       |     |
| 2                                                          | y2             | -0.02849   | -0.05262       |     |
|                                                            | y3             | -0.05804   | -0.08567       |     |
| 3                                                          | y2             | 0.16071    | 0.29588        |     |
|                                                            | y3             | 0.10882    | 0.13002        |     |
| Schematic Representation of<br>Residual Cross Correlations |                |            |                |     |
| Variable/<br>Lag                                           | 0              | 1          | 2              | 3   |
| y2                                                         | ++             | ..         | ..             | ..+ |
| y3                                                         | ++             | ..         | ..             | ..  |
| + is > 2*std error, - is <<br>-2*std error, . is between   |                |            |                |     |
| Portmanteau Test for Residual<br>Cross Correlations        |                |            |                |     |
| To<br>Lag                                                  | Chi-<br>Square | DF         | Prob><br>ChiSq |     |
| 3                                                          | 8.38           | 4          | 0.0787         |     |

**Output 4.2.12.** Diagnostic Checks Continued

| The VARMAX Procedure               |          |        |         |        |
|------------------------------------|----------|--------|---------|--------|
| Univariate Model Diagnostic Checks |          |        |         |        |
| Variable                           | R-square | StdDev | F Value | Prob>F |
| y2                                 | 0.0897   | 0.0119 | 1.08    | 0.3809 |
| y3                                 | 0.2796   | 0.0093 | 4.27    | 0.0011 |

| Univariate Model Diagnostic Checks |       |                    |                |                  |        |
|------------------------------------|-------|--------------------|----------------|------------------|--------|
| Variable                           | DW(1) | Normality<br>ChiSq | Prob><br>ChiSq | ARCH1<br>F Value | Prob>F |
| y2                                 | 2.02  | 14.54              | 0.0007         | 0.49             | 0.4842 |
| y3                                 | 2.13  | 32.27              | <.0001         | 0.08             | 0.7782 |

| Univariate Model Diagnostic Checks |         |        |         |        |         |        |         |        |
|------------------------------------|---------|--------|---------|--------|---------|--------|---------|--------|
| Variable                           | AR1     |        | AR1-2   |        | AR1-3   |        | AR1-4   |        |
|                                    | F Value | Prob>F | F Value | Prob>F | F Value | Prob>F | F Value | Prob>F |
| y2                                 | 0.04    | 0.8448 | 0.04    | 0.9570 | 0.62    | 0.6029 | 0.42    | 0.7914 |
| y3                                 | 0.62    | 0.4343 | 0.62    | 0.5383 | 0.72    | 0.5452 | 0.36    | 0.8379 |

### Example 4.3. Numerous Examples

The following are examples of syntax for model fitting:

```

/* Data 'a' Generated Process */
proc iml;
  sig = {1.0 0.5, 0.5 1.25};
  phi = {1.2 -0.5, 0.6 0.3};
  call varmasim(y,phi) sigma = sig n = 100 seed = 46859;
  cn = {'y1' 'y2'};
  create a from y[colname=cn];
  append from y;
quit;

/* when the series has a linear trend */
proc varmax data=a;
  model y1 y2 / p=1 trend=linear;
run;

/* Fit subset of AR order 1 and 3 */
proc varmax data=a;
  model y1 y2 / p=(1,3);
run;

/* Check if the series is nonstationary */
proc varmax data=a;
  model y1 y2 / p=1 dfest print=(roots);
run;

```

```

/* Fit VAR(1) in differencing */
proc varmax data=a;
  model y1 y2 / p=1 print=(roots) dify=(1);
run;

/* Fit VAR(1) in seasonal differencing */
proc varmax data=a;
  model y1 y2 / p=1 dify=(4) lagmax=5;
run;

/* Fit VAR(1) in both regular and seasonal differencing */
proc varmax data=a;
  model y1 y2 / p=1 dify=(1,4) lagmax=5;
run;

/* Fit VAR(1) in different differencing */
proc varmax data=a;
  model y1 y2 / p=1 dif=(y1(1,4) y2(1)) lagmax=5;
run;

/* Options related prediction */
proc varmax data=a;
  model y1 y2 / p=1 lagmax=3
               print=(impulse covpe(5) decompose(5));
run;

/* Options related tentative order selection */
proc varmax data=a;
  model y1 y2 / p=1 lagmax=5 minic
               print=(parcoef pcancorr pcorr);
run;

/* Automatic selection of the AR order */
proc varmax data=a;
  model y1 y2 / minic=(type=aic p=5);
run;

/* Compare results of LS and Yule-Walker Estimator */
proc varmax data=a;
  model y1 y2 / p=1 print=(yw);
run;

/* BVAR(1) of the nonstationary series y1 and y2 */
proc varmax data=a;
  model y1 y2 / p=1
               prior=(lambda=1 theta=0.2 ivar nrep=200);
run;

/* BVAR(1) of the nonstationary series y1 */
proc varmax data=a;
  model y1 y2 / p=1
               prior=(lambda=0.1 theta=0.15 ivar=(y1) seed=12345);
run;

```

```

/* Data 'b' Generated Process */
proc iml;
  sig = { 0.5  0.14 -0.08 -0.03,  0.14 0.71 0.16 0.1,
          -0.08 0.16  0.65  0.23, -0.03 0.1  0.23 0.16};
  sig = sig * 0.0001;
  phi = {1.2 -0.5 0.  0.1,  0.6 0.3 -0.2  0.5,
          0.4  0. -0.2 0.1, -1.0 0.2  0.7 -0.2};
  call varmasim(y,phi) sigma = sig n = 100 seed = 32567;
  cn = {'y1' 'y2' 'y3' 'y4'};
  create b from y[colname=cn];
  append from y;
quit;

/* Cointegration Rank Test using Trace statistics */
proc varmax data=b;
  model y1-y4 / p=2 lagmax=4 cointtest;
run;

/* Cointegration Rank Test using Max statistics */
proc varmax data=b;
  model y1-y4 / p=2 lagmax=4 cointtest=(johansen=(type=max));
run;

/* Common Trends Test using Filter(Differencing) statistics */
proc varmax data=b;
  model y1-y4 / p=2 lagmax=4 cointtest=(sw);
run;

/* Common Trends Test using Filter(Residual) statistics */
proc varmax data=b;
  model y1-y4 / p=2 lagmax=4 cointtest=(sw=(type=filtres lag=1));
run;

/* Common Trends Test using Kernel statistics */
proc varmax data=b;
  model y1-y4 / p=2 lagmax=4 cointtest=(sw=(type=kernel lag=1));
run;

/* Cointegration Rank Test for I(2) */
proc varmax data=b;
  model y1-y4 / p=2 lagmax=4 cointtest=(johansen=(iorder=2));
run;

/* Fit VECM(2) with rank=3 */
proc varmax data=b;
  model y1-y4 / p=2 lagmax=4 print=(roots iarr)
          ecm=(rank=3 normalize=y1);
run;

```

```

/* Weak Exogenous Testing for each variable */
proc varmax data=b outstat=bbb;
  model y1-y4 / p=2 lagmax=4
             ecm=(rank=3 normalize=y1);
  cointeg rank=3 exogeneity;
run;

/* Hypotheses Testing for long-run and adjustment parameter */
proc varmax data=b outstat=bbb;
  model y1-y4 / p=2 lagmax=4
             ecm=(rank=3 normalize=y1);
  cointeg rank=3 normalize=y1
    h=(1 0 0, 0 1 0, -1 0 0, 0 0 1)
    j=(1 0 0, 0 1 0, 0 0 1, 0 0 0);
run;

/* ordinary regression model */
proc varmax data=grunfeld;
  model y1 y2 = x1-x3;
run;

/* Ordinary regression model with subset lagged terms */
proc varmax data=grunfeld;
  model y1 y2 = x1 / xlag=(1,3);
run;

/* VARX(1,1) with no current time Exogenous Variables */
proc varmax data=grunfeld;
  model y1 y2 = x1 / p=1 xlag=1 nocurrentx;
run;

/* VARX(1,1) with different Exogenous Variables */
proc varmax data=grunfeld;
  model y1 = x3, y2 = x1 x2 / p=1 xlag=1;
run;

/* VARX(1,2) in difference with current time Exogenous Variables */
proc varmax data=grunfeld;
  model y1 y2 = x1 / p=1 xlag=2 difx=(1) dify=(1);
run;

```

---

## References

- Anderson, T. W. (1951), "Estimating Linear Restrictions on Regression Coefficients for Multivariate Normal Distributions," *Annals of Mathematical Statistics*, 22, 327-351.
- Ansley, C. F. and Kohn, R. (1986), "A Note on Reparameterizing a Vector Autoregressive Moving Average Model to Enforce Stationarity", *Journal of Statistical Computation and Simulation*, 24, 99-106.
- Ansley, C. F., Kohn, R., and Shively, T. S. (1992), "Computing  $p$ -values for the Generalized Durbin-Watson and Other Invariant Test Statistics," *Journal of Econo-*

- metrics*, 54, 277–300.
- Brockwell, P. J. and Davis, R. A. (1991), *Time Series: Theory and Methods*, Second Edition, New York: Springer-Verlag.
- Dickey, D. A., Hasza, D. P., and Fuller, W. A. (1984), “Testing for Unit Roots in Seasonal Time Series,” *Journal of the American Statistical Association*, 79, 355–367.
- Doan, T., Litterman, R., and Sims, C. (1984), “Forecasting and Conditional Projection Using Realistic Prior Distributions,” *Econometric Reviews*, 3, 1–144.
- Engle, R. F. and Granger, C. W. J. (1987), “Co-integration and Error Correction: Representation, Estimation and Testing,” *Econometrica*, 55, 251–276.
- Engle, R. F. and Yoo, B. S. (1987), “Forecasting and Testing in Co-Integrated Systems,” *Journal of Econometrics*, 35, 143–159.
- Hamilton, J. D. (1994), *Time Series Analysis*, Princeton: Princeton University Press.
- Hannan, E. J. (1970), *Multiple Time Series*, New York: John Wiley & Sons.
- Hannan, E. J. and Deistler, M. (1988), *The Statistical Theory of Linear Systems*, New York: John Wiley & Sons.
- Hosking, J. R. M. (1980), “The Multivariate Portmanteau Statistic,” *Journal of the American Statistical Association*, 75, 602–608.
- Hurvich, C. M. and Tsai, C. (1993), “A Corrected Akaike Information Criterion for Vector Autoregressive Model Selection,” *Journal of Time Series Analysis*, 14, 271–279.
- Johansen, S. (1988), “Statistical Analysis of Cointegration Vectors,” *Journal of Economic Dynamics and Control*, 12, 231–254.
- Johansen, S. (1992a), “A Representation of Vector Autoregressive Processes Integrated of Order 2,” *Econometric Theory*, 8, 188–202.
- Johansen, S. (1992b), “Testing Weak Exogeneity and the Order of Cointegration in UK Money Demand Data,” *Journal of Policy Modeling*, 14, 313–334.
- Johansen, S. (1995a), “A Statistical Analysis of Cointegration for I(2) Variables,” *Econometric Theory*, 11, 25–59.
- Johansen, S. (1995b), *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*, New York: Oxford University Press.
- Johansen, S. and Juselius, K. (1990), “Maximum Likelihood Estimation and Inference on Cointegration: With Applications to the Demand for Money,” *Oxford Bulletin of Economics and Statistics*, 52, 169–210.
- Johansen, S. and Juselius, K. (1992), “Testing Structural Hypotheses in a Multivariate Cointegration Analysis of the PPP and the UIP for UK,” *Journal of Econometrics*, 53, 211–244.
- Kim, M. (1996), “A Remark on Algorithm AS 279: Computing  $p$ -values for the Generalized Durbin-Watson Statistic and Residual Autocorrelation in Regression,” *Applied Statistics*, 45, 273–274.

- Kohn, R., Shively, T. S., and Ansley, C. F. (1993), "Algorithm AS 279: Computing  $p$ -values for the Generalized Durbin-Watson Statistic and Residual Autocorrelation in Regression," *Applied Statistics*, 42, 249–269.
- Koreisha, S. and Pukkila, T. (1989), "Fast Linear Estimation Methods for Vector Autoregressive Moving Average Models," *Journal of Time Series Analysis*, 10, 325–339.
- Litterman, R. B. (1986), "Forecasting with Bayesian Vector Autoregressions: Five Years of Experience," *Journal of Business & Economic Statistics*, 4, 25–38.
- Lütkepohl, H. (1993), *Introduction to Multiple Time Series Analysis*, Berlin: Springer-Verlag.
- Maddala, G. S. and Kim, I. (1998), *Unit Roots, Cointegration, and Structural Change*, Cambridge: Cambridge University Press.
- Osterwald-Lenum, M. (1992), "A Note with Quantiles of the Asymptotic Distribution of the Maximum Likelihood Cointegration Rank Test Statistics," *Oxford Bulletin of Economics and Statistics*, 54, 461–472.
- Pringle, R. M. and Raynor, D. L. (1971), *Generalized Inverse Matrices with Applications to Statistics*, Second Edition, New York: McGraw-Hill Inc.
- Reinsel, G. C. (1997), *Elements of Multivariate Time Series Analysis*, Second Edition, New York: Springer-Verlag.
- Quinn, B. G. (1980), "Order Determination for a Multivariate Autoregression," *Journal of the Royal Statistical Society*, B, 42, 182–185.
- Spliid, H. (1983), "A Fast Estimation for the Vector Autoregressive Moving Average Models with Exogenous Variables," *Journal of the American Statistical Association*, 78, 843–849.
- Stock, J. H. and Watson, M. W. (1988), "Testing for Common Trends," *Journal of the American Statistical Association*, 83, 1097–1107.
- Todd, R. M. (1984), "Improving Economic Forecasting with Bayesian Vector Autoregression," *Federal Reserve Bank of Minneapolis Quarterly Review*, 4 (Fall), 18–29.





# Chapter 5

## The X12 Procedure

### Chapter Table of Contents

---

|                                  |     |
|----------------------------------|-----|
| <b>OVERVIEW</b>                  | 163 |
| <b>GETTING STARTED</b>           | 163 |
| Basic Seasonal Adjustment        | 164 |
| <b>SYNTAX</b>                    | 166 |
| Functional Summary               | 167 |
| PROC X12 Statement               | 168 |
| ARIMA Statement                  | 169 |
| ESTIMATE Statement               | 170 |
| IDENTIFY Statement               | 170 |
| OUTPUT Statement                 | 171 |
| REGRESSION Statement             | 172 |
| TRANSFORM Statement              | 173 |
| VAR Statement                    | 174 |
| X11 Statement                    | 174 |
| <b>DETAILS</b>                   | 176 |
| Computations                     | 176 |
| Displayed Output/ODS Table Names | 176 |
| <b>EXAMPLES</b>                  | 177 |
| Example 5.1 Model Identification | 177 |
| Example 5.2 Model Estimation     | 180 |
| Example 5.3 Seasonal Adjustment  | 181 |
| <b>ACKNOWLEDGMENTS</b>           | 183 |
| <b>REFERENCES</b>                | 184 |



## Chapter 5

# The X12 Procedure

---

### Overview

The X12 procedure, an adaptation of the U.S. Bureau of the Census X-12-ARIMA Seasonal Adjustment program, seasonally adjusts monthly or quarterly time series. The procedure makes additive or multiplicative adjustments and creates an output data set containing the adjusted time series and intermediate calculations.

The X-12-ARIMA program combines the capabilities of the X-11 program (Shiskin, Young, and Musgrave 1967) and the X-11-ARIMA/88 program (Dagum 1988) and also introduces some new features (Findley et al. 1988). Thus, the X-12-ARIMA program contains methods developed by both the U.S. Census Bureau and Statistics Canada. The four major components of the X-12-ARIMA program are regARIMA modeling, model diagnostics, seasonal adjustment using enhanced X-11 methodology, and post-adjustment diagnostics. Statistics Canada's X-11 method fits an ARIMA model to the original series, then uses the model forecast to extend the original series. This extended series is then seasonally adjusted by the standard X-11 seasonal adjustment method. The extension of the series improves the estimation of the seasonal factors and reduces revisions to the seasonally adjusted series as new data become available.

Seasonal adjustment of a series is based on the assumption that seasonal fluctuations can be measured in the original series,  $O_t$ ,  $t = 1, \dots, n$ , and separated from trend cycle, trading-day, and irregular fluctuations. The seasonal component of this time series,  $S_t$ , is defined as the intrayear variation that is repeated constantly or in an evolving fashion from year to year. The trend cycle component,  $C_t$ , includes variation due to the long-term trend, the business cycle, and other long-term cyclical factors. The trading-day component,  $D_t$ , is the variation that can be attributed to the composition of the calendar. The irregular component,  $I_t$ , is the residual variation. Many economic time series are related in a multiplicative fashion ( $O_t = S_t C_t D_t I_t$ ). Other economic series are related in an additive fashion ( $O_t = S_t + C_t + D_t + I_t$ ). A seasonally adjusted time series,  $C_t I_t$  or  $C_t + I_t$ , consists of only the trend cycle and irregular components.

---

### Getting Started

The most common use of the X12 procedure is to produce a seasonally adjusted series. Eliminating the seasonal component from an economic series facilitates comparison among consecutive months or quarters. A plot of the seasonally adjusted series is often more informative about trends or location in a business cycle than a plot of the unadjusted series.

The following example shows how to use PROC X12 to produce a seasonally adjusted series,  $C_t I_t$ , from an original series  $O_t = S_t C_t D_t I_t$ .

In the multiplicative model, the trend cycle component  $C_t$  keeps the same scale as the original series  $O_t$ , while  $S_t$ ,  $D_t$ , and  $I_t$  vary around 1.0. In all displayed tables, these latter components are expressed as percentages and thus vary around 100.0 (in the additive case, they vary around 0.0). However, in the output data set, the data displayed as percentages will be expressed as the decimal equivalent and thus will vary around 1.0 in the multiplicative case.

The naming convention used in PROC X12 for the tables follows the convention used in the Census Bureau's X-12-ARIMA program; refer to *X-12-ARIMA Reference Manual* and *X-12-ARIMA Quick Reference for Unix* (U.S. Bureau of the Census 1999). Also see the section "Displayed Output" later in this chapter. The table names are outlined in Figure 5.5 on page 177.

The tables corresponding to parts A through C are intermediate calculations. The final estimates of the individual components are found in the D tables: table D10 contains the final seasonal factors, table D12 contains the final trend cycle, and table D13 contains the final irregular series. If you are primarily interested in seasonally adjusting a series without consideration of intermediate calculations or diagnostics, you need only to look at table D11, the final seasonally adjusted series. Tables in part E contain information regarding extreme values and changes in the original and seasonally adjusted series. The tables in part F are seasonal adjustment quality measures. Spectral analysis is performed in part G. For further information concerning the tables produced by the X11 statement, refer to Ladiray and Quenneville (1999).

---

## Basic Seasonal Adjustment

Suppose that you have monthly retail sales data starting in September 1978 in a SAS data set named SALES. At this point, you do not suspect that any calendar effects are present, and there are no prior adjustments that need to be made to the data.

In this simplest case, you need only specify the DATE= variable in the PROC X12 statement and request seasonal adjustment in the X11 statement. The results of the seasonal adjustment are in table D11 (the final seasonally adjusted series) in the displayed output.

```
data sales;
  input sales @@;
  date = intnx( 'month', '01sep78'd, _n_-1 );
  format date monyy.;
  datalines;
  ... datalines omitted ...
run;

proc x12 data=sales date=date;
  var sales;
  x11;
run ;
```

| The X12 Procedure                                                |            |            |            |            |            |            |         |
|------------------------------------------------------------------|------------|------------|------------|------------|------------|------------|---------|
| Table D 11: Final seasonally adjusted data<br>For variable sales |            |            |            |            |            |            |         |
| Year                                                             | JAN<br>JUL | FEB<br>AUG | MAR<br>SEP | APR<br>OCT | MAY<br>NOV | JUN<br>DEC | Total   |
| 1978                                                             | .          | .          | .          | .          | .          | .          |         |
|                                                                  |            |            | 124.560    | 124.649    | 124.920    | 129.002    | 503.131 |
| 1979                                                             | 125.087    | 126.759    | 125.252    | 126.415    | 127.012    | 130.041    |         |
|                                                                  | 128.056    | 129.165    | 127.182    | 133.847    | 133.199    | 135.847    | 1547.86 |
| 1980                                                             | 128.767    | 139.839    | 143.883    | 144.576    | 148.048    | 145.170    |         |
|                                                                  | 140.021    | 153.322    | 159.128    | 161.614    | 167.996    | 165.388    | 1797.75 |
| 1981                                                             | 175.984    | 166.805    | 168.380    | 167.913    | 173.429    | 175.711    |         |
|                                                                  | 179.012    | 182.017    | 186.737    | 197.367    | 183.443    | 184.907    | 2141.71 |
| 1982                                                             | 186.080    | 203.099    | 193.386    | 201.988    | 198.322    | 205.983    |         |
|                                                                  | 210.898    | 213.516    | 213.897    | 218.902    | 227.172    | 240.453    | 2513.69 |
| 1983                                                             | 231.839    | 224.165    | 219.411    | 225.907    | 225.015    | 226.535    |         |
|                                                                  | 221.680    | 222.177    | 222.959    | 212.531    | 230.552    | 232.565    | 2695.33 |
| 1984                                                             | 237.477    | 239.870    | 246.835    | 242.642    | 244.982    | 246.732    |         |
|                                                                  | 251.023    | 254.210    | 264.670    | 266.120    | 266.217    | 276.251    | 3037.03 |
| 1985                                                             | 275.485    | 281.826    | 294.144    | 286.114    | 293.192    | 296.601    |         |
|                                                                  | 293.861    | 309.102    | 311.275    | 319.239    | 319.936    | 323.663    | 3604.44 |
| 1986                                                             | 326.693    | 330.341    | 330.383    | 330.792    | 333.037    | 332.134    |         |
|                                                                  | 336.444    | 341.017    | 346.256    | 350.609    | 361.283    | 362.519    | 4081.51 |
| 1987                                                             | 364.951    | 371.274    | 369.238    | 377.242    | 379.413    | 376.451    |         |
|                                                                  | 378.930    | 375.392    | 374.940    | 373.612    | 368.753    | 364.885    | 4475.08 |
| 1988                                                             | 371.618    | 383.842    | 385.849    | 404.810    | 381.270    | 388.689    |         |
|                                                                  | 385.661    | 377.706    | 397.438    | 404.247    | 414.084    | 416.486    | 4711.70 |
| 1989                                                             | 426.716    | 419.491    | 427.869    | 446.161    | 438.317    | 440.639    |         |
|                                                                  | 450.193    | 454.638    | 460.644    | 463.209    | 427.728    | 485.386    | 5340.99 |
| 1990                                                             | 477.259    | 477.753    | 483.841    | 483.056    | 481.902    | 499.200    |         |
|                                                                  | 484.893    | 485.245    | .          | .          | .          | .          | 3873.15 |
| -----                                                            |            |            |            |            |            |            |         |
| Avg                                                              | 277.330    | 280.422    | 282.373    | 286.468    | 285.328    | 288.657    |         |
|                                                                  | 288.389    | 291.459    | 265.807    | 268.829    | 268.774    | 276.446    |         |
| -----                                                            |            |            |            |            |            |            |         |
| Total:                                                           |            | 40323      | Mean:      | 280.02     | S.D.:      | 111.31     |         |
|                                                                  |            |            | Min:       | 124.56     | Max:       | 499.2      |         |

Figure 5.1. Basic Seasonal Adjustment

You can compare the original series (table A1) and the final seasonally adjusted series (table D11) by plotting them together. These tables are requested in the OUTPUT statement and are written to the OUT= data set. Note that the default variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name.

```
proc x12 data=sales date=date;
  var sales;
  x11;
  output out=out a1 d11;
run;

symbol1 i=join v='star';
symbol2 i=join v='circle';
legend1 label=none value=('original' 'adjusted');

proc gplot data=out;
  plot sales_A1 * date = 1
```

```
sales_D11 * date = 2 / overlay legend=legend1;
run;
```

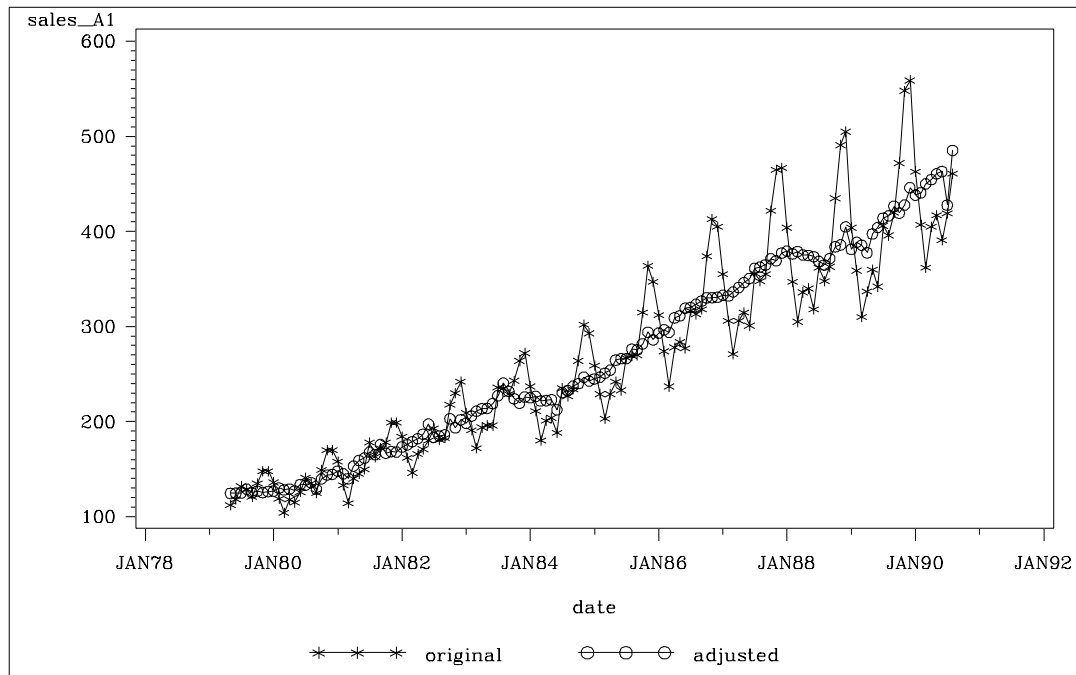


Figure 5.2. Plot of Original and Seasonally Adjusted Data

## Syntax

The X12 procedure uses the following statements:

```
PROC X12 options;  
  TRANSFORM options;  
  ESTIMATE;  
  IDENTIFY options;  
  REGRESSION options;  
  ARIMA options;  
  X11 options;  
  VAR variables;  
  OUTPUT options;
```

The PROC X12 statements perform basically the same function as the Census Bureau's X-12-ARIMA specs. *Specs* or specifications are used in X-12-ARIMA to control the computations and output. The PROC X12 statement performs some of the same functions as the Series spec in the Census Bureau's X-12-ARIMA software. The TRANSFORM, ESTIMATE, IDENTIFY, REGRESSION, ARIMA, and X11 statements are designed to perform the same functions as the corresponding

X-12-ARIMA *specs*, although full compatibility is not yet available. The Census Bureau documentation *X-12-ARIMA Reference Manual* can provide added insight to the functionality of these statements.

## Functional Summary

The following table outlines the options available for the X12 procedure classified by function.

| Description                                                                  | Statement  | Option            |
|------------------------------------------------------------------------------|------------|-------------------|
| <b>Data Set Options</b>                                                      |            |                   |
| specify input data set                                                       | PROC X12   | DATA=             |
| write table values to an output data set                                     | OUTPUT     | OUT=              |
| <b>Printing Control Options</b>                                              |            |                   |
| suppress all printed output                                                  | PROC X12   | NOPRINT           |
| <b>Date Information Options</b>                                              |            |                   |
| specify the date variable                                                    | PROC X12   | DATE=             |
| specify the date of the first observation                                    | PROC X12   | START=            |
| specify the beginning date of the subset                                     | PROC X12   | SPAN=( mmmyy )    |
|                                                                              | PROC X12   | SPAN=( 'yyQq' )   |
| specify the ending date of the subset                                        | PROC X12   | SPAN=( , mmmyy )  |
|                                                                              | PROC X12   | SPAN=( , 'yyQq' ) |
| specify monthly time series                                                  | PROC X12   | INTERVAL=MONTH    |
| specify monthly time series                                                  | PROC X12   | SEASONS=12        |
| specify quarterly time series                                                | PROC X12   | INTERVAL=QTR      |
| specify quarterly time series                                                | PROC X12   | SEASONS= 4        |
| <b>Declaring the Role of Variables</b>                                       |            |                   |
| specify the variables to be seasonally adjusted                              | VAR        |                   |
| <b>Controlling the Table Computations</b>                                    |            |                   |
| transform or prior adjust the series                                         | TRANSFORM  | POWER=            |
| estimate the regARIMA model specified by the REGRESSION and ARIMA statements | ESTIMATE   |                   |
| use differencing to identify the ARIMA part of the model                     | IDENTIFY   |                   |
| specify regression information                                               | REGRESSION | PREDEFINED=       |
| specify the ARIMA part of the model                                          | ARIMA      | MODEL=            |
| specify seasonal adjustment                                                  | X11        |                   |



---

## PROC X12 Statement

### **PROC X12** *options;*

The PROC X12 statement provides information about the time series to be processed by PROC X12. Either the START= or the DATE= option must be specified.

The original series is displayed in table A1. If there are missing values in the original series, and a regARIMA model is specified or automatically selected, then table MV1 is displayed. Table MV1 contains the original series with missing values replaced by the predicted values from the fitted model.

The following options can appear in the PROC X12 statement.

#### **DATA=** *SAS-data-set*

specifies the input SAS data set used. If this option is omitted, the most recently created SAS data set is used.

#### **DATE=** *variable*

#### **DATEVAR=** *variable*

specifies a variable that gives the date for each observation. Unless specified in the SPAN= option, the starting and ending dates are obtained from the first and last values of the DATE= variable, which must contain SAS datetime values. The procedure checks values of the DATE= variable to ensure that the input observations are sequenced correctly in ascending order. If the INTERVAL= or SEASONS= option is specified, its values must agree with the values of the date variable. If neither the INTERVAL= or SEASONS= option is specified, then the procedure tries to determine the type of data from the values of the date variable. This variable is automatically added to the OUTPUT= data set if one is requested and is extrapolated if necessary. If the DATE= option is not specified, the START= option must be specified.

#### **START=** *mmmyy*

#### **START=** *'yyQq'*

#### **STARTDATE=** *mmmyy*

#### **STARTDATE=** *'yyQq'*

gives the date of the first observation. Unless the SPAN= option is used, the starting and ending dates are the dates of the first and last observations, respectively. Either this option or the DATE= option is required. When using this option, use either the INTERVAL= or SEASONS= option to specify monthly or quarterly data. If neither the INTERVAL= or SEASONS= is present, monthly data are assumed. Note that for a quarterly date, the specification must be enclosed in quotes. A 4-digit year can be specified, but if a 2-digit year is given, the value specified in the YEARCUTOFF= SAS system option applies.

#### **SPAN=** *(mmmyy,mmmyy)*

#### **SPAN=** *('yyQq','yyQq')*

gives the dates of the first and last observations to define a subset for processing. A single date in parentheses is interpreted to be the starting date of the subset. To specify only the ending date, use SPAN=(*,mmmyy*). If the starting or ending date is

omitted, then the first or last date, respectively, of the input data set is assumed. A 4-digit year can be specified, but if a 2-digit year is given, the value specified in the YEARCUTOFF= SAS system option applies.

**INTERVAL=** *interval*

specifies the frequency of the input time series. If the input data consist of quarterly observations, then INTERVAL=QTR should be used. If the input data consist of monthly observations, then INTERVAL=MONTH should be used. If the INTERVAL= option is not specified and SEASONS=4, then INTERVAL=QTR is assumed; likewise, SEASONS=12 implies INTERVAL=MONTH. If both are used, the values should not be conflicting. If neither the INTERVAL= nor SEASONS= option is specified and the START= option is specified, then the data are assumed to be monthly. If a date variable is specified using the DATE= option, it is not necessary to specify the INTERVAL= or SEASONS= option; however, if specified, the values of the INTERVAL= or SEASONS= option should not be in conflict with the values of the date variable.

**SEASONS=** *period*

specifies the number of observations in a seasonal cycle. If the SEASONS= option is not specified and INTERVAL=QTR, then SEASONS=4 is assumed. If the SEASONS= option is not specified and INTERVAL=MONTH, then SEASONS=12 is assumed. If the SEASONS= option is specified, its value should not conflict with the values of the INTERVAL= option or the values of the date variable. See the preceding descriptions for the START=, DATE=, and INTERVAL= options for more details.

**NOPRINT**

suppresses any printed output.

---

## ARIMA Statement

**ARIMA** *options;*

The ARIMA statement specifies the ARIMA part of the regARIMA model. This statement defines a pure ARIMA model if the regression statement is omitted. The ARIMA part of the model can include multiplicative seasonal factors.

The following option can appear in the ARIMA statement.

**MODEL=** ( ( *p d q*) ( *P D Q*) *s* )

specifies the ARIMA model. The format follows standard Box-Jenkins notation (Box, Jenkins, and Reinsel 1994). The nonseasonal AR and MA orders are given by *p* and *q*, respectively, while the seasonal AR and MA orders are given by *P* and *Q*. The number of differences and seasonal differences are given by *d* and *D*, respectively. The notation ( *p d q*) and ( *P D Q*) can also be specified as (*p*, *d*, *q*) and (*P*, *D*, *Q*). The lag corresponding to seasonality is *s*. If *s* is omitted, it is set equal to the value used in the PROC X12 SEASONS= statement.

For example,

```
proc x12 data=ICMETI seasons=12 start=jan1968;  
    arima model=((2,1,1)(1,1,0))12;
```

specifies an ARIMA (2,1,1)(1,1,0)<sub>12</sub> model.

---

## ESTIMATE Statement

### **ESTIMATE ;**

The ESTIMATE statement estimates the regARIMA model specified by the REGRESSION and ARIMA statements. Estimation output includes point estimates and standard errors for all estimated AR, MA, and regression parameters; the maximum likelihood estimate of the variance  $\sigma^2$ ;  $t$  statistics for individual regression parameters;  $\chi^2$  statistics for assessing the joint significance of the parameters associated with certain regression effects (if included in the model); and likelihood-based model selection statistics (if the exact likelihood function is used). The regression effects for which  $\chi^2$  statistics are produced are fixed seasonal effects.

Tables displayed in association with estimation are Exact ARMA Likelihood Estimation Iteration Tolerances, Exact ARMA Likelihood Estimation Iteration Summary, Regression Model Parameter Estimates, Exact ARMA Maximum Likelihood Estimation, and Estimation Summary.

No options are currently available for the ESTIMATE statement.

---

## IDENTIFY Statement

### **IDENTIFY options;**

The IDENTIFY statement must be used to produce plots of the sample Autocorrelation Functions (ACFs) and Partial Autocorrelation Functions (PACFs) for identifying the ARIMA part of a regARIMA model. Sample ACFs and PACFs are produced for all combinations of the nonseasonal and seasonal differences of the data specified by the DIFF and SDIFF options. If the REGRESSION statement is present, the ACFs and PACFs are calculated for the specified differences of a series of regression residuals. If the REGRESSION statement is not present, the ACFs and PACFs are calculated for the specified differences of the original data.

Tables printed in association with identification are “Autocorrelation of Model Residuals” and “Partial Autocorrelation of Model Residuals”. If the REGRESSION statement is present, then the “Regression Model Parameter Estimates” table will also be printed.

The following options can appear in the IDENTIFY statement.

**DIFF= (order, order, order)**

specifies orders of nonseasonal differencing. The value 0 specifies no differencing, the value 1 specifies one nonseasonal difference  $(1 - B)$ , the value 2 specifies two nonseasonal differences  $(1 - B)^2$ , and so forth. The ACFs and PACFs are produced for all orders of nonseasonal differencing specified, in combination with all orders of seasonal differencing specified in the SDIFF= option. The default is DIFF=(0). You can specify up to three values for nonseasonal differences.

**SDIFF= (order, order, order)**

specifies orders of seasonal differencing. The value 0 specifies no seasonal differencing, the value 1 specifies one seasonal difference  $(1 - B^s)$ , the value 2 specifies two seasonal differences  $(1 - B^s)^2$ , and so forth. Here the value for  $s$  will correspond to the value of the SEASONS= option in the PROC X12 statement. The value of SEASONS= is supplied explicitly or is implicitly supplied through the INTERVAL= option or the values of the DATE= variable. The ACFs and PACFs are produced for all orders of seasonal differencing specified, in combination with all orders of nonseasonal differencing specified in the DIFF= option. The default is SDIFF=(0). You can specify up to three values for seasonal differences.

For example,

```
identify diff=(1) sdiff=(0, 1);
```

produces ACFs and PACFs for two models:  $(1 - B)$  and  $(1 - B)(1 - B^s)$ .

---

## OUTPUT Statement

```
OUTPUT OUT= SAS-data-set tablename1 tablename2 ... ;
```

The OUTPUT statement creates an output data set containing specified tables. The data set is named by the OUT= option.

**OUT= SAS-data-set**

If the OUT= option is omitted, the SAS System names the new data set using the default DATAn convention.

For each table to be included in the output data set, you must specify the X12 *tablename* keyword. The keyword corresponds to the title label used by the Census Bureau X12-ARIMA software. Currently available tables are A1, A6, A8, B1, C17, D8, D9, D10, D10D, D11, D12, D13, D16, D16B, D18, E5, E6, E7, and MV1. If no table is specified, table A1 will be displayed by default.

The tablename keywords that can be used in the OUTPUT statement are listed in the “Displayed Output/ODS Table Names” section on page 176. The following is an example of a VAR statement and an OUTPUT statement:

```
var sales costs;  
output out=out_x12 b1 d11;
```

Note that the default variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name. The variable `sales_B1` contains the table B1 values for the variable sales, the variable `costs_B1` contains the table B1 values for costs, while the table D11 values for sales are contained in the variable `sales_D11`, and the variable `costs_D11` contains the table D11 values for costs. If necessary, the variable name is shortened so that the table name can be added. Currently, you cannot specify the output variable names. A variable named `_DATE_` is the date identifier.

---

## REGRESSION Statement

### **REGRESSION** *options* ;

The REGRESSION statement includes regression variables in a regARIMA model or specifies regression variables whose effects are to be removed by the IDENTIFY statement to aid in ARIMA model identification. Predefined regression variables are selected with the PREDEFINED option. The only currently available predefined variables are length-of-month, length-of-quarter, and leap year. Table A6 provides information related to trading-day effects. You should note that missing values in the input series automatically create missing value regressors. Combining your model with additional predefined regression variables may result in a singularity problem. If a singularity occurs, then you may need to alter either the model or the choices of the predefined regressors in order to successfully perform the regression.

The following options can appear in the REGRESSION statement.

**PREDEFINED= LOM**

**PREDEFINED= LOQ**

**PREDEFINED= LPYEAR**

lists the predefined regression variables to be included in the model. Data values for these variables are calculated by the program, mostly as functions of the calendar. The values LOM and LOQ are actually equivalent: the actual regression is controlled by the PROC X12 SEASONS= option. Multiple predefined regression variables may be used. The syntax for using both a length-of-month and a leap-year regression could be one of the following forms:

```
regression predefined=lom lpyear;
```

```
regression predefined=(lom lpyear);
```

```
regression predefined=lom predefined=lpyear;
```

**Table 5.1.** Predefined Regression Variables in X-12-ARIMA

| Regression Effect                                   | Variable Definitions                                                                                                                                                 |
|-----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Length-of-Month<br>(monthly flow)<br>LOM            | $m_t - \bar{m}$ where $m_t$ = length of month $t$ (in days)<br>and $\bar{m} = 30.4375$ (average length of month)                                                     |
| Length-of-Quarter<br>(quarterly flow)<br>LOQ        | $q_t - \bar{q}$ where $q_t$ = length of quarter $t$ (in days)<br>and $\bar{q} = 91.3125$ (average length of quarter)                                                 |
| Leap Year<br>(monthly and quarterly flow)<br>LPYEAR | $LY_t = \begin{cases} 0.75 & \text{in leap year February (first quarter)} \\ -0.25 & \text{in other Februaries (first quarter)} \\ 0 & \text{otherwise} \end{cases}$ |

## TRANSFORM Statement

### TRANSFORM options;

The TRANSFORM statement transforms or adjusts the series prior to estimating a regARIMA model. With this statement, the series can be Box-Cox (power) transformed.

The following option can appear in the TRANSFORM statement.

#### POWER= value

Transform the input series  $Y_t$  using a Box-Cox power transformation,

$$Y_t \rightarrow y_t = \begin{cases} \log(Y_t) & \lambda = 0 \\ \lambda^2 + (Y_t^\lambda - 1)/\lambda & \lambda \neq 0 \end{cases}$$

The power  $\lambda$  must be specified (for example, POWER= .33). The default is no transformation ( $\lambda = 1$ ); that is, POWER= 1. The log transformation (POWER= 0), square root transformation (POWER= .5), and the inverse transformation (POWER= -1) are equivalent to the corresponding Census Bureau function argument.

**Table 5.2.** Power Values Related to the Census Bureau Function Argument

| function= | transformation          | range for $Y_t$          | equivalent power argument |
|-----------|-------------------------|--------------------------|---------------------------|
| none      | $Y_t$                   | all values               | $power = 1$               |
| log       | $\log(Y_t)$             | $Y_t > 0$ for all $t$    | $power = 0$               |
| sqrt      | $2(\sqrt{Y_t} - 0.875)$ | $Y_t \geq 0$ for all $t$ | $power = 0.5$             |
| inverse   | $2 - \frac{1}{Y_t}$     | $Y_t \neq 0$ for all $t$ | $power = -1$              |

Note that there are restrictions on the value used in the POWER option when preadjustment factors for seasonal adjustment are generated from a regARIMA model. When seasonal adjustment is requested with the X11 statement, any value of the POWER option can be used for the purpose of forecasting the series with a

regARIMA model. However, this is not the case when factors generated from the regression coefficients are used to adjust either the original series or the final seasonally adjusted series. In this case, the only accepted transformations are the log transformation, which can be specified as POWER = 0 (for multiplicative or log-additive seasonal adjustments) and no transformation, which can be specified as POWER = 1 (for additive seasonal adjustments). If no seasonal adjustment is performed, any POWER transformation can be used.

---

## VAR Statement

**VAR** *variables*;

The VAR statement is used to specify the variables in the input data set that are to be analyzed by the procedure. Only numeric variables can be specified. If the VAR statement is omitted, all numeric variables contained in the input data set are analyzed.

---

## X11 Statement

**X11** *options*;

The X11 statement is an optional statement for invoking seasonal adjustment by an enhanced version of the methodology of the Census Bureau X-11 and X-11Q programs. You can control the type of seasonal adjustment decomposition calculated with the MODE= option. The output includes the final tables and diagnostics for the X-11 seasonal adjustment method listed below in Table 5.3.

**Table 5.3.** Tables Related to X11 Seasonal Adjustment

| Table Name | Description                                                               |
|------------|---------------------------------------------------------------------------|
| B1         | original series, adjusted for prior effects and forecast extended         |
| C17        | final weights for the irregular component                                 |
| D8         | final unmodified SI ratios (differences)                                  |
| D8A        | $F$ tests for stable and moving seasonality, D8                           |
| D9         | final replacement values for extreme SI ratios (differences), D iteration |
| D9A        | moving seasonality ratios for each period                                 |
| D10        | final seasonal factors                                                    |
| D10D       | final seasonal difference                                                 |
| D11        | final seasonally adjusted series                                          |
| D12        | final trend-cycle                                                         |
| D13        | final irregular component                                                 |
| D16        | combined seasonal and trading day factors                                 |
| D16B       | final adjustment differences                                              |
| D18        | combined calendar adjustment factors                                      |
| E4         | ratio of yearly totals of original and seasonally adjusted series         |
| E5         | percent changes (differences) in original series                          |
| E6         | percent changes (differences) in seasonally adjusted series               |
| E7         | percent changes (differences) in final trend component series             |
| F2A - F2I  | X11 diagnostic summary                                                    |
| F3         | monitoring and quality assessment statistics                              |
| G          | spectral plots                                                            |

For more details on the X-11 seasonal adjustment diagnostics, refer to Shiskin, Young, and Musgrave (1967), Lothian and Morry (1978), and Ladiray and Quenneville (1999).

The following options can appear in the X11 statement.

**MODE= ADD**

**MODE= MULT**

**MODE= LOGADD**

**MODE= PSEUDOADD**

determines the mode of the seasonal adjustment decomposition to be performed. There are four choices: multiplicative (MODE=MULT), additive (MODE=ADD), pseudo-additive (MODE=PSEUDOADD), and log-additive (MODE=LOGADD) decomposition. If this option is omitted, the procedure performs multiplicative adjustments. Table 5.4 shows the values of the MODE= option and the corresponding models for the original (O) and the seasonally adjusted (SA) series.

**Table 5.4.** Modes of Seasonal Adjustment and Their Models

| Value of Mode Option | Name            | Model for $O$               | Model for $SA$     |
|----------------------|-----------------|-----------------------------|--------------------|
| mult                 | Multiplicative  | $O = C \times S \times I$   | $SA = C \times I$  |
| add                  | Additive        | $O = C + S + I$             | $SA = C + I$       |
| pseudoadd            | Pseudo-Additive | $O = C \times [S + I - 1]$  | $SA = C \times I$  |
| logadd               | Log-Additive    | $\text{Log}(O) = C + S + I$ | $SA = \exp(C + I)$ |



---

## Details

---

### Computations

---

For more details on the computations used in PROC X12, refer to *X-12-ARIMA Reference Manual* (U.S. Bureau of the Census 1999).

---

### Displayed Output/ODS Table Names

The options specified in PROC X12 control both the tables produced by the procedure and the tables available for output to the OUT= data set specified in the OUTPUT statement.

The displayed output is organized into tables identified by a part letter and a sequence number within the part. The six major parts of the X12 procedure are as follows:

- A prior adjustments (optional)
- B preliminary estimates of irregular component weights and trading-day regression factors
- C final estimates of irregular component weights and trading-day regression factors
- D final estimates of seasonal, trend cycle, and irregular components
- E analytical tables
- F summary measures
- G charts

Table 5.5 describes the individual tables and charts. “P” indicates that the table is displayed only and is not available for output to the OUT= data set. Data from displayed tables can be extracted into data sets using the Output Delivery System (ODS). Refer to Chapter 6, “Using the Output Delivery System,” in the *SAS/ETS User’s Guide*.

**Table 5.5.** Table Names and Descriptions

| Table                   | Description                                           | Notes |
|-------------------------|-------------------------------------------------------|-------|
| A1                      | original series                                       |       |
| RegParameterEstimates   | regression model parameter estimates                  | P     |
| ACF                     | autocorrelation factors                               | P     |
| PACF                    | partial autocorrelation factors                       | P     |
| ARMAIterationTolerances | exact ARMA likelihood estimation iteration tolerances | P     |
| ARMAIterationSummary    | exact ARMA likelihood estimation iteration summary    | P     |
| ARMAParameterEstimates  | exact ARMA maximum likelihood estimation              | P     |
| MLESummary              | estimation summary                                    | P     |
| ForecastCL              | forecasts, standard errors, and confidence limits     | P     |
| MV1                     | original series adjusted for missing value regressors |       |
| A6                      | regARIMA trading day component                        |       |
| A8                      | regARIMA combined outlier component                   |       |
| B1                      | prior adjusted or original series                     |       |
| C17                     | final weight for irregular components                 |       |
| D8                      | final unmodified S-I ratios                           |       |
| D8A                     | seasonality tests                                     | P     |
| D9                      | final replacement values for extreme S-I ratios       |       |
| D9A                     | moving seasonality ratio                              | P     |
| D10                     | final seasonal factors                                |       |
| D10D                    | final seasonal difference                             |       |
| D11                     | final seasonally adjusted series                      |       |
| D12                     | final trend cycle                                     |       |
| D13                     | final irregular series                                |       |
| D16                     | combined adjustment factors                           |       |
| D16B                    | final adjustment differences                          |       |
| D18                     | combined calendar adjustment factors                  |       |
| E4                      | ratios of annual totals                               | P     |
| E5                      | percent changes in original series                    |       |
| E6                      | percent changes in final seasonally adjusted series   |       |
| E7                      | differences in final trend cycle                      |       |
| F2A-I                   | summary measures                                      | P     |
| F3                      | quality assessment statistics                         | P     |
| G                       | spectral analysis                                     | P     |

The actual number of tables displayed depends on the options and statements specified. If a table is not computed, it is not displayed.

---

## Examples

---

### Example 5.1. Model Identification

An example of the statements typically invoked when using PROC X12 for model identification might follow the same format as the following example. This example invokes the X12 procedure and uses the TRANSFORM and IDENTIFY statements.

It specifies the time series data, takes the logarithm of the series (TRANSFORM statement), and generates ACFs and PACFs for the specified levels of differencing (IDENTIFY statement). The same data set can be used as in the section “Basic Seasonal Adjustment” on page 164.

```
proc x12 data=sales seasons=12 start=jul1972;
  var sales;
  transform power=0;
  identify diff=(0,1) sdiff=(0,1);
run ;
```

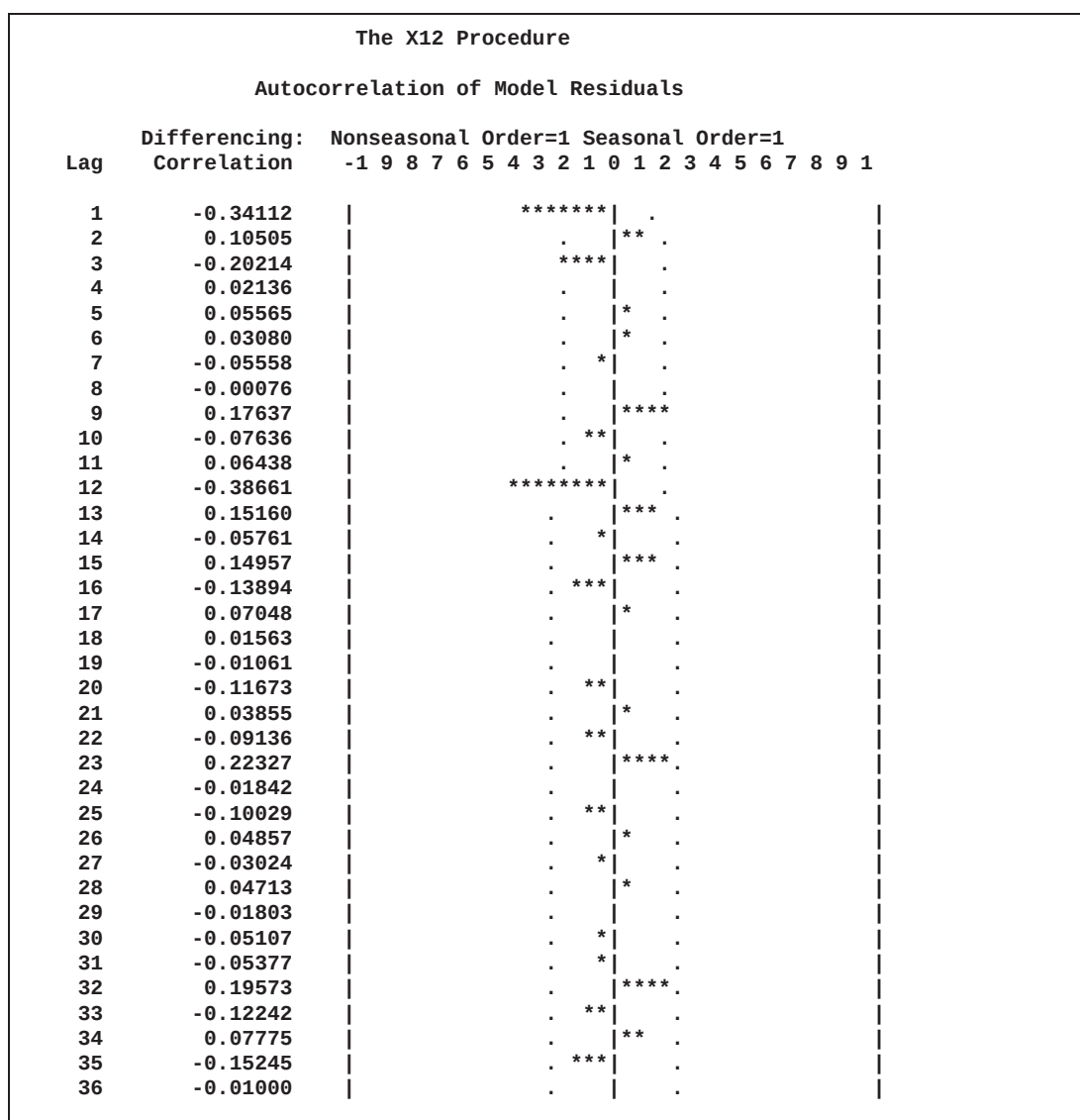


**Figure 5.3.** ACFs (Nonseasonal Order=1 Seasonal Order=1)

| Autocorrelation of Model Residuals                 |                |            |    |            |
|----------------------------------------------------|----------------|------------|----|------------|
| Differencing: Nonseasonal Order=1 Seasonal Order=1 |                |            |    |            |
| Lag                                                | Standard Error | Chi-Square | DF | Pr > ChiSq |
| 1                                                  | 0.08737        | 15.5957    | 1  | <.0001     |
| 2                                                  | 0.09701        | 17.0860    | 2  | 0.0002     |
| 3                                                  | 0.09787        | 22.6478    | 3  | <.0001     |
| 4                                                  | 0.10101        | 22.7104    | 4  | 0.0001     |
| 5                                                  | 0.10104        | 23.1387    | 5  | 0.0003     |
| 6                                                  | 0.10128        | 23.2709    | 6  | 0.0007     |
| 7                                                  | 0.10135        | 23.7050    | 7  | 0.0013     |
| 8                                                  | 0.10158        | 23.7050    | 8  | 0.0026     |
| 9                                                  | 0.10158        | 28.1473    | 9  | 0.0009     |
| 10                                                 | 0.10389        | 28.9869    | 10 | 0.0013     |
| 11                                                 | 0.10432        | 29.5887    | 11 | 0.0018     |
| 12                                                 | 0.10462        | 51.4728    | 12 | <.0001     |
| 13                                                 | 0.11501        | 54.8664    | 13 | <.0001     |
| 14                                                 | 0.11653        | 55.3605    | 14 | <.0001     |
| 15                                                 | 0.11674        | 58.7204    | 15 | <.0001     |
| 16                                                 | 0.11820        | 61.6452    | 16 | <.0001     |
| 17                                                 | 0.11944        | 62.4045    | 17 | <.0001     |
| 18                                                 | 0.11975        | 62.4421    | 18 | <.0001     |
| 19                                                 | 0.11977        | 62.4596    | 19 | <.0001     |
| 20                                                 | 0.11978        | 64.5984    | 20 | <.0001     |
| 21                                                 | 0.12064        | 64.8338    | 21 | <.0001     |
| 22                                                 | 0.12074        | 66.1681    | 22 | <.0001     |
| 23                                                 | 0.12126        | 74.2099    | 23 | <.0001     |
| 24                                                 | 0.12436        | 74.2652    | 24 | <.0001     |
| 25                                                 | 0.12438        | 75.9183    | 25 | <.0001     |
| 26                                                 | 0.12500        | 76.3097    | 26 | <.0001     |
| 27                                                 | 0.12514        | 76.4629    | 27 | <.0001     |
| 28                                                 | 0.12520        | 76.8387    | 28 | <.0001     |
| 29                                                 | 0.12533        | 76.8943    | 29 | <.0001     |
| 30                                                 | 0.12535        | 77.3442    | 30 | <.0001     |
| 31                                                 | 0.12551        | 77.8478    | 31 | <.0001     |
| 32                                                 | 0.12569        | 84.5900    | 32 | <.0001     |
| 33                                                 | 0.12799        | 87.2543    | 33 | <.0001     |
| 34                                                 | 0.12888        | 88.3401    | 34 | <.0001     |
| 35                                                 | 0.12924        | 92.5584    | 35 | <.0001     |
| 36                                                 | 0.13061        | 92.5767    | 36 | <.0001     |

NOTE: The P-values approximate the probability of observing a Q-value at least this large when the model fitted is correct. When DF is positive, small values of P, customarily those below 0.05 indicate model inadequacy.

Figure 5.4. ACFs (Nonseasonal Order=1 Seasonal Order=1)

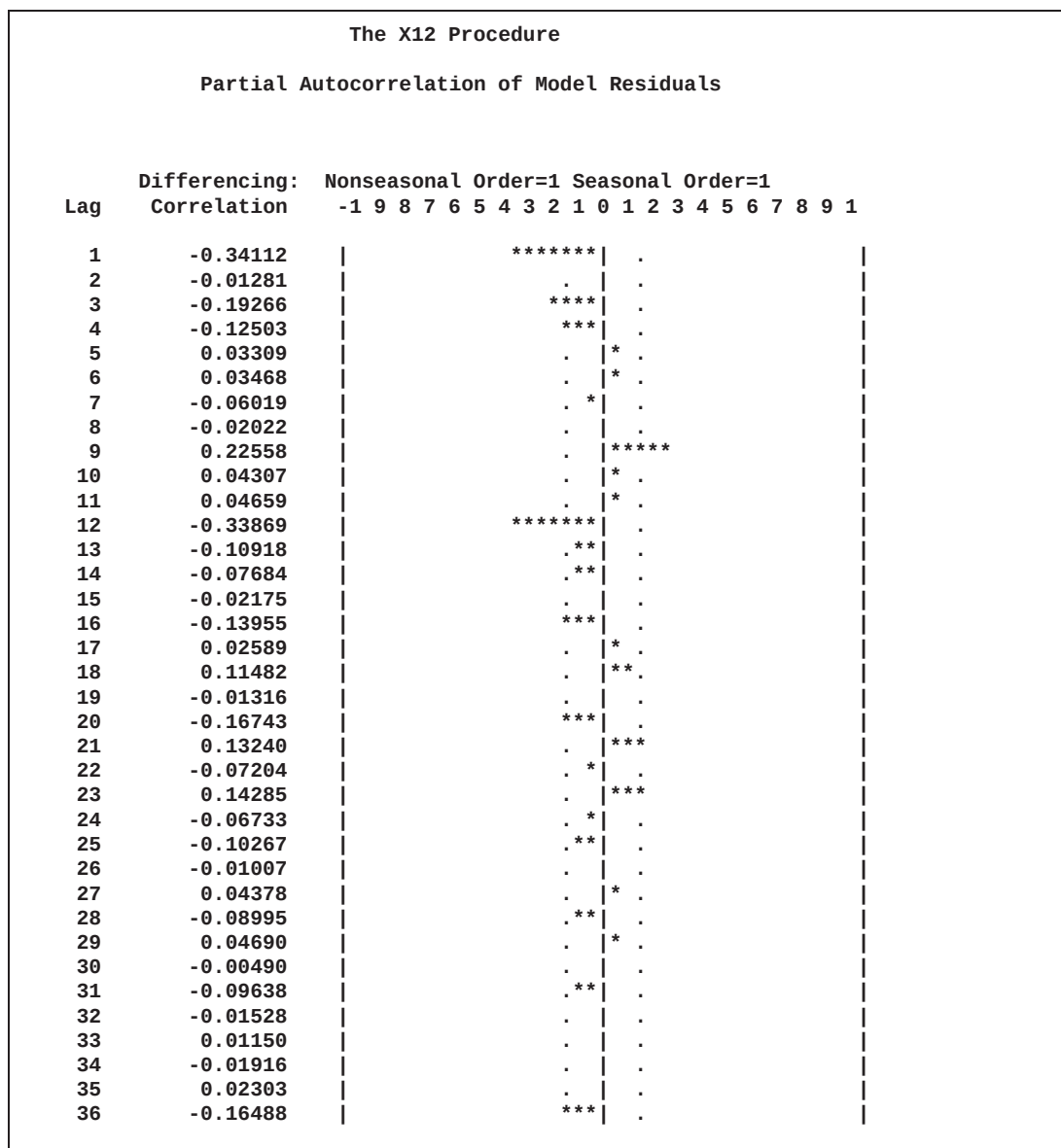


Figure 5.5. PACFs (Nonseasonal Order=1 Seasonal Order=1)

## Example 5.2. Model Estimation

After studying the output from Example 5.1 and identifying the ARIMA part of the model as, for example, (0 1 1)(0 1 1) 12, you can replace the IDENTIFY statement with the ARIMA and ESTIMATE statements. The resulting procedure and statements follow:

```
proc x12 data=sales seasons=12 start=jul1972;
  var sales;
  transform power=0;
  arima model=( 0,1,1)(0,1,1) );
  estimate;
run ;
```

| The X12 Procedure                                             |     |          |                |         |         |
|---------------------------------------------------------------|-----|----------|----------------|---------|---------|
| Exact ARMA Likelihood Estimation Iteration Tolerances         |     |          |                |         |         |
| Maximum Total ARMA Iterations                                 |     |          | 200            |         |         |
| Convergence Tolerance                                         |     |          | 1.0E-05        |         |         |
| Average absolute percentage error in within-sample forecasts: |     |          |                |         |         |
| Last year:                                                    |     |          | 2.81           |         |         |
| Last-1 year:                                                  |     |          | 6.38           |         |         |
| Last-2 year:                                                  |     |          | 7.69           |         |         |
| Last three years:                                             |     |          | 5.63           |         |         |
| Exact ARMA Likelihood Estimation Iteration Summary            |     |          |                |         |         |
| Number of ARMA iterations                                     |     |          | 6              |         |         |
| Number of Function Evaluations                                |     |          | 19             |         |         |
| Exact ARMA Maximum Likelihood Estimation                      |     |          |                |         |         |
| Parameter                                                     | Lag | Estimate | Standard Error | t Value | Pr >  t |
| Nonseasonal MA                                                | 1   | 0.40181  | 0.07887        | 5.09    | <.0001  |
| Seasonal MA                                                   | 12  | 0.55695  | 0.07626        | 7.30    | <.0001  |
| Estimation Summary                                            |     |          |                |         |         |
| Number of Residuals                                           |     |          | 131            |         |         |
| Number of Parameters Estimated                                |     |          | 3              |         |         |
| Variance Estimate                                             |     |          | 1.3E-03        |         |         |
| Standard Error Estimate                                       |     |          | 3.7E-02        |         |         |
| Log likelihood                                                |     |          | 244.6965       |         |         |
| Transformation Adjustment                                     |     |          | -735.2943      |         |         |
| Adjusted Log likelihood                                       |     |          | -490.5978      |         |         |
| AIC                                                           |     |          | 987.1956       |         |         |
| AICC (F-corrected-AIC)                                        |     |          | 987.3845       |         |         |
| Hannan Quinn                                                  |     |          | 990.7005       |         |         |
| BIC                                                           |     |          | 995.8211       |         |         |

Figure 5.6. Estimation Data as Printed in the List File

### Example 5.3. Seasonal Adjustment

Assuming that the model in Example 5.2 is satisfactory, a seasonal adjustment utilizing forecast extension can be performed by adding the X11 statement to the procedure. By default, the data is forecast one year ahead at the end of the series. The resulting PROC X12 step follows:

```
ods output D8A(match_all=list)=SalesD8A_1;
proc x12 data=sales date=date;
  var sales;
  transform power=0;
  arima model=( 0,1,1)(0,1,1) );
  estimate;
  x11;
  run;
```

```

proc print data=SalesD8A_1;
  title 'Stable Seasonality Test';
run;

proc print data=SalesD8A_2;
  title 'Nonparametric Stable Seasonality Test';
run;

proc print data=SalesD8A_3;
  title 'Moving Seasonality Test';
run;

```

The ODS statement in the preceding example directs output from the D8A table into three data sets: SalesD8A\_1, SalesD8A\_2, and SalesD8A\_3. (match\_all=list) is necessary because the three tables associated with table D8A have varying formats.

| The X12 Procedure                                                        |                                 |     |                      |          |    |
|--------------------------------------------------------------------------|---------------------------------|-----|----------------------|----------|----|
| Table D 8.A: F-tests for seasonality                                     |                                 |     |                      |          |    |
| Test for the presence of seasonality assuming stability.                 |                                 |     |                      |          |    |
|                                                                          | Sum of<br>Squares               | DF  | Mean<br>Square       | F-Value  |    |
| Between Months                                                           | 23571.41                        | 11  | 2142.855             | 190.9544 | ** |
| Residual                                                                 | 1481.28                         | 132 | 11.22182             |          |    |
| Total                                                                    | 25052.69                        | 143 |                      |          |    |
| ** Seasonality present at the 0.1 per cent level.                        |                                 |     |                      |          |    |
| Nonparametric Test for the Presence<br>of Seasonality Assuming Stability |                                 |     |                      |          |    |
|                                                                          | Kruskal-<br>Wallis<br>Statistic | DF  | Probability<br>Level |          |    |
|                                                                          | 131.9546                        | 11  | .00%                 |          |    |
| Seasonality present at the one percent level.                            |                                 |     |                      |          |    |
| Moving Seasonality Test                                                  |                                 |     |                      |          |    |
|                                                                          | Sum of<br>Squares               | DF  | Mean<br>Square       | F-Value  |    |
| Between Years                                                            | 259.2517                        | 10  | 25.92517             | 3.370317 | ** |
| Error                                                                    | 846.1424                        | 110 | 7.692204             |          |    |
| **Moving seasonality present at the one percent level.                   |                                 |     |                      |          |    |

Figure 5.7. Table D8.A as Printed in the List File

| Stable Seasonality Test |                |          |       |          |          |        |
|-------------------------|----------------|----------|-------|----------|----------|--------|
| Obs                     | FT_SRC         | FT_SS    | FT_DF | FT_MS    | FT_F     | FT_AST |
| 1                       | Between Months | 23571.41 | 11    | 2142.855 | 190.9544 | **     |
| 2                       | Residual       | 1481.28  | 132   | 11.22182 | .        |        |
| 3                       | Total          | 25052.69 | 143   | .        | .        |        |

| Nonparametric Stable Seasonality Test |          |       |       |
|---------------------------------------|----------|-------|-------|
| Obs                                   | KW_ST    | KW_DF | KW_PR |
| 1                                     | 131.9546 | 11    | .00%  |

| Moving Seasonality Test |               |          |       |          |          |        |
|-------------------------|---------------|----------|-------|----------|----------|--------|
| Obs                     | FT_SRC        | FT_SS    | FT_DF | FT_MS    | FT_F     | FT_AST |
| 1                       | Between Years | 259.2517 | 10    | 25.92517 | 3.370317 | **     |
| 2                       | Error         | 846.1424 | 110   | 7.692204 | .        |        |

Figure 5.8. Table D8.A as Output in a Data Set

---

## Acknowledgments

The X-12-ARIMA procedure was developed by the Time Series Staff of the Statistical Research Division, U.S. Bureau of the Census.

Brian Monsell is the primary programmer for the U.S. Census Bureau's X-12-ARIMA procedure and has been very helpful in the development of PROC X12.

The version of PROC X12 documented here was produced by converting the U.S. Census Bureau's FORTRAN code to the SAS development language and adding typical SAS procedure syntax. This conversion work was performed by SAS Institute, which resulted in the X12 procedure. Although several features were added during the conversion, credit for the statistical aspects and general methodology of the X12 procedure belongs to the U.S. Bureau of the Census.

The X-12-ARIMA seasonal adjustment program contains components developed from Statistics Canada's X-11-ARIMA program.



---

## References

- Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis: Forecasting and Control*, Third Edition, Englewood Cliffs, NJ: Prentice Hall, Inc.
- Dagum, E. B. (1988), *The X-11-ARIMA/88 Seasonal Adjustment Method: Foundations and User's Manual*, Ottawa: Statistics Canada.
- Findley, D. F., Monsell, B. C., Bell, W. R., Otto, M. C., and Chen, B. C. (1998), "New Capabilities and Methods of the X-12-ARIMA Seasonal Adjustment Program," *Journal of Business and Economic Statistics*, 16, 127–176 (with Discussion).
- Ladiray, D. and Quenneville B. (1999), "Understanding the X11 Method," Working Paper, Time Series Research and Analysis Centre, Statistics Canada, [[http://www.census.gov/pub/ts/papers/x11tb\\_en.ps](http://www.census.gov/pub/ts/papers/x11tb_en.ps)].
- Lothian, J. and Morry M., (1978), "A Test of Quality Control Statistics for the X-11-ARIMA Seasonal Adjustment Program," Research Paper, Seasonal Adjustment and Times Series Staff, Statistics Canada.
- Shiskin, J., Young, A. H., and Musgrave, J. C. (1967), "The X-11 Variant of the Census Method II Seasonal Adjustment Program," Technical Paper No. 15, U.S. Department of Commerce, Bureau of the Census.
- U.S. Bureau of the Census (1999), *X-12-ARIMA Reference Manual*, U.S. Department of Commerce, Washington, DC, [<ftp://ftp.census.gov/pub/ts/x12a/>].
- U.S. Bureau of the Census (1999), "X-12-ARIMA Seasonal Adjustment Program," [<ftp://ftp.census.gov/pub/ts/x12a/>].
- U.S. Bureau of the Census (1999), *X-12-ARIMA Quick Reference for Unix*, U.S. Department of Commerce, Washington, DC, [<ftp://ftp.census.gov/pub/ts/x12a/>].

# Part 3

## Investment Analysis

### Table of Contents

---

|                                   |     |
|-----------------------------------|-----|
| Chapter 6. Overview . . . . .     | 187 |
| Chapter 7. Portfolios . . . . .   | 193 |
| Chapter 8. Investments . . . . .  | 201 |
| Chapter 9. Computations . . . . . | 241 |
| Chapter 10. Analyses . . . . .    | 253 |
| Chapter 11. Details . . . . .     | 269 |



## Chapter 6

# Overview

### Chapter Table of Contents

---

|                                               |            |
|-----------------------------------------------|------------|
| <b>ABOUT INVESTMENT ANALYSIS . . . . .</b>    | <b>189</b> |
| <b>STARTING INVESTMENT ANALYSIS . . . . .</b> | <b>190</b> |
| <b>GETTING HELP . . . . .</b>                 | <b>190</b> |
| <b>USING HELP . . . . .</b>                   | <b>191</b> |



# Chapter 6

## Overview

---

### About Investment Analysis

The Investment Analysis system is an interactive environment for the time-value of money of a variety of investments:

- Loans
- Savings
- Depreciations
- Bonds
- Generic cashflows

Various analyses are provided to help analyze the value of investment alternatives: time value, periodic equivalent, internal rate of return, benefit-cost ratio, and breakeven analysis.

These analyses can help answer a number of questions you may have about your investments:

- Which option is more profitable or less costly?
- Is it better to buy or rent?
- Are the extra fees for refinancing at a lower interest rate justified?
- What is the balance of this account after saving this amount periodically for so many years?
- How much is legally tax-deductible?
- Is this a reasonable price?

Investment Analysis can be beneficial to users in many industries for a variety of decisions:

- manufacturing: cost justification of automation or any capital investment, replacement analysis of major equipment, or economic comparison of alternative designs
- government: setting funds for services
- finance: investment analysis and portfolio management for fixed-income securities

---

## Starting Investment Analysis

There are two ways to invoke Investment Analysis from the main SAS window. One way is to select **Solutions** → **Analysis** → **Investment Analysis** from the main SAS menu, as displayed in Figure 6.1.

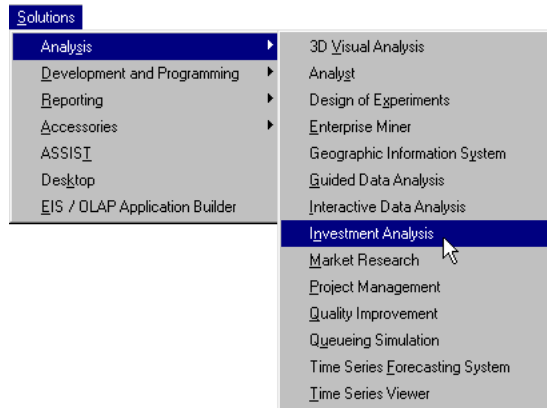


**Figure 6.1.** Initializing Investment Analysis with the Menu Bar

The other way is to type **INVEST** into the toolbar's command prompt, as displayed in Figure 6.2.



**Figure 6.2.** Initializing Investment Analysis with the Toolbar

---

## Getting Help

You can get help in Investment Analysis in three ways. One way is to use the Help Menu, as displayed in Figure 6.3. This is the right-most menu item on the menu bar.

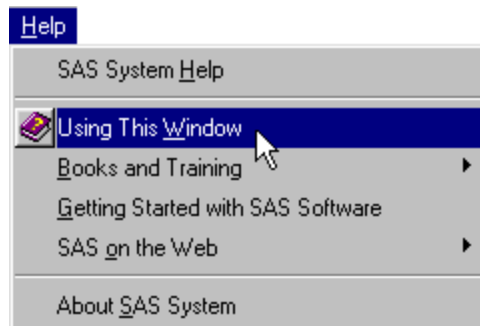


**Figure 6.3.** The Help Menu

Help buttons, as in Figure 6.4, provide another way to access help. Most dialog boxes provide help buttons in their lower-right corners.

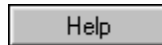


**Figure 6.4.** A Help Button

Also, the toolbar has a button (see Figure 6.5) that invokes the help system. This is the right-most icon on the toolbar.



**Figure 6.5.** The Help Icon

Each of these methods invokes a browser that gives specific help for the active window.

---

## Using Help

The chapters pertaining to Investment Analysis in this document typically have a section that introduces you to a menu and summarizes the options available through the menu. Such chapters then have sections titled Task and Dialog Box Guides. The Task section provides a description of how to perform many useful tasks. The Dialog Box Guide lists all dialog boxes pertinent to those tasks and gives a brief description of each element of each dialog box.





# Chapter 7

## Portfolios

### Chapter Table of Contents

---

|                                                    |     |
|----------------------------------------------------|-----|
| <b>THE FILE MENU</b> . . . . .                     | 195 |
| <b>TASKS</b> . . . . .                             | 196 |
| Creating a New Portfolio . . . . .                 | 196 |
| Saving a Portfolio . . . . .                       | 196 |
| Opening an Existing Portfolio . . . . .            | 196 |
| Saving a Portfolio to a Different Name . . . . .   | 197 |
| Selecting Investments within a Portfolio . . . . . | 197 |
| <b>DIALOG AND UTILITY GUIDE</b> . . . . .          | 198 |
| Investment Analysis . . . . .                      | 198 |
| Menubar Options . . . . .                          | 199 |
| Right-Clicking within the Portfolio Area . . . . . | 199 |



## Chapter 7

# Portfolios

---

### The File Menu

Investment Analysis stores portfolios as catalog entries. Portfolios contain a collection of investments, providing a structure to collect investments with a common purpose or goal (like a retirement or building fund portfolio). It may be advantageous also to collect investments into a common portfolio if they are competing investments you wish to perform a comparative analysis upon. Within this structure you can perform computations and analyses on a collection of investments in a portfolio, just as you would perform them on a single investment.

Investment Analysis provides many tools to aid in your manipulation of portfolios through the **File** menu, shown in Figure 7.1.

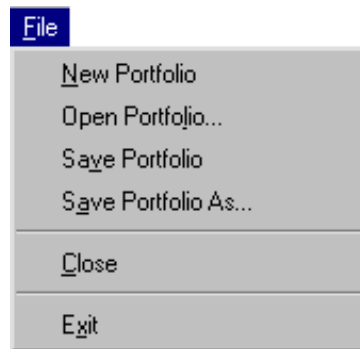


**Figure 7.1.** File Menu

The **File** menu offers the following items:

**New Portfolio** creates an empty portfolio with a new name.

**Open Portfolio...** opens the standard SAS Open dialog box where you select a portfolio to open.

**Save Portfolio** saves the current portfolio to its current name.

**Save Portfolio As...** opens the standard SAS Save As dialog box where you supply a new portfolio name for the current portfolio.

**Close** closes Investment Analysis.

**Exit** closes SAS (Windows only).

## Tasks

### Creating a New Portfolio

From the Investment Analysis dialog box, select **File** → **New Portfolio**.

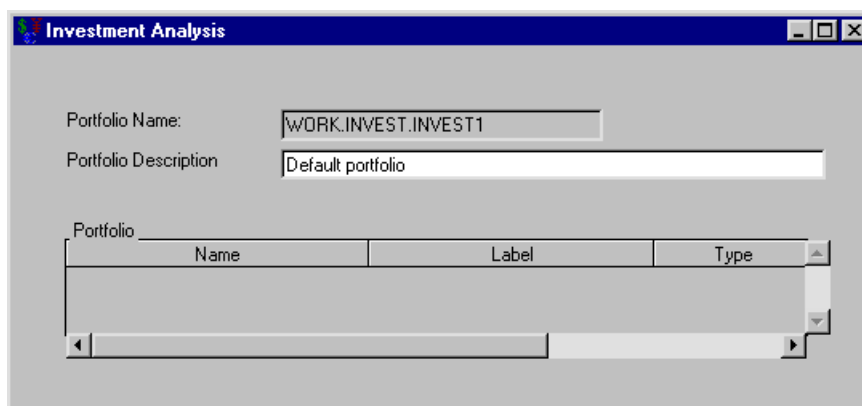


**Figure 7.2.** Creating a New Portfolio

The **Portfolio Name** will be WORK.INVEST.INVEST1 as displayed in Figure 7.2, unless you have saved a portfolio to that name in the past. In that case some other unused portfolio name is given to the new portfolio.

### Saving a Portfolio

From the Investment Analysis dialog box, select **File** → **Save Portfolio**. The portfolio is saved to a catalog-entry with the name in the **Portfolio Name** box.

### Opening an Existing Portfolio

From the Investment Analysis dialog box, select **File** → **Open Portfolio....** This opens the standard SAS Open dialog box. You enter the name of a SAS portfolio to open in the **Entry Name** box. For example, enter SASHELP.INVSAMP.NVST as displayed in Figure 7.3.

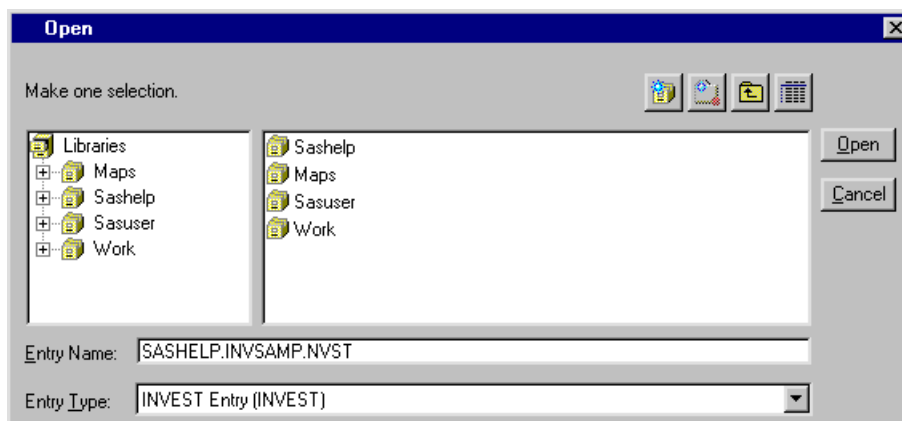


**Figure 7.3.** Opening an Existing Portfolio

Click **Open** to load the portfolio. The portfolio should look like Figure 7.4.

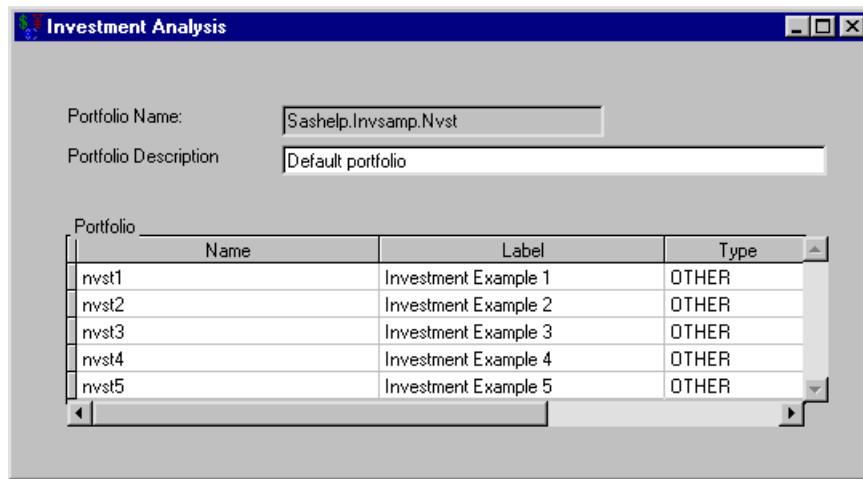


**Figure 7.4.** The Opened Portfolio

---

## Saving a Portfolio to a Different Name

From the Investment Analysis dialog box, select **File** → **Save Portfolio As....**

This opens the standard SAS Save As dialog box. You can enter the name of a SAS portfolio into the **Entry Name** box. For example, enter SASUSER.MY\_PORTS.PORT1 as in Figure 7.5.

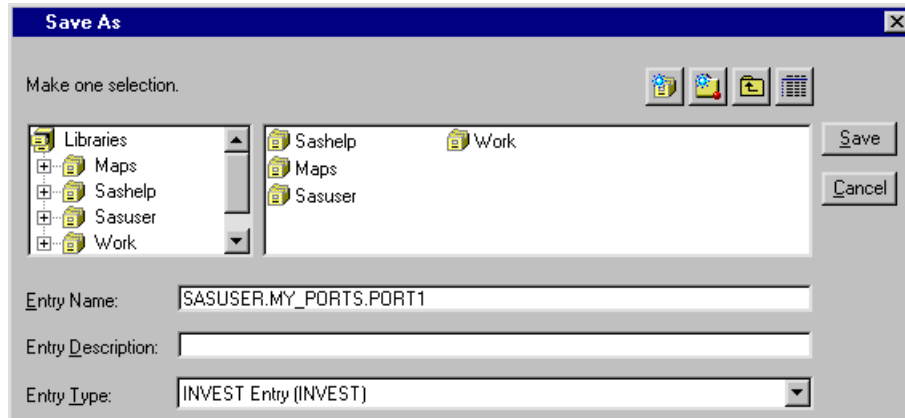


**Figure 7.5.** Saving a Portfolio to a Different Name

Click **Save** to save the portfolio.

---

## Selecting Investments within a Portfolio

To select a single investment in an opened portfolio, click the investment in the Portfolio area within the Investment Analysis dialog box.

To select a list of adjacent investments, do the following: click the first investment, hold down SHIFT, and click the final investment. Once the list of investment is selected, you may release the SHIFT key. The selected investments will appear highlighted as in Figure 7.6.

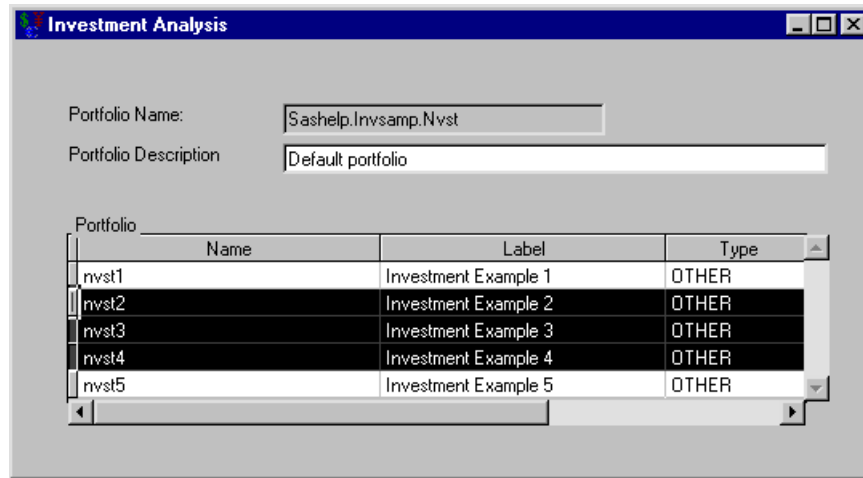


Figure 7.6. Selecting Investments within a Portfolio

---

## Dialog and Utility Guide

---

### Investment Analysis

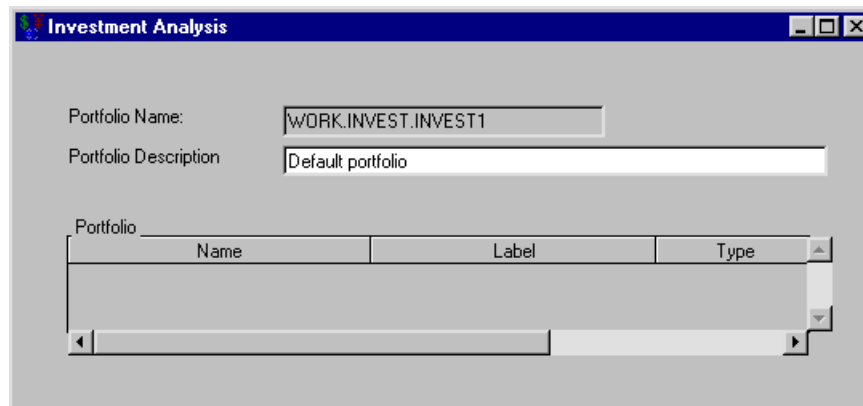


Figure 7.7. Investment Analysis Dialog Box

**Investment Portfolio Name** holds the name of the portfolio. It is of the form `library.catalog_entry.portfolio`. The default portfolio name is `work.invest.invest1`, as in Figure 7.7.

**Portfolio Description** provides a more descriptive explanation of the portfolio's contents. You can edit this description any time this dialog box is active.

The **Portfolio** area contains the list of investments comprising the particular portfolio. Each investment in the **Portfolio** area displays the following attributes:

**Name** is the name of the investment. It must be a valid SAS name. It is used to distinguish investments when performing analyses and computations.

**Label** is a place where you can provide a more descriptive explanation of the investment.

**Type** is the type of investment, which is fixed when you create the investment. It will be one of the following: LOAN, SAVINGS, DEPRECIATION, BOND, or OTHER.

Additional tools to aid in the management of your portfolio are available by selecting from the menu bar and by right-clicking within the **Portfolio** area.

---

## Menubar Options

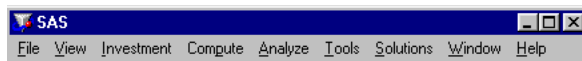


**Figure 7.8.** The Menu bar

The menu bar (shown in Figure 7.8) provides many tools to aid in the management of portfolios and the investments that comprise them. The following menu items provide functionality particular to Investment Analysis:

**File** opens and saves portfolios.

**Investment** creates new investments within the portfolio.

**Compute** performs constant dollar, after tax, and currency conversion computations on generic cashflows.

**Analyze** analyzes investments to aid in decision-making.

**Tools** sets default values of inflation and income tax rates.

---

## Right-Clicking within the Portfolio Area

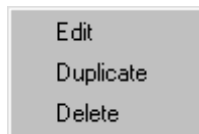


**Figure 7.9.** Right-Clicking

After selecting an investment, right-clicking in the **Portfolio** area pops up a menu (see Figure 7.9) that offers the following options:

**Edit** opens the selected investment within the portfolio.

**Duplicate** creates a duplicate of the selected investment within the portfolio.

**Delete** removes the selected investment from the portfolio.

If you wish to perform one of these actions on a collection of investments, you must select a collection of investments (as described in the section “Selecting Investments within a Portfolio” on page 197) before right-clicking.





# Chapter 8

## Investments

### Chapter Table of Contents

---

|                                                                      |     |
|----------------------------------------------------------------------|-----|
| <b>THE INVESTMENT MENU</b>                                           | 203 |
| <b>TASKS</b>                                                         | 204 |
| Loan Tasks                                                           | 204 |
| Specifying Savings Terms to Create an Account Summary                | 210 |
| Depreciation Tasks                                                   | 211 |
| Bond Tasks                                                           | 214 |
| Generic Cashflow Tasks                                               | 217 |
| <b>DIALOG BOX GUIDE</b>                                              | 223 |
| Loan                                                                 | 223 |
| Loan Initialization Options                                          | 225 |
| Loan Prepayments                                                     | 226 |
| Balloon Payments                                                     | 227 |
| Rate Adjustment Terms                                                | 227 |
| Rounding Off                                                         | 228 |
| Savings                                                              | 229 |
| Depreciation                                                         | 230 |
| Depreciation Table                                                   | 232 |
| Bond                                                                 | 232 |
| Bond Analysis                                                        | 234 |
| Bond Price                                                           | 235 |
| Generic Cashflow                                                     | 236 |
| Right-Clicking within Generic Cashflow's Cashflow Specification Area | 237 |
| Flow Specification                                                   | 238 |
| Forecast Specification                                               | 239 |



## Chapter 8

# Investments

---

### The Investment Menu

Because there are many types of investments, a tool that manages and analyzes collections of investments must be robust and flexible. Providing specifications for four specific investment types and one generic type, Investment Analysis can model almost any real-world investment.

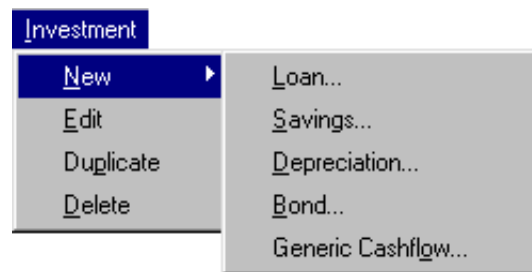


**Figure 8.1.** Investment Menu

The **Investment** menu, shown in Figure 8.1, offers the following items.

**New** → **Loan...** opens the Loan dialog box. Loans are useful for acquiring capital to pursue various interests. Available terms include rate adjustments for variable rate loans, initialization costs, prepayments, and balloon payments.

**New** → **Savings...** opens the Savings dialog box. Savings are necessary when planning for the future, whether for business or personal purposes. Account summary calculations available per deposit include starting balance, deposits, interest earned, and ending balance.

**New** → **Depreciation...** opens the Depreciation dialog box. Depreciations are prevalent for tax purposes. It is advantageous to deduct as much as possible as early as possible. The available depreciation methods are Straight Line, Sum-of-years Digits, Depreciation Table, and Declining Balance. SAS datasets containing the Modified Accelerated Cost Recovery System rates for various half-year conventions are available to load into the **Depreciation** area. Declining Balance with conversion to Straight Line is also provided.

**New** → **Bond...** opens the Bond dialog box. Bonds have widely varying terms depending on the issuer. As bond issuers frequently auction their bonds, the ability to price a bond between the issue date and maturity date is desirable. Fixed-coupon bonds may be analyzed for the following: price versus yield-to-maturity, duration, and convexity. These are available at different times in the bond's life.

**New** → **Generic Cashflow...** opens the Generic Cashflow dialog box. Generic cashflows are the most flexible investments. Only a sequence of date-amount pairs is necessary for specification. You can enter date-amount pairs and load values from SAS datasets to specify any type of investment. You can generate uniform, arithmetic, and geometric cashflows with ease. SAS's forecasting ability is available to forecast future cashflows as well. The new graphical display aids in visualization of the cashflow and enables the user to change the frequency of the cashflow view to aggregate and disaggregate the view.

**Edit** opens the specification dialog box for an investment selected within the portfolio.

**Duplicate** creates a duplicate of an investment selected within the portfolio.

**Delete** removes an investment selected from the portfolio.

If you wish to edit, duplicate, or delete a collection of investments, you must select a collection of investments as described in "Selecting Investments within a Portfolio" on page 197 before performing the menu-option.

---

## Tasks

---

### Loan Tasks

Suppose you want to buy a home that costs \$100,000. You can make a down payment of \$20,000. Hence, you need a loan of \$80,000. You are able to acquire a 30-year loan at 7% interest starting January 1, 2000. Let's use Investment Analysis to specify and analyze this loan.

From the Investment Analysis dialog box, select **Investment** → **New** → **Loan...** from the menu bar to open the Loan dialog box.

#### ***Specifying Loan Terms to Create an Amortization Schedule***

You must specify the loan before generating the amortization table. To specify the loan, follow these steps:

1. Enter **MORTGAGE** for the **Name**.
2. Enter 80000 for the **Loan Amount**.
3. Enter 7 for the **Initial Rate**.
4. Enter 360 for the **Number of Payments**.
5. Enter 01JAN2000 for the **Start Date**.

Once you have specified the loan, click **Create Amortization Schedule** to generate the amortization schedule displayed in Figure 8.2.

The 'Loan' dialog box is shown with the following fields and buttons:

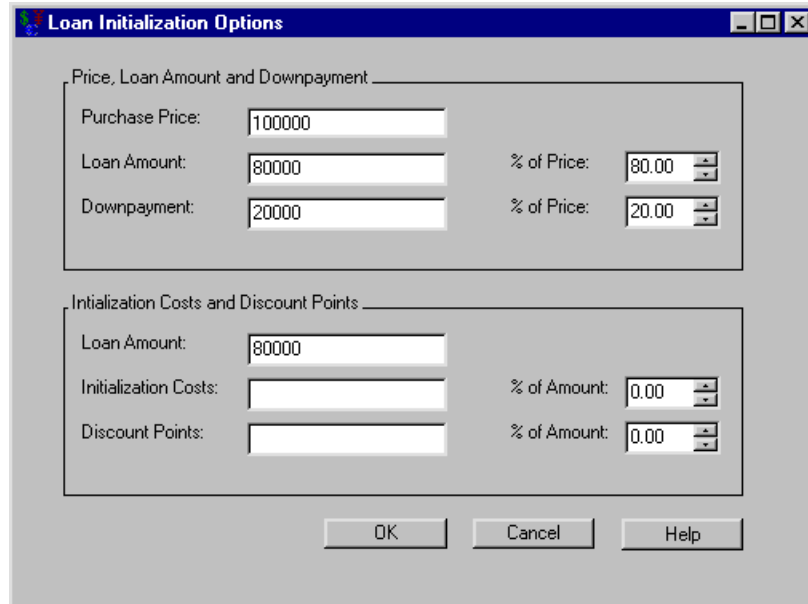
- Name:** MORTGAGE
- Loan Specification:**
  - Loan Amount:** 80000
  - Initial Rate:** 7.00
  - Periodic Payment:** (empty)
  - Start Date:** 01JAN2000
  - Number of Payments:** 360
  - Payment Interval:** MONTH
  - Compounding Interval:** MONTH
- Buttons:** Initialization..., Prepayments..., Balloon Payments..., Rate Adjustments..., Rounding Off...
- Amortization Schedule:**
  - Create Amortization Schedule** button
  - | Date    | Beginning Principal Amount | Periodic Payment Amount | Interest Payment | Principal Rep |
|---------|----------------------------|-------------------------|------------------|---------------|
| JAN2000 | 80000.00                   | 0.00                    | 0.00             | 0.00          |
| FEB2000 | 80000.00                   | 532.24                  | 466.67           | 65.57         |
| MAR2000 | 79934.43                   | 532.24                  | 466.28           | 65.96         |
| APR2000 | 79868.47                   | 532.24                  | 465.90           | 66.34         |
- Bottom Buttons:** Save Data As..., OK, Cancel, Help

**Figure 8.2.** Creating an Amortization Schedule

### Storing Other Loan Terms

Let's include information concerning the purchase price and downpayment. These terms are not necessary to specify the loan, but it may be advantageous to store such information with the loan.

Consider the loan described in "Loan Tasks" on page 204. From the Loan dialog box (Figure 8.2) click **Initialization...** to open the Loan Initialization Options dialog box. Here you can specify the down payment, initialization costs, and discount points. To specify the down payment, enter 100000 for the **Purchase Price**, as shown in Figure 8.3.



The "Loan Initialization Options" dialog box is divided into two sections. The first section, "Price, Loan Amount and Downpayment", contains three input fields: "Purchase Price" (100000), "Loan Amount" (80000), and "Downpayment" (20000). To the right of these are two percentage fields: "% of Price" for the loan amount (80.00) and the downpayment (20.00). The second section, "Initialization Costs and Discount Points", contains three input fields: "Loan Amount" (80000), "Initialization Costs", and "Discount Points". To the right of these are two percentage fields: "% of Amount" for the initialization costs (0.00) and the discount points (0.00). At the bottom are "OK", "Cancel", and "Help" buttons.

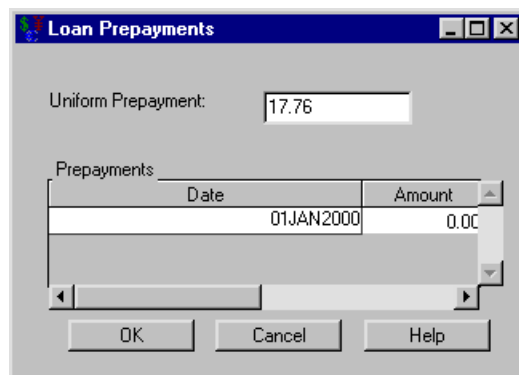
**Figure 8.3.** Including the Purchase Price

Click **OK** to return to the Loan dialog box.

### **Adding Prepayments**

Now let's observe the effect of prepayments on the loan. Consider the loan described in "Loan Tasks" on page 204. You must pay a minimum of \$532.24 each month to keep up with payments. However, let's say you dislike entering this amount in your checkbook. You would rather pay \$550.00 to keep the arithmetic simpler. This would constitute a uniform prepayment of \$17.76 each month.

From the Loan dialog box, click **Prepayments...** that opens the Loan Prepayments dialog box shown in Figure 8.4.



The "Loan Prepayments" dialog box features a "Uniform Prepayment" input field with the value 17.76. Below it is a table titled "Prepayments" with two columns: "Date" and "Amount". The first row shows a date of 01JAN2000 and an amount of 0.00. At the bottom are "OK", "Cancel", and "Help" buttons.

**Figure 8.4.** Specifying the Loan Prepayments

You can specify an arbitrary sequence of prepayments in the **Prepayments** area. If you want a uniform prepayment, clear the **Prepayments** area and enter the uniform payment amount in the **Uniform Prepayment** box. That amount will be added to each payment until the loan is paid off.

To specify this uniform prepayment, follow these steps:

1. Enter 17.76 for the **Uniform Prepayment**.
2. Click **OK** to return to the Loan dialog box.
3. Click **Create Amortization Schedule**, and the amortization schedule updates, as displayed in Figure 8.5.

The screenshot shows the 'Loan' dialog box with the 'Name' field set to 'MORTGAGE'. Under 'Loan Specification', the 'Loan Amount' is 80000, 'Initial Rate' is 7.00, 'Periodic Payment' is empty, 'Start Date' is 01JAN2000, 'Number of Payments' is 360, 'Payment Interval' is MONTH, and 'Compounding Interval' is MONTH. On the right, there are buttons for 'Initialization...', 'Prepayments...', 'Balloon Payments...', 'Rate Adjustments...', and 'Rounding Off...'. Below the specification is the 'Amortization Schedule' section with a 'Create Amortization Schedule' button. The schedule table shows the following data:

| Date    | Beginning Principal Amount | Periodic Payment Amount | Interest Payment | Principal Rep |
|---------|----------------------------|-------------------------|------------------|---------------|
| JAN2000 | 80000.00                   | 0.00                    | 0.00             | 0.00          |
| FEB2000 | 80000.00                   | 550.00                  | 466.67           | 83.33         |
| MAR2000 | 79916.67                   | 550.00                  | 466.18           | 83.82         |
| APR2000 | 79832.85                   | 550.00                  | 465.69           | 84.31         |

At the bottom of the dialog box are buttons for 'Save Data As...', 'OK', 'Cancel', and 'Help'.

**Figure 8.5.** The Amortization Schedule with Loan Prepayments

The last payment is on January 2030 without prepayments and February 2027 with prepayment; you would pay the loan off almost three years earlier with the \$17.76 prepayments.

To continue this example you must remove the prepayments from the loan specification, following these steps:

1. Return to the Prepayments dialog box from the Loan dialog box by clicking **Prepayments....**
2. Enter 0 for **Uniform Prepayment**.
3. Click **OK** to return to the Loan dialog box.

### **Adding Balloon Payments**

Consider the loan described in “Loan Tasks” on page 204. Suppose you cannot afford the payments of \$532.24 each month. To lessen your monthly payment you could pay balloon payments of \$6,000 at the end of 2007 and 2023. You wonder how this would affect your monthly payment.

From the Loan dialog box, click **Balloon Payments...**, which opens the Balloon Payments dialog box shown in Figure 8.6.



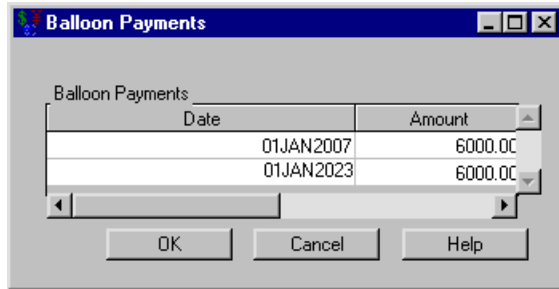


**Figure 8.6.** Defining Loan Balloon Payments

You can specify an arbitrary sequence of balloon payments by adding date-amount pairs to the **Balloon Payments** area.

To specify these balloon payments, follow these steps:

1. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **New**.
2. Set the pair's **Date** to 01JAN2007.
3. Set its **Amount** to 6000.
4. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **New**.
5. Set the new pair's **Date** to 01JAN2023.
6. Set its **Amount** to 6000.

Click **OK** to return to the Loan dialog box. Click **Create Amortization Schedule**, and the amortization schedule updates. Your monthly payment is now \$500.30, a difference of approximately \$32 each month.

To continue this example you must remove the balloon payments from the loan specification, following these steps:

1. Return to the Balloon Payments dialog box.
2. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **Clear**.
3. Click **OK** to return to the Loan dialog box.

### **Handling Rate Adjustments**

Consider the loan described in “Loan Tasks” on page 204. Another option for lowering your payments is to get a variable rate loan. You can acquire a three-year-ARM at 6% with a periodic cap of 1% with a maximum of 9%.

From the Loan dialog box, click **Rate Adjustments...** to open the Rate Adjustment Terms dialog box shown in Figure 8.7.

**Rate Adjustment Terms**

Rate Adjustment Terms:

Life Cap: 3.00

Periodic Cap: 1.00

Adjustment Frequency: 36 months

Rate Adjustment Assumption:

☒ Worst Case

☐ Best Case

☐ Fixed Case

☐ Estimated Case

Estimated Rates

| Date | Rate |
|------|------|
|------|------|

OK Cancel Help

**Figure 8.7.** Setting the Rate Adjustments

To specify these loan adjustment terms, follow these steps:

1. Enter 3 for the **Life Cap**. The **Life Cap** is the maximum deviation from the **Initial Rate**.
2. Enter 1 for the **Periodic Cap**.
3. Enter 36 for the **Adjustment Frequency**.
4. Confirm that **Worst Case** is selected in the Rate Adjustment Assumption area.
5. Click **OK** to return to the Loan dialog box.
6. Enter 6 for the **Initial Rate**.

Click **Create Amortization Schedule**, and the amortization schedule updates. Your monthly payment drops to \$479.64 each month. However, if the worst-case scenario plays out, the payments will increase to \$636.84 in nine years. Figure 8.8 displays amortization table information for the final few months under this scenario.

The screenshot shows a 'Loan' dialog box with the following fields and buttons:

- Name: untitled\_loan
- Loan Specification:
  - Loan Amount: 80000
  - Initial Rate: 6.00
  - Periodic Payment: (empty)
  - Start Date: 01JAN2000
  - Number of Payments: 360
  - Payment Interval: MONTH
  - Compounding Interval: MONTH
- Buttons: Initialization..., Prepayments..., Balloon Payments..., Rate Adjustments..., Rounding Off...
- Amortization Schedule:
 

| Date    | Beginning Principal Amount | Periodic Payment Amount | Interest Payment | Principal Repay |
|---------|----------------------------|-------------------------|------------------|-----------------|
| AUG2029 | 3722.76                    | 636.84                  | 27.92            | 608.92          |
| SEP2029 | 3113.84                    | 636.84                  | 23.35            | 613.49          |
| OCT2029 | 2500.35                    | 636.84                  | 18.75            | 618.09          |
| NOV2029 | 1882.26                    | 636.84                  | 14.12            | 622.72          |
| DEC2029 | 1259.54                    | 636.84                  | 9.45             | 627.39          |
| JAN2030 | 632.15                     | 636.89                  | 4.74             | 632.15          |
- Buttons: Save Data As..., OK, Cancel, Help

**Figure 8.8.** The Amortization Schedule with Rate Adjustments

Click **OK** to return to the Investment Analysis dialog box.

## Specifying Savings Terms to Create an Account Summary

Suppose you put \$500 each month into an account that earns 6% interest for 20 years. What is the balance of the account after those 20 years?

From the Investment Analysis dialog box, select **Investment** → **New** → **Savings...** from the menu bar to open the Savings dialog box.

To specify the savings, follow these steps:

1. Enter **RETIREMENT** for the **Name**.
2. Enter 500 for the **Periodic Deposit**.
3. Enter 240 for the **Number of Deposits**.
4. Enter 6 for the **Initial Rate**.

You must specify the savings before generating the account summary. Once you have specified the savings, click **Create Account Summary** to compute the ending date and balance and to generate the account summary displayed in Figure 8.9.

**Savings**

Name:

Savings Specification

Periodic Deposit:  Start Date:

Number of Deposits:  Deposit Interval:

Initial Rate:  Compounding Interval:

Ending Date:  Balance:

Account Summary

| Date      | StartingBalance | Deposits | InterestEarned | EndingBalance |
|-----------|-----------------|----------|----------------|---------------|
| 01JAN2000 | 0.00            | 500.00   | 0.00           | 500.00        |
| 01FEB2000 | 500.00          | 500.00   | 2.50           | 1002.50       |
| 01MAR2000 | 1002.50         | 500.00   | 5.01           | 1507.51       |

**Figure 8.9.** Creating an Account Summary

Click **OK** to return to the Investment Analysis dialog box.

## Depreciation Tasks

Commercial assets are considered to lose value as time passes. For tax purposes, you want to quantify this loss. This investment structure helps calculate appropriate values.

Suppose you buy a boat that costs \$50,000 for commercial fishing that is considered to have a ten-year useful life. How would you depreciate it?

From the Investment Analysis dialog box, select **Investment** → **New** → **Depreciation...** from the menu bar to open the Depreciation dialog box.

### ***Specifying Depreciation Terms to Create a Depreciation Table***

To specify the depreciation, follow these steps:

1. Enter **FISHING\_BOAT** for the **Name**.
2. Enter 50000 for the **Cost**.
3. Enter 2000 for the **Year of Purchase**.
4. Enter 10 for the **Useful Life**.
5. Enter 0 for the **Salvage Value**.

You must specify the depreciation before generating the depreciation schedule. Once you have specified the depreciation, click **Create Depreciation Schedule** to generate a depreciation schedule like the one displayed in Figure 8.10.

**Depreciation**

Name:

Depreciable Asset Specification

Cost:

Year of Purchase:

Useful Life:

Salvage Value:

Depreciation Method

☐ Straight Line (SL)

☐ Sum-of-years-digits

☐ Depreciation Table....

☒ Declining Balance (DB)

DB Factor: ☒ 2 ☐ 1.5 ☐ 1

Conversion to SL: ☒ Yes ☐ No

Depreciation Schedule

| Year | StartBookValue | Depreciation | EndBookValue |
|------|----------------|--------------|--------------|
| 2000 | 50000.00       | 10000.00     | 40000.00     |
| 2001 | 40000.00       | 8000.00      | 32000.00     |
| 2002 | 32000.00       | 6400.00      | 25600.00     |
| 2003 | 25600.00       | 5120.00      | 20480.00     |
| 2004 | 20480.00       | 4096.00      | 16384.00     |

**Figure 8.10.** Creating a Depreciation Schedule

The default depreciation method is Declining Balance (with Conversion to Straight Line). Try the following methods to see how they each affect the schedule:

- Straight Line
- Sum-of-years Digits
- Declining Balance (without conversion to Straight Line)

It might be useful to compare the value of the boat at 5 years for each method.

A description of these methods is available in “Depreciation Methods” on page 274.

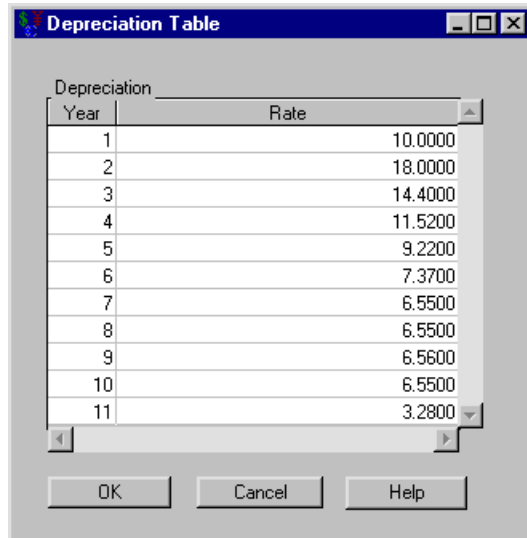
### **Using the Depreciation Table**

Sometimes you want to force the depreciation rates to be certain percentages each year. This option is particularly useful for calculating Modified Accelerated Cost Recovery System (MACRS) Depreciations. The United States’ Tax Reform Act of 1986 set depreciation rates for an asset based on an assumed lifetime for that asset. Since these lists of rates are important to many people, Investment Analysis provides SAS Datasets for situations with yearly rates (using the “half-year convention”). Find them at `SASHELP.MACRS*` where \* refers to the class of the property. For example, use `SASHELP.MACRS15` for a fifteen-year property. (When using the MACRS with the Tax Reform Act tables, you must set the **Salvage Value** to zero.)

Suppose you want to compute the depreciation schedule for the commercial fishing boat described in “Depreciation Tasks” on page 211. The boat is a ten-year property according to the Tax Reform Act of 1986.

To employ the MACRS depreciation from the Depreciation dialog box, follow these steps:

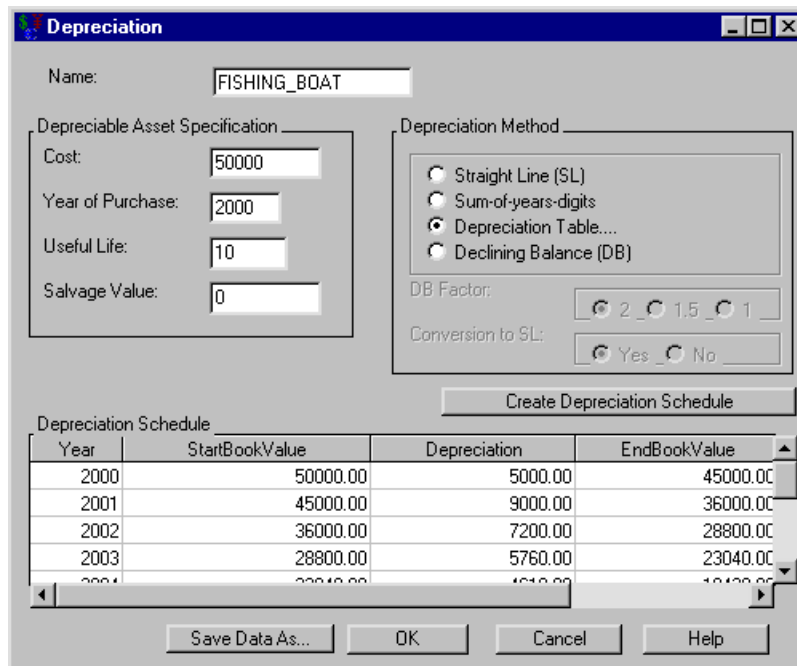
1. Click **Depreciation Table...** within the **Depreciation Method** area. This opens the Depreciation Table dialog box.
2. Right-click within the **Depreciation** area (which pops up a menu) and select **Load**.
3. Enter **SASHELP.MACRS10** for the **Dataset Name**. The dialog box should look like Figure 8.11.



| Year | Rate    |
|------|---------|
| 1    | 10.0000 |
| 2    | 18.0000 |
| 3    | 14.4000 |
| 4    | 11.5200 |
| 5    | 9.2200  |
| 6    | 7.3700  |
| 7    | 6.5500  |
| 8    | 6.5500  |
| 9    | 6.5600  |
| 10   | 6.5500  |
| 11   | 3.2800  |

**Figure 8.11.** MACRS Percentages for a Ten-Year Property

Click **OK** to return to the Depreciation dialog box. Click **Create Depreciation Schedule** and the depreciation schedule fills (see Figure 8.12).



| Year | StartBookValue | Depreciation | EndBookValue |
|------|----------------|--------------|--------------|
| 2000 | 50000.00       | 5000.00      | 45000.00     |
| 2001 | 45000.00       | 9000.00      | 36000.00     |
| 2002 | 36000.00       | 7200.00      | 28800.00     |
| 2003 | 28800.00       | 5760.00      | 23040.00     |
| 2004 | 23040.00       | 4608.00      | 18432.00     |

**Figure 8.12.** Depreciation Table with MACRS10

Note there are eleven entries in this depreciation table. This is because of the half-year convention that enables you to deduct one half of a year the first year which leaves a half year to deduct after the useful life is over.

Click **OK** to return to the Investment Analysis dialog box.

---

## Bond Tasks

Suppose someone offers to sell you a 20-year utility bond. It was issued six years ago. It has a \$1,000 face value and pays semi-year coupons at 2%. You can purchase it for \$780. Would you be satisfied with this bond if you expect an 8% MARR?

From the Investment Analysis dialog box, select **Investment** → **New** → **Bond...** from the menu bar to open the Bond dialog box.

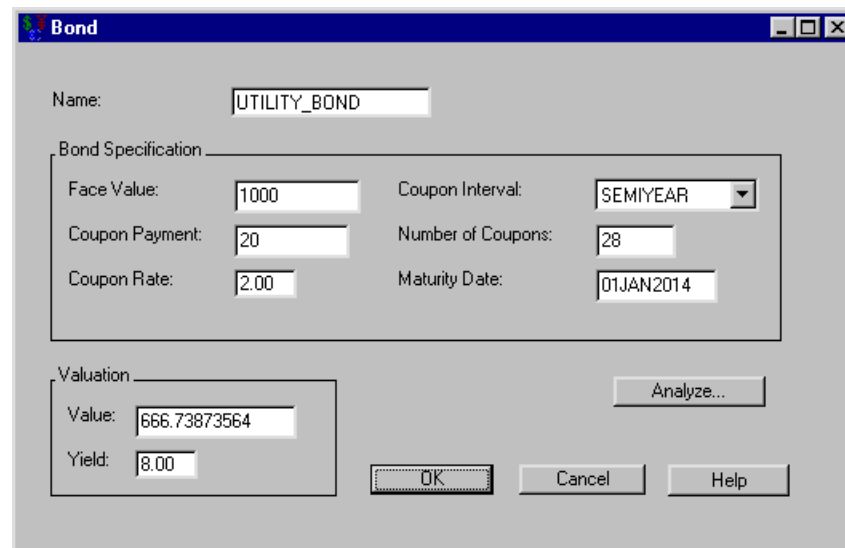
### Specifying Bond Terms

To specify the bond, follow these steps:

1. Enter **UTILITY\_BOND** for the **Name**.
2. Enter 1000 for the **Face Value**.
3. Enter 2 for the **Coupon Rate**. The **Coupon Payment** updates to 20.
4. Select SEMIYEAR for **Coupon Interval**.
5. Enter 28 for the **Number of Coupons**. As 14 years remain before the bond matures, the bond still has 28 semiyear coupons to pay. The **Maturity Date** updates.

### Computing the Price from Yield

Enter 8 for **Yield** within the **Valuation** area. You see the bond's value would be \$666.72 as in Figure 8.13.



The screenshot shows a Windows-style dialog box titled "Bond". It contains two main sections: "Bond Specification" and "Valuation".

**Bond Specification:**

- Name: UTILITY\_BOND
- Face Value: 1000
- Coupon Interval: SEMIYEAR (selected from a dropdown)
- Coupon Payment: 20
- Number of Coupons: 28
- Coupon Rate: 2.00
- Maturity Date: 01JAN2014

**Valuation:**

- Value: 666.73873564
- Yield: 8.00

Buttons at the bottom include "Analyze...", "OK", "Cancel", and "Help".

Figure 8.13. Bond Value

### Computing the Yield from Price

Now enter 780 for **Value** within the **Valuation** area. You see the yield is only 6.5%, as in Figure 8.14. This is not acceptable if you desire an 8% MARR.

The screenshot shows a Windows-style dialog box titled "Bond". It contains two main sections: "Bond Specification" and "Valuation".

**Bond Specification:**

- Name: UTILITY\_BOND
- Face Value: 1000
- Coupon Interval: SEMIYEAR (dropdown menu)
- Coupon Payment: 20
- Number of Coupons: 28
- Coupon Rate: 2.00
- Maturity Date: 01JAN2014

**Valuation:**

- Value: 780
- Yield: 6.51

Buttons: "Analyze...", "OK", "Cancel", and "Help".

Figure 8.14. Bond Yield

### Performing Bond Analysis

To perform bond-pricing analysis, follow these steps:

1. Click **Analyze...** to open the Bond Analysis dialog box.
2. Enter 8.0 as the **Yield to Maturity**.
3. Enter 4.0 as the **+/-**.
4. Enter 0.5 as the **Increment by**.
5. Enter 780 as the **Reference Price**.
6. Click **Create Bond Valuation Summary**.

The **Bond Valuation Summary** area fills and shows you the different values for various yields as in Figure 8.15.



**Bond Analysis**

Analysis Specifications

Yield-to-Maturity: 8.00  
 +/-: 4.00  
 Increment By: 0.50  
 Reference Price: 780.00

Analysis Dates  
 Date: 01JAN2000

Create Bond Valuation Summary

Bond Valuation Summary

| Date      | Yield | Value  | PercentChange | Duration | Convexity |
|-----------|-------|--------|---------------|----------|-----------|
| 01JAN2000 | 12.00 | 463.75 | -40.54        | 8.15     | 94.47     |
| 01JAN2000 | 11.50 | 484.13 | -37.93        | 8.29     | 96.93     |
| 01JAN2000 | 11.00 | 505.75 | -35.16        | 8.43     | 99.43     |

Graphics... Save Data As... Return Help

Figure 8.15. Bond Price Analysis

### Creating a Price versus Yield-to-Maturity Graph

Click **Graphics...** to open the Bond Price dialog box. This contains the price versus yield-to-maturity graph shown in Figure 8.16.

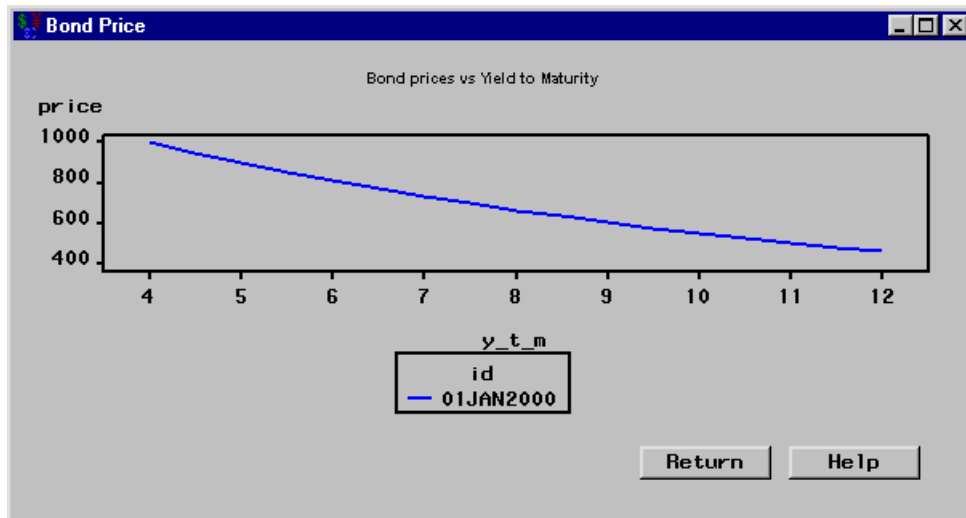


Figure 8.16. Bond Price Graph

Click **Return** to return to the Bond Analysis dialog box. Click **OK** to return to the Bond dialog box. Click **OK** to return to the Investment Analysis dialog box.

## Generic Cashflow Tasks

To specify a generic cashflow, you merely define any sequence of date-amount pairs. The flexibility of generic cashflows enables the user to represent economic alternatives/investments that do not fit into loan, savings, depreciation, or bond specifications.

From the Investment Analysis dialog box, select **Investment** → **New** → **Generic Cashflow...** from the menu bar to open the **Generic Cashflow** dialog box. Enter RETAIL for the **Name** as in Figure 8.17.

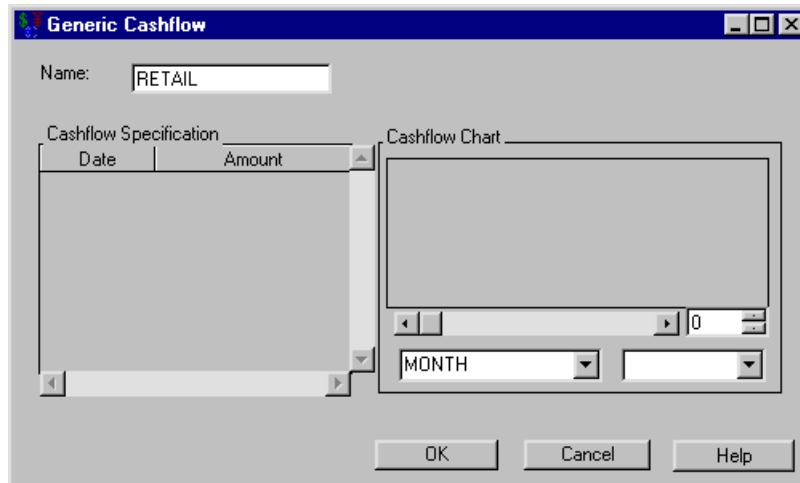


Figure 8.17. Introducing the Generic Cashflow

### *Right-Clicking within the Cashflow Specification Area*

Right-clicking within Generic Cashflow's **Cashflow Specification** area reveals the pop-up menu displayed in Figure 8.18. The menu provides many useful tools to assist you in creating these date-amount pairs.

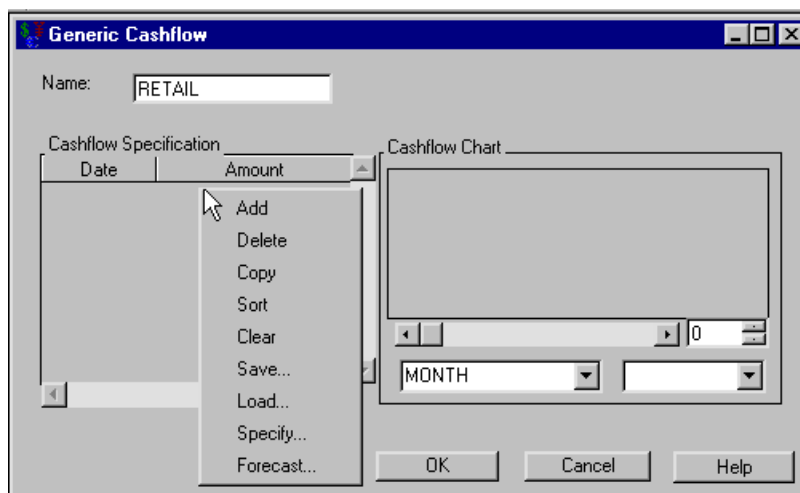


Figure 8.18. Right-Clicking within the Cashflow Specification Area

The following sections describe how to use most of these right-click options. The **Specify...** and **Forecast...** options are described in “Including a Generated Cashflow” and “Including a Forecasted Cashflow”.

#### **Adding a New Date-Amount Pair**

To add a new date-amount pair manually, follow these steps:

1. Right-click in the **Cashflow Specification** area as shown in Figure 8.18, and release on **Add**.
2. Enter 01JAN01 for the date.
3. Enter 100 for the amount.

#### **Copying a Date-Amount Pair**

To copy a selected date-amount pair, follow these steps:

1. Select the pair you just created.
2. Right-click in the **Cashflow Specification** area as shown in Figure 8.18, but this time release on **Copy**.

#### **Sorting All of the Date-Amount Pairs**

Change the second date to 01JAN00. Now the dates are unsorted. Right-click in the **Cashflow Specification** area as shown in Figure 8.18, and release on **Sort**.

#### **Deleting a Date-Amount Pair**

To delete a selected date-amount pair, follow these steps:

1. Select a date-amount pair.
2. Right-click in the **Cashflow Specification** area as shown in Figure 8.18, and release on **Delete**.

#### **Clearing All of the Date-Amount Pairs**

To clear all date-amount pairs, right-click in the **Cashflow Specification** area as shown in Figure 8.18, and release on **Clear**.

#### **Loading Date-Amount Pairs from a Dataset**

To load date-amount pairs from a dataset into the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Load....** This opens the Load Dataset dialog box.
2. Enter SASHELP.RETAIL for **Dataset Name**.
3. Click **OK** to return to the Generic Cashflow dialog box.

If there is a **Date** variable in the dataset, Investment Analysis loads it into the list. If there is no **Date** variable, it loads the first available date or datetime-formatted variable. Investment Analysis then searches the dataset for an **Amount** variable to use. If none exists, it takes the first numeric variable that is not used by the **Date** variable.

### **Saving Date-Amount Pairs to a Dataset**

To save date-amount pairs from the **Cashflow Specification** area to a dataset, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Save....** This opens the Save Dataset dialog box.
2. Enter the name of the dataset for **Dataset Name**.
3. Click **OK** to return to the Generic Cashflow dialog box.

### **Including a Generated Cashflow**

To generate date-amount pairs for the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Specify....** This opens the **Flow Specification** dialog box.
2. Select **YEAR** for the **Time Interval**.
3. Enter today's date for the **Starting Date**.
4. Enter 10 for the **Number of Periods**. The **Ending Date** updates.
5. Enter 100 for the level. You can visualize the specification in the Cashflow Chart area (see Figure 8.19).
6. Click **Add** to add the specified cashflow to the list in the Generic Cashflow dialog box. Clicking **Add** also returns you to the Generic Cashflow dialog box.

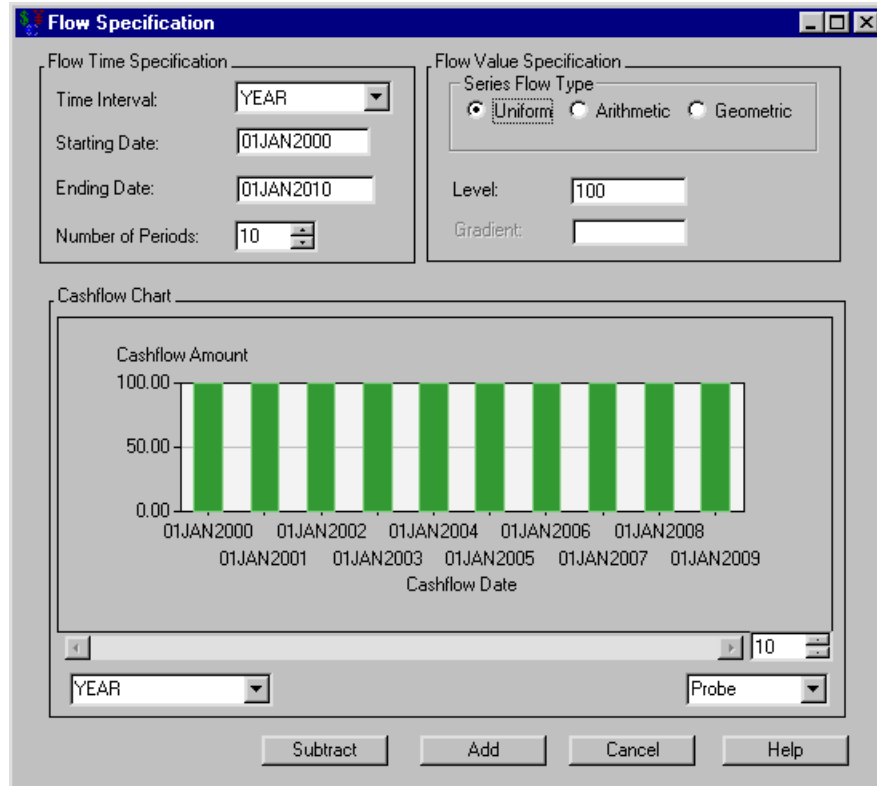


**Figure 8.19.** Uniform Cashflow Specification

Clicking **Subtract** will subtract the current cashflow from the Generic Cashflow dialog box as it returns you to the Generic Cashflow dialog box.

You can generate arithmetic and geometric specifications by clicking them within the **Series Flow Type** area. However, you must enter a value for the **Gradient**. In both cases the **Level** value is the value of the list at the **Starting Date**. With an arithmetic flow type, entries increment by the value **Gradient** each **Time Interval**. With a geometric flow type, entries increase by the factor **Gradient** each **Time Interval**. Figure 8.20 displays an arithmetic cashflow with a **Level** of 100 and a **Gradient** of 12.

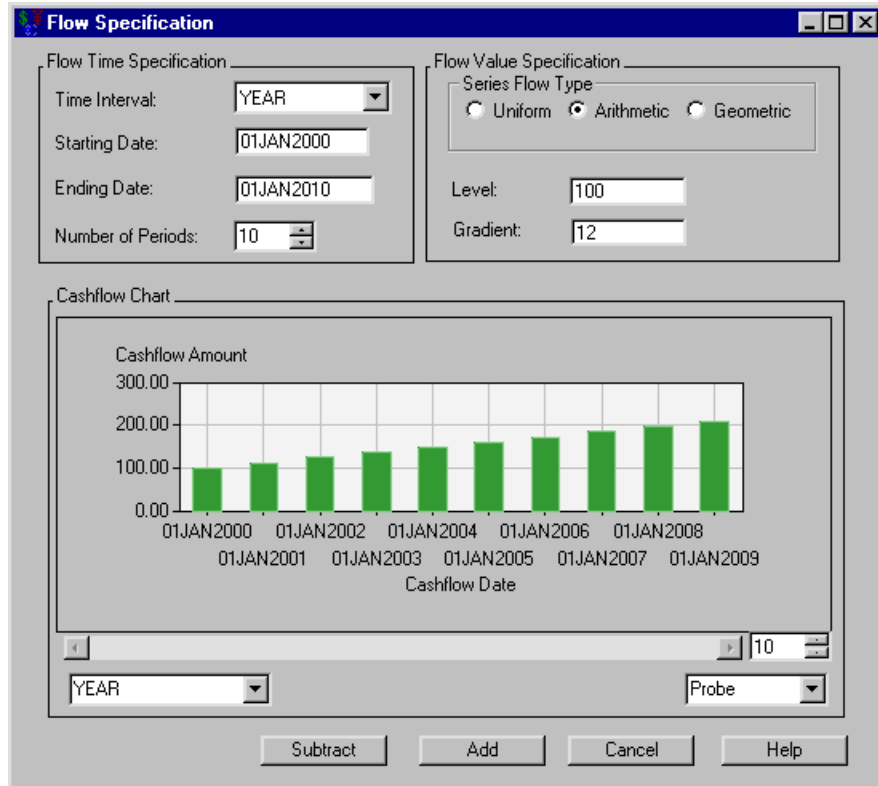


**Figure 8.20.** Arithmetic Cashflow Specification

### ***Including a Forecasted Cashflow***

To generate date-amount pairs for the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Forecast....** This opens the **Forecast Specification** dialog box.
2. Enter SASHELP.RETAIL as the **Data Set**.
3. Select SALES for the **Analysis Variable**.
4. Click **Compute Forecast** to generate the forecast. You can visualize the forecast in the Cashflow Chart area (see Figure 8.21).
5. Click **Add** to add the forecast to the list in the Generic Cashflow dialog box. Clicking **Add** also returns you to the Generic Cashflow dialog box.

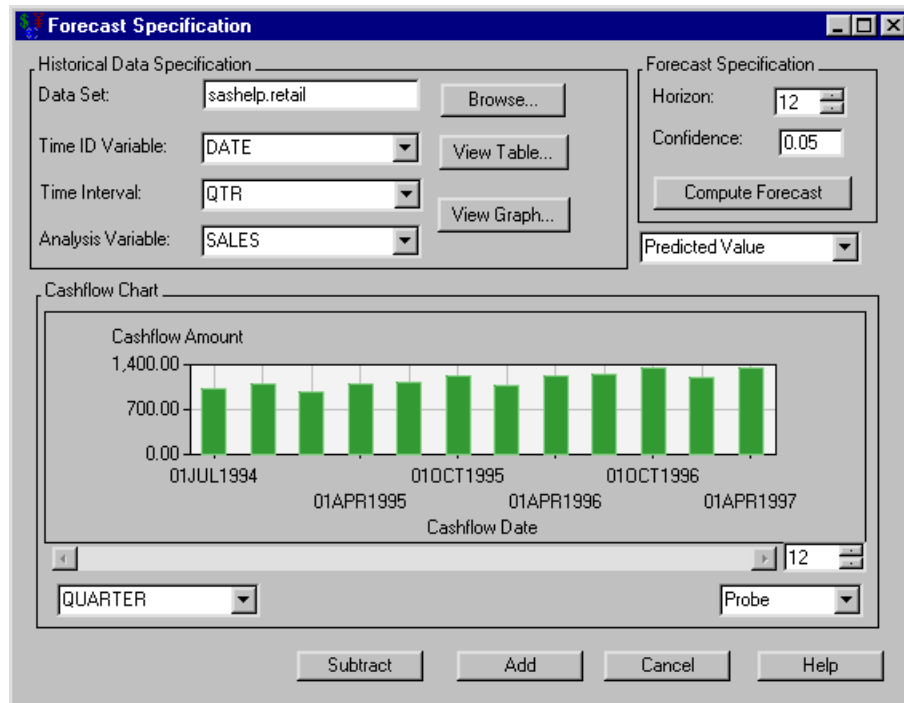


**Figure 8.21.** Cashflow Forecast

Clicking **Subtract** will subtract the current forecast from the Generic Cashflow dialog box as it returns you to the Generic Cashflow dialog box.

To review the values from the dataset you forecast, click **View Table...** or **View Graph...**

You can adjust the following values for the dataset you forecast: **Time ID Variable**, **Time Interval**, and **Analysis Variable**.

You can adjust the following values for the forecast: the **Horizon**, the **Confidence**, and choice of predicted value, lower confidence limit, and upper confidence limit.

### **Using the Cashflow Chart**

Three dialog boxes contain the Cashflow Chart to aide in your visualization of cashflows: Generic Cashflow, Flow Specification, and Forecast Specification. Within this chart, you possess the following tools:

You can click on a bar in the plot and view its **Cashflow Date** and **Cashflow Amount**.

You can change the aggregation period of the view with the box in the lower left corner of the Cashflow Chart. You can take the quarterly sales figures from the previous example, select YEAR as the value for this box, and view the annual sales figures. You can change the number in the box to the right of the horizontal scroll bar to alter the number of entries you wish to view. The number in that box must be no greater than the number of entries in the cashflow list. Lessening this number has the effect of zooming in upon a portion of the cashflow. When the number is less than the number of entries in the cashflow list, you can use the scroll bar at the bottom of the chart to scroll through the chart.

## Dialog Box Guide

### Loan

Selecting **Investment** → **New** → **Loan...** from the Investment Analysis dialog box's menu bar opens the Loan dialog box displayed in Figure 8.22.

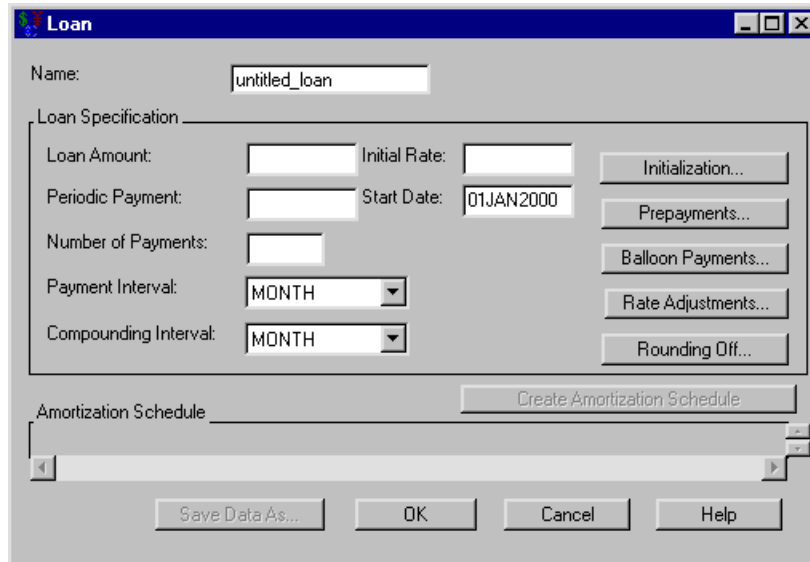


**Figure 8.22.** Loan Dialog Box

The following items are displayed:

**Name** holds the name you assign to the loan. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

The **Loan Specification** area gives access to the values that define the loan.

**Loan Amount** holds the borrowed amount.

**Periodic Payment** holds the value of the periodic payments.

**Number of Payments** holds the number of payments in loan terms.

**Payment Interval** holds the frequency of the **Periodic Payment**.

**Compounding Interval** holds the compounding frequency.

**Initial Rate** holds the interest rate (a nominal percentage between 0 and 120) you pay on the loan.

**Start Date** holds the SAS date when the loan is initialized. The first payment is due one **Payment Interval** after this time.

**Initialization...** opens the Loan Initialization Options dialog box where you can define initialization costs and down-payments relevant to the loan.

**Prepayments...** opens the Loan Prepayments dialog box where you can specify the SAS dates and amounts of any prepayments.



**Balloon Payments...** opens the Balloon Payments dialog box where you can specify the SAS dates and amounts of any balloon payments.

**Rate Adjustments...** opens the Rate Adjustment Terms dialog box where you can specify terms for a variable-rate loan.

**Rounding Off...** opens the Rounding Off dialog box where you can select the number of decimal places for calculations.

**Create Amortization Schedule** becomes available when you adequately define the loan within the **Loan Specification** area. Clicking it generates the amortization schedule.

**Amortization Schedule** fills when you click **Create Amortization Schedule**. The schedule contains a row for the loan's start-date and each payment-date with information about the following:

**Date** is a SAS date, either the loan's start-date or a payment-date.

**Beginning Principal Amount** is the balance at that date.

**Periodic Payment Amount** is the expected payment at that date.

**Interest Payment** is zero for the loan's start-date; otherwise it holds the interest since the previous date.

**Principal Repayment** is the amount of the payment that went toward the principal.

**Ending Principal** is the balance at the end of the payment interval.

**Save Data As...** becomes available when you generate the amortization schedule. Clicking it opens the Save Output Dataset dialog box where you can save the amortization table (or portions thereof) as a SAS Dataset.

**OK** returns you to the Investment Analysis dialog box. If this is a new loan specification, clicking **OK** appends the current loan specification to the portfolio. If this is an existing loan specification, clicking **OK** returns the altered loan specification to the portfolio.

**Cancel** returns you to the Investment Analysis dialog box. If this is a new loan specification, clicking **Cancel** discards the current loan specification. If this is an existing loan specification, clicking **Cancel** discards the current editions.

## Loan Initialization Options

Clicking **Initialization...** in the Loan dialog box opens the Loan Initialization Options dialog box displayed in Figure 8.23.

**Figure 8.23.** Loan Initialization Options Dialog Box

The following items are displayed:

The **Price, Loan Amount and Downpayment** area

**Purchase Price** holds the actual price of the asset. This value equals the loan amount plus the downpayment.

**Loan Amount** holds the loan amount.

**% of Price** (to the right of **Loan Amount**) updates when you enter the **Purchase Price** and either the **Loan Amount** or **Downpayment**. This holds the percentage of the **Purchase Price** that comprises the **Loan Amount**. Setting the percentage manually causes the **Loan Amount** and **Downpayment** to update.

**Downpayment** holds any downpayment paid for the asset.

**% of Price** (to the right of **Downpayment**) updates when you enter the **Purchase Price** and either the **Loan Amount** or **Downpayment**. This holds the percentage of the **Purchase Price** that comprises the **Downpayment**. Setting the percentage manually causes the **Loan Amount** and **Downpayment** to update.

**Initialization Costs and Discount Points** area

**Loan Amount** holds a copy of the **Loan Amount** above.

**Initialization Costs** holds the value of any initialization costs.

**% of Amount** (to the right of **Initialization Costs**) updates when you enter the **Purchase Price** and either the **Initialization Costs** or **Discount Points**. This holds the percentage of the **Loan Amount** that comprises the **Initialization Costs**. Setting the percentage manually causes the **Initialization Costs** to update.

**Discount Points** holds the value of any discount points.

**% of Amount** (to the right of **Discount Points**) updates when you enter the **Purchase Price** and either the **Initialization Costs** or **Discount Points**. This holds the percentage of the **Loan Amount** that comprises the **Discount Points**. Setting the percentage manually causes the **Discount Points** to update.

**OK** returns you to the Loan dialog box, saving the information that is entered.

**Cancel** returns you to the Loan dialog box, discarding any editions since you opened the dialog box.

---

## Loan Prepayments

Clicking **Prepayments...** in the Loan dialog box opens the Loan Prepayments dialog box displayed in Figure 8.24.

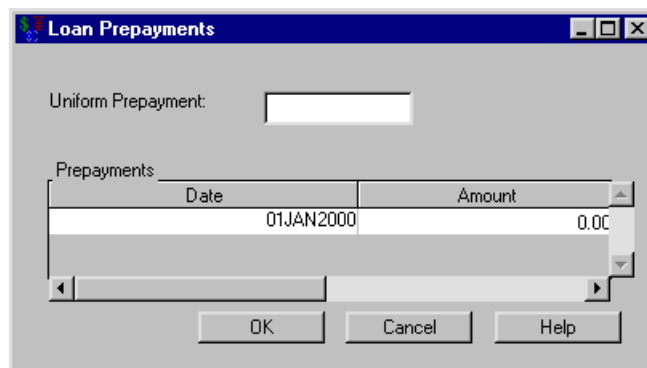


**Figure 8.24.** Loan Prepayments Dialog Box

The following items are displayed:

**Uniform Prepayment** holds the value of a regular prepayment concurrent to the usual periodic payment.

**Prepayments** holds a list of date-amount pairs to accommodate any prepayments. Right-clicking within the **Prepayments** area reveals many helpful tools for managing date-amount pairs.

**OK** returns you to the Loan dialog box, storing the information entered on the prepayments.

**Cancel** returns you to the Loan dialog box, discarding any prepayments entered since you opened the dialog box.

---

## Balloon Payments

Clicking **Balloon Payments...** in the Loan dialog box opens the Balloon Payments dialog box displayed in Figure 8.25.

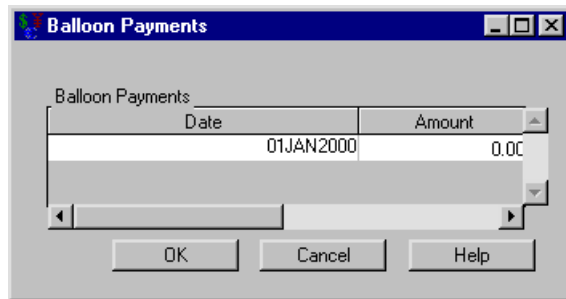


**Figure 8.25.** Balloon Payments Dialog Box

The following items are displayed:

**Balloon Payments** holds a list of date-amount pairs to accommodate any balloon payments. Right-clicking within the **Balloon Payments** area reveals many helpful tools for managing date-amount pairs.

**OK** returns you to the Loan dialog box, storing the information entered on the balloon payments.

**Cancel** returns you to the Loan dialog box, discarding any balloon payments entered since you opened the dialog box.

---

## Rate Adjustment Terms

Clicking **Rate Adjustments...** in the Loan dialog box opens the Rate Adjustment Terms dialog box displayed in Figure 8.26.

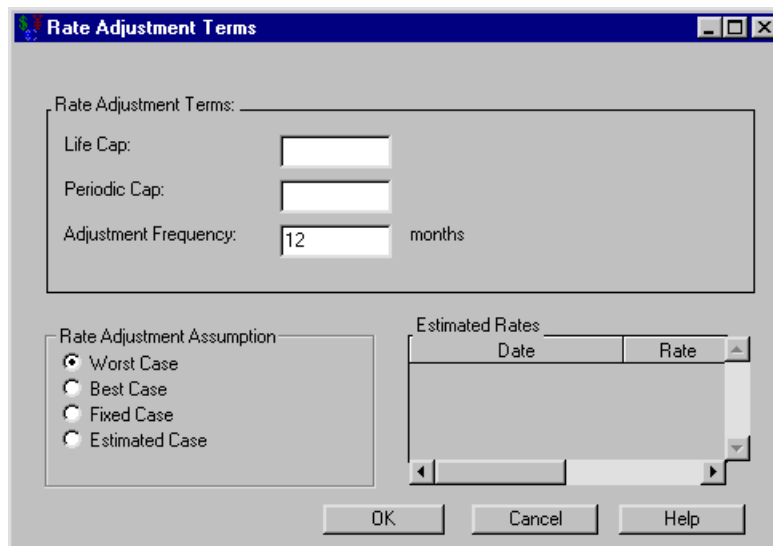


**Figure 8.26.** Rate Adjustment Terms Dialog Box

The following items are displayed:

The **Rate Adjustment Terms** area

**Life Cap** holds the maximum deviation from the **Initial Rate** allowed over the life of the loan.

**Periodic Cap** holds the maximum adjustment allowed per adjustment.

**Adjustment Frequency** holds how often (in months) the lender can adjust the interest rate.

The **Rate Adjustment Assumption** determines the scenario the adjustments will take.

**Worst Case** uses information from the **Rate Adjustment Terms** area to forecast a worst-case scenario.

**Best Case** uses information from the **Rate Adjustment Terms** area to forecast a best-case scenario.

**Fixed Case** specifies a fixed-rate loan.

**Estimated Case** uses information from the **Rate Adjustment Terms** and **Estimated Rate** area to forecast a best-case scenario.

**Estimated Rates** holds a list of date-rate pairs, where each date is a SAS date and the rate is a nominal percentage between 0 and 120. The **Estimated Case** assumption uses these rates for its calculations. Right-clicking within the **Estimated Rates** area reveals many helpful tools for managing date-rate pairs.

**OK** returns you to the Loan dialog box, taking rate adjustment information into account.

**Cancel** returns you to the Loan dialog box, discarding any rate adjustment information provided since opening the dialog box.

---

## Rounding Off

Clicking **Rounding Off...** in the Loan dialog box opens the Rounding Off dialog box displayed in Figure 8.27.

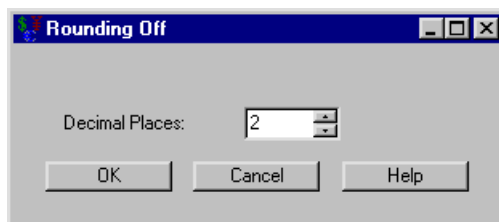


**Figure 8.27.** Rounding Off Dialog Box

The following items are displayed:

**Decimal Places** fixes the number of decimal places your results will display.

**OK** returns you to the Loan dialog box. Numeric values will then be represented with the number of decimals specified in **Decimal Places**.

**Cancel** returns you to the Loan dialog box. Numeric values will be represented with the number of decimals specified prior to opening this dialog box.

## Savings

Selecting **Investment** → **New** → **Savings...** from the Investment Analysis dialog box's menu bar opens the Savings dialog box displayed in Figure 8.28.

**Figure 8.28.** Savings Dialog Box

The following items are displayed:

**Name** holds the name you assign to the savings. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

The **Savings Specification** area

**Periodic Deposit** holds the value of your regular deposits.

**Number of Deposits** holds the number of deposits into the account.

**Initial Rate** holds the interest rate (a nominal percentage between 0 and 120) the savings account earns.

**Start Date** holds the SAS date when deposits begin.

**Deposit Interval** holds the frequency of your **Periodic Deposit**.

**Compounding Interval** holds how often the interest compounds.

**Create Account Summary** becomes available when you adequately define the savings within the **Savings Specification** area. Clicking it generates the account summary.

**Account Summary** fills when you click **Create Account Summary**. The schedule contains a row for each deposit-date with information about the following:

**Date** is the SAS date of a deposit.

**Starting Balance** is the balance at that date.

**Deposits** is the deposit at that date.

**Interest Earned** is the interest earned since the previous date.

**Ending Balance** is the balance after the payment.

**Save Data As...** becomes available when you generate an account summary. Clicking it opens the Save Output Dataset dialog box where you can save the account summary (or portions thereof) as a SAS Dataset.

**OK** returns you to the Investment Analysis dialog box. If this is a new savings, clicking **OK** appends the current savings specification to the portfolio. If this is an existing savings specification, clicking **OK** returns the altered savings to the portfolio.

**Cancel** returns you to the Investment Analysis dialog box. If this is a new savings, clicking **Cancel** discards the current savings specification. If this is an existing savings, clicking **Cancel** discards the current editions.

---

## Depreciation

Selecting **Investment** → **New** → **Depreciation...** from the Investment Analysis dialog box's menu bar opens the Depreciation dialog box displayed in Figure 8.29.

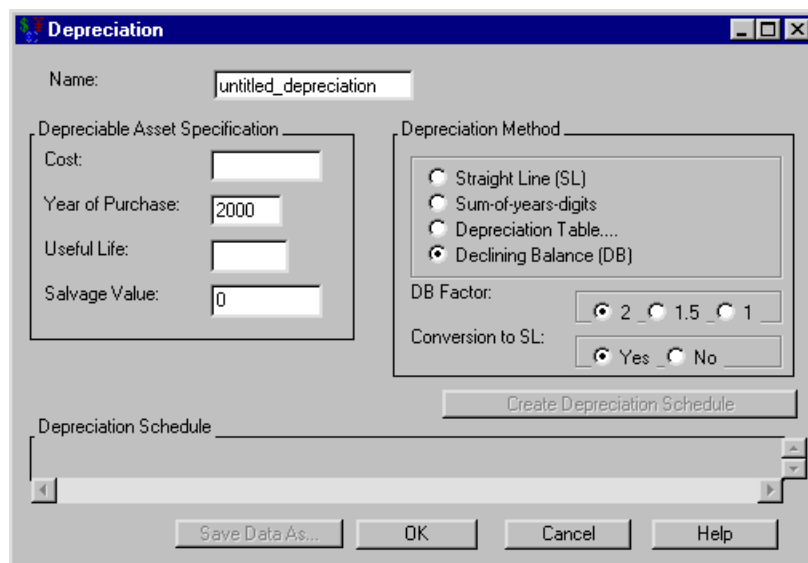


**Figure 8.29.** Depreciation Dialog Box

The following items are displayed:

**Name** holds the name you assign to the depreciation. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

**Depreciable Asset Specification**

**Cost** holds the asset's original cost.

**Year of Purchase** holds the asset's year of purchase.

**Useful Life** holds the asset's useful life (in years).

**Salvage Value** holds the asset's value at the end of its **Useful Life**.

The **Depreciation Method** area holds the depreciation methods available:

- Straight Line
- Sum-of-years Digits
- Depreciation Table...
- Declining Balance
  - DB Factor: choice of 2, 1.5, or 1
  - Conversion to SL: choice of Yes or No

**Create Depreciation Schedule** becomes available when you adequately define the depreciation within the **Depreciation Asset Specification** area. Clicking the **Create Depreciation Schedule** button then fills the **Depreciation Schedule** area.

**Depreciation Schedule** fills when you click **Create Depreciation Schedule**. The schedule contains a row for each year. Each row holds:

**Year** is a year

**Start Book Value** is the starting book value for that year

**Depreciation** is the depreciation value for that year

**End Book Value** is the ending book value for that year

**Save Data As...** becomes available when you generate the depreciation schedule. Clicking it opens the Save Output Dataset dialog box where you can save the depreciation table (or portions thereof) as a SAS Dataset.

**OK** returns you to the Investment Analysis dialog box. If this is a new depreciation specification, clicking **OK** appends the current depreciation specification to the portfolio. If this is an existing depreciation specification, clicking **OK** returns the altered depreciation specification to the portfolio.

**Cancel** returns you to the Investment Analysis dialog box. If this is a new depreciation specification, clicking **Cancel** discards the current depreciation specification. If this is an existing depreciation specification, clicking **Cancel** discards the current editions.



---

## Depreciation Table

Clicking **Depreciation Table...** from **Depreciation Method** area of the Depreciation dialog box opens the Depreciation Table dialog box displayed in Figure 8.30.

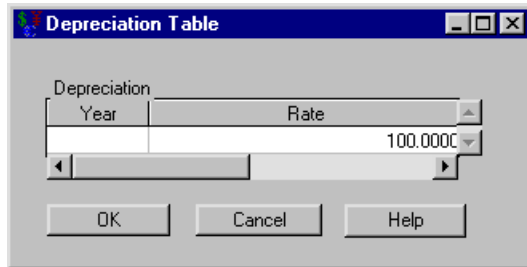


**Figure 8.30.** Depreciation Table Dialog Box

The following items are displayed:

The **Depreciation** area holds a list of year-rate pairs where the rate is an annual depreciation rate (a percentage between 0% and 100%). Right-clicking within the **Depreciation** area reveals many helpful tools for managing year-rate pairs.

**OK** returns you to the Depreciation dialog box with the current list of depreciation rates from the **Depreciation** area.

**Cancel** returns you to the Depreciation dialog box, discarding any editions to the **Depreciation** area since you opened the dialog box.

---

## Bond

Selecting **Investment** → **New** → **Bond...** from the Investment Analysis dialog box's menu bar opens the Bond dialog box displayed in Figure 8.31.

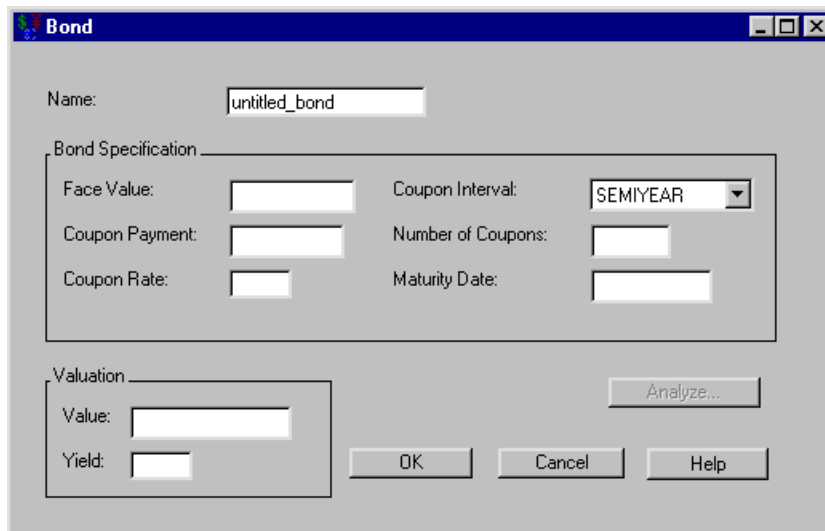


**Figure 8.31.** Bond

The following items are displayed:

**Name** holds the name you assign to the bond. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

### **Bond Specification**

**Face Value** holds the bond's value at maturity.

**Coupon Payment** holds the amount of money you receive periodically as the bond matures.

**Coupon Rate** holds the rate (a nominal percentage between 0% and 120%) of the **Face Value** that defines the **Coupon Payment**.

**Coupon Interval** holds how often the bond pays its coupons.

**Number of Coupons** holds the number of coupons before maturity.

**Maturity Date** holds the SAS date when you can redeem the bond for its **Face Value**.

The **Valuation** area becomes available when you adequately define the bond within the **Bond Specification** area. Entering either the **Value** or the **Yield** causes the calculation of the other. If you respecify the bond after performing a calculation here, you must reenter the **Value** or **Yield** value to update the calculation.

**Value** holds the bond's value if expecting the specified **Yield**.

**Yield** holds the bond's yield if the bond is valued at the amount of **Value**.

You must specify the bond before analyzing it. Once you have specified the bond, clicking **Analyze...** opens the Bond Analysis dialog box where you can compare various values and yields.

**OK** returns you to the Investment Analysis dialog box. If this is a new bond specification, clicking **OK** appends the current bond specification to the portfolio. If this is an existing bond specification, clicking **OK** returns the altered bond specification to the portfolio.

**Cancel** returns you to the Investment Analysis dialog box. If this is a new bond specification, clicking **Cancel** discards the current bond specification. If this is an existing bond specification, clicking **Cancel** discards the current editions.

## Bond Analysis

Clicking **Analyze...** from the Bond dialog box opens the Bond Analysis dialog box displayed in Figure 8.32.

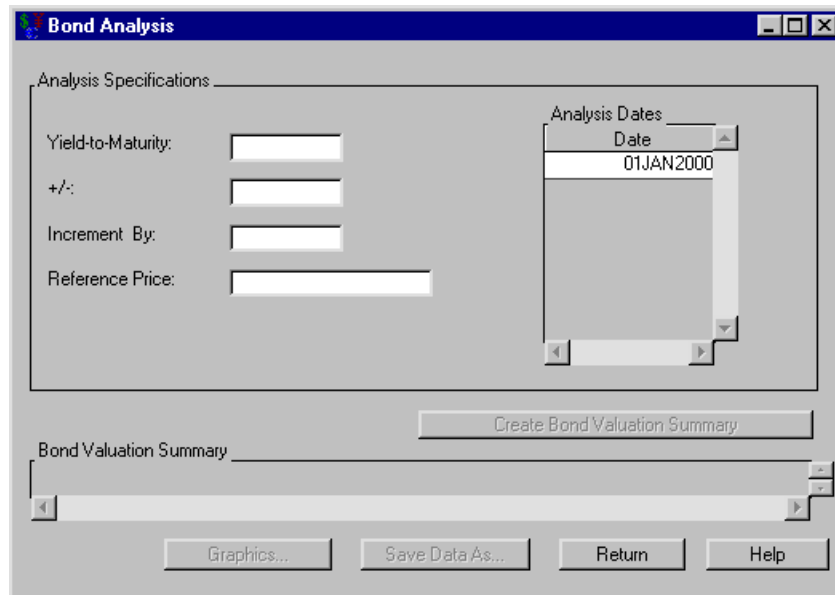


**Figure 8.32.** Bond Analysis

The following items are displayed:

### Analysis Specifications

**Yield-to-maturity** holds the percentage yield upon which to center the analysis.

**+/-** holds the maximum deviation percentage from the **Yield-to-maturity** to consider.

**Increment by** holds the percentage increment by which the analysis is calculated.

**Reference Price** holds the reference price.

**Analysis Dates** holds a list of SAS dates for which you perform the bond analysis.

You must specify the analysis before valuing the bond for the various yields. Once you adequately specify the analysis, click **Create Bond Valuation Summary** to generate the bond valuation summary.

**Bond Valuation Summary** fills when you click **Create Bond Valuation Summary**. The schedule contains a row for each rate with information concerning the following:

**Date** is the SAS date when the **Value** gives the particular **Yield**.

**Yield** is the percent yield that corresponds to the **Value** at the given **Date**.

**Value** is the value of the bond at **Date** for the given **Yield**.

**Percent Change** is the percent change if the **Reference Price** is specified.

**Duration** is the duration.

**Convexity** is the convexity.

**Graphics...** opens the Bond Price graph representing the price versus yield-to-maturity.

**Save Data As...** becomes available when you fill the **Bond Valuation Summary** area. Clicking it opens the Save Output Dataset dialog box where you can save the valuation summary (or portions thereof) as a SAS Dataset.

**Return** takes you back to the Bond dialog box.

---

## Bond Price

Clicking **Graphics...** from the Bond dialog box opens the Bond Price dialog box displayed in Figure 8.33.

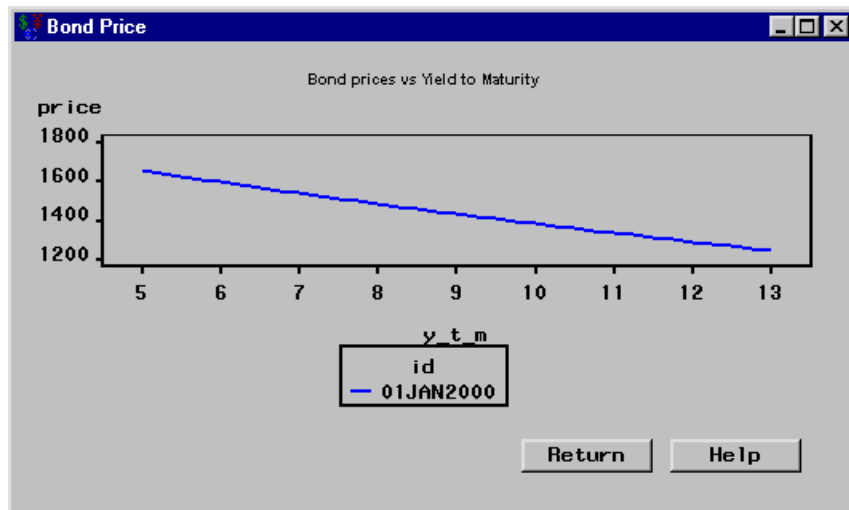


**Figure 8.33.** Bond Price Graph

It possesses the following item:

**Return** takes you back to the Bond Analysis dialog box.

## Generic Cashflow

Selecting **Investment** → **New** → **Generic Cashflow...** from the Investment Analysis dialog box's menu bar opens the Generic Cashflow dialog box displayed in Figure 8.34.

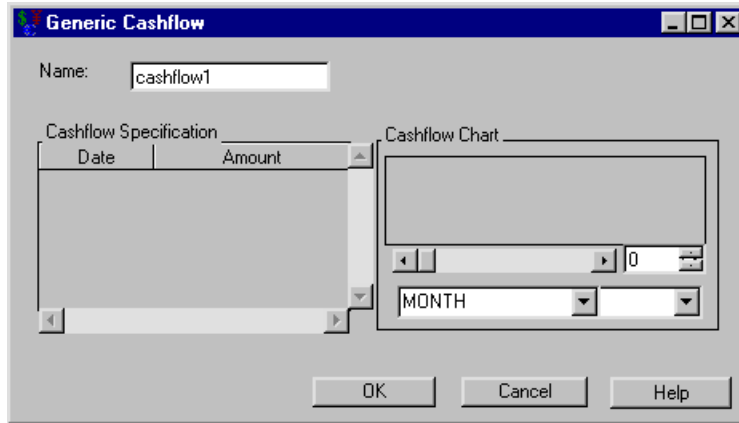


**Figure 8.34.** Generic Cashflow

The following items are displayed:

**Name** holds the name you assign to the generic cashflow. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

**Cashflow Specification** holds date-amount pairs corresponding to deposits and withdrawals (or benefits and costs) for the cashflow. Each date is a SAS date. Right-clicking within the **Cashflow Specification** area reveals many helpful tools for managing date-amount pairs.

The **Cashflow Chart** fills with a graph representing the cashflow when the **Cashflow Specification** area is nonempty. The box to the right of the scroll bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency for drilling purposes.

**OK** returns you to the Investment Analysis dialog box. If this is a new generic cashflow specification, clicking **OK** appends the current cashflow specification to the portfolio. If this is an existing cashflow specification, clicking **OK** returns the altered cashflow specification to the portfolio.

**Cancel** returns you to the Investment Analysis dialog box. If this is a new cashflow specification, clicking **Cancel** discards the current cashflow specification. If this is an existing cashflow specification, clicking **Cancel** discards the current editions.

---

## Right-Clicking within Generic Cashflow's Cashflow Specification Area

Right-click within the **Cashflow Specification** area of the Generic Cashflow dialog box pops up the menu displayed in Figure 8.35.

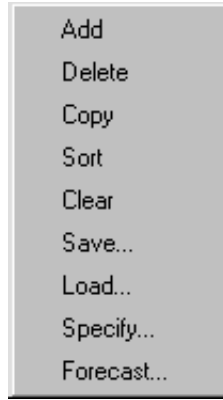


**Figure 8.35.** Right-Clicking

**Add** creates a blank pair.

**Delete** removes the currently highlighted pair.

**Copy** duplicates the currently selected pair.

**Sort** arranges the entered pairs in chronological order.

**Clear** empties the area of all pairs.

**Save...** opens the Save Dataset dialog box where you can save the entered pairs as a SAS Dataset for later use.

**Load...** opens the Load Dataset dialog box where you select a SAS Dataset to populate the area.

**Specify...** opens the **Flow Specification** dialog box where you can generate date-rate pairs to include in your cashflow.

**Forecast...** opens the **Forecast Specification** dialog box where you can generate the forecast of a dataset to include in your cashflow.

If you wish to perform one of these actions on a collection of pairs, you must select a collection of pairs before right-clicking. To select an adjacent list of pairs, do the following: click the first pair, hold down SHIFT, and click the final pair. Once the list of pairs is selected, you may release the SHIFT key.

## Flow Specification

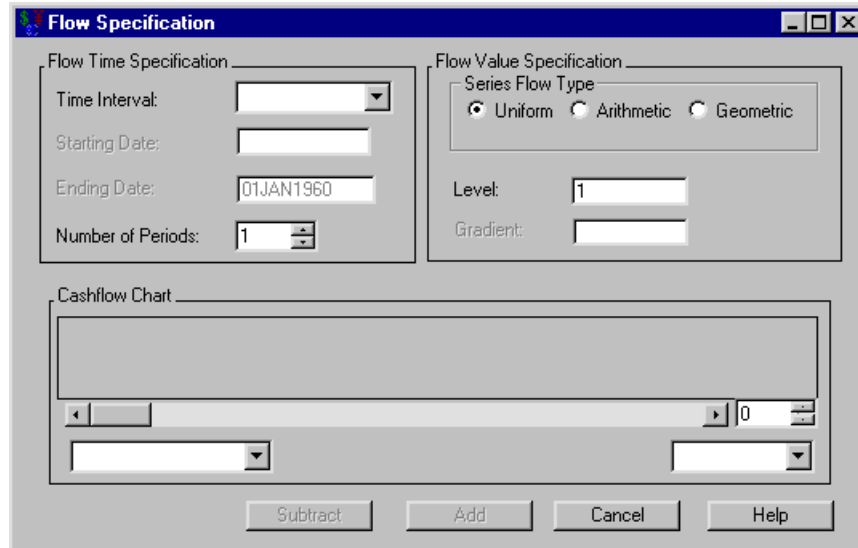
The image shows a 'Flow Specification' dialog box with a blue title bar. It is divided into three main sections. The top-left section, 'Flow Time Specification', contains four fields: 'Time Interval' (a dropdown menu), 'Starting Date' (a text box), 'Ending Date' (a text box containing '01JAN1960'), and 'Number of Periods' (a spinner box set to '1'). The top-right section, 'Flow Value Specification', contains a 'Series Flow Type' group box with three radio buttons: 'Uniform' (selected), 'Arithmetic', and 'Geometric'. Below this are 'Level' (a text box with '1') and 'Gradient' (a text box). The bottom section, 'Cashflow Chart', features a large empty rectangular area for a graph, a horizontal scrollbar on the right, and two small dropdown menus at the bottom. At the very bottom of the dialog are four buttons: 'Subtract', 'Add', 'Cancel', and 'Help'.

Figure 8.36. Flow Specification

The following items are displayed:

### Flow Time Specification

**Time Interval** holds the uniform frequency of the entries.

You can set the **Starting Date** when you set the **Time Interval**. It holds the SAS date the entries will start.

You can set the **Ending Date** when you set the **Time Interval**. It holds the SAS date the entries will end.

**Number of Periods** holds the number of entries.

### Flow Value Specification

**Series Flow Type** describes the movement the entries can assume:

- **Uniform** assumes all entries are equal.
- **Arithmetic** assumes the entries begin at **Level** and increase by the value of **Gradient** per entry.
- **Geometric** assumes the entries begin at **Level** and increase by a factor of **Gradient** per entry.

**Level** holds the starting amount for all **Flow Types**.

You can set the **Gradient** when you select either **Arithmetic** or **Geometric Gradient**. It holds the arithmetic and geometric gradients, respectively, for the **Arithmetic** and **Geometric Flow Types**.

The **Cashflow Chart** fills with a graph displaying the dates and values of the entries when the cashflow entries are adequately defined. The box to the right of the scroll

bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency for drilling purposes.

**Subtract** becomes available when the collection of entries is adequately specified. Clicking **Subtract** then returns you to the Generic Cashflow dialog box subtracting the entries from the current cashflow.

**Add** becomes available when the collection of entries is adequately specified. Clicking **Add** then returns you to the Generic Cashflow dialog box adding the entries to the current cashflow.

**Cancel** returns you to Generic Cashflow dialog box without editing the cashflow.

## Forecast Specification

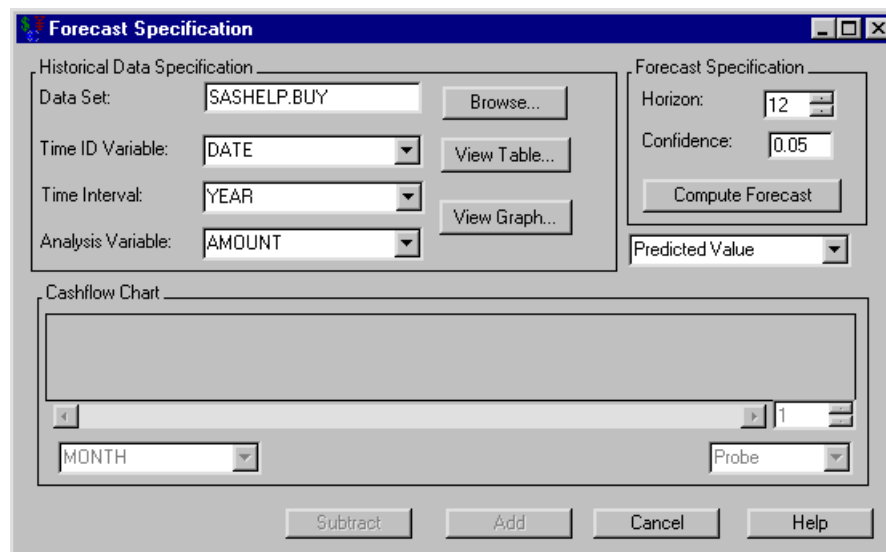


**Figure 8.37.** Forecast Specification

The following items are displayed:

### Historical Data Specification

**Data Set** holds the name of the dataset to forecast.

**Browse...** opens the standard SAS **Open** dialog box to help select a dataset to forecast.

**Time ID Variable** holds the time ID variable to forecast over.

**Time Interval** fixes the time interval for the **Time ID Variable**.

**Analysis Variable** holds the data variable upon which to forecast.

**View Table...** opens a table that displays in a list the contents of the specified dataset.

**View Graph...** opens the Time Series Viewer that graphically displays the contents of the specified dataset.



### Forecast Specification

**Horizon** holds the number of periods into the future you wish to forecast.

**Confidence** holds the confidence limit for applicable forecasts.

**Compute Forecast** fills the **Cashflow Chart** with the forecast.

The box below holds the type of forecast you wish to generate:

- Predicted Value
- Lower Confidence Limit
- Upper Confidence Limit

The **Cashflow Chart** fills when you click **Compute Forecast**. The box to the right of the scroll bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency for drilling purposes.

**Subtract** becomes available when the collection of entries is adequately specified. Clicking **Subtract** then returns you to the Generic Cashflow dialog box subtracting the forecast from the current cashflow.

**Add** becomes available when the collection of entries is adequately specified. Clicking **Add** then returns you to the Generic Cashflow adding the forecast to the current cashflow.

**Cancel** returns to Generic Cashflow dialog box without editing the cashflow.

# Chapter 9

## Computations

### Chapter Table of Contents

---

|                                |     |
|--------------------------------|-----|
| <b>THE COMPUTE MENU</b>        | 243 |
| <b>TASKS</b>                   | 243 |
| Taxing a Cashflow              | 244 |
| Converting Currency            | 245 |
| Inflating Cashflows            | 247 |
| <b>DIALOG BOX GUIDE</b>        | 249 |
| After Tax Cashflow Computation | 249 |
| Currency Conversion            | 250 |
| Constant Dollar Calculation    | 251 |



## Chapter 9

# Computations

---

### The Compute Menu

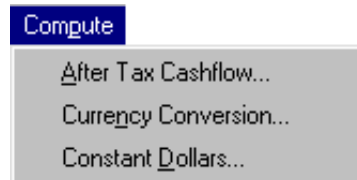


**Figure 9.1.** The Compute Menu

The **Compute** menu, shown in Figure 9.1, offers the following options to apply to generic cashflows:

Computing an **After Tax Cashflow** is useful when taxes affect investment alternatives differently. Comparing after tax cashflows provide a more accurate determination of the cashflows' profitabilities. You can set default values for income tax rates by selecting **Tools** → **Define Rate** → **Income Tax Rate...** from the Investment Analysis dialog box. This opens the Income Tax Specification dialog box where you can enter the tax rates.

**Currency Conversion** is necessary when investments are in different currencies. For data concerning currency conversion rates, consider <http://dsbb.imf.org/>, the International Monetary Fund's Dissemination Standards Bulletin Board.

A **Constant Dollar** (inflation adjusted monetary value) calculation takes a cashflow and inflation information and discounts the cashflow to a level where the buying power of the monetary unit is "constant" over time. Groups quantify inflation (in the form of price indices and inflation rates) for countries and industries by averaging the growth of prices for various products and sectors of the economy. For data concerning price indices, consider the United States Department of Labor at <http://www.dol.gov/> and the International Monetary Fund's Dissemination Standards Bulletin Board at <http://dsbb.imf.org/>. You can set default values for inflation rates by clicking **Tools** → **Define Rate** → **Inflation...** from the Investment Analysis dialog box. This opens the Inflation Specification dialog box where you can enter the inflation rates.

---

## Tasks

The next few sections show how to perform computations for the following situation. Suppose you buy a \$10,000 certificate of deposit that pays 12% interest a year for five years. Your earnings are taxed at a rate of 30% federally and 7% locally. Also, you want to transfer all the money to an account in England. British pounds converts to

American dollars at an exchange rate of \$1.00 to £0.60. The inflation rate in England is 3%. The instructions in this example assume familiarity with the following:

- The right-clicking options of the **Cashflow Specification** area in the Generic Cashflow dialog box (described in “Right-Clicking within Generic Cashflow’s Cashflow Specification Area” on page 237.)
- The **Save Data As...** button located in many dialog boxes (described in “Saving Output to SAS Datasets” on page 271.)

## Taxing a Cashflow

Consider the example described in “The Compute Menu” on page 243. To create the earnings, follow these steps:

1. Select **Investment** → **New** → **Generic Cashflow** to create a generic cashflow.
2. Enter CD\_INTEREST for the **Name**.
3. Enter 1200 for each of the five years starting one year from today as displayed in Figure 9.2.
4. Click **OK** to return to the Investment Analysis dialog box.

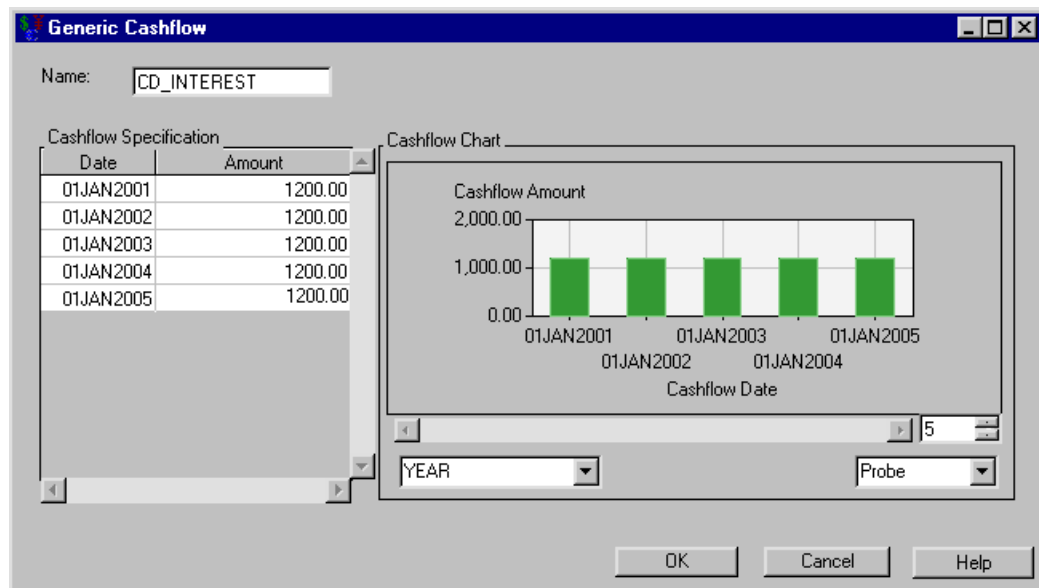


**Figure 9.2.** Computing the Interest on the CD

To compute the tax on the earnings, follow these steps:

1. Select CD\_INTEREST from the **Portfolio** area.
2. Select **Compute** → **After Tax Cashflow** from the pull-down menu.
3. Enter 30 for **Federal Tax**.
4. Enter 7 for **Local Tax**. Note that **Combined Tax** updates.

- Click **Create After Tax Cashflow** and the **After Tax Cashflow** area fills, as displayed in Figure 9.3.

**After Tax Cashflow Calculation**

Name:

Federal Tax:

Local Tax:

Combined Tax:

After Tax Cashflow

| Date      | Amount |
|-----------|--------|
| 01JAN2001 | 781.20 |
| 01JAN2002 | 781.20 |
| 01JAN2003 | 781.20 |
| 01JAN2004 | 781.20 |
| 01JAN2005 | 781.20 |

**Figure 9.3.** Computing the Interest After Taxes

Save the taxed earnings to a dataset named WORK.CD\_AFTERTAX. Click **Return** to return to the Investment Analysis dialog box.

## Converting Currency

Consider the example described in “The Compute Menu” on page 243. To create the cashflow to convert, follow these steps:

- Select **Investment** → **New** → **Generic Cashflow...** to open a new generic cashflow.
- Enter CD\_DOLLARS for the **Name**.
- Load WORK.CD\_AFTERTAX into its **Cashflow Specification**.
- Add -10,000 for today and +10,000 for five years from today to the cashflow as displayed in Figure 9.4.
- Sort the transactions by date to aid your reading.
- Click **OK** to return to the Investment Analysis dialog box.

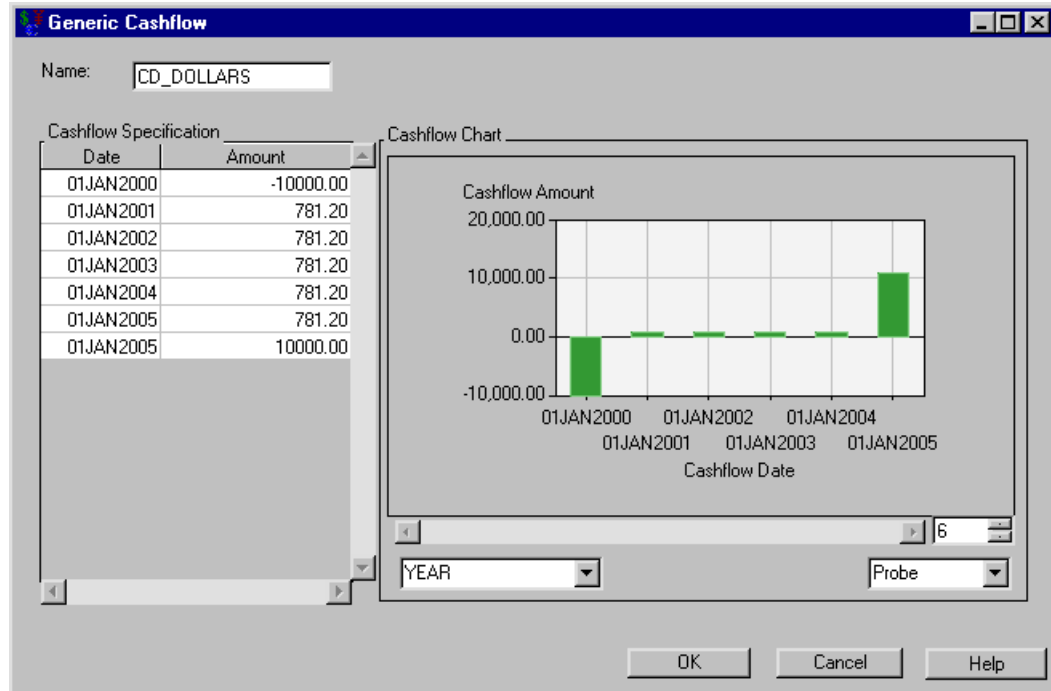


**Figure 9.4.** The CD in Dollars

To convert from British pounds to American dollars, follow these steps:

1. Select CD\_DOLLARS from the portfolio.
2. Select **Compute** → **Currency Conversion** from the pull-down menu. This opens the Currency Conversion dialog box.
3. Select USD for the **From Currency**.
4. Select GBP for the **To Currency**.
5. Enter 0.60 for the **Exchange Rate**.
6. Click **Apply Currency Conversion** to fill the **Currency Conversion** area as displayed in Figure 9.5.

**Currency Conversion**

Name:

From Currency:

To Currency:

Exchange Rate:

| Currency Conversion |            |          |
|---------------------|------------|----------|
| Date                | CD_DOLLARS | GBP      |
| 01JAN2000           | -10000.00  | -6000.00 |
| 01JAN2001           | 781.20     | 468.72   |
| 01JAN2002           | 781.20     | 468.72   |
| 01JAN2003           | 781.20     | 468.72   |
| 01JAN2004           | 781.20     | 468.72   |
| 01JAN2005           | 781.20     | 468.72   |
| 01JAN2005           | 10000.00   | 6000.00  |

**Figure 9.5.** Converting the CD to Pounds

Save the converted values to a dataset named WORK.CD\_POUNDS. Click **Return** to return to the Investment Analysis dialog box.

## Inflating Cashflows

Consider the example described in “The Compute Menu” on page 243. To create the cashflow to deflate, follow these steps:

1. Select **Investment** → **New** → **Generic Cashflow...** to open a new generic cashflow.
2. Enter CD\_DEFLATED for **Name**.
3. Load WORK.CD\_POUNDS into its **Cashflow Specification** (see Figure 9.6).
4. Click **OK** to return to the Investment Analysis dialog box.



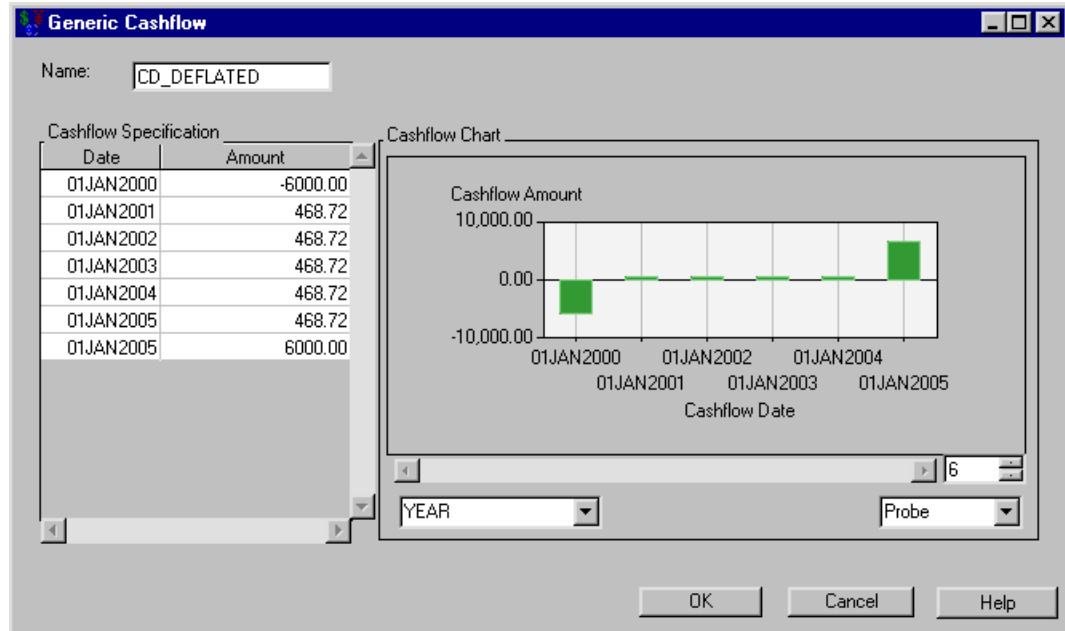


**Figure 9.6.** The CD before Deflation

To deflate the values, follow these steps:

1. Select CD\_DEFLATED from the portfolio.
2. Select **Compute** → **Constant Dollars** from the pull-down menu. This opens the Constant Dollar Calculation dialog box.
3. Clear the **Variable Inflation List** area.
4. Enter 3 for the **Constant Inflation Rate**.
5. Click **Create Constant Dollar Equivalent** to generate a constant dollar equivalent summary (see Figure 9.7).

The dialog box titled "Constant Dollar Calculation" contains the following fields and controls:

- Name:** CD\_DEFLATED
- Constant Inflation Rate:** 3.00
- Variable Inflation List:** A table with columns "Date" and "Rate".
- Dates:** A list box containing "01JAN2000".
- Create Constant Dollar Equivalent:** A button.
- Constant Dollar Equivalent Summary:** A table with columns "Date" and "JAN2000".
- Buttons:** "Save Data As...", "Return", and "Help".

| Date | Rate |
|------|------|
|      |      |
|      |      |
|      |      |
|      |      |
|      |      |
|      |      |

| Date      | JAN2000  |
|-----------|----------|
| 01JAN2000 | -6000.00 |
| 01JAN2001 | 455.07   |
| 01JAN2002 | 441.81   |
| 01JAN2003 | 428.95   |
| 01JAN2004 | 416.45   |
| 01JAN2005 | 404.32   |
| 01JAN2005 | 5175.65  |

**Figure 9.7.** CD Values after Deflation

You can save the deflated cashflow to a dataset for use in an internal rate of return analysis or breakeven analysis.

Click **Return** to return to the Investment Analysis dialog box.

---

## Dialog Box Guide

---

### After Tax Cashflow Computation

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an after tax calculation, select **Compute** → **After Tax...** from the Investment Analysis dialog box's menu bar. This opens the After Tax Cashflow Calculation dialog box displayed in Figure 9.8.

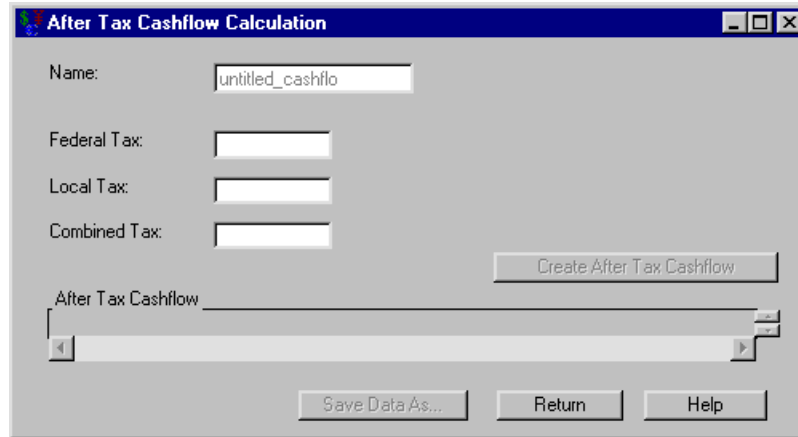


**Figure 9.8.** After Tax Cashflow Computation Dialog Box

The following items are displayed:

**Name** holds the name of the investment for which you are computing the after tax cashflow.

**Federal Tax** holds the federal tax rate (a percentage between 0% and 100%).

**Local Tax** holds the local tax rate (a percentage between 0% and 100%).

**Combined Tax** holds the effective tax rate from federal and local income taxes.

**Create After Tax Cashflow** becomes available when **Combined Tax** is non-empty. Clicking **Create After Tax Cashflow** then fills the **After Tax Cashflow** area.

**After Tax Cashflow** fills when you click **Create After Tax Cashflow**. It holds a list of date-amount pairs where the amount is the amount retained after taxes for that date.

**Save Data As...** becomes available when you fill the after tax cashflow. Clicking it opens the Save Output Dataset dialog box where you can save the resulting cashflow (or portions thereof) as a SAS Dataset.

**Return** returns you to the Investment Analysis dialog box.

---

## Currency Conversion

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a currency conversion, select **Compute** → **Currency Conversion...** from the Investment Analysis dialog box's menu bar. This opens the Currency Conversion dialog box displayed in Figure 9.9.

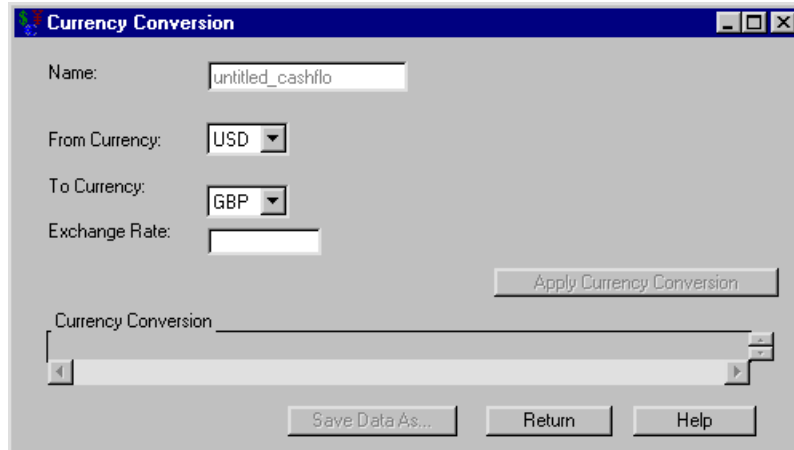


**Figure 9.9.** Currency Conversion Dialog Box

The following items are displayed:

**Name** holds the name of the investment to which you are applying the currency conversion.

**From Currency** holds the name of the currency the cashflow currently represents.

**To Currency** holds the name of the currency to which you wish to convert.

**Exchange Rate** holds the rate of exchange between the **From Currency** and the **To Currency**.

**Apply Currency Conversion** becomes available when you fill **Exchange Rate**. Clicking **Apply Currency Conversion** fills the **Currency Conversion** area.

**Currency Conversion** fills when you click **Apply Currency Conversion**. The schedule contains a row for each cashflow item with the following information:

- **Date** is a SAS date within the cashflow.
- The **From Currency** value is the amount in the original currency at that date.
- The **To Currency** value is the amount in the new currency at that date.

**Save Data As...** becomes available when you fill the **Currency Conversion** area. Clicking it opens the Save Output Dataset dialog box where you can save the conversion table (or portions thereof) as a SAS Dataset.

**Return** returns you to the Investment Analysis dialog box.

---

## Constant Dollar Calculation

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a constant dollar calculation, select **Compute** → **Constant Dollars...** from the Investment Analysis dialog box's menu bar. This opens the Constant Dollar Calculation dialog box displayed in Figure 9.10.

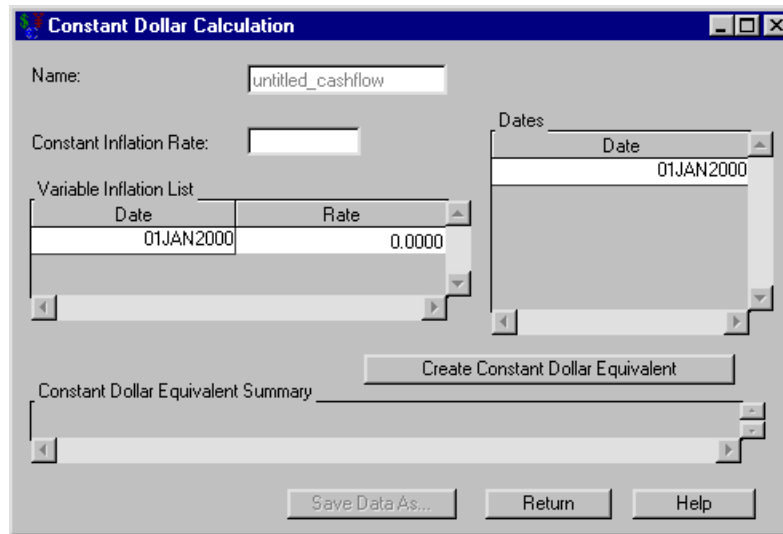


**Figure 9.10.** Constant Dollar Calculation Dialog Box

The following items are displayed:

**Name** holds the name of the investment for which you are computing the constant dollars value.

**Constant Inflation Rate** holds the constant inflation rate (a percentage between 0% and 120%). This value is used if the **Variable Inflation List** area is empty.

**Variable Inflation List** holds date-rate pairs that describe how inflation varies over time. Each date is a SAS date, and the rate is a percentage between 0% and 120%. Each date refers to when that inflation rate begins. Right-clicking within the **Variable Inflation** area reveals many helpful tools for managing date-rate pairs. If you assume a fixed inflation rate, just insert that rate in **Constant Rate**.

**Dates** holds the SAS date(s) at which you wish to compute the constant dollar equivalent. Right-clicking within the **Dates** area reveals many helpful tools for managing date lists.

**Create Constant Dollar Equivalent** becomes available when you enter inflation rate information. Clicking it fills the constant dollar equivalent summary with the computed constant dollar values.

**Constant Dollar Equivalent Summary** fills with a summary when you click **Create Constant Dollar Equivalent**. The first column lists the dates of the generic cashflow. The second column contains the constant dollar equivalent of the original generic cashflow item of that date.

**Save Data As...** becomes available when you fill the constant dollar equivalent summary. Clicking it opens the Save Output Dataset dialog box where you can save the Constant Dollar Equivalent Summary (or portions thereof) as a SAS Dataset.

**Return** returns you to the Investment Analysis dialog box.

# Chapter 10

## Analyses

### Chapter Table of Contents

---

|                                                    |     |
|----------------------------------------------------|-----|
| <b>THE ANALYZE MENU</b> . . . . .                  | 255 |
| <b>TASKS</b> . . . . .                             | 256 |
| Performing Time Value Analysis . . . . .           | 256 |
| Computing an Internal Rate of Return . . . . .     | 257 |
| Performing a Benefit-Cost Ratio Analysis . . . . . | 258 |
| Computing a Uniform Periodic Equivalent . . . . .  | 259 |
| Performing a Breakeven Analysis . . . . .          | 260 |
| <b>DIALOG BOX GUIDE</b> . . . . .                  | 262 |
| Time Value Analysis . . . . .                      | 262 |
| Uniform Periodic Equivalent . . . . .              | 263 |
| Internal Rate of Return . . . . .                  | 264 |
| Benefit-Cost Ratio Analysis . . . . .              | 264 |
| Breakeven Analysis . . . . .                       | 266 |
| Breakeven Graph . . . . .                          | 267 |



## Chapter 10

# Analyses

---

### The Analyze Menu



**Figure 10.1.** Analyze Menu

The **Analyze** menu, shown in Figure 10.1, offers the following options for use on applicable investments:

**Time Value** analysis involves moving money through time across a defined MARR so that you can compare value at a consistent date. The MARR can be constant or variable over time.

Uniform **Periodic Equivalent** analysis determines the payment needed to convert a cashflow to uniform amounts over time, given a periodicity, a number of periods, and a MARR. This option helps when making comparisons where one alternative is uniform (such as renting) and another is not (such as buying).

The **Internal Rate of Return** of a cashflow is the interest rate that makes the time value equal to 0. This calculation assumes uniform periodicity of the cashflow. It is particularly applicable where the choice of MARR would be difficult.

The **Benefit-Cost Ratio** divides the time value of the benefits by the time value of the costs. For example, governments often use this analysis when deciding whether to commit to a public works project.

**Breakeven Analysis** computes time values at various MARRs to compare. It is advantageous when it is difficult to determine a MARR. This helps you determine how the cashflow's profitability varies with your choice of MARR. A graph displaying the relationships between time value and MARR is also available.



## Tasks

### Performing Time Value Analysis

Suppose a rock quarry needs equipment to use the next five years. It has two alternatives:

- A box loader and conveyer system that has a one-time cost of \$264,000.
- A two-shovel loader which costs \$84,000 but has a yearly operating cost of \$36,000. This loader has a service life of three years, which necessitates the purchase of a new loader for the final two years of the rock quarry project. Assume the second loader also costs \$84,000 and its salvage value after its two-year service is \$10,000. A SAS dataset that describes this is available at SASHELP.ROCKPIT.

You expect a 13% MARR. Which is the better alternative?

To create the cashflows, follow these steps:

1. Create a cashflow with the single amount -264,000. Date the amount 01JAN1998 to be consistent with the dataset you load.
2. Load SASHELP.ROCKPIT into a second cashflow, as displayed in Figure 10.2.

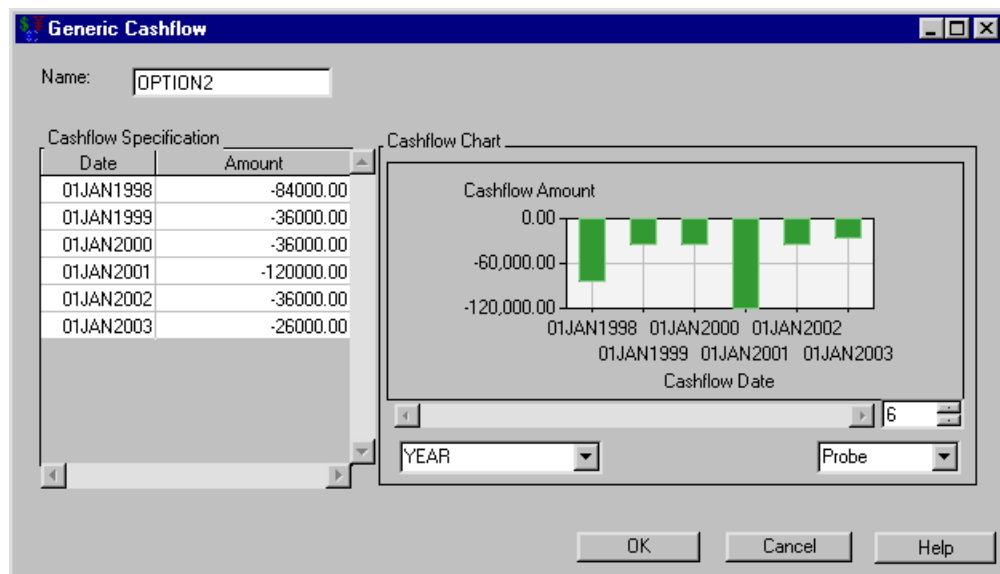


**Figure 10.2.** The contents of SASHELP.ROCKPIT

To compute the time values of these investments, follow these steps:

1. Select both cashflows.
2. Select **Analyze** → **Time Value....** This opens the Time Value Analysis dialog box.

3. Enter the date 01JAN1998 into the **Dates** area.
4. Enter 13 for the **Constant MARR**.
5. Click **Create Time Value Summary**.

**Time Value Analysis**

Analysis Specifications

Dates: 01JAN1998

Constant MARR: 13.00

MARR List

Time Value Summary

| Date      | OPTION1    | OPTION2    |
|-----------|------------|------------|
| 01JAN1998 | -264000.00 | -263408.94 |

Create Time Value Summary

Save Data As... Return Help

**Figure 10.3.** Performing the Time Value Analysis

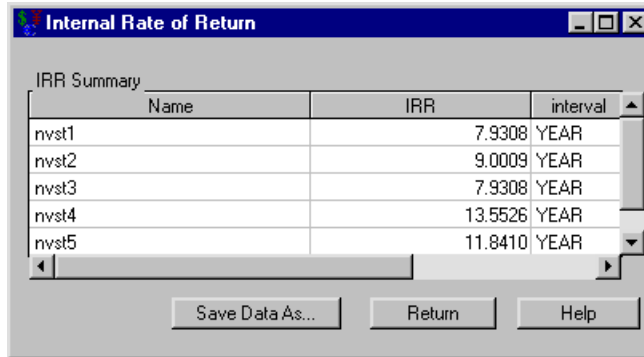
As shown in Figure 10.3, option 1 has a time value of -\$264,000.00 naturally on 01JAN1998. However, option 2 has a time value of -\$263,408.94, which is slightly less expensive.

## Computing an Internal Rate of Return

You are choosing between five investments. A portfolio containing these investments is available at SASHELP.INVSAMP.NVST. Which investments are acceptable if you expect a MARR of 9%?

Open the portfolio SASHELP.INVSAMP.NVST and compare the investments. Note that Internal Rate of Return computations assume regular periodicity of the cashflow. To compute the internal rates of return, follow these steps:

1. Select all five investments.
2. Select **Analyze** → **Internal Rate of Return....**



The screenshot shows a window titled "Internal Rate of Return" with a sub-header "IRR Summary". Below this is a table with three columns: "Name", "IRR", and "interval". The table lists five investments: nvst1, nvst2, nvst3, nvst4, and nvst5. The IRR values are 7.9308, 9.0009, 7.9308, 13.5526, and 11.8410 respectively, all with an interval of "YEAR". At the bottom of the window are three buttons: "Save Data As...", "Return", and "Help".

| Name  | IRR     | interval |
|-------|---------|----------|
| nvst1 | 7.9308  | YEAR     |
| nvst2 | 9.0009  | YEAR     |
| nvst3 | 7.9308  | YEAR     |
| nvst4 | 13.5526 | YEAR     |
| nvst5 | 11.8410 | YEAR     |

**Figure 10.4.** Computing an Internal Rate of Return

The results displayed in Figure 10.4 indicate that the internal rates of return for investments 2, 4, and 5 are greater than 9%. Hence, each of these is acceptable.

---

## Performing a Benefit-Cost Ratio Analysis

Suppose a municipality has excess funds to invest. It is choosing between the same investments described in the previous example. Government agencies often compute benefit-cost ratios to decide which investment to pursue. Which is best in this case?

Open the portfolio SASHELP.INVSAMP.NVST and compare the investments.

To compute the benefit-cost ratios, follow these steps:

1. Select all five investments.
2. Select **Analyze** → **Benefit-Cost Ratio...**
3. Enter 01JAN1996 for the **Date**.
4. Enter 9 for **Constant MARR**.
5. Click **Create Benefit-Cost Ratio Summary** to fill the **Benefit-Cost Ratio Summary** area.

**Benefit-Cost Ratio Analysis**

Analysis Specifications

Dates

| Date      |
|-----------|
| 01JAN2000 |

Constant MARR: 9.00

MARR List

| Date | MARR |
|------|------|
|------|------|

Create Benefit-Cost Ratio Summary

Benefit-Cost Ratio Summary

| Date      | nvst1  | nvst2  | nvst3  | nvst4  | nvst5  |
|-----------|--------|--------|--------|--------|--------|
| 01JAN2000 | 0.9724 | 1.0000 | 0.9724 | 1.1349 | 1.0807 |

Save Data As... Return Help

**Figure 10.5.** Performing a Benefit-Cost Ratio Analysis

The results displayed in Figure 10.5 indicate that investments 2, 4, and 5 have ratios greater than 1. Therefore, each is profitable with a MARR of 9%.

## Computing a Uniform Periodic Equivalent

Suppose you need a warehouse for ten years. You have two options:

- Pay rent for ten years at \$23,000 per year.
- Build a two-stage facility that you will maintain and which you intend to sell at the end of those ten years.

Datasets describing these scenarios are available in the portfolio `SASHELP.INVSAMP.BUYRENT`. Which option is more financially sound if you desire a 12% MARR?

Open the portfolio `SASHELP.INVSAMP.BUYRENT` and compare the options.

To perform the periodic equivalent, follow these steps:

1. Load the portfolio `SASHELP.INVSAMP.BUYRENT`.
2. Select both cashflows.
3. Select **Analyze** → **Periodic Equivalent....** This opens the Uniform Periodic Equivalent dialog box.
4. Enter 01JAN1996 for the **Start Date**.
5. Enter 10 for the **Number of Periods**.
6. Select YEAR for the **Interval**.
7. Enter 12 for the **Constant MARR**.
8. Click **Create Time Value Summary**.

Uniform Periodic Equivalent

Analysis Specifications

Start Date: 01JAN1996 Interval: YEAR

Number of Periods: 10 Constant MARR: 12.00

Create Periodic Equivalent Summary

Periodic Equivalent Summary

| Name | Amount    |
|------|-----------|
| buy  | -21868.44 |
| rent | -20535.71 |

Save Data As... Return Help

**Figure 10.6.** Computing a Uniform Periodic Equivalent

Figure 10.6 indicates that to rent costs about \$1,300 less each year. Hence, renting is more financially sound. Notice the periodic equivalent for renting is not \$23,000. This is because the the \$23,000 per year does not account for the MARR.

---

## Performing a Breakeven Analysis

In the previous example you computed the uniform periodic equivalent for a rent-buy scenario. Now let's perform a breakeven analysis to see how the MARR affects the time values.

To perform the breakeven analysis, follow these steps:

1. Select both options.
2. Select **Analyze** → **Breakeven Analysis....**
3. Enter 01JAN1996 for the **Date**.
4. Enter 12.0 for **Value**.
5. Enter 4.0 for **(+/-)**.
6. Enter 0.5 for **Increment by**.
7. Click **Create Breakeven Analysis Summary** to fill the **Breakeven Analysis Summary** area as displayed in Figure 10.7.

**Breakeven Analysis**

Analysis Specifications

Analysis:  Variable:

Date:  Value:

+/-:

Increment by:

Breakeven Analysis Summary

| Date      | buy        | rent       | MARR    |
|-----------|------------|------------|---------|
| 01JAN1996 | -138119.27 | -111164.23 | 16.0000 |
| 01JAN1996 | -138200.93 | -113265.33 | 15.5000 |
| 01JAN1996 | -138270.89 | -115431.68 | 15.0000 |
| 01JAN1996 | -138328.24 | -117665.87 | 14.5000 |
| 01JAN1996 | -138371.98 | -119970.66 | 14.0000 |

**Figure 10.7.** Performing a Breakeven Analysis

Click **Graphics...** to view a plot displaying the relationship between time value and MARR.

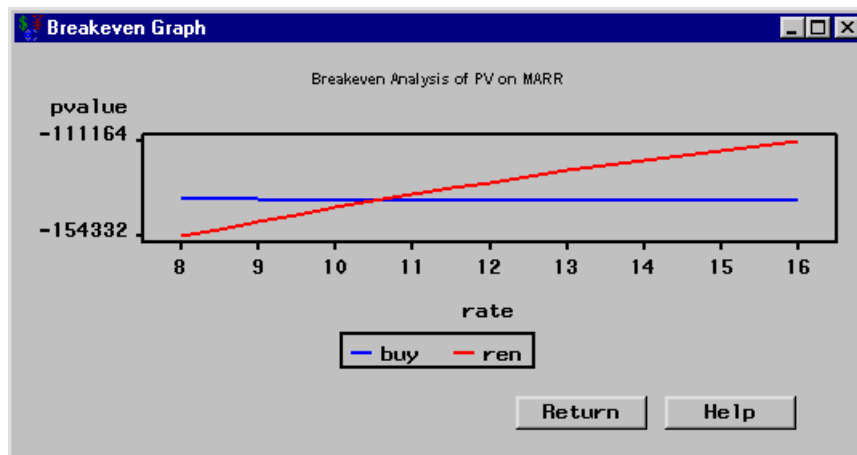


**Figure 10.8.** Viewing a Breakeven Graph

As shown in Figure 10.8 renting is better if you want a MARR of 12%. However, if your MARR should drop to 10.5%, buying would be better.

With a single investment, knowing where the graph has a time value of 0 tells the MARR when a venture switches from being profitable to a loss. With multiple investments, knowing where the graphs for the various investments cross each other tells at what MARR a particular investment becomes more profitable than another.

## Dialog Box Guide

### Time Value Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an time value analysis, select **Analyze** → **Time Value...** from the Investment Analysis dialog box's menu bar. This opens the Time Value Analysis dialog box displayed in Figure 10.9.

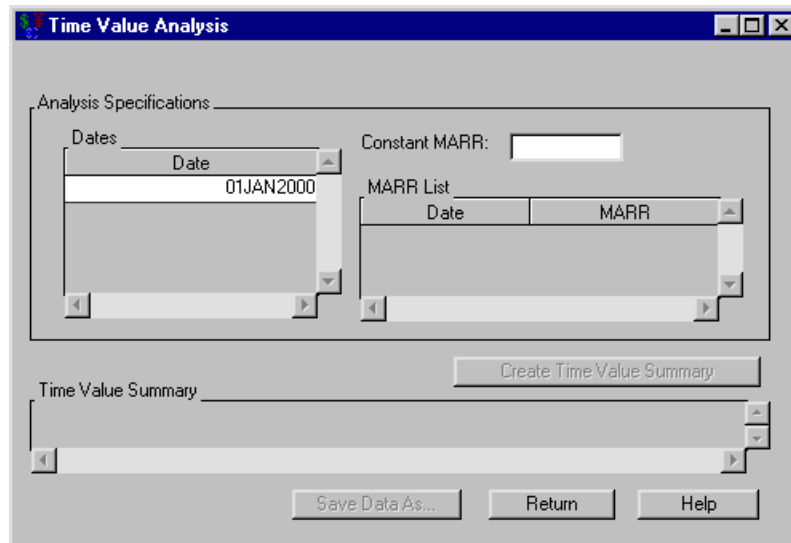


**Figure 10.9.** Time Value Analysis Dialog Box

The following items are displayed:

#### Analysis Specifications

**Dates** holds the list of dates as of which to perform the time value analysis. Right-clicking within the **Dates** area reveals many helpful tools for managing date lists.

**Constant MARR** holds the desired MARR for the time value analysis. This value is used if the **MARR List** area is empty.

**MARR List** holds date-rate pairs that express your desired MARR as it changes over time. Each date refers to when that expected MARR begins. Right-clicking within the **MARR List** area reveals many helpful tools for managing date-rate pairs.

**Create Time Value Summary** becomes available when you adequately specify the analysis within the **Analysis Specifications** area. Clicking **Create Time Value Summary** then fills the **Time Value Summary** area.

**Time Value Summary** fills when you click **Create Time Value Summary**. The table contains a row for each date in the **Dates** area. The remainder of each row holds the time values at that date, one value for each investment selected.

**Save Data As...** becomes available when you fill the time value summary. Clicking it opens the Save Output Dataset dialog box where you can save the **Time Value Summary** (or portions thereof) as a SAS Dataset.

**Return** takes you back to the Investment Analysis dialog box.

## Uniform Periodic Equivalent

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a uniform periodic equivalent, select **Analyze** → **Periodic Equivalent...** from the Investment Analysis dialog box's menu bar. This opens the Uniform Periodic Equivalent dialog box displayed in Figure 10.10.

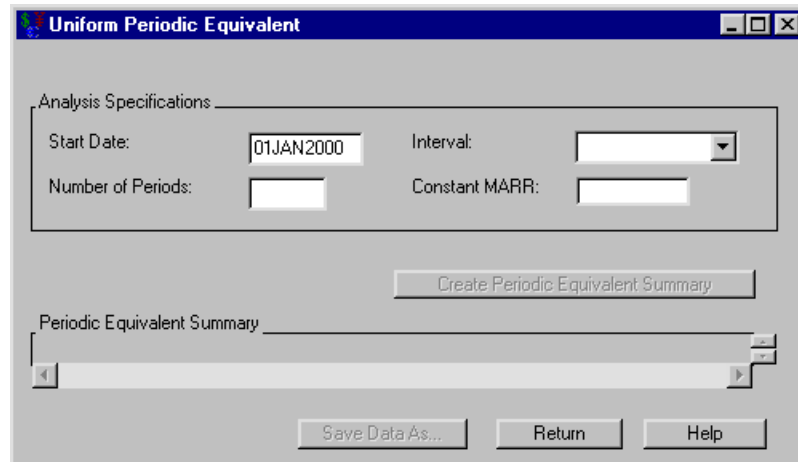


**Figure 10.10.** Uniform Periodic Equivalent Dialog Box

The following items are displayed:

### Analysis Specifications

**Start Date** holds the date the uniform periodic equivalents begin.

**Number of Periods** holds the number of uniform periodic equivalents.

**Interval** holds how often the uniform periodic equivalents occur.

**Constant MARR** holds the Minimum Attractive Rate of Return.

**Create Periodic Equivalent Summary** becomes available when you adequately fill the **Analysis Specification** area. Clicking **Create Periodic Equivalent Summary** then fills the periodic equivalent summary.

**Periodic Equivalent Summary** fills with two columns when you click **Create Periodic Equivalent Summary**. The first column lists the investments selected. The second column lists the computed periodic equivalent amount.

**Save Data As...** becomes available when you generate the periodic equivalent summary. Clicking it opens the Save Output Dataset dialog box where you can save the Periodic Equivalent Summary (or portions thereof) as a SAS Dataset.

**Return** takes you back to the Investment Analysis dialog box.



---

## Internal Rate of Return

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an internal rate of return calculation, select **Analyze** → **Internal Rate of Return...** from the Investment Analysis dialog box's menu bar. This opens the Internal Rate of Return dialog box displayed in Figure 10.11.

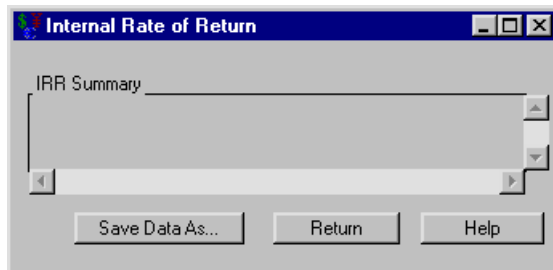


**Figure 10.11.** Internal Rate of Return Dialog Box

The following items are displayed:

**IRR Summary** contains a row for each deposit. Each row holds:

**Name** holds the name of the investment.

**IRR** holds the internal rate of return for that investment.

**interval** holds the interest rate interval for that **IRR**.

**Save Data As...** Clicking it opens the Save Output Dataset dialog box where you can save the IRR summary (or portions thereof) as a SAS Dataset.

**Return** takes you back to the Investment Analysis dialog box.

---

## Benefit-Cost Ratio Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to compute a benefit-cost ratio, select **Analyze** → **Benefit-Cost Ratio...** from the Investment Analysis dialog box's menu bar. This opens the Benefit-Cost Ratio Analysis dialog box displayed in Figure 10.12.

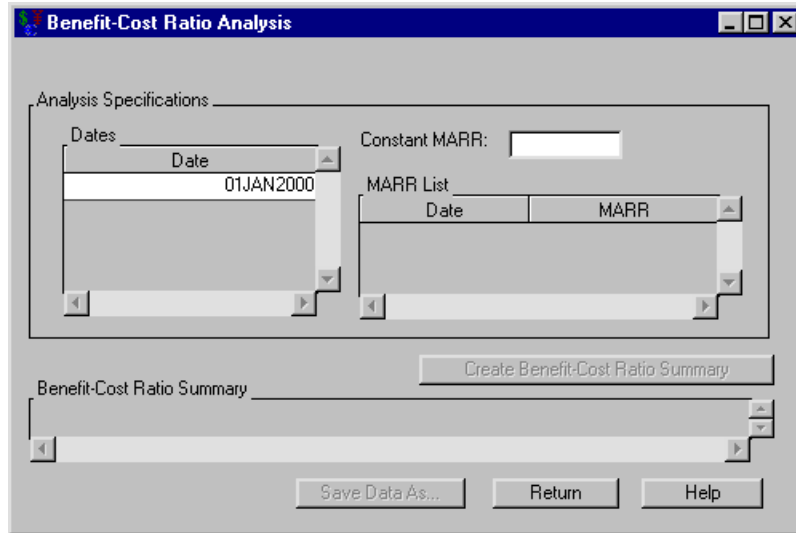


**Figure 10.12.** Benefit-Cost Ratio Analysis Dialog Box

The following items are displayed:

#### Analysis Specifications

**Dates** holds the dates as of which to compute the Benefit-Cost ratios.

**Constant MARR** holds the desired MARR. This value is used if the **MARR List** area is empty.

**MARR List** holds date-rate pairs that express your desired MARR as it changes over time. Each date refers to when that expected MARR begins. Right-clicking within the **MARR List** area reveals many helpful tools for managing date-rate pairs.

**Create Benefit-Cost Ratio Summary** becomes available when you adequately specify the analysis. Clicking **Create Benefit-Cost Ratio Summary** fills the benefit-cost ratio summary.

**Benefit-Cost Ratio Summary** fills when you click **Exchange the Rates**. The area contains a row for each date in the **Dates** area. The remainder of each row holds the benefit-cost ratios at that date, one value for each investment selected.

**Save Data As...** becomes available when you generate the benefit-cost ratio summary. Clicking it opens the Save Output Dataset dialog box where you can save the Benefit-Cost Summary (or portions thereof) as a SAS Dataset.

**Return** takes you back to the Investment Analysis dialog box.

## Breakeven Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a breakeven analysis, select **Analyze** → **Breakeven Analysis...** from the Investment Analysis dialog box's menu bar. This opens the Breakeven Analysis dialog box displayed in Figure 10.13.

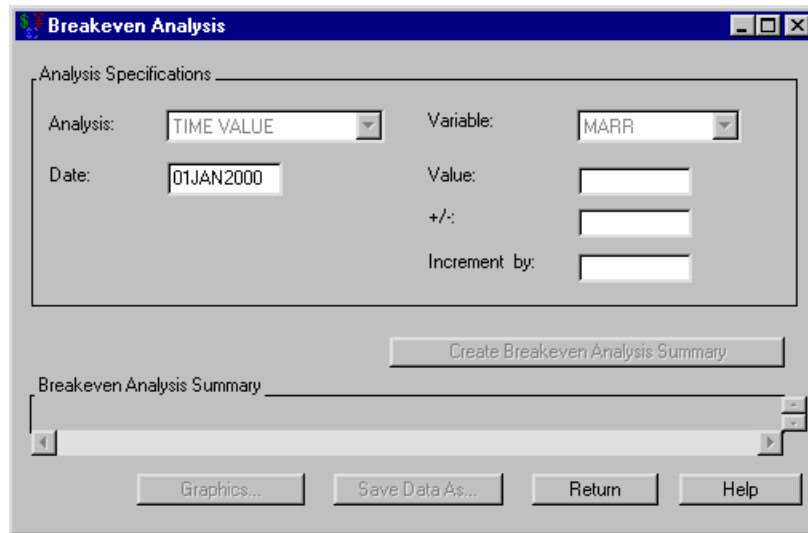


**Figure 10.13.** Breakeven Analysis Dialog Box

The following items are displayed:

### Analysis Specification

**Analysis** holds the analysis type. Only Time Value is currently available.

**Date** holds the date for which you perform this analysis.

**Variable** holds the variable upon which the breakeven analysis will vary. Only MARR is currently available.

**Value** holds the desired rate upon which to center the analysis.

**+/-** holds the maximum deviation from the **Value** to consider.

**Increment by** holds the increment by which the analysis is calculated.

**Create Breakeven Analysis Summary** becomes available when you adequately specify the analysis. Clicking **Create Breakeven Analysis Summary** then fills the **Breakeven Analysis Summary** area.

**Breakeven Analysis Summary** fills when you click **Create Breakeven Analysis Summary**. The schedule contains a row for each MARR and date.

**Graphics...** becomes available when you fill the **Breakeven Analysis Summary** area. Clicking it opens the Breakeven Graph graph representing the time value versus MARR.

**Save Data As...** becomes available when you generate the breakeven analysis summary. Clicking it opens the Save Output Dataset dialog box where you can save the Breakeven Analysis Summary (or portions thereof) as a SAS Dataset.

**Return** takes you back to the Investment Analysis dialog box.

---

## Breakeven Graph

Suppose you perform a breakeven analysis in the Breakeven Analysis dialog box. Once you create the breakeven analysis summary, you can click the **Graphics...** button to open the Breakeven Graph dialog box displayed in Figure 10.14.

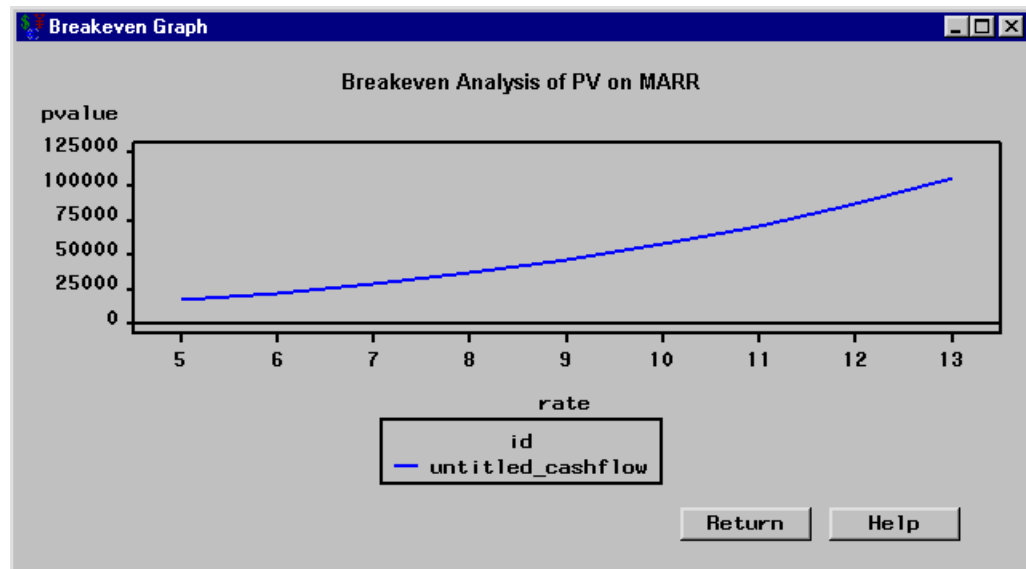


**Figure 10.14.** Breakeven Graph Dialog Box

The following item is displayed:

**Return** takes you back to the Breakeven Analysis dialog box.



# Chapter 11

## Details

### Chapter Table of Contents

---

|                                                    |     |
|----------------------------------------------------|-----|
| <b>INVESTMENTS AND DATASETS</b> . . . . .          | 271 |
| Saving Output to SAS Datasets . . . . .            | 271 |
| Loading a SAS Dataset into a List . . . . .        | 272 |
| Saving Data from a List to a SAS Dataset . . . . . | 273 |
| <b>RIGHT MOUSE BUTTON OPTIONS</b> . . . . .        | 274 |
| <b>DEPRECIATION METHODS</b> . . . . .              | 274 |
| Straight Line (SL) . . . . .                       | 274 |
| Sum-of-years Digits . . . . .                      | 275 |
| Declining Balance (DB) . . . . .                   | 276 |
| Comparison of Depreciation Methods . . . . .       | 278 |
| <b>RATE INFORMATION</b> . . . . .                  | 278 |
| The Tools Menu . . . . .                           | 278 |
| Dialog Box Guide . . . . .                         | 279 |
| Minimum Attractive Rate of Return (MARR) . . . . . | 279 |
| Income Tax Specification . . . . .                 | 280 |
| Inflation Specification . . . . .                  | 281 |
| <b>REFERENCE</b> . . . . .                         | 281 |



## Chapter 11

# Details

---

## Investments and Datasets

Investment Analysis provides tools to assist you in moving data between SAS datasets and lists you can use within Investment Analysis.

---

### Saving Output to SAS Datasets

Many investment specifications have a button that reads **Save Data As....** Clicking that button opens the Save Output Dataset dialog box (see Figure 11.1). This dialog box enables you to save all or part of the area generated by the specification.

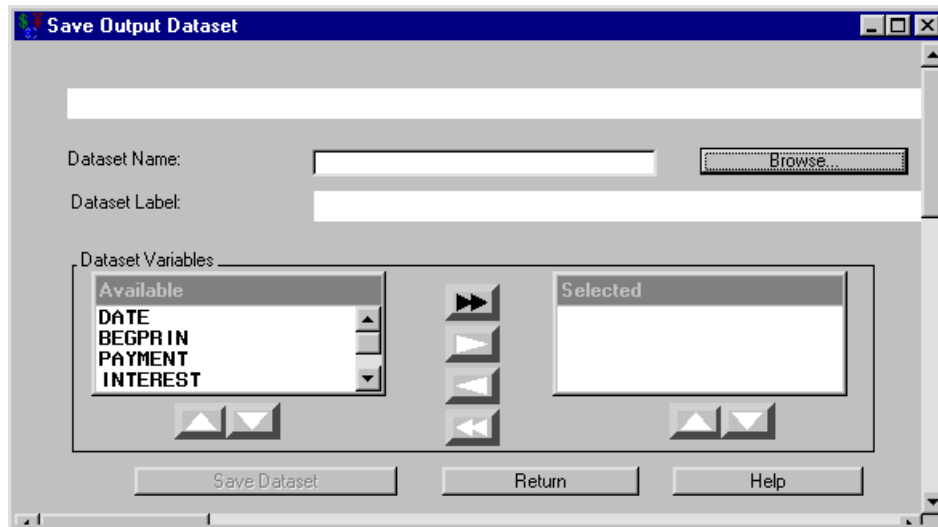


**Figure 11.1.** Saving to a Dataset

The following items are displayed:

**Dataset Name** holds the SAS dataset name to which you wish to save.

**Browse...** opens the standard SAS **Open** dialog box, which enables you to select an existing dataset to overwrite.

**Dataset Label** holds the dataset's label.



**Dataset Variables** organizes variables. The variables listed in the **Selected** area will be included in the dataset.

- You can select variables one at a time, by clicking the single right-arrow after each selection to move it to the **Selected** area.
- If the desired dataset has many variables you wish to save, it may be simpler to follow these steps:
  1. Click the double right arrow to select all available variables.
  2. Remove any unwanted variable by selecting it from the **Selected** area and clicking the single left arrow.
- The double left arrow removes all selected variables from the proposed dataset.
- The up and down arrows below the **Available** and **Selected** boxes enable you to scroll up and down the list of variables in their respective boxes.

**Save Dataset** attempts to save the dataset. If the Dataset name exists, you are asked if you want to replace the dataset, append to the existing Dataset, or cancel the current save attempt. You then return to this dialog box ready to create another dataset to save.

**Return** takes you back to the specification dialog box.

---

## Loading a SAS Dataset into a List

Right-click in the area you wish to load the list and release on **Load...** This opens the Load Dataset dialog box (see Figure 11.2).

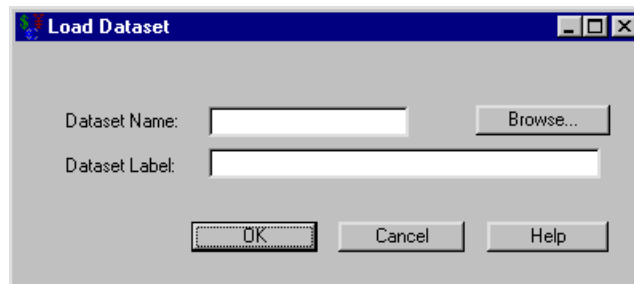


**Figure 11.2.** Load Dataset Dialog Box

The following items are displayed:

**Dataset Name** holds the name of the SAS dataset you wish to load.

**Browse...** opens the standard SAS **Open** dialog box, which aids in finding a dataset to load. If there is a **Date** variable in the dataset, Investment Analysis loads it into the list. If there is no **Date** variable, it loads the first available time-formatted variable. If an amount or rate variable is needed, Investment Analysis searches the dataset for a **Amount** or **Rate** variable to use. Otherwise it takes the first numeric variable that is not used by the **Date** variable.

**Dataset Label** holds a dataset label.

**OK** attempts to load the dataset specified in **Dataset Name**. If the specified dataset exists, clicking **OK** returns you to the calling dialog box with the selected dataset filling the list. If the specified dataset does not exist and you click **OK**, you receive an error message and no dataset is loaded.

**Cancel** returns you to the calling dialog box without loading a dataset. To load values from a dataset into a list, follow these steps:

---

## Saving Data from a List to a SAS Dataset

Right-click in the area you wish to hold the list, and release on **Save....** This opens the Save Dataset dialog box.

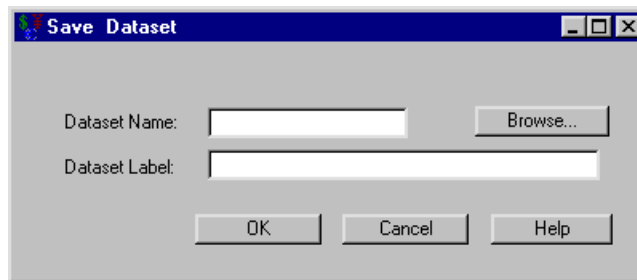


**Figure 11.3.** Save Dataset Dialog Box

The following items are displayed:

**Dataset Name** holds the SAS dataset name to which you wish to save.

**Browse...** opens the standard SAS **Save As** dialog box, which enables you to find an existing dataset to overwrite.

**Dataset Label** holds a user-defined description to be saved as the label of the dataset.

**OK** saves the current data to the dataset specified in **Data set Name**. If the specified dataset does not already exist, clicking **OK** saves the dataset and returns you to the calling dialog box. If the specified dataset does already exist, clicking **OK** warns you and enables you to replace the old dataset with the new dataset or cancel the save attempt.

**Cancel** aborts the save process. Clicking **Cancel** returns you to the calling dialog box without attempting to save.

---

## Right Mouse Button Options

A pop-up menu often appears when you right-click within table editors. The menus offer tools to aid in the management of the table's entries. Most table editors provide the following options.

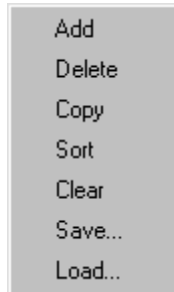


**Figure 11.4.** Right-Clicking Options

**Add** creates a blank row.

**Delete** removes any currently selected row.

**Copy** duplicates the currently selected row.

**Sort** arranges the rows in chronological order according to the date variable.

**Clear** empties the table of all rows.

**Save...** opens the Save Dataset dialog box where you can save the all rows to a SAS Dataset for later use.

**Load...** opens the Load Dataset dialog box where you select a SAS Dataset to fill the rows.

If you wish to perform one of these actions on a collection of rows, you must select a collection of rows before right-clicking. To select an adjacent list of rows, do the following: click the first pair, hold down SHIFT, and click the final pair. Once the list of rows is selected, you may release the SHIFT key.

---

## Depreciation Methods

Suppose an asset's price is \$20,000 and it has a salvage value of \$5,000 in five years. The following sections describe various methods to quantify the depreciation.

---

### Straight Line (SL)

This method assumes a constant depreciation value per year.

Assuming the price of a depreciating asset is  $P$  and its salvage value after  $N$  years is  $S$ ,

$$\text{Annual Depreciation} = \frac{P-S}{N}$$

For our example, the annual depreciation would be

$$\frac{\$20,000 - \$5,000}{5} = \$3,000$$

## Sum-of-years Digits

An asset often loses more of its value early in its lifetime. A method that exhibits this dynamic is desirable.

Assume an asset depreciates from price  $P$  to salvage value  $S$  in  $N$  years. First compute the value:  $\text{sum-of-years} = 1 + 2 + \cdots + N$ . The depreciation for the years after the asset's purchase is:

**Table 11.1.** Sum-of-years General Example

| year number | annual depreciation                      |
|-------------|------------------------------------------|
| first       | $\frac{N}{\text{sum-of-years}}(P - S)$   |
| second      | $\frac{N-1}{\text{sum-of-years}}(P - S)$ |
| third       | $\frac{N-2}{\text{sum-of-years}}(P - S)$ |
| $\vdots$    | $\vdots$                                 |
| final       | $\frac{1}{\text{sum-of-years}}(P - S)$   |

For the  $i$ th year of the asset's use this equation generalizes to

$$\text{Annual Depreciation} = \frac{N+1-i}{\text{sum-of-years}}(P - S)$$

For our example,  $N = 5$  and the sum of years is  $1 + 2 + 3 + 4 + 5 = 15$ . The depreciation during the first year is

$$(\$20,000 - \$5,000) \frac{5}{15} = \$5,000$$

Table 11.2 describes how Declining Balance would depreciate the asset.

**Table 11.2.** Sum-of-years Example

| Year | Depreciation                                  | Year-end Value |
|------|-----------------------------------------------|----------------|
| 1    | $(\$20,000 - \$5,000) \frac{5}{15} = \$5,000$ | \$15,000.00    |
| 2    | $(\$20,000 - \$5,000) \frac{4}{15} = \$4,000$ | \$11,000.00    |
| 3    | $(\$20,000 - \$5,000) \frac{3}{15} = \$3,000$ | \$8,000.00     |
| 4    | $(\$20,000 - \$5,000) \frac{2}{15} = \$2,000$ | \$6,000.00     |
| 5    | $(\$20,000 - \$5,000) \frac{1}{15} = \$1,000$ | \$5,000.00     |

And as expected, the value after  $N$  years is  $S$ .

$$\begin{aligned}
 \text{Value after 5 years} &= P - (5 \text{ years' depreciation}) \\
 &= P - \left( \frac{5}{10}(P - S) + \frac{4}{10}(P - S) + \frac{3}{10}(P - S) + \right. \\
 &\quad \left. \frac{2}{10}(P - S) + \frac{1}{10}(P - S) \right) \\
 &= P - (P - S) \\
 &= S
 \end{aligned}$$

## Declining Balance (DB)

Recall that the Straight Line method assumes a constant depreciation value. Conversely, the Declining Balance method assumes a constant depreciation rate per year. And like the Sum-of-years method, more depreciation tends to occur earlier in the asset's life.

Assume the price of a depreciating asset is  $P$  and its salvage value after  $N$  years is  $S$ . You could assume the asset depreciates by a factor of  $\frac{1}{N}$  (or a rate of  $\frac{100}{N}\%$ ). This method is known as Single Declining Balance. In an equation this looks like:

$$\text{Annual Depreciation} = \frac{1}{N} \text{ Previous year's value}$$

So for our example, the depreciation during the first year is

$$\frac{\$20,000}{5} = \$4,000$$

Table 11.3 describes how Declining Balance would depreciate the asset.

**Table 11.3.** Declining Balance Example

| Year | Depreciation                         | Year-end Value |
|------|--------------------------------------|----------------|
| 1    | $\frac{\$20,000.00}{5} = \$4,000.00$ | \$16,000.00    |
| 2    | $\frac{\$16,000.00}{5} = \$3,200.00$ | \$12,800.00    |
| 3    | $\frac{\$12,800.00}{5} = \$2,560.00$ | \$10,240.00    |
| 4    | $\frac{\$10,240.00}{5} = \$2,048.00$ | \$8,192.00     |
| 5    | $\frac{\$8,192.00}{5} = \$1,638.40$  | \$6,553.60     |

### DB Factor

You could also accelerate the depreciation by increasing the factor (and hence the rate) at which depreciation occurs. Other commonly accepted depreciation rates are  $\frac{200}{N}\%$  (called Double Declining Balance as the depreciation factor becomes  $\frac{2}{N}$ ) and  $\frac{150}{N}\%$ . Investment Analysis enables you to choose between these three types for Declining Balance: 2 (with  $\frac{200}{N}\%$  depreciation), 1.5 (with  $\frac{150}{N}\%$ ), and 1 (with  $\frac{100}{N}\%$ ).

### Declining Balance and the Salvage Value

The Declining Balance method assumes that depreciation is faster earlier in an asset's life; this is what you wanted. But notice the final value is greater than the salvage

value. Even if the salvage value were greater than \$6,553.60, the final year-end value would not change. The salvage value never enters the calculation, so there is no way for the salvage value to force the depreciation to assume its value. Newnan and Lavelle (1998) describe two ways to adapt the Declining Balance method to assume the salvage value at the final time. One way is as follows:

Suppose you call the depreciated value after  $i$  years  $V(i)$ . This sets  $V(0) = P$  and  $V(N) = S$ .

- If  $V(N) > S$  according to the usual calculation for  $V(N)$ , redefine  $V(N)$  to equal  $S$ .
- If  $V(i) < S$  according to the usual calculation for  $V(i)$  for some  $i$  (and hence for all subsequent  $V(i)$  values), you can redefine all such  $V(i)$  to equal  $S$ .

This alteration to Declining Balance forces the depreciated value of the asset after  $N$  years to be  $S$  and keeps  $V(i)$  no less than  $S$ .

### **Conversion to SL**

The second (and preferred) way to force Declining Balance to assume the salvage value is by Conversion to Straight Line. If  $V(N) > S$ , the first way redefines  $V(N)$  to equal  $S$ ; you can think of this as converting to the Straight Line method for the last timestep.

If the  $V(N)$  value supplied by DB is appreciably larger than  $S$ , then the depreciation in the final year would be unrealistically large. An alternate way is to compute the DB and SL step at each timestep and take whichever step gives a larger depreciation (unless DB drops below the salvage value).

Once SL assumes a larger depreciation, it continues to be larger over the life of the asset. This forces the value at the final time to equal the salvage value as SL forces this. As an algorithm, this looks like

```

V(0) = P;
for i=1 to N
    if DB step > SL step from (i,V(i))
        take a DB step to make V(i);
    else
        break;
for j = i to N
    take a SL step to make V(j);

```

The MACRS discussed in Depreciation Table... is actually a variation on the Declining Balance with conversion to Straight Line method.

## Comparison of Depreciation Methods

Table 11.4 displays the depreciation for four depreciation methods.

**Table 11.4.** Comparison of Depreciation Methods

| year | sbvalue  | deprectn | ebvalue  |
|------|----------|----------|----------|
| 1999 | 20000.00 | 3000.00  | 17000.00 |
| 2000 | 17000.00 | 3000.00  | 14000.00 |
| 2001 | 14000.00 | 3000.00  | 11000.00 |
| 2002 | 11000.00 | 3000.00  | 8000.00  |
| 2003 | 8000.00  | 3000.00  | 5000.00  |

Straight Line

| year | sbvalue  | deprectn | ebvalue  |
|------|----------|----------|----------|
| 1999 | 20000.00 | 4000.00  | 16000.00 |
| 2000 | 16000.00 | 6400.00  | 9600.00  |
| 2001 | 9600.00  | 3840.00  | 5760.00  |
| 2002 | 5760.00  | 2304.00  | 3456.00  |
| 2003 | 3456.00  | 2304.00  | 1152.00  |
| 2004 | 1152.00  | 1152.00  | 0.00     |

Depreciation Table

| year | sbvalue  | deprectn | ebvalue  |
|------|----------|----------|----------|
| 1999 | 20000.00 | 5000.00  | 15000.00 |
| 2000 | 15000.00 | 4000.00  | 11000.00 |
| 2001 | 11000.00 | 3000.00  | 8000.00  |
| 2002 | 8000.00  | 2000.00  | 6000.00  |
| 2003 | 6000.00  | 1000.00  | 5000.00  |

Sum-of-years Digits

| year | sbvalue  | deprectn | ebvalue  |
|------|----------|----------|----------|
| 1999 | 20000.00 | 8000.00  | 12000.00 |
| 2000 | 12000.00 | 4800.00  | 7200.00  |
| 2001 | 7200.00  | 2200.00  | 5000.00  |
| 2002 | 5000.00  | 0.00     | 5000.00  |
| 2003 | 5000.00  | 0.00     | 5000.00  |

Declining Balance

- Under Depreciation Table, realize a 5-year class MACRS Depreciation actually lasts 6 years.
- The Declining Balance method is Double Declining Balance with conversion to Straight Line.

For further reference, consider Newnan and Lavelle (1998). They offer explanations and graphs for the individual depreciation methods. They also analyze the differences between the various methods.

## Rate Information

### The Tools Menu

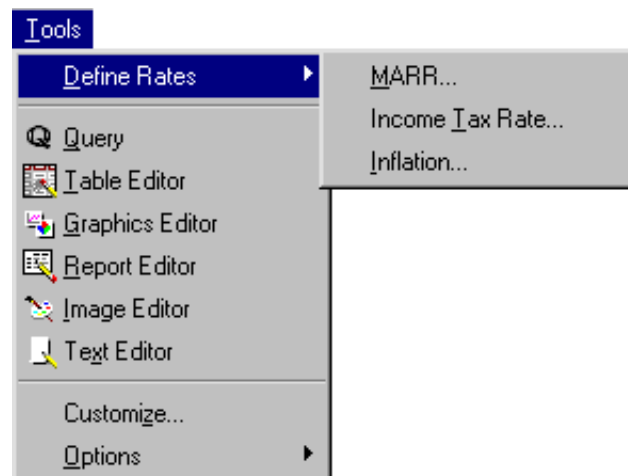


**Figure 11.5.** The Tools Menu

The **Tools** → **Define Rates** menu offers the following options:

**MARR...** opens the Minimum Attractive Rate of Return (MARR) dialog box.

**Income Tax Rate...** opens the Income Tax Specification dialog box.

**Inflation...** opens the Inflation Specification dialog box.

---

## Dialog Box Guide

### *Minimum Attractive Rate of Return (MARR)*

Selecting **Tools** → **Define Rates** → **MARR** from the Investment Analysis dialog box menu bar opens the MARR dialog box displayed in Figure 11.6.

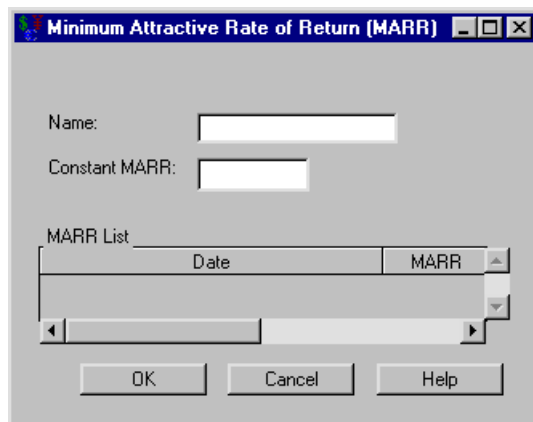


**Figure 11.6.** MARR Dialog Box

**Name** holds the name you assign to the MARR specification. This must be a valid SAS name.

**Constant MARR** holds the numeric value you choose to be the constant MARR. This value is used if the **MARR List** table editor is empty.

**MARR List** holds date-MARR pairs where the date refers to when the particular MARR value begins. Each date is a SAS date.

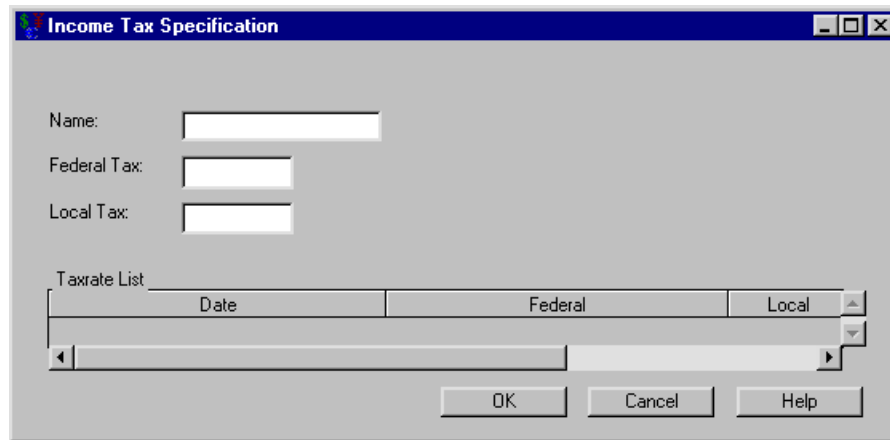
**OK** returns you to the Investment Analysis dialog box. Pressing it causes the MARR specification above to be assumed when you do not specify MARR rates in a dialog box that needs MARR rates.

**Cancel** returns you to the Investment Analysis dialog box, discarding any work done in the MARR dialog box.



### Income Tax Specification

Selecting **Tools** → **Define Rates** → **Income Tax Rate** from the Investment Analysis dialog box menu bar opens the Income Tax Specification dialog box displayed in Figure 11.7.

The dialog box is titled "Income Tax Specification" and has a standard Windows-style title bar with minimize, maximize, and close buttons. It contains three input fields: "Name:", "Federal Tax:", and "Local Tax:", each followed by a text box. Below these is a "Taxrate List" section which is a table with three columns: "Date", "Federal", and "Local". The table has a scroll bar on the right. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

| Date | Federal | Local |
|------|---------|-------|
|      |         |       |

**Figure 11.7.** Income Tax Specification Dialog Box

**Name** holds the name you assign to the Income Tax specification. This must be a valid SAS name.

**Federal Tax** holds the numeric value you desire to be the constant Federal Tax.

**Local Tax** holds the numeric value you desire to be the constant Local Tax.

**Taxrate List** holds date-Income Tax triples where the date refers to when the particular Income Tax value begins. Each date is a SAS date, and the value is a percentage between 0% and 100%.

**OK** returns you to the Investment Analysis dialog box. Clicking it causes the income tax specification above to be the default income tax rates when using the After Tax Cashflow Calculation dialog box.

**Cancel** returns you to the Investment Analysis dialog box, discarding any editions since this dialog box was opened.

### ***Inflation Specification***

Selecting **Tools** → **Define Rates** → **Inflation** from the Investment Analysis dialog box menu bar opens the Inflation Specification dialog box displayed in Figure 11.8.

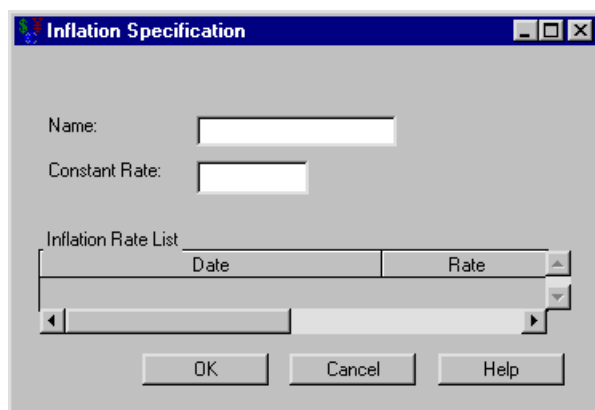


**Figure 11.8.** Inflation Specification Dialog Box

**Name** holds the name you assign to the Inflation specification. This must be a valid SAS name.

**Constant Rate** holds the numeric value you desire to be the constant inflation rate. This value is used if the **Inflation Rate List** table editor is empty.

**Inflation Rate List** holds date-rate pairs where the date refers to when the particular inflation rate begins. Each date is a SAS date and the rate is a percentage between 0% and 120%.

**OK** returns you to the Investment Analysis dialog box. Pressing it causes the inflation specification above to be assumed when you use the Constant Dollar Calculation dialog box and do not specify inflation rates.

**Cancel** returns you to the Investment Analysis dialog box, discarding any editions since this dialog box was opened.

---

## **Reference**

Newnan, Donald G. and Lavelle, Jerome P. (1998), *Engineering Economic Analysis*, Austin, Texas: Engineering Press.



# Subject Index

## A

After Tax Cashflow Calculation, 244  
Annuity  
    See Uniform Periodic Equivalent

## B

Bayesian vector autoregressive models  
    VARMAX procedure, 51, 78  
Bayesian vector error correction models  
    VARMAX procedure, 56  
Benefit-Cost Ratio Analysis, 258  
Bond, 214  
Breakeven Analysis, 260

## C

Cashflow  
    Generic Cashflow  
causality testing  
    VARMAX procedure, 62  
Census X-12 method  
    See X12 procedure  
cointegration testing  
    VARMAX procedure, 53  
confidence limits  
    VARMAX procedure, 128  
Constant Dollar Calculation, 247  
CONTENTS procedure  
    SASECRSP engine, 14  
    SASEFAME engine, 24  
convert option  
    SASEFAME engine, 23  
creating a FAME view  
    See SASEFAME engine  
Currency Conversion, 245

## D

Depreciation, 211  
Dickey-Fuller test  
    VARMAX procedure, 78  
differencing  
    VARMAX procedure, 70  
DRI data files in FAME.db  
    See SASEFAME engine  
DRI/McGraw-Hill data files in FAME.db  
    See SASEFAME engine  
DROP in the DATA step  
    SASEFAME engine, 30

## F

FAME data files  
    See SASEFAME engine  
FAME Information Services Databases  
    See SASEFAME engine

## G

Generic Cashflow, 217

## I

Internal Rate of Return, 257  
Investment Portfolio, 198

## K

KEEP in the DATA step  
    SASEFAME engine, 30

## L

LIBNAME interface engine for FAME database  
    See SASEFAME engine  
libname statement  
    SASECRSP engine, 13  
LIBNAME statement  
    SASEFAME engine, 23  
Loan, 204

## M

main economic indicators (OECD) data files in  
    FAME.db  
    See SASEFAME engine  
MARR, 279

## N

national accounts data files (OECD) in FAME.db  
    See SASEFAME engine

## O

OECD data files in FAME.db  
    See SASEFAME engine  
Organization for Economic Cooperation and Development data files in FAME.db  
    See SASEFAME engine

## P

parameter estimation and testing under restrictions  
    VARMAX procedure, 60  
Periodic Equivalent  
    See Uniform Periodic Equivalent  
Portfolio

- Investment Portfolio
- predicted values
  - VARMAX procedure, 128
- Present Value Analysis
  - See Time Value Analysis

**R**

- reading from a FAME data base
  - SASEFAME engine, 23
- reading from CRSP data files
  - SASECRSP engine, 13
- RENAME in the DATA step
  - SASEFAME engine, 30
- residuals
  - VARMAX procedure, 128

**S**

- sample correlations
  - VARMAX procedure, 72
- SAS DATA step
  - SASECRSP engine, 13
  - SASEFAME engine, 23
- SAS output data set
  - SASECRSP engine, 15
  - SASEFAME engine, 25
- SASECRSP engine
  - CONTENTS procedure, 14
  - libname statement, 13
  - reading from CRSP data files, 13
  - SAS DATA step, 13
  - SAS output data set, 15
  - setid option, 13
  - SQL procedure, creating a view, 14
  - SQL procedure, using clause, 14
- SASEFAME engine
  - CONTENTS procedure, 24
  - convert option, 23
  - creating a FAME view, 23
  - DRI data files in FAME.db, 23
  - DRI/McGraw-Hill data files in FAME.db, 23
  - DROP in the DATA step, 30
  - FAME data files, 23
  - FAME Information Services Databases, 23
  - KEEP in the DATA step, 30
  - LIBNAME interface engine for FAME databases, 23
  - LIBNAME statement, 23
  - main economic indicators (OECD) data files in FAME.db, 23
  - national accounts data files (OECD) in FAME.db, 23
  - OECD data files in FAME.db, 23
  - Organization for Economic Cooperation and Development data files in FAME.db, 23
  - reading from a FAME data base, 23
  - RENAME in the DATA step, 30
  - SAS DATA step, 23
  - SAS output data set, 25
  - SQL procedure, using clause, 24

- SQL procedure, creating a view, 24
- viewing a FAME database, 23
- WHERE in the DATA step, 31

- Savings, 210

- seasonal adjustment
  - time series data, 163
  - X12 procedure, 163

- seasonal component
  - X12 procedure, 163

- setid option
  - SASECRSP engine, 13
- SQL procedure, using clause
  - SASEFAME engine, 24

- SQL procedure, creating a view
  - SASECRSP engine, 14
  - SASEFAME engine, 24
- SQL procedure, using clause
  - SASECRSP engine, 14

**T**

- time intervals
  - VARMAX procedure, 69
- time series data
  - seasonal adjustment, 163
- Time Value Analysis, 256

**U**

- Uniform Periodic Equivalent, 259

**V**

- VARMAX procedure
  - Bayesian vector autoregressive models, 51, 78
  - Bayesian vector error correction models, 56
  - causality testing, 62
  - cointegration testing, 53
  - confidence limits, 128
  - Dickey-Fuller test, 78
  - differencing, 70
  - parameter estimation and testing under restrictions, 60
  - predicted values, 128
  - residuals, 128
  - sample correlations, 72
  - time intervals, 69
  - vector autoregressive models, 45
  - vector autoregressive models with exogenous variables, 57
  - vector error correction models, 51, 80
- vector autoregressive models
  - VARMAX procedure, 45
- vector autoregressive models with exogenous variables
  - VARMAX procedure, 57
- vector error correction models
  - VARMAX procedure, 51, 80
- viewing a FAME database
  - See SASEFAME engine

## **W**

WHERE in the DATA step  
    SASEFAME engine, 31

## **X**

X-12 seasonal adjustment method  
    See X12 procedure  
X-12-ARIMA seasonal adjustment method  
    See X12 procedure  
X12 procedure  
    Census X-12 method, 163  
    seasonal adjustment, 163  
    seasonal component, 163  
    X-12 seasonal adjustment method, 163  
    X-12-ARIMA seasonal adjustment method, 163



# Syntax Index

## A

ALIGN= option  
    ID statement (VARMAX), 69  
ALPHA= option  
    OUTPUT statement (VARMAX), 80  
ARIMA statement  
    X12 procedure, 169

## B

BACK= option  
    OUTPUT statement (VARMAX), 80  
BY statement  
    VARMAX procedure, 66

## C

CAUSAL statement  
    VARMAX procedure, 66  
CENTER option  
    MODEL statement (VARMAX), 70  
COINTEG statement  
    VARMAX procedure, 67  
CONVERT= option  
    LIBNAME statement (SASEFAME), 24

## D

DATA= option  
    PROC VARMAX statement, 65  
    PROC X12 statement, 168  
DATE= option  
    PROC X12 statement, 168  
DFTEST option  
    MODEL statement (VARMAX), 78  
DIF= option  
    MODEL statement (VARMAX), 70  
DIFF= option  
    IDENTIFY statement (X12), 171  
DIFX= option  
    MODEL statement (VARMAX), 70  
DIFY= option  
    MODEL statement (VARMAX), 70

## E

ECM= option  
    MODEL statement (VARMAX), 80  
ESTIMATE statement  
    X12 procedure, 170  
EXOGENEITY option  
    COINTEG statement (VARMAX), 67

## H

H= option  
    COINTEG statement (VARMAX), 67

## I

ID statement  
    VARMAX procedure, 69  
IDENTIFY statement  
    X12 procedure, 170  
INSET= option  
    LIBNAME statement (SASECRSP), 15  
INTERVAL= option  
    ID statement (VARMAX), 69  
    PROC X12 statement, 169

## J

J= option  
    COINTEG statement (VARMAX), 68

## L

LAGMAX= option  
    MODEL statement (VARMAX), 71  
LEAD= option  
    OUTPUT statement (VARMAX), 80  
LIBNAME *libref* SASEFAME statement, 24  
LIBNAME *libref* SASECRSP statement, 14

## M

METHOD= option  
    MODEL statement (VARMAX), 70  
MINIC= option  
    MODEL statement (VARMAX), 75  
MODE= option  
    X11 statement (X12), 175  
MODEL statement  
    VARMAX procedure, 69  
MODEL= option  
    ARIMA statement (X12), 169

## N

NOCURRENTX option  
    MODEL statement (VARMAX), 71  
NOINT option  
    MODEL statement (VARMAX), 71  
NOPRINT option  
    MODEL statement (VARMAX), 72  
    OUTPUT statement (VARMAX), 80  
NORMALIZE= option  
    COINTEG statement (VARMAX), 68



NSEASON= option  
 MODEL statement (VARMAX), 71

## O

OUT= option  
 OUTPUT statement (VARMAX), 80  
 OUTPUT statement (X12), 171  
 OUTCOV  
 PROC VARMAX statement, 66  
 OUTEST= option  
 PROC VARMAX statement, 65  
 OUTPUT statement  
 VARMAX procedure, 80  
 X12 procedure, 171  
 OUTSTAT= option  
 PROC VARMAX statement, 66

## P

P= option  
 MODEL statement (VARMAX), 75  
 PERMNO= option  
 LIBNAME statement (SASECRSP), 15  
 POWER= option  
 TRANSFORM statement (X12), 173  
 PREDEFINED= option  
 PREDEFINED statement (X12), 172  
 PRINT= option  
 MODEL statement (VARMAX), 72  
 PRINTALL option  
 MODEL statement (VARMAX), 72  
 PRINTFORM= option  
 MODEL statement (VARMAX), 72  
 PRIOR option  
 MODEL statement (VARMAX), 78  
 PROC VARMAX statement, 65  
 PROC X12 statement, 168

## R

RANGE= option  
 LIBNAME statement (SASECRSP), 15  
 RANK= option  
 COINTEG statement (VARMAX), 68  
 REGRESSION statement  
 X12 procedure, 172  
 RESTRICT statement  
 VARMAX procedure, 81

## S

SCENTER option  
 MODEL statement (VARMAX), 71  
 SDIFF= option  
 IDENTIFY statement (X12), 171  
 SEASONS= option  
 PROC X12 statement, 169  
 SETID= option  
 LIBNAME statement (SASECRSP), 14  
 SPAN= option  
 PROC X12 statement, 168  
 START= option  
 PROC X12 statement, 168

## T

TEST statement  
 VARMAX procedure, 82  
 TRANSFORM statement  
 X12 procedure, 173  
 TREND= option  
 MODEL statement (VARMAX), 71

## V

VAR statement  
 X12 procedure, 174  
 VARDEF= option  
 MODEL statement (VARMAX), 71  
 VARMAX procedure, 63  
 syntax, 63

## W

WILDCARD= option  
 LIBNAME statement (SASEFAME), 25

## X

X11 statement  
 X12 procedure, 174  
 X12 procedure, 166  
 syntax, 166  
 XLAG= option  
 MODEL statement (VARMAX), 75

# Your Turn

---

If you have comments or suggestions about *SAS/ETS® Software: Changes and Enhancements, Release 8.1*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute  
Publications Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [suggest@sas.com](mailto:suggest@sas.com)



Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido

## **SAS® Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

- 1.** Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
- 2.** Leave the "Subject" line blank.
- 3.** Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.

# **SAS/IML<sup>®</sup> Software: Changes and Enhancements, Release 8.1**

---

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/IML® Software: Changes and Enhancements, Release 8.1*, Cary, NC: SAS Institute Inc., 2000

**SAS/IML® Software: Changes and Enhancements, Release 8.1**

Copyright © 2000 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-707-0

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, May 2000

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. indicates USA registration.

IBM® and all other International Business Machines Corporation product or service names are registered trademarks or trademarks of International Business Machines Corporation in the USA and other countries.

Oracle® and all other Oracle Corporation product or service names are registered trademarks or trademarks of Oracle Corporation in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

Chapter 1. Changes and Enhancements . . . . . 1

Subject Index . . . . . 37

Syntax Index . . . . . 39





# Chapter 1

## Changes and Enhancements

### Chapter Table of Contents

---

|                                                    |           |
|----------------------------------------------------|-----------|
| <b>OVERVIEW . . . . .</b>                          | <b>3</b>  |
| <b>FINANCIAL FUNCTIONS . . . . .</b>               | <b>3</b>  |
| <b>ROBUST REGRESSION . . . . .</b>                 | <b>10</b> |
| <b>MULTIVARIATE TIME SERIES ANALYSIS . . . . .</b> | <b>29</b> |
| <b>REFERENCES . . . . .</b>                        | <b>36</b> |



# Chapter 1

## Changes and Enhancements

---

### Overview

New financial functions, robust regression subroutines, and subroutines related to multivariate time series analysis have been added. All necessary details such as arguments and operands are included.

---

### Financial Functions

The following functions that compute information about financial quantities have been added:

---

#### CONVEXIT Function

**calculates and returns a scalar containing the convexity of a noncontingent cash flow**

**CONVEXIT(*times,flows,ytm*)**

The CONVEXIT function calculates and returns a scalar containing the convexity of a noncontingent cash flow.

|              |                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------|
| <i>times</i> | is an $n$ -dimensional column vector of times. Elements should be non-negative.                       |
| <i>flows</i> | is an $n$ -dimensional column vector of cash flows.                                                   |
| <i>ytm</i>   | is the per-period yield-to-maturity of the cash-flow stream. This is a scalar and should be positive. |

Convexity is essentially a measure of how duration, the sensitivity of price to yield, changes as interest rates change:

$$C = \frac{1}{P} \frac{d^2 P}{dy^2}$$

With cash flows that are not yield-sensitive, and the assumption of parallel shifts to a flat term-structure, convexity is given by

$$C = \frac{\sum_{k=1}^K t_k (t_k + 1) \frac{c(k)}{(1+y)^{t_k}}}{P(1+y)^2}$$

where  $P$  is the present value,  $y$  is the effective per period yield-to-maturity,  $K$  is the number of cash flows, and the  $k$ -th cash flow is  $c(k)$   $t_k$  periods from the present.

The statements

```
timesn=T(do(1,100,1));
flows=repeat(10,100);
ytm=0.1;
convexit=convexit(timesn,flows,ytm);
print convexit;
```

result in the following output:

```
CONVEXIT
199.26229
```

---

## DURATION Function

**calculates and returns a scalar containing the modified duration of a noncontingent cash flow**

**DURATION(times,flows,ytm)**

The DURATION function returns the modified duration of a noncontingent cash flow as a scalar.

|       |                                                                                                       |
|-------|-------------------------------------------------------------------------------------------------------|
| times | is an $n$ -dimensional column vector of times. Elements should be non-negative.                       |
| flows | is an $n$ -dimensional column vector of cash flows.                                                   |
| ytm   | is the per-period yield-to-maturity of the cash-flow stream. This is a scalar and should be positive. |

Duration of a security is generally defined as

$$D = -\frac{\frac{dP}{P}}{dy}$$

In other words, it is the relative change in price for a unit change in yield. Since prices move in the opposite direction to yields, the sign change preserves positivity for convenience. With cash flows that are not yield-sensitive, and the assumption of parallel shifts to a flat term-structure, duration is given by

$$D_{\text{mod}} = \frac{\sum_{k=1}^K t_k \frac{c(k)}{(1+y)^{t_k}}}{P(1+y)}$$

where  $P$  is the present value,  $y$  is the per period effective yield-to-maturity,  $K$  is the number of cash flows, and the  $k$ -th cash flow is  $c(k)$ ,  $t_k$  periods from the present. This

measure is referred to as *modified duration* to differentiate it from the first duration measure ever proposed, *Macaulay duration*:

$$D_{\text{Mac}} = \frac{\sum_{k=1}^K t_k \frac{c(k)}{(1+y)^{t_k}}}{P}$$

This expression also reveals the reason for the name duration, since it is a present-value-weighted average of the duration (that is, timing) of all the cash flows and is hence an “average time-to-maturity” of the bond.

For example, the statements below

```
times={1};
ytm={0.1};
flow={10};
duration=duration(times,flow,ytm);
print duration;
```

produce the output

```
DURATION
0.9090909
```

---

## FORWARD Function

calculates a column vector of forward rates given vectors of spot rates and times

**FORWARD**(*times,spot\_rates*)

The FORWARD function returns an  $n \times 1$  vector of forward rates.

**times** is an  $n \times 1$  column vector of times in consistent units. Elements should be non-negative.

**spot\_rates** is an  $n \times 1$  column vector of corresponding per-period spot rates. Elements should be positive.

The FORWARD function transforms the given spot rates as

$$f_1 = s_1$$

$$f_i = \left( \frac{(1 + s_i)^{t_i}}{(1 + s_{i-1})^{t_{i-1}}} \right)^{\frac{1}{t_i - t_{i-1}}} - 1; \quad i = 2, \dots, n$$

For example, the following statements

```

spt={0.75};
times={1};
forward=forward(times,spt);
print forward;

```

produce the following output:

```

FORWARD
0.75

```

---

## PV Function

**calculates the present value of a vector of cash flows and returns a scalar**

**PV(times,flows,freq,rates)**

The PV function returns a scalar containing the present value of the cash flows based on the specified frequency and rates.

|       |                                                                                                                                                                                                                                                                                                                                      |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| times | is an $n \times 1$ column vector of times. Elements should be non-negative.                                                                                                                                                                                                                                                          |
| flows | is an $n \times 1$ column vector of cash flows.                                                                                                                                                                                                                                                                                      |
| freq  | is a scalar that represents the base of the rates to be used for discounting the cash flows. If positive, it represents discrete compounding as the reciprocal of the number of compoundings. If zero, it represents continuous compounding. If -1, it represents per-period discount factors. No other negative values are allowed. |
| rates | is an $n \times 1$ column vector of rates to be used for discounting the cash flows. Elements should be positive.                                                                                                                                                                                                                    |

A general present value relationship can be written as

$$P = \sum_{k=1}^K c(k)D(t_k)$$

where  $P$  is the present value of the asset,  $\{c(k)\}_{k=1, \dots, K}$  is the sequence of cash flows from the asset,  $t_k$  is the time to the  $k$ -th cash flow in periods from the present, and  $D(t)$  is the discount function for time  $t$ .

With per-unit-time-period discount factors  $d_t$ :

$$D(t) = d_t^t$$

With continuous compounding:

$$D(t) = e^{-r_t t}$$

With discrete compounding:

$$D(t) = (1 + fr)^{-t/f}$$

where  $f > 0$  is the frequency, the reciprocal of the number of compoundings per unit time period.

The following code presents an example of the PV function:

```
timesn=T(do(1,100,1));
flows=repeat(10,100);
freq=50;
rate=repeat(0.10,100);
pv=pv(timesn,flows,freq,rate);
print pv;
```

The result is

```
PV
266.4717
```

---

## RATES Function

**calculates a column vector of interest rates converted from one base to another**

**RATES(rates,oldfreq,newfreq)**

The RATES function returns an  $n \times 1$  vector of interest rates converted from one base to another.

|         |                                                                                                                                                                                                                                                                                   |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rates   | is an $n \times 1$ column vector of rates. Elements should be positive.                                                                                                                                                                                                           |
| oldfreq | is a scalar that represents the old base. If positive, it represents discrete compounding as the reciprocal of the number of compoundings. If zero, it represents continuous compounding. If -1, it represents discount factors. No other negative values are allowed.            |
| newfreq | is a scalar that represents the new base. If positive, it represents discrete compounding as the reciprocal of the number of compoundings. If zero, it represents continuous compounding. If -1, it represents per-period discount factors. No other negative values are allowed. |

Let  $D(t)$  be the discount function, which is the present value of a unit amount to be received  $t$  periods from now. The discount function can be expressed in three different ways:

with per-unit-time-period discount factors  $d_t$ :

$$D(t) = d_t^t$$



with continuous compounding:

$$D(t) = e^{-r_t t}$$

with discrete compounding:

$$D(t) = (1 + fr)^{-t/f}$$

where  $f > 0$  is the frequency, the reciprocal of the number of compoundings per unit time period. The RATES function converts between these three representations.

For example, the following code uses the RATES function:

```
rates=T(do(0.1,0.3,0.1));
oldfreq=0;
newfreq=0;
rates=rates(rates,oldfreq,newfreq);
print rates;
```

The output is

```
RATES
0.1
0.2
0.3
```

---

## SPOT Function

**calculates a column vector of spot rates given vectors of forward rates and times**

**SPOT(times,forward\_rates)**

The SPOT function returns an  $n \times 1$  vector of spot rates.

**times** is an  $n \times 1$  column vector of times in consistent units. Elements should be non-negative.

**forward\_rates** is an  $n \times 1$  column vector of corresponding per-period forward rates. Elements should be positive.

The SPOT function transforms the given spot rates as

$$s_1 = f_1$$

$$s_i = \left( \prod_{j=1}^{j=i} (1 + f_j)^{t_j - t_{j-1}} \right)^{\frac{1}{t_i}} - 1; \quad i = 2, \dots, n$$

where, by convention,  $t_0 = 0$ .

For example, the following code

```
fwd={0.05};
times={1};
spot=spot(times,fwd);
print spot;
```

produces the following output:

```
SPOT
0.05
```

---

## YIELD Function

calculates yield-to-maturity of a cash-flow stream and returns a scalar

**YIELD**(*times,flows,freq,value*)

The YIELD function returns a scalar containing yield-to-maturity of a cash flow stream based on frequency and value specified.

|              |                                                                                                                                                                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>times</i> | is an $n$ -dimensional column vector of times. Elements should be non-negative.                                                                                                                                                                                              |
| <i>flows</i> | is an $n$ -dimensional column vector of cash flows.                                                                                                                                                                                                                          |
| <i>freq</i>  | is a scalar that represents the base of the rates to be used for discounting the cash flows. If positive, it represents discrete compounding as the reciprocal of the number of compoundings. If zero, it represents continuous compounding. No negative values are allowed. |
| <i>value</i> | is a scalar that is the discounted present value of the cash flows.                                                                                                                                                                                                          |

The present value relationship can be written as

$$P = \sum_{k=1}^K c(k)D(t_k)$$

where  $P$  is the present value of the asset,  $\{c(k)\}_{k=1, \dots, K}$  is the sequence of cash flows from the asset,  $t_k$  is the time to the  $k$ -th cash flow in periods from the present, and  $D(t)$  is the discount function for time  $t$ .

With continuous compounding:

$$D(t) = e^{-yt}$$

With discrete compounding:

$$D(t) = (1 + fy)^{-t/f}$$

where  $f > 0$  is the frequency, the reciprocal of the number of compoundings per unit time period, and  $y$  is the yield-to-maturity. The YIELD function solves for  $y$ .

For example, the following code

```
timesn=T(do(1,100,1));
flows=repeat(10,100);
freq=50;
value=682.31027;
yield=yield(timesn,flows,freq,value);
print yield;
```

produces the following output:

```
YIELD
0.0100001
```

---

## Robust Regression

These are the new algorithms for robust regression analysis:

---

### LTS Call

**performs robust regression**

```
CALL LTS(sc, coef, wgt, opt, y <, < x ><, sorb>>);
```

A new algorithm, FAST-LTS, was added to the LTS subroutine in SAS/IML Release 8.1. The FAST-LTS algorithm is set as the default algorithm. The original algorithm is kept for convenience, and can be used temporarily by specifying optn[9]=1. Eventually the original algorithm will be replaced with the new FAST-LTS algorithm.

The original algorithm for the LTS subroutine and the algorithm used in the LMS subroutine are based on the PROGRESS program by Rousseeuw and Leroy (1987). Rousseeuw and Hubert (1996) prepared a new version of PROGRESS to facilitate its inclusion in SAS software, and they have incorporated several recent developments. Among other things, the new version of PROGRESS now yields the exact LMS for simple regression, and the program uses a new definition of the robust coefficient of determination ( $R^2$ ). Therefore, the outputs may differ slightly from those given in Rousseeuw and Leroy (1987) or those obtained from software based on the older version of PROGRESS.

### **FAST-LTS**

Least trimmed squares (LTS) regression is based on the subset of  $h$  cases (out of  $n$ ) whose least squares fit possesses the smallest sum of squared residuals. The coverage  $h$  may be set between  $n/2$  and  $n$ . The LTS method was proposed by Rousseeuw (1984, p. 876) as a highly robust regression estimator, with breakdown value  $(n-h)/n$ . It turned out that the computation time of the previous LTS algorithm grew too fast with the size of the data set, precluding their use for data mining. Rousseeuw and

Van Driessen (1998) developed a new algorithm called FAST-LTS. The basic idea is an inequality involving order statistics and sums of squared residuals. Based on this inequality, techniques called “selective iteration” and “nested extensions” are developed in Rousseeuw and Van Driessen (1998). The new LTS algorithm implements these techniques to achieve faster computation. The intercept adjustment technique is also used in this new algorithm. For small data sets, FAST-LTS typically finds the exact LTS, whereas for larger data sets it gives more accurate results than the previous LTS algorithm and is faster by orders of magnitude. The new algorithm is described briefly as follows; refer to Rousseeuw and Van Driessen (1998) for details:

1. The default  $h$  is  $(n + p + 1)/2$ , where  $p$  is the number of the independent variables. You can choose any integer  $h$  with  $[(n + p + 1)/2] \leq h \leq n$ . The LTS’s breakdown point  $(n - h + 1)/n$  is reported. If you are sure that the data contains less than 25% of contamination, you can obtain a good compromise between breakdown value and statistical efficiency by putting  $h = \lceil .75n \rceil$ .
2. If  $p = 1$  (univariate data), then compute the LTS estimator by the exact algorithm of Rousseeuw and Leroy (1987, pp. 171-172) and stop.
3. From here on,  $p \geq 2$ . If  $n < 600$ , draw a random  $p$ -subset and compute the regression coefficients using these  $p$  points (if the regression is degenerate, draw another  $p$ -subset). Compute the absolute residuals for all points in the data set and select the first  $h$  points with smallest absolute residuals. From this selected  $h$ -subset, carry out C-steps (Concentration step; refer to Rousseeuw and Van Driessen [1998] for details) until convergence. Repeat this procedure 500 or  $\binom{n}{p}$  times (as determined by  $opt[5]$  of LTS Call) and find the ten (at most) solutions with the lowest sums of  $h$  squared residuals. For each of these ten best solutions, take C-steps until convergence and find the best final solution.
4. If  $n > 600$ , construct up to five disjoint random subsets with sizes as equal as possible, but not to exceed 300. Inside each subset, repeat the procedure in step 3  $500/5 = 100$  times and keep the ten best solutions. Pool the subsets, yielding the merged set of size  $n_{\text{merged}}$ . In the merged set, for each of the  $5 \times 10 = 50$  best solutions, carry out two C-steps using  $n_{\text{merged}}$  and  $h_{\text{merged}} = \lceil n_{\text{merged}}(h/n) \rceil$  and keep the ten best solutions. In the full data set, for each of these ten best solutions, take C-steps using  $n$  and  $h$  until convergence and find the best final solution.

### **OPTION Changes**

The previous LTS algorithm is used if  $optn[9] = 1$ ; the FAST-LTS algorithm is set as default (or  $optn[9] = 0$ ).

### **OUTPUT Changes**

Because of the change in algorithm, the output from the FAST-LTS algorithm is different:

1. The “Complete Enumeration for LTS” table and the “Resistant Diagnostic” table do not apply for the FAST-LTS algorithm and are not displayed.
2. The analysis based on “Observations of Best Subset” of size  $p$ , where  $p$  is the number of the independent variables, is replaced by the analysis based on

observations of the best  $h$  of the entire data set obtained after full iteration, which gives the exact LTS estimator and covariances for small data set ( $\binom{n}{p} < 500$ ). LTS Objective Function, Preliminary LTS Scale, Robust R Squared, and Final LTS Scale are also reported for the LTS estimator.

3. The LTS residuals are changed, because of the change of the LTS estimator.
4. The “Coef” vector does not include the best subset.

See the following illustrative example for details.

### **Illustrative Example**

The following example shows the difference between the two algorithms. The second output is generated by the FAST-LTS algorithm.

```

title1 'Compare two algorithms for LTS';

proc iml;
  reset noname;

  x = {42, 37, 37, 28, 18, 18, 19, 20, 15};
  y = {80, 80, 75, 62, 62, 62, 62, 62, 58};
  optn = j(9, 1, .);
  optn[1]= 0;      /* --- with intercept      --- */
  optn[2]= 4;      /* --- print all output      --- */
  optn[3]= 3;      /* --- compute LS and WLS    --- */
  optn[8]= 3;      /* --- covariance matrices    --- */
  optn[9]= 1;      /* --- Version 7 LTS          --- */

  call lts(sc, coef, wgt, optn, y, x);

  print "sc is " sc, "coef is " coef;

  optn[9]= 0;      /* --- FAST-LTS algorithm    --- */

  call lts(sc, coef, wgt, optn, y, x);

  print "sc is " sc, "coef is " coef;

quit;

```

Comparison of the outputs is summarized as follows:

1. The summary statistics for dependent and independent variables and the results of the classical (unweighted) least-squares estimation (outputs in Page 1 and first half of Page 2) do not change.
2. The “Complete Enumeration for LTS” table on page 1 and the “Resistant Diagnostic” table on page 3 of the first output is eliminated. The analysis based on “Observations of Best Subset” of size  $p$ , where  $p$  is the number of the independent variables, is replaced by the analysis based on observations of the best  $h$  of the entire data set obtained after full iteration.

3. The best ten (at most) estimates before the final search are available by setting a proper option (see pages 2 and 3 of the second output). The first one is the best with the smallest objective value.
4. The results of the weighted least squared estimation (page 4 for the first output, page 5 for the second output) do not change because the two algorithms detect the same outliers in this simple example. Results will be changed if they detect different outliers.
5. The “Coef” vector does not include the best subset (page 6 of the second output).

### Output from Previous Algorithm for LTS

## Compare two algorithms for LTS

1

**LTS:** The sum of the 6 smallest squared residuals will be minimized.

### Median and Mean

|          | Median | Mean |
|----------|--------|------|
| VAR1     | 20     | 26   |
| Intercep | 1      | 1    |
| Response | 62     | 67   |

### Dispersion and Standard Deviation

|          | Dispersion   | StdDev       |
|----------|--------------|--------------|
| VAR1     | 7.4130110925 | 10.22252415  |
| Intercep | 0            | 0            |
| Response | 7.3806276975 | 8.7177978871 |

## Unweighted Least-Squares Estimation

### LS Parameter Estimates

| Variable | Estimate   | Approx<br>Std Err | t Value | Pr >  t | Lower WCI  | Upper WCI  |
|----------|------------|-------------------|---------|---------|------------|------------|
| VAR1     | 0.80502392 | 0.10637482        | 7.57    | 0.0001  | 0.59653312 | 1.01351473 |
| Intercep | 46.069378  | 2.94965086        | 15.62   | <.0001  | 40.2881685 | 51.8505874 |

Sum of Squares = 66.218899522  
Degrees of Freedom = 7  
LS Scale Estimate = 3.0756857429

### Cov Matrix of Parameter Estimates

|          | VAR1         | Intercep     |
|----------|--------------|--------------|
| VAR1     | 0.0113156014 | -0.294205637 |
| Intercep | -0.294205637 | 8.7004402045 |

R-squared = 0.8910873363  
F(1,7) Statistic = 57.271681208  
Probability = 0.0001297174

## Compare two algorithms for LTS

2

## LS Residuals

| N | Observed  | Estimated | Residual  | Res / S   |
|---|-----------|-----------|-----------|-----------|
| 1 | 80.000000 | 79.880383 | 0.119617  | 0.038891  |
| 2 | 80.000000 | 75.855263 | 4.144737  | 1.347581  |
| 3 | 75.000000 | 75.855263 | -0.855263 | -0.278072 |
| 4 | 62.000000 | 68.610048 | -6.610048 | -2.149130 |
| 5 | 62.000000 | 60.559809 | 1.440191  | 0.468251  |
| 6 | 62.000000 | 60.559809 | 1.440191  | 0.468251  |
| 7 | 62.000000 | 61.364833 | 0.635167  | 0.206512  |
| 8 | 62.000000 | 62.169856 | -0.169856 | -0.055226 |
| 9 | 58.000000 | 58.144737 | -0.144737 | -0.047058 |

## Distribution of Residuals

| MinRes       | 1st Qu.      | Median       |
|--------------|--------------|--------------|
| -6.610047847 | -0.512559809 | 0.1196172249 |
| Mean         | 3rd Qu.      | MaxRes       |
| -2.36848E-15 | 1.0376794258 | 4.1447368421 |

(no change)

There are 36 subsets of 2 cases out of 9 cases.

All 36 subsets will be considered.

## Complete Enumeration for LTS

| Subset | Singular | Best<br>Criterion | Percent |
|--------|----------|-------------------|---------|
| 10     | 1        | 0.084493          | 27      |
| 18     | 1        | 0.084493          | 50      |
| 28     | 2        | 0.084493          | 77      |
| 36     | 2        | 0.084493          | 100     |

Minimum Criterion= 0.0844927545

(eliminated)

Least Trimmed Squares (LTS) Method  
 Minimizing Sum of 6 Smallest Squared Residuals.  
 Highest Possible Breakdown Value = 44.44 %  
 Selection of All 36 Subsets of 2 Cases Out of 9  
 Among 36 subsets 2 are singular.

## Observations of Best Subset

1 5

## Estimated Coefficients

| VAR1 | Intercep    |
|------|-------------|
| 0.75 | 47.91666667 |



Compare two algorithms for LTS

3

LTS Objective Function = 0.6236095645

Preliminary LTS Scale = 1.189465671

Robust R Squared = 0.825

Final LTS Scale = 0.8595864639

## LTS Residuals

| N | Observed  | Estimated | Residual  | Res / S   |
|---|-----------|-----------|-----------|-----------|
| 1 | 80.000000 | 79.416667 | 0.583333  | 0.678621  |
| 2 | 80.000000 | 75.666667 | 4.333333  | 5.041184  |
| 3 | 75.000000 | 75.666667 | -0.666667 | -0.775567 |
| 4 | 62.000000 | 68.916667 | -6.916667 | -8.046505 |
| 5 | 62.000000 | 61.416667 | 0.583333  | 0.678621  |
| 6 | 62.000000 | 61.416667 | 0.583333  | 0.678621  |
| 7 | 62.000000 | 62.166667 | -0.166667 | -0.193892 |
| 8 | 62.000000 | 62.916667 | -0.916667 | -1.066404 |
| 9 | 58.000000 | 59.166667 | -1.166667 | -1.357242 |

## Distribution of Residuals

| MinRes       | 1st Qu.      | Median       |
|--------------|--------------|--------------|
| -6.916666667 | -1.041666667 | -0.166666667 |
| Mean         | 3rd Qu.      | MaxRes       |
| -0.416666667 | 0.583333333  | 4.333333333  |

(replaced)

## Resistant Diagnostic

| N | U         | Resistant Diagnostic |
|---|-----------|----------------------|
| 1 | 12.521981 | 5.600000             |
| 2 | 12.521981 | 5.600000             |
| 3 | 9.167879  | 4.100000             |
| 4 | 13.339459 | 5.965588             |
| 5 | 1.709204  | 0.764379             |
| 6 | 1.709204  | 0.764379             |
| 7 | 0.642081  | 0.287147             |
| 8 | 1.697749  | 0.759257             |
| 9 | 2.236068  | 1.000000             |

Median(U)= 2.2360679775

(eliminated)

## Compare two algorithms for LTS

4

## Weighted Least-Squares Estimation

## RLS Parameter Estimates Based on LTS

| Variable | Estimate   | Approx<br>Std Err | t Value | Pr ><br> t | Lower WCI  | Upper WCI  |
|----------|------------|-------------------|---------|------------|------------|------------|
| VAR1     | 0.76455907 | 0.03125286        | 24.46   | <.0001     | 0.70330458 | 0.82581356 |
| Intercep | 47.3985025 | 0.815574          | 58.12   | <.0001     | 45.8000068 | 48.9969982 |

Weighted Sum of Squares = 3.3544093178

Degrees of Freedom = 5

RLS Scale Estimate = 0.819073784

## Cov Matrix of Parameter Estimates

|          | VAR1         | Intercep     |
|----------|--------------|--------------|
| VAR1     | 0.0009767415 | -0.02358133  |
| Intercep | -0.02358133  | 0.6651609492 |

Weighted R-squared = 0.9917145853

F(1,5) Statistic = 598.47009637

Probability = 2.1279132E-6

There are 7 points with nonzero weight.

Average Weight = 0.777777778

## Weighted LS Residuals

| N | Observed  | Estimated | Residual  | Res / S   | Weight   |
|---|-----------|-----------|-----------|-----------|----------|
| 1 | 80.000000 | 79.509983 | 0.490017  | 0.598257  | 1.000000 |
| 2 | 80.000000 | 75.687188 | 4.312812  | 5.265474  | 0        |
| 3 | 75.000000 | 75.687188 | -0.687188 | -0.838982 | 1.000000 |
| 4 | 62.000000 | 68.806156 | -6.806156 | -8.309577 | 0        |
| 5 | 62.000000 | 61.160566 | 0.839434  | 1.024858  | 1.000000 |
| 6 | 62.000000 | 61.160566 | 0.839434  | 1.024858  | 1.000000 |
| 7 | 62.000000 | 61.925125 | 0.074875  | 0.091414  | 1.000000 |
| 8 | 62.000000 | 62.689684 | -0.689684 | -0.842029 | 1.000000 |
| 9 | 58.000000 | 58.866889 | -0.866889 | -1.058377 | 1.000000 |

## Distribution of Residuals

| MinRes       | 1st Qu.      | Median      |
|--------------|--------------|-------------|
| -6.806156406 | -0.77828619  | 0.074875208 |
| Mean         | 3rd Qu.      | MaxRes      |
| -0.27703827  | 0.6647254576 | 4.31281198  |

The run has been executed successfully.

(no change)

```

Compare two algorithms for LTS                                     5

sc is      6
           36
           2
           7
           0.6236096
           1.1894657
           0.8595865
           0.825
           1.9073884
           .
           0.8190738

           3.3544093
           0.9917146
           598.4701
           .
           .
           .
           .
           .
           .

coef is      0.75 47.916667
            1      5 ----->eliminated
           0.7645591 47.398502
           0.0312529 0.815574
           24.463648 58.11674
           2.1279E-6 2.8538E-8
           0.7033046 45.800007
           0.8258136 48.996998

```

## Output from FAST-LTS Algorithm

Compare two algorithms for LTS

1

LTS: The sum of the 6 smallest squared residuals will be minimized.

## Median and Mean

|          | Median | Mean |
|----------|--------|------|
| VAR1     | 20     | 26   |
| Intercep | 1      | 1    |
| Response | 62     | 67   |

## Dispersion and Standard Deviation

|          | Dispersion   | StdDev       |
|----------|--------------|--------------|
| VAR1     | 7.4130110925 | 10.22252415  |
| Intercep | 0            | 0            |
| Response | 7.3806276975 | 8.7177978871 |

## Unweighted Least-Squares Estimation

## LS Parameter Estimates

| Variable | Estimate   | Approx<br>Std Err | t Value | Pr ><br> t | Lower WCI  | Upper WCI  |
|----------|------------|-------------------|---------|------------|------------|------------|
| VAR1     | 0.80502392 | 0.10637482        | 7.57    | 0.0001     | 0.59653312 | 1.01351473 |
| Intercep | 46.069378  | 2.94965086        | 15.62   | <.0001     | 40.2881685 | 51.8505874 |

Sum of Squares = 66.218899522

Degrees of Freedom = 7

LS Scale Estimate = 3.0756857429

## Cov Matrix of Parameter Estimates

|          | VAR1         | Intercep     |
|----------|--------------|--------------|
| VAR1     | 0.0113156014 | -0.294205637 |
| Intercep | -0.294205637 | 8.7004402045 |

R-squared = 0.8910873363

F(1,7) Statistic = 57.271681208

Probability = 0.0001297174

## Compare two algorithms for LTS

2

## LS Residuals

| N | Observed  | Estimated | Residual  | Res / S   |
|---|-----------|-----------|-----------|-----------|
| 1 | 80.000000 | 79.880383 | 0.119617  | 0.038891  |
| 2 | 80.000000 | 75.855263 | 4.144737  | 1.347581  |
| 3 | 75.000000 | 75.855263 | -0.855263 | -0.278072 |
| 4 | 62.000000 | 68.610048 | -6.610048 | -2.149130 |
| 5 | 62.000000 | 60.559809 | 1.440191  | 0.468251  |
| 6 | 62.000000 | 60.559809 | 1.440191  | 0.468251  |
| 7 | 62.000000 | 61.364833 | 0.635167  | 0.206512  |
| 8 | 62.000000 | 62.169856 | -0.169856 | -0.055226 |
| 9 | 58.000000 | 58.144737 | -0.144737 | -0.047058 |

## Distribution of Residuals

| MinRes       | 1st Qu.      | Median       |
|--------------|--------------|--------------|
| -6.610047847 | -0.512559809 | 0.1196172249 |
| Mean         | 3rd Qu.      | MaxRes       |
| -2.36848E-15 | 1.0376794258 | 4.1447368421 |

(no change)

## Least Trimmed Squares (LTS) Method

The (at most) 10 Best Estimates

Objective Value [1]: 0.0428203092

## Estimated Coefficients

| VAR1         | Intercep     |
|--------------|--------------|
| 0.7521545304 | 0.1250675364 |

Objective Value [2]: 0.0454534796

## Estimated Coefficients

| VAR1         | Intercep     |
|--------------|--------------|
| 0.7773132092 | 0.0679381212 |

Objective Value [3]: 0.0458503276

Compare two algorithms for LTS

3

## Estimated Coefficients

| VAR1         | Intercep     |
|--------------|--------------|
| 0.7857569344 | 0.0875053435 |

Objective Value [4]: 0.0470161862

## Estimated Coefficients

| VAR1         | Intercep     |
|--------------|--------------|
| 0.7454450208 | 0.1033087106 |

Objective Value [5]: 0.0504563846

## Estimated Coefficients

| VAR1        | Intercep     |
|-------------|--------------|
| 0.987904817 | 0.1606655488 |

Objective Value [6]: 0.1790484378

## Estimated Coefficients

| VAR1         | Intercep     |
|--------------|--------------|
| 0.1926222834 | -0.081665104 |

Objective Value [7]: 1.797693E308

(added)

Least Trimmed Squares (LTS) Method  
 Minimizing Sum of 6 Smallest Squared Residuals.  
 Highest Possible Breakdown Value = 44.44 %  
 Selection of All 36 Subsets of 2 Cases Out of 9  
 Among 36 subsets 2 are singular.

## Compare two algorithms for LTS

4

The best half of the entire data set obtained after full iteration consists of the cases:

1            3            5            6            7            8

## Estimated Coefficients

VAR1            Intercep

0.7488687783      47.945701357

LTS Objective Function = 0.6235087791

Preliminary LTS Scale = 1.1892734341

Robust R Squared = 0.819730444

Final LTS Scale = 0.8627851118

## LTS Residuals

| N | Observed  | Estimated | Residual  | Res / S   |
|---|-----------|-----------|-----------|-----------|
| 1 | 80.000000 | 79.398190 | 0.601810  | 0.697520  |
| 2 | 80.000000 | 75.653846 | 4.346154  | 5.037354  |
| 3 | 75.000000 | 75.653846 | -0.653846 | -0.757832 |
| 4 | 62.000000 | 68.914027 | -6.914027 | -8.013614 |
| 5 | 62.000000 | 61.425339 | 0.574661  | 0.666053  |
| 6 | 62.000000 | 61.425339 | 0.574661  | 0.666053  |
| 7 | 62.000000 | 62.174208 | -0.174208 | -0.201914 |
| 8 | 62.000000 | 62.923077 | -0.923077 | -1.069880 |
| 9 | 58.000000 | 59.178733 | -1.178733 | -1.366195 |

## Distribution of Residuals

| MinRes       | 1st Qu.      | Median       |
|--------------|--------------|--------------|
| -6.914027149 | -1.050904977 | -0.174208145 |
| Mean         | 3rd Qu.      | MaxRes       |
| -0.416289593 | 0.5746606335 | 4.3461538462 |

(changed)

## Compare two algorithms for LTS

5

## Weighted Least-Squares Estimation

## RLS Parameter Estimates Based on LTS

| Variable | Estimate   | Approx<br>Std Err | t Value | Pr ><br> t | Lower WCI  | Upper WCI  |
|----------|------------|-------------------|---------|------------|------------|------------|
| VAR1     | 0.76455907 | 0.03125286        | 24.46   | <.0001     | 0.70330458 | 0.82581356 |
| Intercep | 47.3985025 | 0.815574          | 58.12   | <.0001     | 45.8000068 | 48.9969982 |

Weighted Sum of Squares = 3.3544093178

Degrees of Freedom = 5

RLS Scale Estimate = 0.819073784

## Cov Matrix of Parameter Estimates

|          | VAR1         | Intercep     |
|----------|--------------|--------------|
| VAR1     | 0.0009767415 | -0.02358133  |
| Intercep | -0.02358133  | 0.6651609492 |

Weighted R-squared = 0.9917145853

F(1,5) Statistic = 598.47009637

Probability = 2.1279132E-6

There are 7 points with nonzero weight.

Average Weight = 0.7777777778

## Weighted LS Residuals

| N | Observed  | Estimated | Residual  | Res / S   | Weight   |
|---|-----------|-----------|-----------|-----------|----------|
| 1 | 80.000000 | 79.509983 | 0.490017  | 0.598257  | 1.000000 |
| 2 | 80.000000 | 75.687188 | 4.312812  | 5.265474  | 0        |
| 3 | 75.000000 | 75.687188 | -0.687188 | -0.838982 | 1.000000 |
| 4 | 62.000000 | 68.806156 | -6.806156 | -8.309577 | 0        |
| 5 | 62.000000 | 61.160566 | 0.839434  | 1.024858  | 1.000000 |
| 6 | 62.000000 | 61.160566 | 0.839434  | 1.024858  | 1.000000 |
| 7 | 62.000000 | 61.925125 | 0.074875  | 0.091414  | 1.000000 |
| 8 | 62.000000 | 62.689684 | -0.689684 | -0.842029 | 1.000000 |
| 9 | 58.000000 | 58.866889 | -0.866889 | -1.058377 | 1.000000 |

## Distribution of Residuals

| MinRes       | 1st Qu.      | Median      |
|--------------|--------------|-------------|
| -6.806156406 | -0.77828619  | 0.074875208 |
| Mean         | 3rd Qu.      | MaxRes      |
| -0.27703827  | 0.6647254576 | 4.31281198  |

The run has been executed successfully.

(no change)



```

Compare two algorithms for LTS
6

sc is      6
           36
           2
           7
0.6235088
1.1892734
0.8627851
0.8197304
1.879
.
0.8190738
3.3544093
0.9917146
598.4701
.
.
.
.
.
.

coef is 0.7488688 47.945701
0.7645591 47.398502
0.0312529 0.815574
24.463648 58.11674
2.1279E-6 2.8538E-8
0.7033046 45.800007
0.8258136 48.996998
.
.

(changed)

```

## MCD Call

**finds the minimum covariance determinant estimator**

**CALL MCD(*sc*, *coef*, *dist*, *opt*, *x*);**

The MCD call is the robust (resistant) estimation of multivariate location and scatter, defined by minimizing the determinant of the covariance matrix computed from  $h$  points. The algorithm for the MCD subroutine is based on the FAST-MCD algorithm given by Rousseeuw and Van Driessen (1999).

The MCD subroutine computes the minimum covariance determinant estimator. These robust locations and covariance matrices can be used to detect multivariate outliers and leverage points. For this purpose, the MCD subroutine provides a table of robust distances.

In the following discussion,  $N$  is the number of observations and  $n$  is the number of regressors. The inputs to the MCD subroutine are as follows:

*opt*      refers to an options vector with the following components (missing values

are treated as default values):

*opt*[1] specifies the amount of printed output. Higher option values request additional output and include the output of lower values.

*opt*[1]=0 prints no output except error messages.

*opt*[1]=1 prints most of the output.

*opt*[1]=2 additionally prints case numbers of the observations in the best subset and some basic history of the optimization process.

*opt*[1]=3 additionally prints how many subsets result in singular linear systems.

The default is *opt*[1]=0.

*opt*[2] specifies whether the classical, initial, and final robust covariance matrices are printed. The default is *opt*[2]=0. Note that the final robust covariance matrix is always returned in *coef*.

*opt*[3] specifies whether the classical, initial, and final robust correlation matrices are printed or returned:

*opt*[3]=0 does not return or print.

*opt*[3]=1 prints the robust correlation matrix.

*opt*[3]=2 returns the final robust correlation matrix in *coef*.

*opt*[3]=3 prints and returns the final robust correlation matrix.

*opt*[4] specifies the quantile  $h$  used in the objective function. The default is *opt*[5]=  $h = \lceil \frac{N+n+1}{2} \rceil$ . If the value of  $h$  is specified outside the range  $\frac{N}{2} + 1 \leq h \leq \frac{3N}{4} + \frac{n+1}{4}$ , it is reset to the closest boundary of this region.

*opt*[5] specifies the number  $N_{\text{Rep}}$  of subset generations. This option is the same as described for the LTS subroutines. Due to computer time restrictions, not all subset combinations can be inspected for larger values of  $N$  and  $n$ .

When *opt*[5] is zero or missing:

If  $N > 600$ , construct up to five disjoint random subsets with sizes as equal as possible, but not to exceed 300. Inside each subset, choose  $500/5 = 100$  subset combinations of  $n$  observations.

If  $N < 600$ , the number of subsets is taken from the following table.

| n                  | 1   | 2  | 3  | 4  | 5  | 6  | 7 | 8 | 9 | 10 |
|--------------------|-----|----|----|----|----|----|---|---|---|----|
| $N_{\text{lower}}$ | 500 | 50 | 22 | 17 | 15 | 14 | 0 | 0 | 0 | 0  |

| n                  | 11 | 12 | 13 | 14 | 15 |
|--------------------|----|----|----|----|----|
| $N_{\text{lower}}$ | 0  | 0  | 0  | 0  | 0  |

If the number of cases (observations)  $N$  is smaller than  $N_{\text{lower}}$ , then all possible subsets are used; otherwise, 500 subsets are cho-

sen randomly. This means that an exhaustive search is performed for  $opt[5]=-1$ . If  $N$  is larger than  $N_{upper}$ , a note is printed in the log file indicating how many subsets exist.

$x$  refers to an  $N \times n$  matrix  $\mathbf{X}$  of regressors.

Missing values are not permitted in  $x$ . Missing values in  $opt$  cause the default value to be used.

The MCD subroutine returns the following values:

$sc$  is a column vector containing the following scalar information:

|         |                                                                                                           |
|---------|-----------------------------------------------------------------------------------------------------------|
| $sc[1]$ | the quantile $h$ used in the objective function                                                           |
| $sc[2]$ | number of subsets generated                                                                               |
| $sc[3]$ | number of subsets with singular linear systems                                                            |
| $sc[4]$ | number of nonzero weights $w_i$                                                                           |
| $sc[5]$ | lowest value of the objective function $F_{MCD}$ attained (smallest determinant)                          |
| $sc[6]$ | Mahalanobis-like distance used in the computation of the lowest value of the objective function $F_{MCD}$ |
| $sc[7]$ | the cutoff value used for the outlier decision                                                            |

$coef$  is a matrix with  $n$  columns containing the following results in its rows:

|                    |                                                                  |
|--------------------|------------------------------------------------------------------|
| $coef[1]$          | location of ellipsoid center                                     |
| $coef[2]$          | eigenvalues of final robust scatter matrix                       |
| $coef[3:2+n]$      | the final robust scatter matrix for $opt[2]=1$ or $opt[2]=3$     |
| $coef[2+n+1:2+2n]$ | the final robust correlation matrix for $opt[3]=1$ or $opt[3]=3$ |

$dist$  is a matrix with  $N$  columns containing the following results in its rows:

|           |                                                       |
|-----------|-------------------------------------------------------|
| $dist[1]$ | Mahalanobis distances                                 |
| $dist[2]$ | robust distances based on the final estimates         |
| $dist[3]$ | weights (=1 for small, =0 for large robust distances) |

### Example

Consider Brownlee's (1965) stackloss data used in the example for the MVE subroutine.

For  $N = 21$  and  $n = 4$  (three explanatory variables including intercept), you obtain a total of 5,985 different subsets of 4 observations out of 21. If you decide not to specify  $optn[5]$ , the MCD algorithm chooses 500 random sample subsets:

```

/* X1 X2 X3 Y Stackloss data */
aa = { 1 80 27 89 42,
        1 80 27 88 37,
        1 75 25 90 37,
        1 62 24 87 28,
        1 62 22 87 18,
        1 62 23 87 18,
        1 62 24 93 19,
        1 62 24 93 20,
        1 58 23 87 15,
        1 58 18 80 14,
        1 58 18 89 14,
        1 58 17 88 13,
        1 58 18 82 11,
        1 58 19 93 12,
        1 50 18 89 8,
        1 50 18 86 7,
        1 50 19 72 8,
        1 50 19 79 8,
        1 50 20 80 9,
        1 56 20 82 15,
        1 70 20 91 15 };

a = aa[,2:4];
optn = j(8,1,.);
optn[1]= 2; /* ipri */
optn[2]= 1; /* pcov: print COV */
optn[3]= 1; /* pcor: print CORR */

CALL MCD(sc,xmcd,dist,optn,a);

```

The first part of the output of this program is a summary of the MCD algorithm and the final  $h$  points selected:

#### Fast MCD by Rousseeuw and Van Driessen

|                                      |       |
|--------------------------------------|-------|
| Number of Variables                  | 3     |
| Number of Observations               | 21    |
| Default Value for h                  | 12    |
| Specified Value for h                | 12    |
| Breakdown Value                      | 42.86 |
| - Highest Possible Breakdown Value - |       |

The best half of the entire data set obtained after full iteration consists of the cases:

4      5      6      7      8      9      10      11      12      13      14      20

The second part of the output is the MCD estimators of the location, scatter matrix, and correlation matrix:

**MCD Location Estimate**

| VAR1                          | VAR2         | VAR3         |
|-------------------------------|--------------|--------------|
| 59.5                          | 20.833333333 | 87.333333333 |
| Average of 12 Selected Points |              |              |

**MCD Scatter Matrix Estimate**

|                                         | VAR1         | VAR2         | VAR3         |
|-----------------------------------------|--------------|--------------|--------------|
| VAR1                                    | 5.1818181818 | 4.8181818182 | 4.7272727273 |
| VAR2                                    | 4.8181818182 | 7.6060606061 | 5.0606060606 |
| VAR3                                    | 4.7272727273 | 5.0606060606 | 19.151515152 |
| Determinant = 238.07387929              |              |              |              |
| Covariance Matrix of 12 Selected Points |              |              |              |

**MCD Correlation Matrix**

|      | VAR1         | VAR2         | VAR3         |
|------|--------------|--------------|--------------|
| VAR1 | 1            | 0.7674714142 | 0.4745347313 |
| VAR2 | 0.7674714142 | 1            | 0.4192963398 |
| VAR3 | 0.4745347313 | 0.4192963398 | 1            |

The MCD scatter matrix is multiplied by a factor to make it consistent when all the data come from a single Gaussian distribution.

**Consistent Scatter Matrix**

|                            | VAR1         | VAR2         | VAR3         |
|----------------------------|--------------|--------------|--------------|
| VAR1                       | 8.6578437815 | 8.0502757968 | 7.8983838007 |
| VAR2                       | 8.0502757968 | 12.708297013 | 8.4553211199 |
| VAR3                       | 7.8983838007 | 8.4553211199 | 31.998580526 |
| Determinant = 397.77668436 |              |              |              |

The final output presents a table containing the classical Mahalanobis distances, the robust distances, and the weights identifying the outlying observations (that is, leverage points when explaining  $y$  with these three regressor variables):

| Classical Distances and Robust (Rousseeuw) Distances |                       |                  |          |
|------------------------------------------------------|-----------------------|------------------|----------|
| Unsquared Mahalanobis Distance and                   |                       |                  |          |
| Unsquared Rousseeuw Distance of Each Observation     |                       |                  |          |
| N                                                    | Mahalanobis Distances | Robust Distances | Weight   |
| 1                                                    | 2.253603              | 12.173282        | 0        |
| 2                                                    | 2.324745              | 12.255677        | 0        |
| 3                                                    | 1.593712              | 9.263990         | 0        |
| 4                                                    | 1.271898              | 1.401368         | 1.000000 |

|    |          |          |          |
|----|----------|----------|----------|
| 5  | 0.303357 | 1.420020 | 1.000000 |
| 6  | 0.772895 | 1.291188 | 1.000000 |
| 7  | 1.852661 | 1.460370 | 1.000000 |
| 8  | 1.852661 | 1.460370 | 1.000000 |
| 9  | 1.360622 | 2.120590 | 1.000000 |
| 10 | 1.745997 | 1.809708 | 1.000000 |
| 11 | 1.465702 | 1.362278 | 1.000000 |
| 12 | 1.841504 | 1.667437 | 1.000000 |
| 13 | 1.482649 | 1.416724 | 1.000000 |
| 14 | 1.778785 | 1.988240 | 1.000000 |
| 15 | 1.690241 | 5.874858 | 0        |
| 16 | 1.291934 | 5.606157 | 0        |
| 17 | 2.700016 | 6.133319 | 0        |
| 18 | 1.503155 | 5.760432 | 0        |
| 19 | 1.593221 | 6.156248 | 0        |
| 20 | 0.807054 | 2.172300 | 1.000000 |
| 21 | 2.176761 | 7.622769 | 0        |

Robust distances are based on reweighted estimates.

The cutoff value is the square root of the 0.975 quantile of the chi square distribution with 3 degrees of freedom.

Points whose robust distance exceeds 3.0575159206 have received a zero weight in the last column above.

There were 9 such points in the data.

These may include boundary cases.

Only points whose robust distance is substantially larger than the cutoff should be considered outliers.

---

## Multivariate Time Series Analysis

These are the new functions for the creation and analysis of multivariate time series:

---

### VARMACOV Call

computes the theoretical cross-covariance matrices for a stationary VARMA( $p, q$ ) model

**CALL VARMACOV(*cov, phi, theta, sigma* <, *p, q, lag*>);**

The inputs to the VARMACOV subroutine are as follows:

*phi* specifies a  $km_p \times k$  matrix,  $\Phi$ , containing the autoregressive coefficient matrices, where  $m_p$  is the number of elements in the subset of the AR order and  $k \geq 2$  is the number of variables. All the roots of  $|\Phi(B)| = 0$  should be greater than one in absolute value, where  $\Phi(B)$  is the finite order matrix polynomial in the backshift operator  $B$ , such that  $B^j \mathbf{y}_t = \mathbf{y}_{t-j}$ . You must

specify either *phi* or *theta*.

- theta* specifies a  $km_q \times k$  matrix containing the moving-average coefficient matrices, where  $m_q$  is the number of the elements in the subset of the MA order. You must specify either *phi* or *theta*.
- sigma* specifies a  $k \times k$  symmetric positive-definite covariance matrix of the innovation series. If *sigma* is not specified, then an identity matrix is used.
- p* specifies the subset of the AR order. The quantity  $m_p$  is defined as

$$m_p = \text{nrow}(\text{phi}) / \text{ncol}(\text{phi})$$

where  $\text{nrow}(\text{phi})$  is the number of rows of the matrix *phi* and  $\text{ncol}(\text{phi})$  is the number of columns of the matrix *phi*.

If you do not specify *p*, the default subset is  $p = \{1, 2, \dots, m_p\}$ .

For example, consider a four-dimensional vector time series, and *phi* is a  $4 \times 4$  matrix. If you specify  $p=1$  (the default, since  $m_p = 4/4 = 1$ ), the VARMA COV subroutine computes the theoretical cross-covariance matrices of VAR(1) as  $\mathbf{y}_t = \Phi \mathbf{y}_{t-1} + \epsilon_t$ .

If you specify  $p=2$ , the VARMA COV subroutine computes the cross-covariance matrices of VAR(2) as  $\mathbf{y}_t = \Phi \mathbf{y}_{t-2} + \epsilon_t$ .

Let  $\text{phi} = [\Phi'_1, \Phi'_2]'$  be an  $8 \times 4$  matrix. If you specify  $p = \{1, 3\}$ , the VARMA COV subroutine computes the cross-covariance matrices of VAR(3) as  $\mathbf{y}_t = \Phi_1 \mathbf{y}_{t-1} + \Phi_2 \mathbf{y}_{t-3} + \epsilon_t$ . If you do not specify *p*, the VARMA COV subroutine computes the cross-covariance matrices of VAR(2) as  $\mathbf{y}_t = \Phi_1 \mathbf{y}_{t-1} + \Phi_2 \mathbf{y}_{t-2} + \epsilon_t$ .

- q* specifies the subset of the MA order. The quantity  $m_q$  is defined as

$$m_q = \text{nrow}(\text{theta}) / \text{ncol}(\text{theta})$$

where  $\text{nrow}(\text{theta})$  is the number of rows of matrix *theta* and  $\text{ncol}(\text{theta})$  is the number of columns of matrix *theta*.

If you do not specify *q*, the default subset is  $q = \{1, 2, \dots, m_q\}$ .

The usage of *q* is the same as that of *p*.

- lag* specifies the length of lags, which must be a positive number. If  $\text{lag} = h$ , the VARMA COV computes the cross-covariance matrices from lag zero to lag *h*. By default,  $\text{lag} = 12$ .

The VARMA COV subroutine returns the following value:

- cov* is a  $k(\text{lag} + 1) \times k$  matrix that contains the theoretical cross-covariance matrices of the VARMA(*p*, *q*) model.

To compute the cross-covariance matrices of a bivariate ( $k = 2$ ) VARMA(1,1) model

$$\mathbf{y}_t = \Phi \mathbf{y}_{t-1} + \epsilon_t - \Theta \epsilon_{t-1}$$

where

$$\Phi = \begin{bmatrix} 1.2 & -0.5 \\ 0.6 & 0.3 \end{bmatrix} \quad \Theta = \begin{bmatrix} -0.6 & 0.3 \\ 0.3 & 0.6 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.25 \end{bmatrix}$$

you can specify

```
phi = { 1.2 -0.5, 0.6 0.3 };
theta= {-0.6 0.3, 0.3 0.6 };
sigma= { 1.0 0.5, 0.5 1.25};
call varmacov(cov, phi, theta, sigma) lag=5;
```

---

## VARMALIK Call

computes the log-likelihood function for a VARMA( $p, q$ ) model

**CALL VARMALIK(*lnl*, *series*, *phi*, *theta*, *sigma* <, *p*, *q*, *opt*> );**

The inputs to the VARMALIK subroutine are as follows:

- |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>series</i> | specifies an $n \times k$ matrix containing the vector time series (assuming mean zero), where $n$ is the number of observations and $k \geq 2$ is the number of variables.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
| <i>phi</i>    | specifies a $km_p \times k$ matrix containing the autoregressive coefficient matrices, where $m_p$ is the number of the elements in the subset of the AR order. You must specify either <i>phi</i> or <i>theta</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
| <i>theta</i>  | specifies a $km_q \times k$ matrix containing the moving-average coefficient matrices, where $m_q$ is the number of the elements in the subset of the MA order. You must specify either <i>phi</i> or <i>theta</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
| <i>sigma</i>  | specifies a $k \times k$ covariance matrix of the innovation series. If you do not specify <i>sigma</i> , an identity matrix is used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
| <i>p</i>      | specifies the subset of the AR order. See the VARMACOV subroutine.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
| <i>q</i>      | specifies the subset of the MA order. See the VARMACOV subroutine.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
| <i>opt</i>    | specifies the method of computing the log-likelihood function: <table border="0" style="margin-left: 20px;"> <tr> <td style="vertical-align: top; padding-right: 10px;"><i>opt</i>=0</td> <td>requests the multivariate innovations algorithm. This algorithm requires that the time series is stationary and does not contain missing observations.</td> </tr> <tr> <td style="vertical-align: top; padding-right: 10px;"><i>opt</i>=1</td> <td>requests the conditional log-likelihood function. This algorithm requires that the number of the observations in the time series must be greater than <math>p+q</math> and that the series does not contain missing observations.</td> </tr> <tr> <td style="vertical-align: top; padding-right: 10px;"><i>opt</i>=2</td> <td>requests the Kalman filtering algorithm. This is the default and is used if the required conditions in <i>opt</i>=0 and <i>opt</i>=1 are not satisfied.</td> </tr> </table> | <i>opt</i> =0 | requests the multivariate innovations algorithm. This algorithm requires that the time series is stationary and does not contain missing observations. | <i>opt</i> =1 | requests the conditional log-likelihood function. This algorithm requires that the number of the observations in the time series must be greater than $p+q$ and that the series does not contain missing observations. | <i>opt</i> =2 | requests the Kalman filtering algorithm. This is the default and is used if the required conditions in <i>opt</i> =0 and <i>opt</i> =1 are not satisfied. |
| <i>opt</i> =0 | requests the multivariate innovations algorithm. This algorithm requires that the time series is stationary and does not contain missing observations.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
| <i>opt</i> =1 | requests the conditional log-likelihood function. This algorithm requires that the number of the observations in the time series must be greater than $p+q$ and that the series does not contain missing observations.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |
| <i>opt</i> =2 | requests the Kalman filtering algorithm. This is the default and is used if the required conditions in <i>opt</i> =0 and <i>opt</i> =1 are not satisfied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |                                                                                                                                                        |               |                                                                                                                                                                                                                        |               |                                                                                                                                                           |



The VARMA1IK subroutine returns the following value:

*lnl* is a  $3 \times 1$  matrix containing the log-likelihood function, the sum of log determinant of the innovation variance, and the weighted sum of squares of residuals. The log-likelihood function is computed as  $-0.5 \times$  (the sum of last two terms).

The options  $opt=0$  and  $opt=2$  are equivalent for stationary time series without missing values. Setting  $opt=0$  is useful for a small number of the observations and a high order of  $p$  and  $q$ ;  $opt=1$  is useful for a high order of  $p$  and  $q$ ;  $opt=2$  is useful for a low order of  $p$  and  $q$ , or for missing values in the observations.

To compute the log-likelihood function of a bivariate ( $k = 2$ ) VARMA(1,1) model

$$\mathbf{y}_t = \Phi \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t - \Theta \boldsymbol{\epsilon}_{t-1}$$

where

$$\Phi = \begin{bmatrix} 1.2 & -0.5 \\ 0.6 & 0.3 \end{bmatrix} \quad \Theta = \begin{bmatrix} -0.6 & 0.3 \\ 0.3 & 0.6 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.25 \end{bmatrix}$$

you can specify

```
phi = { 1.2 -0.5, 0.6 0.3 };
theta= {-0.6 0.3, 0.3 0.6 };
sigma= { 1.0 0.5, 0.5 1.25};
call varmasim(yt, phi, theta) sigma=sigma;
call varmalik(lnl, yt, phi, theta, sigma);
```

---

## VARMASIM Call

generates a VARMA( $p,q$ ) time series

**CALL VARMA SIM**(*series, phi, theta, mu, sigma, n <, p, q, initial, seed>*);

The inputs to the VARMA SIM subroutine are as follows:

- phi* specifies a  $km_p \times k$  matrix containing the autoregressive coefficient matrices, where  $m_p$  is the number of the elements in the subset of the AR order and  $k \geq 2$  is the number of variables. You must specify either *phi* or *theta*.
- theta* specifies a  $km_q \times k$  matrix containing the moving-average coefficient matrices, where  $m_q$  is the number of the elements in the subset of the MA order. You must specify either *phi* or *theta*.
- mu* specifies a  $k \times 1$  (or  $1 \times k$ ) mean vector of the series. If *mu* is not specified, a zero vector is used.
- sigma* specifies a  $k \times k$  covariance matrix of the innovation series. If *sigma* is not specified, an identity matrix is used.

- n* specifies the length of the series. If *n* is not specified,  $n = 100$  is used.
- p* specifies the subset of the AR order. See the VARMA COV subroutine.
- q* specifies the subset of the MA order. See the VARMA COV subroutine.
- initial* specifies the initial values of random variables. If *initial* =  $a_0$ , then  $\mathbf{y}_{-p+1}, \dots, \mathbf{y}_0$  and  $\boldsymbol{\epsilon}_{-q+1}, \dots, \boldsymbol{\epsilon}_0$  all take the same value  $a_0$ . If the *initial* option is not specified, the initial values are estimated for the stationary vector time series; the initial values are assumed as zero for the nonstationary vector time series.
- seed* specifies the random number seed. See the VNORMAL subroutine.

The VARMA SIM subroutine returns the following value:

- series* is an  $n \times k$  matrix containing the generated VARMA( $p, q$ ) time series. When either the *initial* option is specified or zero initial values are used, these initial values are not included in *series*.

To generate a bivariate ( $k = 2$ ) stationary VARMA(1,1) time series

$$\mathbf{y}_t - \boldsymbol{\mu} = \Phi(\mathbf{y}_{t-1} - \boldsymbol{\mu}) + \boldsymbol{\epsilon}_t - \Theta\boldsymbol{\epsilon}_{t-1}$$

where

$$\Phi = \begin{bmatrix} 1.2 & -0.5 \\ 0.6 & 0.3 \end{bmatrix} \quad \Theta = \begin{bmatrix} -0.6 & 0.3 \\ 0.3 & 0.6 \end{bmatrix} \quad \boldsymbol{\mu} = \begin{bmatrix} 10 \\ 20 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.25 \end{bmatrix}$$

you can specify

```
phi = { 1.2 -0.5, 0.6 0.3 };
theta= {-0.6 0.3, 0.3 0.6 };
mu = { 10, 20 };
sigma= { 1.0 0.5, 0.5 1.25};
call varmasim(yt, phi, theta, mu, sigma, 100);
```

To generate a bivariate ( $k = 2$ ) nonstationary VARMA(1,1) time series with the same  $\boldsymbol{\mu}$ ,  $\Sigma$ , and  $\Theta$  in the previous example and the AR coefficient

$$\Phi = \begin{bmatrix} 1.0 & 0 \\ 0 & 0.3 \end{bmatrix}$$

you can specify

```
phi = { 1.0 0.0, 0.0 0.3 };
call varmasim(yt, phi, theta, mu, sigma, 100) initial=3;
```

---

## VNORMAL Call

**generates a multivariate normal random series**

**CALL VNORMAL(*series*, *mu*, *sigma*, *n* <, *seed*>);**

The inputs to the VNORMAL subroutine are as follows:

- mu* specifies a  $k \times 1$  (or  $1 \times k$ ) mean vector, where  $k \geq 2$  is the number of variables. You must specify either *mu* or *sigma*. If *mu* is not specified, a zero vector is used.
- sigma* specifies a  $k \times k$  symmetric positive-definite covariance matrix. By default, *sigma* is an identity matrix with dimension  $k$ . You must specify either *mu* or *sigma*. If *sigma* is not specified, an identity matrix is used.
- n* specifies the length of the series. If *n* is not specified,  $n = 100$  is used.
- seed* specifies the random number seed. If it is not supplied, the system clock is used to generate the seed. If it is negative, then the absolute value is used as the starting seed; otherwise, subsequent calls ignore the value of *seed* and use the last seed generated internally.

The VNORMAL subroutine returns the following value:

*series* is an  $n \times k$  matrix that contains the generated normal random series.

To generate a bivariate ( $k = 2$ ) normal random series with mean  $\mu$  and covariance matrix  $\Sigma$ , where

$$\mu = \begin{bmatrix} 10 \\ 20 \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.25 \end{bmatrix}$$

you can specify

```
mu    = { 10, 20 };
sigma= { 1.0  0.5, 0.5 1.25};
call vnormal(et, mu, sigma, 100);
```

---

## VTSROOT Call

**calculates the characteristic roots of the model from AR and MA characteristic functions**

**CALL VTSROOT(*root*, *phi*, *theta*<, *p*, *q*>);**

The inputs to the VTSROOT subroutine are as follows:

- phi* specifies a  $km_p \times k$  matrix containing the autoregressive coefficient matrices, where  $m_p$  is the number of the elements in the subset of the AR order and  $k \geq 2$  is the number of variables. You must specify either *phi* or *theta*.
- theta* specifies a  $km_q \times k$  matrix containing the moving-average coefficient matrices, where  $m_q$  is the number of the elements in the subset of the MA order. You must specify either *phi* or *theta*.
- p* specifies the subset of the AR order. See the VARMA COV subroutine.
- q* specifies the subset of the MA order. See the VARMA COV subroutine.

The VTSROOT subroutine returns the following value:

*root* is a  $k(p_{max} + q_{max}) \times 5$  matrix, where  $p_{max}$  is the maximum order of the AR characteristic function and  $q_{max}$  is the maximum order of the MA characteristic function. The first  $kp_{max}$  rows refer to the results of the AR characteristic function; the last  $q_{max}$  rows refer to the results of the MA characteristic function.

The first column contains the real parts,  $x$ , of eigenvalues of companion matrix associated with the AR( $p_{max}$ ) or MA( $q_{max}$ ) characteristic function; the second column contains the imaginary parts,  $y$ , of the eigenvalues; the third column contains the moduli of the eigenvalues,  $\sqrt{x^2 + y^2}$ ; the fourth column contains the arguments ( $\arctan(y/x)$ ) of the eigenvalues, measured in radians from the positive real axis. The fifth column contains the arguments expressed in degrees rather than radians.

To compute the roots of the characteristic functions,  $\Phi(B) = I - \Phi B$  and  $\Theta(B) = I - \Theta B$ , where  $I$  is an identity matrix with dimension 2 and

$$\Phi = \begin{bmatrix} 1.2 & -0.5 \\ 0.6 & 0.3 \end{bmatrix} \quad \Theta = \begin{bmatrix} -0.6 & 0.3 \\ 0.3 & 0.6 \end{bmatrix}$$

you can specify

```
phi = { 1.2 -0.5, 0.6 0.3 };
theta= {-0.6 0.3, 0.3 0.6 };
call vtsroot(root, phi, theta);
```

---

## References

- Brownlee, K.A. (1965), *Statistical Theory and Methodology in Science and Engineering*, New York: John Wiley & Sons, Inc.
- Rousseeuw, P.J. (1984), “Least Median of Squares Regression,” *Journal of the American Statistical Association*, 79, 871–880.
- Rousseeuw, P.J. and Hubert, M. (1997), “Recent Developments in PROGRESS,”  *$L_1$ -Statistical Procedures and Related Topics*, ed. by Y. Dodge, IMS Lecture Notes, Monograph Series, No. 31, 201–214.
- Rousseeuw, P.J. and Leroy, A.M. (1987), *Robust Regression and Outlier Detection*, New York: John Wiley & Sons, Inc.
- Rousseeuw, P.J. and Van Driessen, K. (1998), “Computing LTS Regression for Large Data Sets,” Technical Report, University of Antwerp, submitted.
- Rousseeuw, P.J. and Van Driessen, K. (1999), “A Fast Algorithm for the Minimum Covariance Determinant Estimator,” *Technometrics*, 41, 212–223.

# Subject Index

## C

CONVEXIT function  
calculating convexity of noncontingent cash flows,  
3

## D

DURATION function  
calculating modified duration of noncontingent  
cash flows, 4

## F

forward rates, 5

## L

LTS call  
performs robust regression, 10

## M

MCD call, 24

## P

PV function  
calculating present value, 6

## R

RATES function  
converting interest rates, 7

## S

SPOT function  
calculating spot rates, 8

## V

VARMACOV Call  
computing cross-covariance matrices, 29  
VARMALIK Call  
computing log-likelihood function, 31  
VARMASIM Call  
generating VARMA( $p,q$ ) time series, 32  
VNORMAL Call  
generating multivariate normal random series, 34  
VTSROOT Call  
calculating characteristic roots, 34

## Y

YIELD function  
calculating yield-to-maturity of a cash-flow stream,



# Syntax Index

## **C**

CONVEXIT function, 3

## **D**

DURATION function, 4

## **F**

FORWARD function, 5

## **L**

LTS call, 10

## **M**

MCD call, 24

## **P**

PV function, 6

## **R**

RATES function, 7

## **S**

SPOT function, 8

## **V**

VARMACOV Call, 29

VARMALIK Call, 31

VARMASIM Call, 32

VNORMAL Call, 34

VTSROOT Call, 34

## **Y**

YIELD function, 9





# Your Turn

---

If you have comments or suggestions about *SAS/IML® Software: Changes and Enhancements, Release 8.1*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute  
Publications Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** yourturn@sas.com

For suggestions about the software, please return the photocopy to

SAS Institute  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** suggest@sas.com



*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS® Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

1. Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
2. Leave the "Subject" line blank.
3. Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.





# **SAS/QC<sup>®</sup> Software: Changes and Enhancements, Release 8.1**

---



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/QC® Software: Changes and Enhancements, Release 8.1*, Cary, NC: SAS Institute Inc., 2000

**SAS/QC® Software: Changes and Enhancements, Release 8.1**

Copyright © 2000 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-708-9

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, May 2000

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. indicates USA registration.

IBM® and all other International Business Machines Corporation product or service names are registered trademarks or trademarks of International Business Machines Corporation in the USA and other countries.

Oracle® and all other Oracle Corporation product or service names are registered trademarks or trademarks of Oracle Corporation in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

Chapter 1. The SHEWHART Procedure . . . . . 1

Chapter 2. The TRELIABILITY Procedure . . . . . 5

Chapter 3. The ADX Interface for Design and Analysis of Experiments . . . . . 19



# Chapter 1

## The SHEWHART Procedure

### Chapter Table of Contents

---

|                  |   |
|------------------|---|
| SYNTAX . . . . . | 3 |
|------------------|---|



# Chapter 1

## The SHEWHART Procedure

---

### SYNTAX

**BOXSTYLE=POINTSSCHEMATIC**

specifies a schematic box chart that is overlaid with points plotting all observations in the subgroups. This box style combines the POINTS and SCHEMATIC box styles. This option is available in the BOXCHART statement.

**MRRESTART****MRRESTART=value**

causes the moving range computation on the IRCHART to be restarted when a missing value is encountered. Without the MRRESTART option, a missing value is simply skipped, and the moving range for the next non-missing subgroup is computed using the most recent previous non-missing value. MRRESTART restarts the moving range computation, so only the observations after the missing value are used in subsequent moving range computations. MRRESTART restarts the moving range computation on any missing value; you can also specify MRRESTART=value to restart only on a particular missing value. For example, MRRESTART=R will restart the computation only when the missing value “.R” is encountered. This option is available in the IRCHART statement.

**TESTRESET=variable****TESTRESET=value**

allows tests for special causes to be reset in a primary chart. The specified variable must be a character variable of length 8, or length 16 if customized tests are requested. The variable values have the same format as those of the \_TESTS\_ variable in a TABLE= data set. A test that is flagged by the TESTRESET= value for a given subgroup is reset starting with that subgroup. That means that a positive result for the test can include the given subgroup only if it is the first subgroup in the pattern. For example, the value “12345678” for the TESTRESET= variable will reset all standard tests for special causes. This option is available in all chart statements.

**TEST2RESET=variable****TEST2RESET=value**

allows tests for special causes to be reset in a secondary chart. The specified variable must be a character variable of length 8, or length 16 if customized tests are requested. The variable values have the same format as those of the \_TESTS\_ variable in a TABLE= data set. A test that is flagged by the TEST2RESET= value for a given subgroup is reset starting with that subgroup. That means a positive result for the test can include the given subgroup only if it is the first subgroup in the pattern. For example, the value “12345678” for the TEST2RESET= variable will reset all standard tests for special causes. This option is available with the MRCHART, RCHART, SCHART, XRCHART, and XSCHART statements.



# Chapter 2

## The TRELIABILITY Procedure

### Chapter Table of Contents

---

|                                                           |    |
|-----------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                 | 7  |
| <b>SYNTAX</b> . . . . .                                   | 7  |
| LOGSCALE Statement . . . . .                              | 7  |
| MCFPLOT Statement . . . . .                               | 8  |
| MODEL Statement . . . . .                                 | 8  |
| PROBPLOT Statement . . . . .                              | 9  |
| RELATIONPLOT Statement . . . . .                          | 9  |
| <b>EXAMPLES</b> . . . . .                                 | 12 |
| Regression Model With Non-Constant Scale . . . . .        | 12 |
| Regression Model With Two Independent Variables . . . . . | 15 |
| <b>REFERENCES</b> . . . . .                               | 18 |





## Chapter 2

# The TRELIABILITY Procedure

---

### Overview

The TRELIABILITY procedure is an experimental version of the RELIABILITY procedure. The following are the major changes and enhancements available in the TRELIABILITY procedure:

- You can specify the logarithm of the distribution scale parameter as a function of explanatory variables in regression models.
- You can construct simultaneous confidence bands on probability plots.
- The default confidence intervals on probability plots are pointwise asymptotic normal parametric intervals on cumulative failure probability. The default intervals in PROC RELIABILITY are pointwise parametric intervals on distribution percentiles.
- Multiple overlaying symbols on plots are labeled with the count of points instead of being jittered as in PROC RELIABILITY.
- You can plot non-linear relationships on relation plots.
- You can construct relation plots with two independent variables.

---

### Syntax

The following statements and statement options are available in the TRELIABILITY procedure in addition to the statements and options in the RELIABILITY procedure.

---

#### LOGSCALE Statement

**LOGSCALE** *<effect-list>* *</options>* ;

LOGSCALE is a new statement in the TRELIABILITY procedure. You use the LOGSCALE statement to model the logarithm of the distribution scale parameter as a function of explanatory variables. A MODEL statement must be present to specify the model for the distribution location parameter. *effect-list* is a list of variables in the input data set representing the values of the independent variables in the model for each observation, and combinations of variables representing interaction terms. It can contain any variables or combination of variables in the input data set. It can contain the same variables as the MODEL statement, or it can contain different variables. The variables in the *effect-list* can be any combination of indicator variables named

in a CLASS statement and continuous variables. The coefficients of the explanatory variables are estimated by maximum likelihood.

The following *options* are available for the LOGSCALE statement.

**Table 2.1.** LOGSCALE Statement Options

| Option                               | Option Description                                                                                      |
|--------------------------------------|---------------------------------------------------------------------------------------------------------|
| INITIAL= <i>number list</i>          | specifies initial values for log-scale regression parameters other than the location, or intercept term |
| INTERCEPT= <i>number</i> < INTINIT > | specifies initial or fixed value of the intercept parameter, depending on whether INTINIT is present    |

## MCFPLOT Statement

The following options are new or can take additional values.

**Table 2.2.** MCFPLOT Statement Options

| Option                                                | Option Description                                   |
|-------------------------------------------------------|------------------------------------------------------|
| PLOTSYMBOL= <i>symbol</i>  <br>( <i>symbol list</i> ) | symbols representing events in an MCF plot           |
| PLOTCOLOR= <i>color</i>  <br>( <i>color list</i> )    | colors of symbols representing events in an MCF plot |

## MODEL Statement

The following are new options in the MODEL statement.

**Table 2.3.** New MODEL Statement Options

| Option                                                                                                                                                                                      | Option Description                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RELATION=ARRHENIUS  <br>ARRHENIUS2  <br>POWER  <br>LOGISTIC<br>RELATION=( ARRHENIUS  <br>ARRHENIUS2  <br>POWER<br>LOGISTIC<br>< , ><br>ARRHENIUS  <br>ARRHENIUS2  <br>POWER  <br>LOGISTIC ) | the <i>logistic</i> function, defined as $T(x) = \log\left(\frac{x}{1-x}\right)$ , has been added as an optional transformation of the independent variable in a regression model. The logistic transformation is useful when the variable is naturally between 0 and 1. Values of 0, 1, and values outside 0 and 1 are treated as missing values. |

## PROBPLOT Statement

The following options are new or can take additional values.

**Table 2.4.** PROBPLOT Statement Options

| Option                                                  | Option Description                                                                                                                                                                                                                                                    |
|---------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CFIT= <i>color</i>   ( <i>color list</i> )              | color for fit lines and confidence curves in a probability plot                                                                                                                                                                                                       |
| LFIT= <i>linetype</i>   ( <i>linetype list</i> )        | line styles for fit lines and confidence curves in a probability plot. The <i>linetype list</i> is a list of numbers from 1 to 46 representing different linetypes, and can be separated by blanks or commas or can be a list in the form $n_1$ to $n_2$ <by $n_3$ >. |
| NOPPOS                                                  | suppresses plotting of symbols for failures in a probability plot                                                                                                                                                                                                     |
| NPINTERVALS=POINTWIASE  <br>SIMULTANEOUS                | type of nonparametric confidence interval displayed in a probability plot                                                                                                                                                                                             |
| PINTERVALS=PROBABILITY  <br>PERCENTILES  <br>LIKELIHOOD | type of parametric pointwise confidence interval displayed in a probability plot. The default type is PROBABILITY, pointwise confidence intervals on cumulative failure probability.                                                                                  |
| PPOSSYMBOL= <i>symbol</i>   ( <i>symbol list</i> )      | symbols representing failures on a probability plot                                                                                                                                                                                                                   |
| PPOSCOLOR= <i>color</i>   ( <i>color list</i> )         | colors of symbols representing failures on a probability plot                                                                                                                                                                                                         |
| SHOWMULTIPLES                                           | display the count for multiple overlaying symbols                                                                                                                                                                                                                     |

## RELATIONPLOT Statement

The following options are new or can take additional values.

**Table 2.5.** RELATIONPLOT Statement Options

| Option                                         | Option Description                                                                          |
|------------------------------------------------|---------------------------------------------------------------------------------------------|
| CFIT= <i>color</i>   ( <i>color list</i> )     | color for fit lines and confidence curves in a probability plot                             |
| CPLOTFIT= <i>color</i>   ( <i>color list</i> ) | colors for percentile lines                                                                 |
| FITTYPE=                                       | specifies method of estimating distribution parameters                                      |
| LSYX                                           | -least squares fit to the probability plot. The probability axis is the dependent variable. |
| LSXY                                           | -least squares fit to the probability plot. The lifetime axis is the dependent variable.    |
| MLE                                            | -maximum likelihood (default)                                                               |

**Table 2.5.** RELATIONPLOT Statement Options (continued)

| Option                                                        | Option Description                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MODEL                                                         | -use the fit from the preceding MODEL statement                                                                                                                                                                                                                                                     |
| NONE                                                          | -no fit is computed                                                                                                                                                                                                                                                                                 |
| REGRESSION                                                    | -use the fit from the preceding MODEL statement. Non-linear relations and percentiles from models using two independent variables can be plotted.                                                                                                                                                   |
| WEIBAYES                                                      | -Weibayes method                                                                                                                                                                                                                                                                                    |
| LFIT= <i>linetype</i>   ( <i>linetype list</i> )              | line styles for fit lines and confidence curves in a probability plot. The <i>linetype list</i> is a list of numbers from 1 to 46 representing different linetypes, and can be separated by blanks or commas or can be a list in the form $n_1$ to $n_2$ <by $n_3$ >.                               |
| LPLOTFIT= <i>linetype</i>   ( <i>linetype list</i> )          | line styles for percentile lines. <i>linetype list</i> is a list of numbers representing different line-types, and can be separated by blanks or commas or can be a list in the form $n_1$ to $n_2$ <by $n_3$ >.                                                                                    |
| NOPPOS                                                        | suppresses plotting of symbols for failures in a probability plot                                                                                                                                                                                                                                   |
| PINTERVALS=PROBABILITY  <br>PERCENTILES  <br>LIKELIHOOD       | type of parametric pointwise confidence interval displayed in a probability plot. The default type is PROBABILITY, pointwise confidence intervals on cumulative failure probability.                                                                                                                |
| RCENCOLOR= <i>color</i>   ( <i>color list</i> )               | colors for the symbols representing uncensored, right censored, and left censored observations in a relation plot                                                                                                                                                                                   |
| RCENSYMBOL= <i>symbol</i>   ( <i>symbol list</i> )            | symbols representing right censored and left censored observations in a relation plot. The <i>symbol</i> is one of the symbol names (plus, star, square, diamond, triangle, hash, paw, point, dot, circle) or a letter (A–Z).                                                                       |
| RELATION=ARRHENIUS  <br>ARRHENIUS2  <br>POWER  <br>LOGISTIC ) | the <i>logistic</i> function, defined as $T(x) = \log\left(\frac{x}{1-x}\right)$ , has been added as an optional transformation of the stress axis. The logistic transformation is useful when the variable is naturally between 0 and 1. Values of 0, 1, and values outside 0 and 1 are not valid. |
| SHOWMULTIPLES                                                 | display the count for multiple overlaying symbols                                                                                                                                                                                                                                                   |

**Table 2.5.** RELATIONPLOT Statement Options (continued)

| Option                      | Option Description                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>variable=number list</i> | allows plots of percentiles from a regression model when two independent variables are used in a MODEL statement <i>effect list</i> . The FIT=REGRESSION option must be used with this option. Percentile plots are created for each value of the independent <i>variable</i> in the <i>number list</i> . <i>number list</i> is a list of numeric values separated by blanks or commas, or in the form of a list $n_1$ to $n_2$ <by $n_3$ >. |

## Examples

The examples in this section illustrate some of the new features in the TRELIABILITY procedure.

### Regression Model With Non-Constant Scale

Nelson (1990, p. 272) and Meeker and Escobar (1998, p. 439) analyzed data from a strain-controlled fatigue test on 26 specimens of a type of superalloy. The following SAS statements create a SAS data set containing for each specimen the level of pseudo-stress (PSTRESS), the number of cycles (in thousands) (KCYCLES) until failure or removal from the test, and a variable to indicate whether a specimen failed (F) or was right censored (C) (STATUS):

```
data alloy;
  input pstress kcycles status$ @@;
  cen = ( status = 'C' );
  datalines;
80.3 211.629 F 99.8 43.331 F
80.6 200.027 F 100.1 12.076 F
80.8 57.923 C 100.5 13.181 F
84.3 155.000 F 113.0 18.067 F
85.2 13.949 F 114.8 21.300 F
85.6 112.968 C 116.4 15.616 F
85.8 152.680 F 118.0 13.030 F
86.4 156.725 F 118.4 8.489 F
86.7 138.114 C 118.6 12.434 F
87.2 56.723 F 120.4 9.750 F
87.3 121.075 F 142.5 11.865 F
89.7 122.372 C 144.5 6.705 F
91.3 112.002 F 145.9 5.733 F
;
run;
```

The following statements use PROC TRELIABILITY to fit a Weibull regression model with the number of cycles to failure as the response variable. The data set=RESIDS contains standardized residuals created with the ODS OUTPUT statement. The MODEL statement specifies a model quadratic in the log of pseudo-stress for the extreme value location parameter. The quadratic model in pseudo-stress PSTRESS is specified in the MODEL statement, and the RELATION=POW option specifies the log transformation be applied to PSTRESS in the MODEL statement and the LOGSCALE statement. The LOGSCALE statement specifies the log of the scale parameter as a linear function of the log of PSTRESS. The RPLOT statement specifies a plot of the data and the fitted regression model versus the variable PSTRESS. The FIT=REGRESSION option specifies plotting the regression model fitted with the preceding MODEL statement. The RELATION=POW option specifies using a log stress axis. The PLOTFIT option specifies plotting the 10th, 50th, and 90th percentiles of the regression model at each stress level. The SLOWER, SUPPER, and LUPPER options control limits on the stress and lifetime axes:

```

ods output ModObstats = Resids;
proc treliability data = alloy;
  distribution Weibull;
  model kcycles*cen(1) = pstress pstress*pstress / relation = pow;
  logscale pstress;
  rplot kcycles*cen(1) = pstress / fit=regression
                                relation = pow
                                plotfit 10 50 90
                                slower=60 supper=160
                                lupper=500;

  label pstress = "Pseudo-Stress";
  label kcycles = "Thousands of Cycles";
run;

```

Figure 2.1 displays the parameter estimates from the fitted regression model. Parameter estimates for both the model for the location parameter and the scale parameter models are shown. Standard errors and confidence limits for all parameter estimates are included.

| The RELIABILITY Procedure     |          |                |                                         |          |
|-------------------------------|----------|----------------|-----------------------------------------|----------|
| Weibull Parameter Estimates   |          |                |                                         |          |
| Parameter                     | Estimate | Standard Error | Asymptotic Normal 95% Confidence Limits |          |
|                               |          |                | Lower                                   | Upper    |
| Intercept                     | 243.1681 | 58.0666        | 129.3596                                | 356.9766 |
| pstress                       | -96.5240 | 24.7075        | -144.9498                               | -48.0983 |
| pstress*pstress               | 9.6653   | 2.6247         | 4.5210                                  | 14.8095  |
| Log-Scale Parameter Estimates |          |                |                                         |          |
| Parameter                     | Estimate | Standard Error | Asymptotic Normal 95% Confidence Limits |          |
|                               |          |                | Lower                                   | Upper    |
| Intercept                     | 4.4666   | 4.0724         | -3.5152                                 | 12.4485  |
| pstress                       | -1.1757  | 0.8731         | -2.8870                                 | 0.5355   |

**Figure 2.1.** Parameter Estimates for Fitted Regression Model



Figure 2.2 displays the plot of the data and fitted regression model.

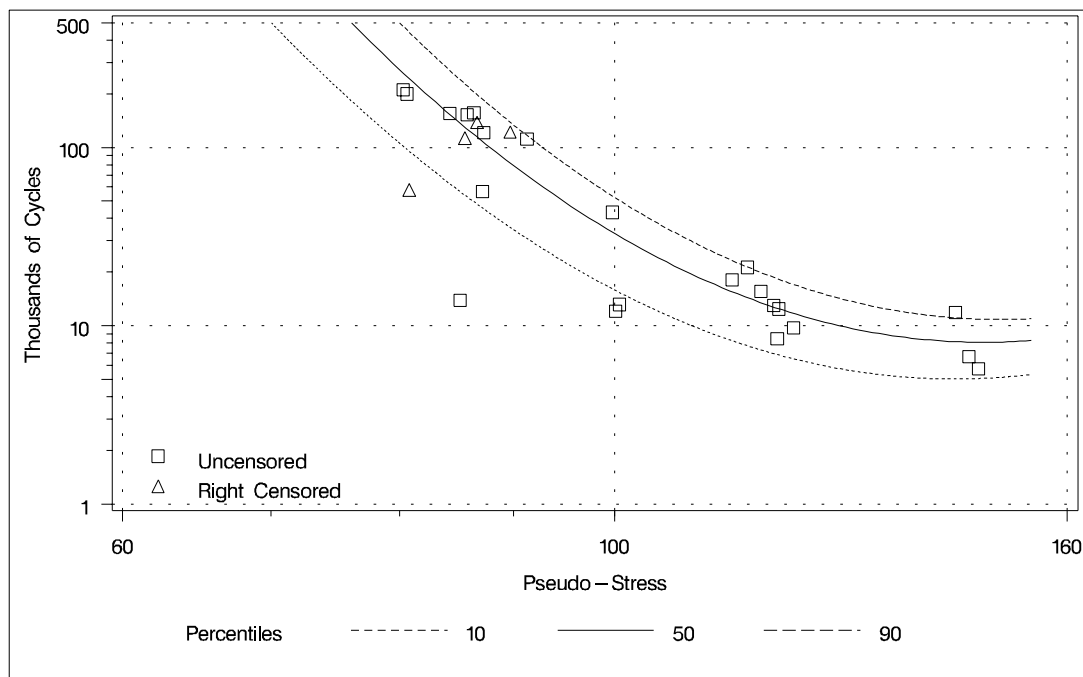


**Figure 2.2.** Superalloy Fatigue Data with Fitted Regression Model

The following SAS statements create an extreme values probability plot of standardized residuals from the regression model shown in Figure 2.3:

```
proc treliability data = Resids;
  distribution ev;
  pplot sresid*cen(1) / nofit;
run;
```

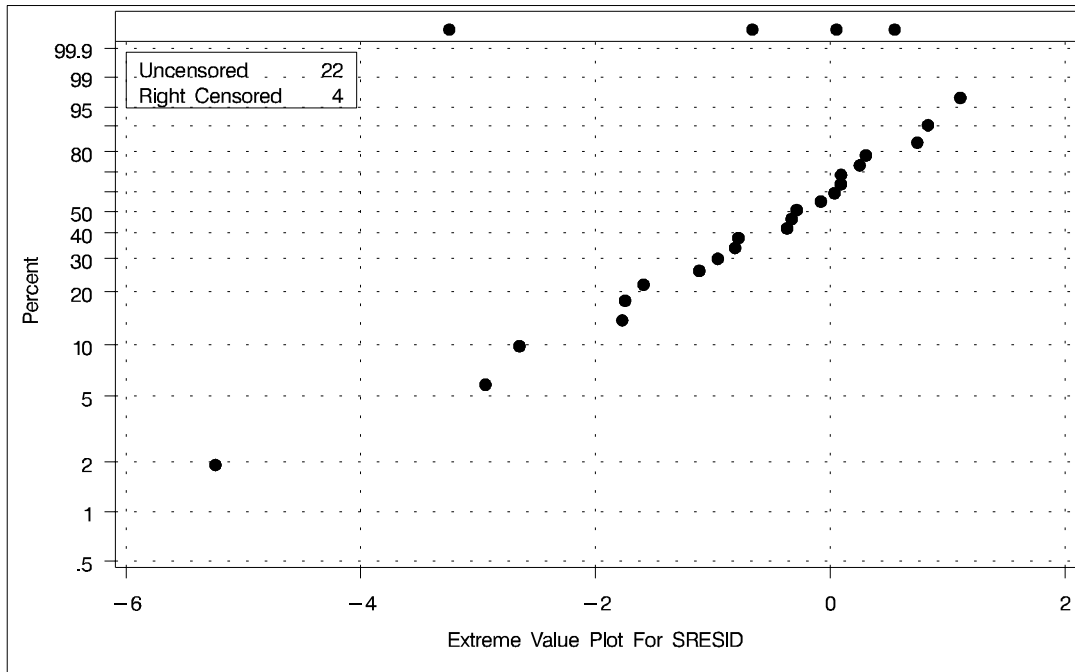


**Figure 2.3.** Residuals for Superalloy Fatigue Data Regression Model

## Regression Model With Two Independent Variables

Meeker and Escobar (1998, p. 447) analyze data from an accelerated test on the lifetimes of glass capacitors as a function of operating voltage and temperature. The following SAS statements create a SAS data set containing the data. There are four lifetimes for each of eight combinations and four censored observations after the fourth failure for each combination:

```
data glass;
  input Temp Voltage @;
  do i = 1 to 4;
    Cen = 0;
    input Hours @;
    output;
  end;
  do i = 1 to 4;
    Cen = 1;
    output;
  end;
  datalines;
170 200 439 904 1092 1105
170 250 572 690 904 1090
170 300 315 315 439 628
170 350 258 258 347 588
180 200 959 1065 1065 1087
180 250 216 315 455 473
180 300 241 315 332 380
180 350 241 241 435 455
;
```

The following statements use PROC TRELIABILITY to analyze the capacitor data. The MODEL statement fits a regression model with TEMP and VOLTAGE as independent variables. Parameter estimates from the fitted regression model are shown in Figure 2.4. An interaction term between TEMP and VOLTAGE is included. The PPLOT statement creates a Weibull probability plot shown in Figure 2.5 with all temperature-voltage combinations overlaid on the same plot. The regression model fit is also plotted. The RPLOT statement creates the plot shown in Figure 2.6 of the data and Weibull distribution percentiles from the regression model as a function of voltage for values of temperature of 150, 170, and 180:

```
proc treliability data = glass;
  distribution Weibull;
  model Hours*Cen(1) = temp voltage temp * voltage;

  pplot Hours*Cen(1) = ( temp voltage ) / fit = model
                                overlay
                                noconf
                                lupper = 2000
                                lfit = ( 1 to 8 );

  rplot Hours*Cen(1) = voltage / fit = regression
                                plotfit
                                temp = 150, 170, 180;

run;
```

| The RELIABILITY Procedure   |          |                   |                                            |         |
|-----------------------------|----------|-------------------|--------------------------------------------|---------|
| Weibull Parameter Estimates |          |                   |                                            |         |
| Parameter                   | Estimate | Standard<br>Error | Asymptotic Normal<br>95% Confidence Limits |         |
|                             |          |                   | Lower                                      | Upper   |
| Intercept                   | 9.4135   | 10.5402           | -11.2449                                   | 30.0719 |
| Temp                        | -0.0062  | 0.0598            | -0.1235                                    | 0.1110  |
| Voltage                     | 0.0086   | 0.0374            | -0.0648                                    | 0.0820  |
| Temp*Voltage                | -0.0001  | 0.0002            | -0.0005                                    | 0.0003  |
| EV Scale                    | 0.3624   | 0.0553            | 0.2687                                     | 0.4887  |
| Weibull Shape               | 2.7593   | 0.4210            | 2.0461                                     | 3.7209  |

**Figure 2.4.** Parameter Estimates for Fitted Regression Model

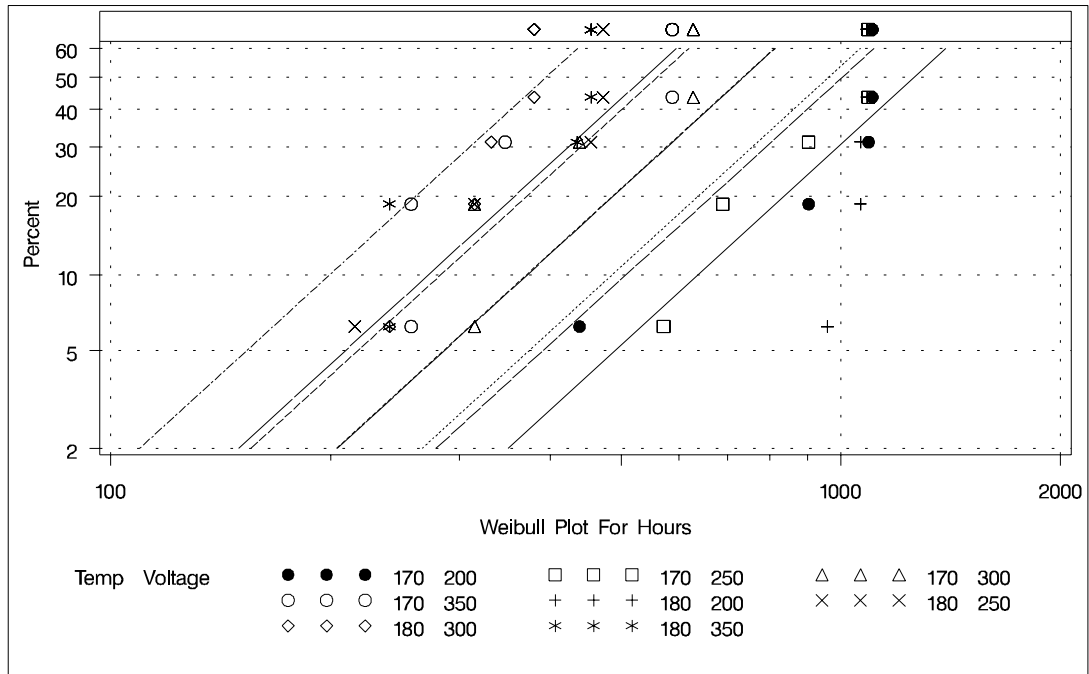


**Figure 2.5.** Probability Plot for Glass Capacitor Regression Model

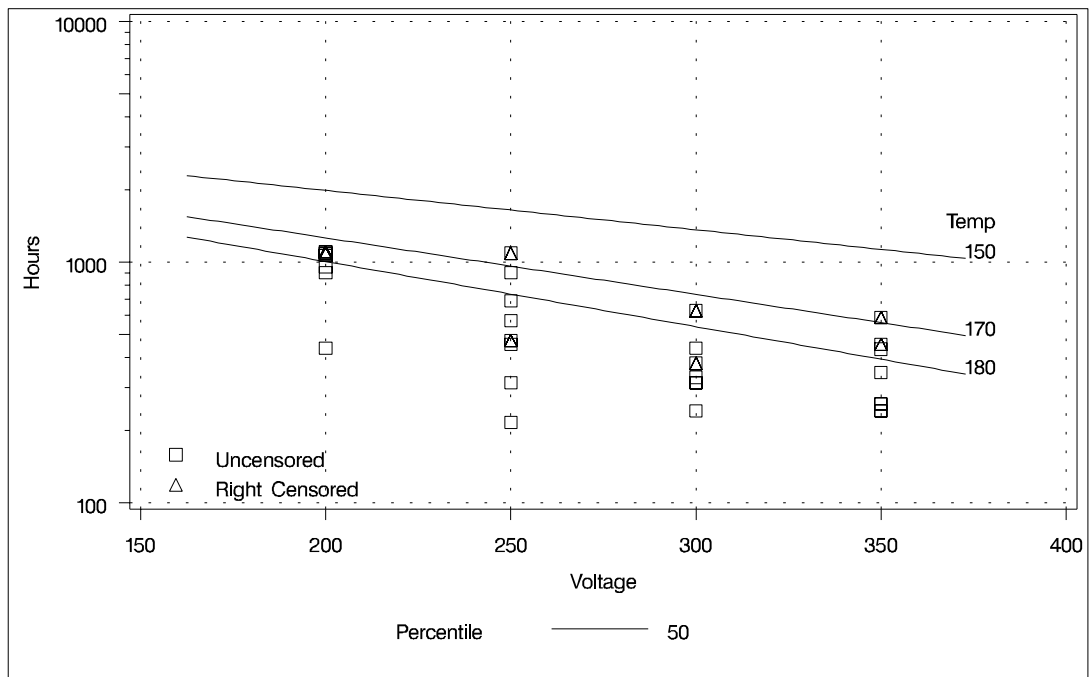


**Figure 2.6.** Plot of Data and Fitted Weibull Percentiles for Glass Capacitor Regression Model

---

## References

Meeker, W.Q. and Escobar, L.A.(1998), *Statistical Methods for Reliability Data*, New York: John Wiley & Sons.

Nelson, W. (1990), *Accelerated Testing: Statistical Models, Test Plans, and Data Analyses*, New York: John Wiley & Sons.

# Chapter 3

## The ADX Interface for Design and Analysis of Experiments

### Chapter Table of Contents

---

|                                               |           |
|-----------------------------------------------|-----------|
| <b>OVERVIEW . . . . .</b>                     | <b>21</b> |
| <b>CONSTRAINED MIXTURE DESIGNS . . . . .</b>  | <b>21</b> |
| <b>SPLIT-PLOT DESIGNS . . . . .</b>           | <b>21</b> |
| <b>NUMERICAL OPTIMIZATION . . . . .</b>       | <b>21</b> |
| <b>DESIGN AUGMENTATION . . . . .</b>          | <b>22</b> |
| <b>CANONICAL AND RIDGE ANALYSIS . . . . .</b> | <b>22</b> |
| <b>GRAPHICAL DISPLAYS . . . . .</b>           | <b>22</b> |



## Chapter 3

# The ADX Interface for Design and Analysis of Experiments

---

## Overview

The ADX Interface is a point-and-click interface for the design and analysis of experiments. Designed for the desktop environment of PCs and workstations, the Interface is intended primarily for engineers and researchers who require a guided interface for every step in the experimental process.

The ADX Interface has been enhanced in several ways. Mixture designs have been extended to include constraints to restrict the region of experimentation, and analysis of split-plot data is now supported. A simple grid search is available to perform a numerical optimization of response variables, and methods are now available to augment a design. N-way effects plots, 3-D surface plots, and box plots are three new interactive graphical displays available.

---

## Constrained Mixture Designs

Sometimes the nature of a mixture experiment does not allow the researcher to set factor levels freely. Constraints on the factors usually do not allow classical design of experiment analysis. Analysis of constrained mixture designs has been added that uses optimal design techniques. This requires a more delicate analysis than that used in the classical case.

---

## Split-plot Designs

In some situations, complete randomization of the order of runs within a block is not possible. In such cases, a split-plot design is appropriate. Typically in a split-plot design each block is divided into whole plots and each whole plot is divided into subplots (or split-plots). The ADX Interface now allows analysis of split-plot data. However, the split-plot design needs to be created outside the Interface and then imported.

---

## Numerical Optimization

The ADX Interface offers a simple grid search to perform a numerical optimization of response variables. A wizard-style approach is used where the response and optimization objective are first specified. The user then has a choice to enter limits on the non-optimized responses. Later the factor space region where the search is to be focused is specified. The ADX Interface computes the response predictions in that



factor space region (for each response variable using the respective predictive model) and then sorts the results based on the response objective.

---

## Design Augmentation

New to the ADX Interface are three methods to augment a design. The user can choose to fold-over a design by adding a new fraction with signs reversed to an existing fractional factorial design to break certain alias links. In some situations where a fold-over design is not optimal, the Interface can search for the minimum runs that can break the specified alias links. Finally, a two-level design can be augmented to a central composite design.

---

## Canonical and Ridge Analysis

Canonical analysis can be performed to determine the shape of the fitted response and determine if the estimated stationary point is a maximum, minimum, or saddle point. This type of analysis can be used to answer the following questions:

- Is the surface shaped like a hill, a valley, a saddle surface, or a flat surface?
- If there is a unique optimum combination of factor values, where is it?
- To which factor or factors are the predicted responses most sensitive?

Ridge analysis can be performed to determine the direction in which further experimentation should be conducted.

---

## Graphical Displays

Three new graphical displays are available in the ADX Interface. The n-way effect plot allows the experimenter to look in detail at potential 3- and 4-way interactions. These can be viewed from main effect up to 4-way interaction effects using the n-way effect plot. The 3-D surface plot is a representation of the response surface system. The 3-D surface plot can be spun to view the surface from various points by using the spin-control palette available at the top left corner of the plot. Box plots are a popular tool to display the differences in the response distribution between different levels of factors. Not only do the box plots show differences in response mean between different levels, thus suggesting significant main effects, they can also show differences in the data spread.

# Your Turn

---

If you have comments or suggestions about *SAS/QC® Software: Changes and Enhancements, Release 8.1*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute  
Publications Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [suggest@sas.com](mailto:suggest@sas.com)



*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS<sup>®</sup> Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

- 1.** Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
- 2.** Leave the "Subject" line blank.
- 3.** Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.

# **SAS/OR<sup>®</sup> Software: Changes and Enhancements, Release 8.1**

---

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/OR® Software: Changes and Enhancements, Release 8.1*, Cary, NC: SAS Institute Inc., 2000

**SAS/OR® Software: Changes and Enhancements, Release 8.1**

Copyright © 2000 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-656-2

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, May 2000

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. indicates USA registration.

IBM® and all other International Business Machines Corporation product or service names are registered trademarks or trademarks of International Business Machines Corporation in the USA and other countries.

Oracle® and all other Oracle Corporation product or service names are registered trademarks or trademarks of Oracle Corporation in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

|                                             |     |
|---------------------------------------------|-----|
| Chapter 1. The BOM Procedure . . . . .      | 1   |
| Chapter 2. The CPM Procedure . . . . .      | 35  |
| Chapter 3. The DTREE Procedure . . . . .    | 53  |
| Chapter 4. The GANTT Procedure . . . . .    | 57  |
| Chapter 5. The INTPOINT Procedure . . . . . | 61  |
| Chapter 6. The NETDRAW Procedure . . . . .  | 189 |
| Chapter 7. The NETFLOW Procedure . . . . .  | 193 |
| Chapter 8. The PM Procedure . . . . .       | 211 |
| Subject Index . . . . .                     | 217 |
| Syntax Index . . . . .                      | 219 |





# Chapter 1

## The BOM Procedure

### Chapter Table of Contents

---

|                                                                   |    |
|-------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                         | 3  |
| <b>GETTING STARTED</b> . . . . .                                  | 3  |
| <b>SYNTAX</b> . . . . .                                           | 11 |
| Functional Summary . . . . .                                      | 11 |
| PROC BOM Statement . . . . .                                      | 11 |
| STRUCTURE Statement . . . . .                                     | 12 |
| <b>DETAILS</b> . . . . .                                          | 14 |
| Single-Level Bills of Material Data Set . . . . .                 | 14 |
| Indented Bills of Material Data Set . . . . .                     | 15 |
| Summarized Bills of Material Data Set . . . . .                   | 17 |
| Missing Values in the Input Data Set . . . . .                    | 18 |
| <b>EXAMPLES</b> . . . . .                                         | 19 |
| Example 1.1 Bill of Material with Lead Time Information . . . . . | 19 |
| Example 1.2 Planning Bill of Material . . . . .                   | 25 |
| Example 1.3 Bills of Material Verification . . . . .              | 31 |
| <b>REFERENCES</b> . . . . .                                       | 33 |



# Chapter 1

## The BOM Procedure

---

### Overview

The BOM procedure is an experimental procedure for performing bill-of-material processing. It composes a series of single-level bills of material into a multilevel tree-structured bill of material, determines the level of each part in the bill, and represents the multilevel bill of material structure in the form of an indented bill of material. PROC BOM can also output a summarized bill of material.

A *bill of material* (BOM) is a list of all parts, ingredients, or materials needed to make one production run of a product. The bill of material may also be called the *formula*, *recipe*, or *ingredients list* in certain process industries (Cox and Blackstone 1998). The way in which the bill of material data are organized and presented is called the *structure* of the bill of material or the structure of the product.

A variety of display formats is available for bills of material. The simplest format is the *single-level BOM*. It consists of a list of all components that are directly used in a parent item. A *multilevel bill of material* provides a display of all components that are directly or indirectly used in a parent item. When an item is a subcomponent, blend, intermediate, etc., all of its components, including purchased parts and raw materials, are also exhibited. A multilevel structure can be illustrated by a tree with several levels. An *indented bill of material* is a form of multilevel BOM. It exhibits the final product as level 0 and all its components as level 1. The level numbers increase as you look down the tree structure. If an item is used in more than one parent within a given product structure, it appears more than once, under every subassembly in which it is used. The *summarized bill of material* is another form of a multilevel bill of material. It lists all the parts and their quantities required in a given product structure. Unlike the indented bill of material, it does not list the levels of parts and does not illustrate the part-component relationships. Moreover, the summarized bill of material lists each item only once for the total quantity required. Refer to Cox and Blackstone (1998) for further details.

The indented bill of material data generated by PROC BOM are organized in such a manner that they can be easily retrieved and manipulated to generate reports, and can be used by other SAS/OR procedures to perform additional analysis. The summarized bill of material data are quite useful in gross requirements planning and other applications.

---

### Getting Started

The ABC Lamp Company product structure example from Fogarty, Blackstone, and Hoffmann (1991) illustrates the basic features of PROC BOM. The single-level bill

of material data are depicted in Figure 1.1. This consists of a list of each part in the product structure, the components that are directly used in the part, and the quantity of each component needed to make one unit of that particular part. The data set also includes a short verbal description and the unit of measure for each part. The SAS DATA step code to create and display the single-level BOM data set is as follows:

```

/* Single-level BOM data */
data SLBOM0;
  input Part      $8.
        Desc      $24.
        Unit      $8.
        Component $8.
        QtyPer    8.0
        ;
  datalines;
LA01    Lamp LA          Each    B100          1
                                           S100          1
                                           A100          1
B100    Base assembly    Each    1100          1
                                           1200          1
                                           1300          1
                                           1400          4
S100    Black shade      Each
A100    Socket assembly  Each    1500          1
                                           1600          1
                                           1700          1
1100    Finished shaft    Each    2100         26
1200    7-Diameter steel plate Each
1300    Hub              Each
1400    1/4-20 Screw      Each
1500    Steel holder      Each    1400          2
1600    One-way socket    Each
1700    Wiring assembly   Each    2200         12
                                           2300          1
2100    3/8 Steel tubing   inches
2200    16-Gauge lamp cord Feet
2300    Standard plug terminal Each
;

/* Display the input data set */
proc print data=SLBOM0 noobs;
  title 'ABC Lamp Company';
  title3 'Single-level Bill of Material';
run;

```

| ABC Lamp Company              |                        |        |           |         |
|-------------------------------|------------------------|--------|-----------|---------|
| Single-level Bill of Material |                        |        |           |         |
| Part                          | Desc                   | Unit   | Component | Qty Per |
| LA01                          | Lamp LA                | Each   | B100      | 1       |
|                               |                        |        | S100      | 1       |
|                               |                        |        | A100      | 1       |
|                               |                        |        |           |         |
| B100                          | Base assembly          | Each   | 1100      | 1       |
|                               |                        |        | 1200      | 1       |
|                               |                        |        | 1300      | 1       |
|                               |                        |        | 1400      | 4       |
|                               |                        |        |           |         |
| S100                          | Black shade            | Each   |           | .       |
| A100                          | Socket assembly        | Each   | 1500      | 1       |
|                               |                        |        | 1600      | 1       |
|                               |                        |        | 1700      | 1       |
| 1100                          | Finished shaft         | Each   | 2100      | 26      |
| 1200                          | 7-Diameter steel plate | Each   |           | .       |
| 1300                          | Hub                    | Each   |           | .       |
| 1400                          | 1/4-20 Screw           | Each   |           | .       |
| 1500                          | Steel holder           | Each   | 1400      | 2       |
| 1600                          | One-way socket         | Each   |           | .       |
| 1700                          | Wiring assembly        | Each   | 2200      | 12      |
|                               |                        |        | 2300      | 1       |
| 2100                          | 3/8 Steel tubing       | inches |           | .       |
| 2200                          | 16-Gauge lamp cord     | Feet   |           | .       |
| 2300                          | Standard plug terminal | Each   |           | .       |

**Figure 1.1.** Single-level Bill of Material

The best way to describe a multilevel product structure is to represent it as a *family tree*. The following code produces the indented bill of material and the summarized bill of material. It also uses PROC NETDRAW to draw a tree diagram for illustrating the multilevel product structure. For further details about the NETDRAW procedure see Chapter 6, “The NETDRAW Procedure.” The standard way of displaying multilevel bill of material structure is the indented parts list report. The following code also displays the indented BOM and summarized BOM:

```

/* Create the indented BOM and the summarized BOM */
proc bom data=SlBOM0 out=IndBOM0 summaryout=SumBOM0;
  structure / part=Part
             component=Component
             quantity=QtyPer
             id=(Desc Unit);
run;

/* Draw a tree diagram for illustrating the product structure */
/* Each record denotes a node in the tree */
data IndBOM0a(drop=Part_ID);
  set IndBOM0;
  Paren_ID=Part_ID;
run;

```

```

/* Extract the Parent - Part information */
data IndBOM0b;
  set IndBOM0(keep=Paren_ID Part_ID);
run;

/* Prepare the data set for running NETDRAW */
data TreBOM0;
  set IndBOM0a IndBOM0b;
run;

/* Specify graphics options */
options border;
pattern1 v=s c=gray;

title h=3 j=c 'Multilevel Bill of Material';
title2 ' ';
footnote2 ' ';
footnote3 h=1 j=l
  'Node shows ID Number, Part Number, and Quantity Required';

/* Invoke PROC NETDRAW to display BOM tree */
proc netdraw data=TreBOM0( where=(Paren_ID NE .) );
  actnet / act=Paren_ID succ=Part_ID id=(Paren_ID _Part_ QtyPer)
    ctext=white font=swiss htext=3 cars=black
    ybetween=3 xbetween=18 centerid
    tree pcompress rotatetext rotate
    arrowhead=0 rectilinear nodefid nlabel;
run;

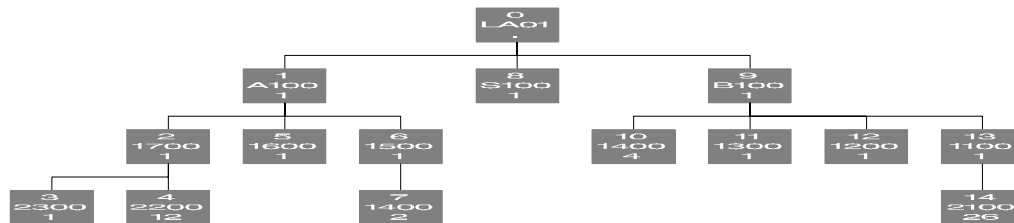
/* Display the indented BOM data */
proc print data=IndBOM0(where=(Part_ID NE .)) noobs;
  var _Level_ _Part_ Part_ID Desc QtyPer Qty_Prod
    Unit _Parent_ Paren_ID _Prod_;
  title 'ABC Lamp Company';
  title3 'Indented Bill of Material, Part LA01';
run;

/* Sort and display the summarized BOM data */
proc sort data=SumBOM0;
  by _Part_;
run;

proc print data=SumBOM0 noobs;
  title 'ABC Lamp Company';
  title3 'Summarized Bill of Material, Part LA01';
run;

```

# Multilevel Bill of Material



Node shows ID Number, Part Number, and Quantity Required

**Figure 1.2.** Multilevel Product Structure

Figure 1.2 displays the tree structure of the ABC Lamp Company. Corresponding to this tree structure is the indented parts list report displayed in Figure 1.3. Each record in the indented BOM data set is associated with a node in the tree structure, and the node or record can be uniquely identified by the `Part_ID` variable. The `Part_ID` information is needed in case some items are required at more than one place in the BOM. For example, the item '1400' (1/4-20 Screw) is required at 'B100' (Base assembly) and '1500' (Steel holder). In this situation the part number is not sufficient to identify each node in the tree structure. The `_Level_` variable denotes the level number of each node. The final product, 'LA01', is at level 0, and the level numbers increase as you look down the tree. The `Part`, `QtyPer`, and `ID` variables (`Desc` and `Unit`) in the Single-level BOM data set, `SIBOM0`, are also included in the indented BOM data set. The name of the `Part` variable is changed to `_Part_` in the indented BOM data set. In addition, the `Qty_Prod` variable contains the quantity of the component required in order to make one unit of the final product 'LA01'. The `_Parent_` variable denotes the parent item in which the part (identified by the `_Part_` variable) is directly used. The `Paren_ID` variable identifies the parent node in the tree structure and the `_Prod_` variable denotes the final product of the structure.



| ABC Lamp Company                     |      |        |                        |     |      |        |      |        |             |
|--------------------------------------|------|--------|------------------------|-----|------|--------|------|--------|-------------|
| Indented Bill of Material, Part LA01 |      |        |                        |     |      |        |      |        |             |
| Level                                | Part | Parent | Description            | Qty | Unit | Parent | Part | Parent | Description |
| 0                                    | LA01 | 0      | Lamp LA                | .   | 1    | Each   | .    | LA01   |             |
| 1                                    | A100 | 1      | Socket assembly        | 1   | 1    | Each   | LA01 | 0      | LA01        |
| 2                                    | 1700 | 2      | Wiring assembly        | 1   | 1    | Each   | A100 | 1      | LA01        |
| 3                                    | 2300 | 3      | Standard plug terminal | 1   | 1    | Each   | 1700 | 2      | LA01        |
| 3                                    | 2200 | 4      | 16-Gauge lamp cord     | 12  | 12   | Feet   | 1700 | 2      | LA01        |
| 2                                    | 1600 | 5      | One-way socket         | 1   | 1    | Each   | A100 | 1      | LA01        |
| 2                                    | 1500 | 6      | Steel holder           | 1   | 1    | Each   | A100 | 1      | LA01        |
| 3                                    | 1400 | 7      | 1/4-20 Screw           | 2   | 2    | Each   | 1500 | 6      | LA01        |
| 1                                    | S100 | 8      | Black shade            | 1   | 1    | Each   | LA01 | 0      | LA01        |
| 1                                    | B100 | 9      | Base assembly          | 1   | 1    | Each   | LA01 | 0      | LA01        |
| 2                                    | 1400 | 10     | 1/4-20 Screw           | 4   | 4    | Each   | B100 | 9      | LA01        |
| 2                                    | 1300 | 11     | Hub                    | 1   | 1    | Each   | B100 | 9      | LA01        |
| 2                                    | 1200 | 12     | 7-Diameter steel plate | 1   | 1    | Each   | B100 | 9      | LA01        |
| 2                                    | 1100 | 13     | Finished shaft         | 1   | 1    | Each   | B100 | 9      | LA01        |
| 3                                    | 2100 | 14     | 3/8 Steel tubing       | 26  | 26   | inches | 1100 | 13     | LA01        |

**Figure 1.3.** Indented Bill of Material

From Figure 1.2 and Figure 1.3, you can easily see that the Indented BOM data set lists all of the parts in the tree structure in *depth-first* order (Aho, Hopcroft, and Ullman 1983). Unlike the Single-level BOM data set, the Indented BOM data set provides the “parent-part” information rather than the “part-component” relationship. The part-component information can still be easily retrieved. For example, the following code creates a list of components that are directly used in the part ‘LA01’:

```

/* Display the components that are directly used */
/* in item LA01 */
proc print data=IndBOM0(where=(_Parent_='LA01')
                        rename=(_Part_=Component))
    noobs;
    var Component Desc QtyPer Unit _Prod_;
    title 'ABC Lamp Company';
    title3 'Single-level Bill of Material Retrieval, Part LA01';
run;
```

| ABC Lamp Company                                   |                 |            |      |        |
|----------------------------------------------------|-----------------|------------|------|--------|
| Single-level Bill of Material Retrieval, Part LA01 |                 |            |      |        |
| Component                                          | Desc            | Qty<br>Per | Unit | _Prod_ |
| A100                                               | Socket assembly | 1          | Each | LA01   |
| S100                                               | Black shade     | 1          | Each | LA01   |
| B100                                               | Base assembly   | 1          | Each | LA01   |

**Figure 1.4.** Components That Are Directly Used in Part LA01

You can also create *single-level where-used* reports from the Indented BOM data set. For example, the following code creates a list of parent items in which the part '1400' is directly used and in what quantity:

```

/* Create the where-used data set */
data Used0a(keep=_Parent_ Paren_ID QtyPer Unit _Prod_);
  set IndBOM0(where=(Part_='1400'));
run;

/* Get the part description from the IndBOM0 data set */
proc sql;
  create table Used0b as
    select Used0a._Parent_, IndBOM0.Desc,
           Used0a.QtyPer, Used0a.Unit, Used0a._Prod_
    from Used0a left join IndBOM0
      on Used0a.Paren_ID=IndBOM0.Part_ID;
quit;

/* Display the where-used data set */
proc print data=Used0b noobs;
  var _Parent_ Desc QtyPer Unit _Prod_;
  title 'ABC Lamp Company';
  title3 'Single-level Where-used Report, Part 1400';
run;

```

| ABC Lamp Company                          |               |            |      |        |
|-------------------------------------------|---------------|------------|------|--------|
| Single-level Where-used Report, Part 1400 |               |            |      |        |
| _Parent_                                  | Desc          | Qty<br>Per | Unit | _Prod_ |
| 1500                                      | Steel holder  | 2          | Each | LA01   |
| B100                                      | Base assembly | 4          | Each | LA01   |

**Figure 1.5.** Parents in Which The Part 1400 Is Directly Used

The Summarized BOM data set, as displayed in Figure 1.6, lists all the parts and their quantity required in the given ABC Lamp Company product structure. Since the Single-level BOM data set, SIBOM0, does not contain any gross requirement information, PROC BOM assumes the final product, 'LA01', is the only master schedule item. The *master schedule* (MPS) items are items selected to be planned

by the master scheduler. These items are deemed critical in their impact on lower-level components or resources. Therefore, the master scheduler, not the computer, maintains the plan for these items. The BOM procedure also assumes that the gross requirement of 'LA01' is 1 unit and that there are no items currently on hand. The net requirement of the part 'LA01' is the gross requirement (1) minus the quantity on hand (0). The gross requirements of the other items are then calculated sequentially, level by level, as follows:

**'B100' Base assembly** (1 per lamp)

|                   |                  |
|-------------------|------------------|
| Gross requirement | $1 \times 1 = 1$ |
| Quantity on hand  | -0               |
| Net requirement   | <hr/> 1          |

**'1100' Finished shaft** (1 per base assembly)

|                   |                  |
|-------------------|------------------|
| Gross requirement | $1 \times 1 = 1$ |
| Quantity on hand  | -0               |
| Net requirement   | <hr/> 1          |

**'2100' 3/8 Steel tubing** (26 inches per shaft)

|                   |                    |
|-------------------|--------------------|
| Gross requirement | $1 \times 26 = 26$ |
| Quantity on hand  | -0                 |
| Net requirement   | <hr/> 26           |

If a part (like item '1400') is required at more than one assembly, the gross requirement is the total needed in all assemblies where it is required.

The Net\_Req variable denotes the net requirement and the Low\_Code variable denotes the *low-level code* of the part. The low-level code is a number that indicates the lowest level in any bill of material at which the item appears. The low-level codes are necessary to make sure the net requirement for a given item is not calculated until all the gross requirements have been calculated down to that level.

| ABC Lamp Company                       |          |          |         |         |                        |        |
|----------------------------------------|----------|----------|---------|---------|------------------------|--------|
| Summarized Bill of Material, Part LA01 |          |          |         |         |                        |        |
| _Part_                                 | Low_Code | Gros_Req | On_Hand | Net_Req | Desc                   | Unit   |
| 1100                                   | 2        | 1        | 0       | 1       | Finished shaft         | Each   |
| 1200                                   | 2        | 1        | 0       | 1       | 7-Diameter steel plate | Each   |
| 1300                                   | 2        | 1        | 0       | 1       | Hub                    | Each   |
| 1400                                   | 3        | 6        | 0       | 6       | 1/4-20 Screw           | Each   |
| 1500                                   | 2        | 1        | 0       | 1       | Steel holder           | Each   |
| 1600                                   | 2        | 1        | 0       | 1       | One-way socket         | Each   |
| 1700                                   | 2        | 1        | 0       | 1       | Wiring assembly        | Each   |
| 2100                                   | 3        | 26       | 0       | 26      | 3/8 Steel tubing       | inches |
| 2200                                   | 3        | 12       | 0       | 12      | 16-Gauge lamp cord     | Feet   |
| 2300                                   | 3        | 1        | 0       | 1       | Standard plug terminal | Each   |
| A100                                   | 1        | 1        | 0       | 1       | Socket assembly        | Each   |
| B100                                   | 1        | 1        | 0       | 1       | Base assembly          | Each   |
| LA01                                   | 0        | 1        | 0       | 1       | Lamp LA                | Each   |
| S100                                   | 1        | 1        | 0       | 1       | Black shade            | Each   |

Figure 1.6. Summarized Bill of Material

## Syntax

The following statements are available in PROC BOM.

```
PROC BOM options ;
      STRUCTURE / options ;
```

## Functional Summary

The following table outlines the options available for the BOM procedure.

**Table 1.1.** PROC BOM Options

| Task                                                                                                                                 | Statement                           | Option                                  |
|--------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|-----------------------------------------|
| <b>Data Set Specification</b><br>Single-level BOM data set<br>Indented BOM data set<br>Summarized BOM data set                       | BOM<br>BOM<br>BOM                   | DATA=<br>OUT=<br>SUMMARYOUT=            |
| <b>Output Control Options</b><br>ID variables                                                                                        | STRUCTURE                           | ID=                                     |
| <b>Part Information Specification</b><br>lead time variable<br>gross requirement variable<br>quantity on hand variable               | STRUCTURE<br>STRUCTURE<br>STRUCTURE | LEADTIME=<br>REQUIREMENT=<br>QTYONHAND= |
| <b>Product Structure Specification</b><br>component variables<br>part variable<br>quantity of component needed to make one part item | STRUCTURE<br>STRUCTURE<br>STRUCTURE | COMPONENT=<br>PART=<br>QUANTITY=        |

## PROC BOM Statement

```
PROC BOM options ;
```

The PROC BOM statement invokes the procedure. You can specify the following options in the PROC BOM statement.

**DATA=SAS-data-set**

names the input SAS data set. This data set (also referred to as the **Single-level BOM** data set) contains all of the single-level bill of material information. See the “Single-Level Bills of Material Data Set” section on page 14 for information about the variables that can be included in this data set. If you do not specify the DATA= option, PROC BOM uses the most recently created SAS data set as the input data set.

**OUT=SAS-data-set**

specifies a name for the output SAS data set that contains all of the indented bill of material information. This data set is also referred to as the **Indented BOM** data set. See the “Indented Bills of Material Data Set” section on page 15 for information about the variables that are included in this data set. If the OUT= option is omitted, the SAS System still creates a data set and names it according to the DATA $n$  naming convention.

**SUMMARYOUT=SAS-data-set**

specifies a name for the output SAS data set that contains all of the summarized bill of material information. This data set is also referred to as the **Summarized BOM** data set. See the “Summarized Bills of Material Data Set” section on page 17 for information about the variables that are included in this data set. If the SUMMARYOUT= option is omitted, PROC BOM does not produce the summarized bill of material.

---

## STRUCTURE Statement

**STRUCTURE** / options ;

The STRUCTURE statement lists the variables in the input data set (the Single-level BOM data set). You should specify the Part variable and at least one Component variable. All other variables are optional.

**COMPONENT=(variables)****COMP=(variables)**

specifies variables in the input data set that name all of the components directly used in the item identified by the Part variable. These variables are also referred to as Component variables. The Component variables must be of the same type and length as the Part variable. At least one Component variable must be specified.

**ID=(variables)**

identifies all variables not specified in the COMPONENT=, LEADTIME=, PART=, QTYONHAND=, QUANTITY=, or REQUIREMENT= options that are to be included in the Indented BOM and the Summarized BOM output data sets. These variables are also referred to as ID variables. The ID= option is useful for carrying any relevant information about each part (identified by the Part variable) from the Single-level BOM data set to the output data sets. The ID variables cannot be named as: Gros\_Req, \_Level\_, Low\_Code, Net\_Req, On\_Hand, Qty\_Per, Qty\_Prod, \_Parent\_, Paren\_ID, \_Part\_, Part\_ID, \_Prod\_, and Tot\_Lead.

**LEADTIME=variable****DUR=variable**

identifies the variable in the input data set that contains the lead time information about the part identified by the Part variable. This variable is also referred to as the LeadTime variable. The *lead time* of a part is the length of time between recognition of the need for an order of this part and the receipt of the parts. The variable specified must be numeric. If you do not specify this option, PROC BOM assumes the lead

time for each part to be 0. The information in the `LeadTime` variable is carried to the Indented BOM output data set. Moreover, the procedure creates a new variable, `Tot_Lead`, for the Indented BOM data set if this option is specified.

**PART=***variable*

**ITEM=***variable*

specifies the variable in the input data set that names the parent item. This variable is also referred to as the `Part` variable. The `Part` variable can be either character or numeric. You must specify the `PART=` option in the `STRUCTURE` statement. All the information specified by the `ID` variables corresponds to the part identified by the `Part` variable. All the information of the `Part` variable, except the name of the variable, is carried to the Indented BOM and the Summarized BOM output data sets. The name of this variable is changed to `_Part_` in both output data sets.

**QTYONHAND=***variable*

**INVENTORY=***variable*

identifies the variable in the input data set that contains the quantity currently on hand for the part identified by the `Part` variable. This variable is also referred to as the `QtyOnHand` variable. The `QtyOnHand` variable must be numeric. If you do not specify the `QtyOnHand` variable, the procedure assumes that you do not have any items on hand. The information of the `QtyOnHand` variable is carried to the Summarized BOM output data set if the `SUMMARYOUT=` option is specified. A new variable, `On_Hand`, is created for the Summarized BOM data set if this option is omitted.

**QUANTITY=***(variables)*

**QTYPER=***(variables)*

identifies variables in the input data set that contain the quantity of items identified by the `Component` variables required to make 1 unit of the item identified by the `Part` variable. These variables are also referred to as the `Quantity` or `QtyPer` variables. The variables specified must be numeric and the number of `Quantity` variables must be equal to the number of `Component` variables. If you do not specify these variables, the procedure assumes that you need 1 unit for each component to make 1 unit of any parent item. In addition, the procedure creates a new variable, `Qty_Per`, for the Indented BOM data set. The information of the `Quantity` variables is carried to the Indented BOM output data set if this option is specified.

**REQUIREMENT=***variable*

**GROSSREQ=***variable*

identifies the variable in the input data set that contains the gross requirement of the part identified by the `Part` variable. This variable is also referred to as the `Requirement` variable. The `Requirement` variable must be numeric. If you do not specify this option, the procedure assumes that the gross requirement for all parts are missing except for the final products. All final products are assumed to have gross requirements of 1 unit. In addition, if you do not specify this option but the `SUMMARYOUT=` option is specified, PROC BOM creates a new variable, `Gros_Req`, for the Summarized BOM output data set.

## Details

### Single-Level Bills of Material Data Set

The BOM procedure uses the Single-level BOM data set as input data with key variable names being used to identify the appropriate information. Table 1.2 lists all of the variables in this input data set, with their type and their interpretation by the BOM procedure. It also lists the options in the STRUCTURE statement that are used to identify those variables.

**Table 1.2.** Single-level BOM Data Set and Associated Variables

| Variable    | Type                 | Option       | Interpretation                                      |
|-------------|----------------------|--------------|-----------------------------------------------------|
| Component   | same as Part         | COMPONENT=   | Component name or number                            |
| ID          | character or numeric | ID=          | Additional information about the part               |
| LeadTime    | numeric              | LEADTIME=    | Lead time of the part                               |
| Part        | character or numeric | PART=        | Part name or number                                 |
| QtyOnHand   | numeric              | QTYONHAND=   | Quantity of the part that is currently on hand      |
| Quantity    | numeric              | QUANTITY=    | Quantity of the component required per unit of part |
| Requirement | numeric              | REQUIREMENT= | Gross requirement of the part                       |

The value of the **Requirement** variable should be missing for most parts, except master schedule items. A *master schedule item* is a part that is selected to be planned by the master scheduler. In general, final products are master schedule items. Some companies like to select a few items that are deemed critical in their impact on lower level components or resources as master schedule items. The requirements of the master schedule items are also called *independent demands*. In the Single-level BOM data set, if the value of the **Requirement** variable is greater than or equal to 0, then the procedure treats the item identified by the **Part** variable as a master schedule item and assumes that its gross requirement is planned by the master scheduler. The gross requirements of other parts (also known as the *dependent demands*) are determined by the procedure as the value of the net requirements of their parent items times the quantity required to make one unit of the parent items. This process is called the *dependent demand process* (Clement, Coldrick, and Sari 1992).

In general, the information in the **Part** and the **LeadTime** variables and all **ID** variables are stored in the company's part master data file. The *part master data file* gives various details of fairly constant data associated with the individual parts. It might contain the following kinds of data (Clement, Coldrick, and Sari 1992):

- part number or name
- part description
- engineering drawing number
- revision level
- commodity code
- cost
- product code
- lead time
- order quantity
- item type or status
- latest engineering change dates
- lot size
- planner/buyer codes
- unit of measure
- make or buy

The information of the **Part**, **Component**, and **Quantity** variables can usually be found in the product structure data. The *product structure data* describe the material content of a product at each stocking level in the manufacturing process. The quantity of each part that is currently on hand is contained in the company's inventory data, and the gross requirements of master schedule items are from the *master production schedule* (MPS). The Single-level BOM data set contains information from all these data sets. You can combine the part master data file, inventory data file, and master production schedule, based on the common variable **Part**, to a new data set. Then you can either append the product structure data (part-component relationship information) to the end of the new data set (as in Example 1.1 on page 19) or merge them together (as in Example 1.2 on page 25).

---

## Indented Bills of Material Data Set

The Indented BOM data set produced by PROC BOM contains all the information in the Single-level BOM data set, plus a few more variables, to describe the structure of the products. The **Part**, **LeadTime**, **Quantity**, and all ID variables in the Single-level BOM data set are carried to this output data set. The name of the **Part** variable is changed to **\_Part\_**. Like the Single-level BOM data set, the information contained in those variables is associated with the item identified by the **Part** variable. A few new variables are also added to the Indented BOM data set.



The `_Level_` variable contains the indenture level of the item identified by the `Part` variable. The top-most parts (the final products) have level 0, and all components that are directly used by these parts have level 1. All subsequent components of those items have the level number increased by 1. This process continues until there are no subsequent components for those highest level parts. The `_Parent_` variable contains the part name or number of the parent item. The `_Prod_` variable contains the part name or number of the final product in the product structure. If the input data set contains the lead time information for each part, PROC BOM measures the *cumulative lead time* from the product (identified by the `_Prod_` variable) to the part (identified by the `Part` variable) and puts it in the `Tot_Lead` variable. The `Qty_Prod` variable denotes the quantity of the part (identified by the `Part` variable) required to make one unit of the product identified by the `_Prod_` variable.

There are two more variables that are added to the output data set: variable `Part_ID` and variable `Paren_ID`. Note that if a item is used in more than one parent item, it appears more than once, under every subassembly in which it is used. In order to distinguish those parts that have multiple appearances, PROC BOM assigns a unique number to each part in the multilevel tree structure. The `Part_ID` variable contains this identification number of the part (identified by the `Part` variable) and the `Paren_ID` variable contains the identification number of the parent item. These two variables are useful when you use the Indented BOM data set as the input data set for other SAS/OR procedures, such as PROC NETDRAW and PROC CPM, which require unique identification of each node.

Table 1.3 lists all of the variables in the Indented BOM data set. It also lists the type and a brief description of these variables.

**Table 1.3.** Indented BOM Data Set and Associated Variables

| Variable | Type                 | Interpretation                                         |
|----------|----------------------|--------------------------------------------------------|
| ID       | character or numeric | Additional information about the part                  |
| LeadTime | numeric              | Lead time of the part                                  |
| _Level_  | numeric              | Level number of the part                               |
| _Parent_ | same as Part         | Parent item of the part                                |
| Paren_ID | numeric              | Identification number of the parent                    |
| Part     | character or numeric | Part name or number                                    |
| Part_ID  | numeric              | Identification number of the part                      |
| _Prod_   | same as Part         | Name or number of the final product                    |
| Qty_Prod | numeric              | Quantity required to produce one unit of final product |
| Quantity | numeric              | Quantity per assembly                                  |
| Tot_Lead | numeric              | Cumulative lead time                                   |

The records in the Indented BOM data set are organized so that the final product item is always listed first. In addition, each item is listed directly after its parent and before any siblings of the parent. For example, from the Indented BOM data as shown in Figure 1.3 on page 8, you can easily see that the final product 'LA01' is the first record of the Indented BOM data set, and the part '1700' is listed directly after its parent, 'A100', and before the siblings of the part: 'A100', 'S100', and 'B100'.

The single-level (part-component) and multilevel relationships of the product structure are clearly shown in the Indented BOM data. For example, the three components that go into the parent 'LA01' are easily determined by the level 1 identifiers. It is also easy to see that the part 'B100' (Base assembly) takes three components at level 2.

---

## Summarized Bills of Material Data Set

If the SUMMARYOUT= option in the PROC BOM statement is specified, the BOM procedure creates a Summarized BOM data set. The summarized bill of material data set lists all the parts and their quantities required and currently on hand. Unlike the indented bill of material data set, it does not list the parent-part relationships and the levels of manufacture, and it lists a part only once for the total quantity used.

The Part, QtyOnHand, and all ID variables in the Single-level BOM data set are carried to this output data set. The name of the Part variable is changed to \_Part\_. If the item identified by the Part variable is a master schedule item, the value of the Requirement variable (the independent demand) is also carried to this data set. The value of the Requirement variable (dependent demand) for those items that are not master schedule items is determined by the dependent demand process as

$$\text{gross requirement of the current item} = \text{net requirement of the parent item} \times \text{quantity needed to make 1 unit of parent item}$$

The new variable, Net\_Req, denotes the net requirement of the item identified by the Part variable. The value of the Net\_Req variable is determined as

$$\text{Net\_Req} = \text{Requirement} - \text{QtyOnHand}$$

The Low\_Code variable denotes the low-level code of the item identified by the Part variable, that is, the lowest level in any bill of material at which the item appears. The low-level codes are necessary to make sure the net requirement of any item is not calculated until all the gross requirements have been calculated down to that level.

Table 1.4 lists all of the variables in the Summarized BOM data set. It also lists the type and a brief description of these variables.

**Table 1.4.** Summarized BOM Data Set and Associated Variables

| Variable    | Type                 | Interpretation                                 |
|-------------|----------------------|------------------------------------------------|
| ID          | character or numeric | Additional information about the part          |
| Low_Code    | numeric              | Low-level code of the part                     |
| Net_Req     | numeric              | Net requirement of the part                    |
| Part        | character or numeric | Part name or number                            |
| QtyOnHand   | numeric              | Quantity of the part that is currently on hand |
| Requirement | numeric              | Gross requirement of the part                  |

## Missing Values in the Input Data Set

The following table summarizes the treatment of missing values for variables in the Single-level BOM input data set.

**Table 1.5.** Treatment of Missing Values in the BOM Procedure

| Variable    | Value Used / Assumption Made / Action Taken                                                                                                                                      |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Component   | value ignored                                                                                                                                                                    |
| ID          | missing, if Part is not missing;<br>otherwise ignored                                                                                                                            |
| LeadTime    | 0, if Part is not missing; otherwise ignored                                                                                                                                     |
| Part        | input error: procedure stops with error message, if this is the first record;<br>otherwise, treat this record as a continuation of the previous record                           |
| QtyOnHand   | 0, if Part is not missing;<br>otherwise ignored                                                                                                                                  |
| Quantity    | 1, if corresponding Component variable is not missing;<br>otherwise ignored                                                                                                      |
| Requirement | ignored if Part is missing;<br>1, if Part is not missing and the part identified by the Part variable is a final product;<br>otherwise, the value is determined by the procedure |

Note that a missing value is allowed for the **Part** variable only when the information of an item is specified in more than one record (see Figure 1.1 on page 5 as an example). If the **Part** variable is missing, the values for the **ID**, **LeadTime**, **QtyOnHand**, and **Requirement** variables are ignored by the procedure. If the **Part** variable is not missing and the item identified by the **Part** variable is not a master schedule item

(the Requirement variable is missing or the value of the Requirement variable is less than 0), the gross requirement of this part is determined based on the dependent demand process.

---

## Examples

This section contains examples that illustrate the features of the BOM procedure. Most of the examples use the data from the ABC Lamp Company product structure described in the “Getting Started” section beginning on page 3.

---

### Example 1.1. Bill of Material with Lead Time Information

As explained in the section “Single-Level Bills of Material Data Set” beginning on page 14, most companies store the relatively constant information of parts in a part master file. In this example, you start with two different data files, Pmaster1 and ParComp1. The Pmaster1 data set, as shown in Output 1.1.1, is a simplified version of the ABC Lamp Company’s part master file. Note that the data set contains the description, the lead time, and the unit of measure information for each part. The data set ParComp1 (as shown in Output 1.1.2) lists the part-component relationship of each part along with the quantity of the component item needed to make one unit of that particular part.

The SAS DATA step code to accomplish this is as follows:

```

/* Part Master data */
data PMaster1;
  input Part      $8.
         Desc     $24.
         Unit     $8.
         LeadTime 8.0
        ;

  datalines;
1100    Finished shaft           Each           2
1200    6-Diameter steel plate  Each           3
1300    Hub                      Each           2
1400    1/4-20 Screw             Each           1
1500    Steel holder             Each           2
1600    One-way socket           Each           2
1700    Wiring assembly          Each           1
2100    3/8 Steel tubing          inches         3
2200    16-Gauge lamp cord        Feet           2
2300    Standard plug terminal    Each           1
A100    Socket assembly           Each           1
B100    Base assembly             Each           1
LA01    Lamp LA                   Each           2
S100    Black shade               Each           2
;

```

```

/* Part-Component relationship data */
data ParComp1;
  input Part      $8.
         Component $8.
         QtyPer   8.0 ;
datalines;
LA01    B100      1
        S100      1
        A100      1
B100    1100      1
        1200      1
        1300      1
        1400      4
A100    1500      1
        1600      1
        1700      1
1100    2100      26
1500    1400      2
1700    2200      12
        2300      1
;

/* Display the part master data */
proc print data=PMaster1 noobs;
  title 'ABC Lamp Company';
  title3 'Part Master Data';
run;

/* Display the part-component relationship data */
proc print data=ParComp1 noobs;
  title 'ABC Lamp Company';
  title3 'Part-Component Relationship Data';
run;

```

Output 1.1.1. Part Master Data Set

| ABC Lamp Company |                        |        |           |
|------------------|------------------------|--------|-----------|
| Part Master Data |                        |        |           |
| Part             | Desc                   | Unit   | Lead Time |
| 1100             | Finished shaft         | Each   | 2         |
| 1200             | 6-Diameter steel plate | Each   | 3         |
| 1300             | Hub                    | Each   | 2         |
| 1400             | 1/4-20 Screw           | Each   | 1         |
| 1500             | Steel holder           | Each   | 2         |
| 1600             | One-way socket         | Each   | 2         |
| 1700             | Wiring assembly        | Each   | 1         |
| 2100             | 3/8 Steel tubing       | inches | 3         |
| 2200             | 16-Gauge lamp cord     | Feet   | 2         |
| 2300             | Standard plug terminal | Each   | 1         |
| A100             | Socket assembly        | Each   | 1         |
| B100             | Base assembly          | Each   | 1         |
| LA01             | Lamp LA                | Each   | 2         |
| S100             | Black shade            | Each   | 2         |

**Output 1.1.2.** Part-Component Relationship Data Set

| ABC Lamp Company                 |           |            |
|----------------------------------|-----------|------------|
| Part-Component Relationship Data |           |            |
| Part                             | Component | Qty<br>Per |
| LA01                             | B100      | 1          |
|                                  | S100      | 1          |
|                                  | A100      | 1          |
| B100                             | 1100      | 1          |
|                                  | 1200      | 1          |
|                                  | 1300      | 1          |
|                                  | 1400      | 4          |
| A100                             | 1500      | 1          |
|                                  | 1600      | 1          |
|                                  | 1700      | 1          |
| 1100                             | 2100      | 26         |
| 1500                             | 1400      | 2          |
| 1700                             | 2200      | 12         |
|                                  | 2300      | 1          |

The SIBOM1 data set is created by appending the part-component data set, ParComp1, to the end of the data set PMaster1 to form an input data set for the procedure:

```

/* Append the Part-Component information to */
/* the part master data                      */
data SIBOM1;
    set PMaster1 ParComp1;
run;

```

The following code produces the Indented BOM data set and displays it in Output 1.1.3. A new variable, Tot\_Lead, has been added to the output data set. This variable denotes the cumulative lead time from the product 'LA01' to the item identified by the \_Part\_ variable:

```

/* Create the indented BOM with lead time */
proc bom data=SIBOM1 out=IndBOM1;
    structure / part=Part
               leadtime=LeadTime
               component=Component
               quantity=QtyPer
               id=(Desc Unit);
run;

/* Display the indented BOM data */
proc print data=IndBOM1(where=(Part_ID NE .)) noobs;
    var _Level_ _Part_ Desc QtyPer Qty_Prod
        Unit LeadTime Tot_Lead _Parent_ _Prod_;
    title 'ABC Lamp Company';
    title3 'Indented Bill of Material, Part LA01';
run;

```

**Output 1.1.3.** Indented Bill of Material with Lead Time

| ABC Lamp Company                     |      |                        |    |    |        |   |   |      |      |
|--------------------------------------|------|------------------------|----|----|--------|---|---|------|------|
| Indented Bill of Material, Part LA01 |      |                        |    |    |        |   |   |      |      |
| —                                    | —    | —                      | Q  | Q  | —      | L | T | —    | —    |
| L                                    | P    | D                      | t  | y  | U      | e | o | P    | P    |
| e                                    | a    | e                      | y  | P  | n      | a | L | a    | r    |
| v                                    | r    | s                      | P  | r  | i      | T | e | r    | o    |
| e                                    | t    | c                      | e  | o  | t      | i | a | e    | d    |
| l                                    | —    | —                      | r  | d  | —      | m | — | —    | —    |
| —                                    | —    | —                      | —  | —  | —      | e | d | —    | —    |
| 0                                    | LA01 | Lamp LA                | .  | 1  | Each   | 2 | 2 |      | LA01 |
| 1                                    | A100 | Socket assembly        | 1  | 1  | Each   | 1 | 3 | LA01 | LA01 |
| 2                                    | 1700 | Wiring assembly        | 1  | 1  | Each   | 1 | 4 | A100 | LA01 |
| 3                                    | 2300 | Standard plug terminal | 1  | 1  | Each   | 1 | 5 | 1700 | LA01 |
| 3                                    | 2200 | 16-Gauge lamp cord     | 12 | 12 | Feet   | 2 | 6 | 1700 | LA01 |
| 2                                    | 1600 | One-way socket         | 1  | 1  | Each   | 2 | 5 | A100 | LA01 |
| 2                                    | 1500 | Steel holder           | 1  | 1  | Each   | 2 | 5 | A100 | LA01 |
| 3                                    | 1400 | 1/4-20 Screw           | 2  | 2  | Each   | 1 | 6 | 1500 | LA01 |
| 1                                    | S100 | Black shade            | 1  | 1  | Each   | 2 | 4 | LA01 | LA01 |
| 1                                    | B100 | Base assembly          | 1  | 1  | Each   | 1 | 3 | LA01 | LA01 |
| 2                                    | 1400 | 1/4-20 Screw           | 4  | 4  | Each   | 1 | 4 | B100 | LA01 |
| 2                                    | 1300 | Hub                    | 1  | 1  | Each   | 2 | 5 | B100 | LA01 |
| 2                                    | 1200 | 6-Diameter steel plate | 1  | 1  | Each   | 3 | 6 | B100 | LA01 |
| 2                                    | 1100 | Finished shaft         | 1  | 1  | Each   | 2 | 5 | B100 | LA01 |
| 3                                    | 2100 | 3/8 Steel tubing       | 26 | 26 | inches | 3 | 8 | 1100 | LA01 |

The following code uses the Indented BOM data set produced in the previous invocation of PROC BOM to produce a data set that can be used by the NETDRAW procedure:

```

/* Prepare the data set for running NETDRAW */
data IndBOM1a(drop=Part_ID);
  set IndBOM1;
  Paren_ID=Part_ID;
run;

data IndBOM1b;
  set IndBOM1(keep=Paren_ID Part_ID);
run;

data TreBOM1;
  set IndBOM1a IndBOM1b;
  LTnQP = put(LeadTime, f3.)||" "||put(QtyPer, f3.);
run;

```

Then, PROC NETDRAW is invoked with the NODISPLAY option to create a data set (Layout1) that contains all the layout information of the tree structure. The OUT= option specifies the name of the layout data set:

```

/* Specify graphics options */
title h=1.5 j=c 'Multilevel Bill of Material with Lead-time Offset';
title2 ' ';
footnote2 ' ';
footnote3 h=0.5 j=l
          'Node shows Part Number, Lead-time, and Quantity Required';
pattern1 v=e c=gray;

/* get the layout information of the BOM tree */
proc netdraw data=TreBOM1( where=(Paren_ID NE .) )
    out=Layout1 nodisplay;
    actnet / act=Paren_ID succ=Part_ID id=( _Part_ LTnQP Tot_Lead)
            ybetween=3 xbetween=15 tree
            rectilinear nodefid nodelabel;
run;

```

In the next invocation, PROC NETDRAW uses a modified layout of the nodes to produce a diagram where the nodes are aligned according to the cumulative lead time. Refer to Chapter 6, “The NETDRAW Procedure,” for details of the NETDRAW procedure.

The resulting tree diagram is shown in Output 1.1.4:

```

/* Lead time offset the X coordinate of each node */
data TreBOM1a;
    set Layout1;
    if _seq_ = 0;
    drop _seq_;
    _x_ = Tot_Lead;
run;

/* display the BOM tree with lead-time offset */
proc netdraw data=TreBOM1a;
    actnet / id=( _Part_ LTnQP)
            font=swiss ctext=black htext=3 carcs=black
            align=Tot_Lead frame pcompress
            xbetween=15 ybetween=3
            arrowhead=0 rectilinear nodefid nodelabel;
run;

```





## Example 1.2. Planning Bill of Material

The previous examples dealt with the bill of material from the perspective of the manufacturing process. In this example, let's turn to another type of BOM that is often useful in planning and handling engineering charges. It is referred to as a *planning BOM*, a *pseudo BOM*, *super BOM*, or a *phantom BOM*.

Let's assume that the ABC Lamp Company sells lamps with 8 different shades (either '14 inches' or '15 inches' with color of 'Black', 'White', 'Cream', or 'Yellow'), 3 alternate base plates ('6 inches', '7 inches', or '8 inches'), and 2 types of sockets ('One-way' and 'Three-way'). Working with all different options the company now has 48 ( $= 8 \times 3 \times 2$ ) different final products. In other words, the ABC Lamp Company has 48 different end items to forecast and plan. This might not sound too complicated; however, this example is only a simplified case. In the real world, some manufacturing companies making automobiles, computers, or machine tools can easily make over 5,000 different final products. So many final products are obviously impossible to forecast accurately and plan with any hope of validity. However, a reasonable forecast for total lamp sales could be made, for example, 2,000 per week. Some parts like the '1100 Finished shaft' are common to all configurations. Although these parts are produced independently of one another, they can be grouped as common parts on the BOM for administrative purposes. The new part that is created to group all common parts is never stocked and hence it is called a *phantom part*. Phantom items are generally assigned 0 lead times and lot-for-lot order quantity. Besides the common parts phantom item, other phantom items are created for planning different options. These items are also referred to as *model items*. This process is known as the *modularized bill of material construction*.

The following SAS code creates three data sets: The Phantom2 data set contains the generic part 'LAXX' and all phantom items. The data set Option2 lists all options of each model item that are not already in the part master data set PMaster1 (described in Example 1.1 on page 19). Finally, the ParComp2 data set contains the re-grouped part-component relationship information:

```

/* Generic and phantom part data */
data Phantom2;
  input Part      $8.
         Desc     $24.
         Req      8.0
         Unit     $8.
         LeadTime 4.0
        ;
  datalines;
LAXX    Lamp LA                      10000 Each    3
4000    Common parts                  . Each    0
A10X    Socket assembly options      . Each    0
B10X    Base assembly options        . Each    0
S10X    Shade options                . Each    0
  ;

```

```

/* Additional options and alternative parts */
data Option2;
  input Part      $8.
        Desc      $24.
        Req       8.0
        Unit      $8.
        LeadTime  4.0
  ;
datalines;
A101  Three-way socket assem.      . Each      1
B101  7in Base assembly            . Each      1
B102  8in Base assembly            . Each      1
S101  14in White shade              . Each      2
S102  14in Cream shade              . Each      2
S103  14in Yellow shade             . Each      2
S104  15in Black shade              . Each      2
S105  15in White shade              . Each      2
S106  15in Cream shade              . Each      2
S107  15in Yellow shade             . Each      2
1201  7-Diameter steel plate        . Each      3
1202  8-Diameter steel plate        . Each      3
1601  Three-way socket              . Each      2
;

/* part-component relationship data */
data ParComp2;
  input Part      $8.
        Component $8.
        QtyPer    8.2
  ;
datalines;
LAXX  4000      1.00
LAXX  B10X      1.00
LAXX  S10X      1.00
LAXX  A10X      1.00
4000  1100      1.00
4000  1300      1.00
4000  1400      4.00
4000  1500      1.00
4000  1700      1.00
B10X  B100      0.32
B10X  B101      0.41
B10X  B102      0.33
S10X  S100      0.07
S10X  S101      0.18
S10X  S102      0.24
S10X  S103      0.10
S10X  S104      0.06
S10X  S105      0.14
S10X  S106      0.22
S10X  S107      0.10
A10X  A100      0.11
A10X  A101      0.92
B100  1200      1.00

```

|      |      |       |
|------|------|-------|
| B101 | 1201 | 1.00  |
| B102 | 1202 | 1.00  |
| A100 | 1600 | 1.00  |
| A101 | 1601 | 1.00  |
| 1100 | 2100 | 26.00 |
| 1500 | 1400 | 2.00  |
| 1700 | 2200 | 12.00 |
| 1700 | 2300 | 1.00  |

;

The part identified as '4000' is the phantom item for common parts '1100', '1300', '1400', '1500', and '1700'. Each available option is structured as a model item with a quantity per parent (identified as the QtyPer variable) that represents its forecasted popularity or option percentage. Note that the total of percentage of each option in this example is more than 100 percent (for example, the total percentage of the 'A10X: Socket assembly options' is  $0.11 + 0.92 = 1.03$ ). This extra percentage is used to cover the uncertainty of the exact percentage split. Using this procedure to cover possible high side demand for each option is called *option overplanning* (Fogarty, Blackstone, and Hoffmann 1991).

The new part master data set PMaster2 combines the generic and phantom part data, the additional options data, and the old part master file shown in Output 1.1.1 on page 20. The modularized single-level BOM data set as shown in Output 1.2.1 combines the data set PMaster2 with the part-component data set ParComp2 by using PROC SQL. The SAS code to accomplish this is as follows:

```

/* Append the old part master data to the new */
/* phantom item and additional option data sets */
data PMaster2;
  set Phantom2
      Option2
      PMaster1(where=(Part NE 'LA01'));
run;

proc sort data=PMaster2;
  by Part;
run;

proc sort data=ParComp2;
  by Part;
run;

/* Merge the Part-Component information to */
/* the new part master data to create an */
/* input data set */
proc sql;
  create table SlBOM2 as
    select PMaster2.*,
           ParComp2.Component, ParComp2.QtyPer
    from PMaster2 left join ParComp2
      on PMaster2.Part=ParComp2.Part;
quit;

```

```

/* Display the single-level BOM data */
proc print data=SlBOM2 noobs;
  title 'ABC Lamp Company';
  title3 'Modularized Single-level Bills of Material';
run;

```

Output 1.2.1. Modularized Single-level Bills of Material

| ABC Lamp Company                           |                         |       |        |           |           |         |
|--------------------------------------------|-------------------------|-------|--------|-----------|-----------|---------|
| Modularized Single-level Bills of Material |                         |       |        |           |           |         |
| Part                                       | Desc                    | Req   | Unit   | Lead Time | Component | Qty Per |
| 1100                                       | Finished shaft          | .     | Each   | 2         | 2100      | 26.00   |
| 1200                                       | 6-Diameter steel plate  | .     | Each   | 3         |           | .       |
| 1201                                       | 7-Diameter steel plate  | .     | Each   | 3         |           | .       |
| 1202                                       | 8-Diameter steel plate  | .     | Each   | 3         |           | .       |
| 1300                                       | Hub                     | .     | Each   | 2         |           | .       |
| 1400                                       | 1/4-20 Screw            | .     | Each   | 1         |           | .       |
| 1500                                       | Steel holder            | .     | Each   | 2         | 1400      | 2.00    |
| 1600                                       | One-way socket          | .     | Each   | 2         |           | .       |
| 1601                                       | Three-way socket        | .     | Each   | 2         |           | .       |
| 1700                                       | Wiring assembly         | .     | Each   | 1         | 2200      | 12.00   |
| 1700                                       | Wiring assembly         | .     | Each   | 1         | 2300      | 1.00    |
| 2100                                       | 3/8 Steel tubing        | .     | inches | 3         |           | .       |
| 2200                                       | 16-Gauge lamp cord      | .     | Feet   | 2         |           | .       |
| 2300                                       | Standard plug terminal  | .     | Each   | 1         |           | .       |
| 4000                                       | Common parts            | .     | Each   | 0         | 1100      | 1.00    |
| 4000                                       | Common parts            | .     | Each   | 0         | 1300      | 1.00    |
| 4000                                       | Common parts            | .     | Each   | 0         | 1400      | 4.00    |
| 4000                                       | Common parts            | .     | Each   | 0         | 1500      | 1.00    |
| 4000                                       | Common parts            | .     | Each   | 0         | 1700      | 1.00    |
| A100                                       | Socket assembly         | .     | Each   | 1         | 1600      | 1.00    |
| A101                                       | Three-way socket assem. | .     | Each   | 1         | 1601      | 1.00    |
| A10X                                       | Socket assembly options | .     | Each   | 0         | A100      | 0.11    |
| A10X                                       | Socket assembly options | .     | Each   | 0         | A101      | 0.92    |
| B100                                       | Base assembly           | .     | Each   | 1         | 1200      | 1.00    |
| B101                                       | 7in Base assembly       | .     | Each   | 1         | 1201      | 1.00    |
| B102                                       | 8in Base assembly       | .     | Each   | 1         | 1202      | 1.00    |
| B10X                                       | Base assembly options   | .     | Each   | 0         | B100      | 0.32    |
| B10X                                       | Base assembly options   | .     | Each   | 0         | B101      | 0.41    |
| B10X                                       | Base assembly options   | .     | Each   | 0         | B102      | 0.33    |
| LAXX                                       | Lamp LA                 | 10000 | Each   | 3         | 4000      | 1.00    |
| LAXX                                       | Lamp LA                 | 10000 | Each   | 3         | B10X      | 1.00    |
| LAXX                                       | Lamp LA                 | 10000 | Each   | 3         | S10X      | 1.00    |
| LAXX                                       | Lamp LA                 | 10000 | Each   | 3         | A10X      | 1.00    |
| S100                                       | Black shade             | .     | Each   | 2         |           | .       |
| S101                                       | 14in White shade        | .     | Each   | 2         |           | .       |
| S102                                       | 14in Cream shade        | .     | Each   | 2         |           | .       |
| S103                                       | 14in Yellow shade       | .     | Each   | 2         |           | .       |
| S104                                       | 15in Black shade        | .     | Each   | 2         |           | .       |
| S105                                       | 15in White shade        | .     | Each   | 2         |           | .       |
| S106                                       | 15in Cream shade        | .     | Each   | 2         |           | .       |
| S107                                       | 15in Yellow shade       | .     | Each   | 2         |           | .       |
| S10X                                       | Shade options           | .     | Each   | 0         | S100      | 0.07    |
| S10X                                       | Shade options           | .     | Each   | 0         | S101      | 0.18    |
| S10X                                       | Shade options           | .     | Each   | 0         | S102      | 0.24    |
| S10X                                       | Shade options           | .     | Each   | 0         | S103      | 0.10    |
| S10X                                       | Shade options           | .     | Each   | 0         | S104      | 0.06    |
| S10X                                       | Shade options           | .     | Each   | 0         | S105      | 0.14    |
| S10X                                       | Shade options           | .     | Each   | 0         | S106      | 0.22    |
| S10X                                       | Shade options           | .     | Each   | 0         | S107      | 0.10    |

The following code invokes PROC BOM to generate the planning BOM and the summarized BOM from the modularized single-level BOM:

```

/* Generate the indented BOM and summarized BOM data sets */
proc bom data=SlBOM2 out=IndBOM2 summaryout=SumBOM2;
  structure / part=Part
             requirement=Req
             leadtime=LeadTime
             component=Component
             quantity=QtyPer
             id=(Desc Unit);
run;

/* Display the indented BOM data */
proc print data=IndBOM2(where=(Part_ID NE .)) noobs;
  var _Level_ _Part_ Desc QtyPer Qty_Prod
      Unit LeadTime Tot_Lead _Parent_ _Prod_;
  title 'ABC Lamp Company';
  title3 'Indented Bill of Material, Part LAXX';
run;

proc sort data=SumBOM2;
  by _Part_;
run;

/* Display the summarized BOM data */
proc print data=SumBOM2 noobs;
  title 'ABC Lamp Company';
  title3 'Summarized Bill of Material, Part LAXX';
run;

```

The indented bill of material is shown in Output 1.2.2. Output 1.2.3 lists the quantity of each part that is needed to build 10,000 lamps.

## Output 1.2.2. Planning Bill of Material with Option Overplanning

| ABC Lamp Company                     |      |                         |       |       |        |     |      |        |         |
|--------------------------------------|------|-------------------------|-------|-------|--------|-----|------|--------|---------|
| Indented Bill of Material, Part LAXX |      |                         |       |       |        |     |      |        |         |
| Level                                | Part | Description             | Qty   | Unit  | Parent | Qty | Unit | Parent | Product |
| —                                    | —    | —                       | —     | —     | —      | —   | —    | —      | —       |
| 0                                    | LAXX | Lamp LA                 | .     | 1.00  | Each   | 3   | 3    |        | LAXX    |
| 1                                    | A10X | Socket assembly options | 1.00  | 1.00  | Each   | 0   | 3    | LAXX   | LAXX    |
| 2                                    | A101 | Three-way socket assem. | 0.92  | 0.92  | Each   | 1   | 4    | A10X   | LAXX    |
| 3                                    | 1601 | Three-way socket        | 1.00  | 0.92  | Each   | 2   | 6    | A101   | LAXX    |
| 2                                    | A100 | Socket assembly         | 0.11  | 0.11  | Each   | 1   | 4    | A10X   | LAXX    |
| 3                                    | 1600 | One-way socket          | 1.00  | 0.11  | Each   | 2   | 6    | A100   | LAXX    |
| 1                                    | S10X | Shade options           | 1.00  | 1.00  | Each   | 0   | 3    | LAXX   | LAXX    |
| 2                                    | S107 | 15in Yellow shade       | 0.10  | 0.10  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S106 | 15in Cream shade        | 0.22  | 0.22  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S105 | 15in White shade        | 0.14  | 0.14  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S104 | 15in Black shade        | 0.06  | 0.06  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S103 | 14in Yellow shade       | 0.10  | 0.10  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S102 | 14in Cream shade        | 0.24  | 0.24  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S101 | 14in White shade        | 0.18  | 0.18  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S100 | Black shade             | 0.07  | 0.07  | Each   | 2   | 5    | S10X   | LAXX    |
| 1                                    | B10X | Base assembly options   | 1.00  | 1.00  | Each   | 0   | 3    | LAXX   | LAXX    |
| 2                                    | B102 | 8in Base assembly       | 0.33  | 0.33  | Each   | 1   | 4    | B10X   | LAXX    |
| 3                                    | 1202 | 8-Diameter steel plate  | 1.00  | 0.33  | Each   | 3   | 7    | B102   | LAXX    |
| 2                                    | B101 | 7in Base assembly       | 0.41  | 0.41  | Each   | 1   | 4    | B10X   | LAXX    |
| 3                                    | 1201 | 7-Diameter steel plate  | 1.00  | 0.41  | Each   | 3   | 7    | B101   | LAXX    |
| 2                                    | B100 | Base assembly           | 0.32  | 0.32  | Each   | 1   | 4    | B10X   | LAXX    |
| 3                                    | 1200 | 6-Diameter steel plate  | 1.00  | 0.32  | Each   | 3   | 7    | B100   | LAXX    |
| 1                                    | 4000 | Common parts            | 1.00  | 1.00  | Each   | 0   | 3    | LAXX   | LAXX    |
| 2                                    | 1700 | Wiring assembly         | 1.00  | 1.00  | Each   | 1   | 4    | 4000   | LAXX    |
| 3                                    | 2300 | Standard plug terminal  | 1.00  | 1.00  | Each   | 1   | 5    | 1700   | LAXX    |
| 3                                    | 2200 | 16-Gauge lamp cord      | 12.00 | 12.00 | Feet   | 2   | 6    | 1700   | LAXX    |
| 2                                    | 1500 | Steel holder            | 1.00  | 1.00  | Each   | 2   | 5    | 4000   | LAXX    |
| 3                                    | 1400 | 1/4-20 Screw            | 2.00  | 2.00  | Each   | 1   | 6    | 1500   | LAXX    |
| 2                                    | 1400 | 1/4-20 Screw            | 4.00  | 4.00  | Each   | 1   | 4    | 4000   | LAXX    |
| 2                                    | 1300 | Hub                     | 1.00  | 1.00  | Each   | 2   | 5    | 4000   | LAXX    |
| 2                                    | 1100 | Finished shaft          | 1.00  | 1.00  | Each   | 2   | 5    | 4000   | LAXX    |
| 3                                    | 2100 | 3/8 Steel tubing        | 26.00 | 26.00 | inches | 3   | 8    | 1100   | LAXX    |

**Output 1.2.3. Summarized Part Requirement**

| ABC Lamp Company                       |                 |            |                |                |                         |             |
|----------------------------------------|-----------------|------------|----------------|----------------|-------------------------|-------------|
| Summarized Bill of Material, Part LAXX |                 |            |                |                |                         |             |
| <u>Part_</u>                           | <u>Low_Code</u> | <u>Req</u> | <u>On_Hand</u> | <u>Net_Req</u> | <u>Desc</u>             | <u>Unit</u> |
| 1100                                   | 2               | 10000      | 0              | 10000          | Finished shaft          | Each        |
| 1200                                   | 3               | 3200       | 0              | 3200           | 6-Diameter steel plate  | Each        |
| 1201                                   | 3               | 4100       | 0              | 4100           | 7-Diameter steel plate  | Each        |
| 1202                                   | 3               | 3300       | 0              | 3300           | 8-Diameter steel plate  | Each        |
| 1300                                   | 2               | 10000      | 0              | 10000          | Hub                     | Each        |
| 1400                                   | 3               | 60000      | 0              | 60000          | 1/4-20 Screw            | Each        |
| 1500                                   | 2               | 10000      | 0              | 10000          | Steel holder            | Each        |
| 1600                                   | 3               | 1100       | 0              | 1100           | One-way socket          | Each        |
| 1601                                   | 3               | 9200       | 0              | 9200           | Three-way socket        | Each        |
| 1700                                   | 2               | 10000      | 0              | 10000          | Wiring assembly         | Each        |
| 2100                                   | 3               | 260000     | 0              | 260000         | 3/8 Steel tubing        | inches      |
| 2200                                   | 3               | 120000     | 0              | 120000         | 16-Gauge lamp cord      | Feet        |
| 2300                                   | 3               | 10000      | 0              | 10000          | Standard plug terminal  | Each        |
| 4000                                   | 1               | 10000      | 0              | 10000          | Common parts            | Each        |
| A100                                   | 2               | 1100       | 0              | 1100           | Socket assembly         | Each        |
| A101                                   | 2               | 9200       | 0              | 9200           | Three-way socket assem. | Each        |
| A10X                                   | 1               | 10000      | 0              | 10000          | Socket assembly options | Each        |
| B100                                   | 2               | 3200       | 0              | 3200           | Base assembly           | Each        |
| B101                                   | 2               | 4100       | 0              | 4100           | 7in Base assembly       | Each        |
| B102                                   | 2               | 3300       | 0              | 3300           | 8in Base assembly       | Each        |
| B10X                                   | 1               | 10000      | 0              | 10000          | Base assembly options   | Each        |
| LAXX                                   | 0               | 10000      | 0              | 10000          | Lamp LA                 | Each        |
| S100                                   | 2               | 700        | 0              | 700            | Black shade             | Each        |
| S101                                   | 2               | 1800       | 0              | 1800           | 14in White shade        | Each        |
| S102                                   | 2               | 2400       | 0              | 2400           | 14in Cream shade        | Each        |
| S103                                   | 2               | 1000       | 0              | 1000           | 14in Yellow shade       | Each        |
| S104                                   | 2               | 600        | 0              | 600            | 15in Black shade        | Each        |
| S105                                   | 2               | 1400       | 0              | 1400           | 15in White shade        | Each        |
| S106                                   | 2               | 2200       | 0              | 2200           | 15in Cream shade        | Each        |
| S107                                   | 2               | 1000       | 0              | 1000           | 15in Yellow shade       | Each        |
| S10X                                   | 1               | 10000      | 0              | 10000          | Shade options           | Each        |

**Example 1.3. Bills of Material Verification**

The previous example illustrated how some companies may have more than one type of bills of material. It is important that while the different types of BOM may differ in the way parts are grouped together, they must have the same component items (except those phantom parts) in the same quantities. This example demonstrates a simple method that compares the different types of BOM and detects any differences.

The transaction data set **Trans3** contains the new values of the gross requirements of the lamp 'LAXX' and the options. Note that the options 'B100', 'S100', and 'A100' that are associated with the lamp 'LA01' described in the "Getting Started" section on page 3, as well as the lamp 'LAXX', have the gross requirements of 1. All other options have 0 gross requirements. Therefore, the item 'LAXX' and all of the selected options are master schedule items. See the "Single-Level Bills of Material Data Set" section beginning on page 14 for a detailed description about master schedule items.

The following SAS data step code updates the value of the **Req** variable in the **SIBOM2** data set (shown in Output 1.2.1 on page 28) with the value of the variable



in the transaction data set Trans3:

```

/* Gross requirement transaction data set */
data Trans3;
    input Part      $8.
           Req      8.0
           ;
datalines;
LAXX      1
B100      1
B101      0
B102      0
S100      1
S101      0
S102      0
S103      0
S104      0
S105      0
S106      0
S107      0
A100      1
A101      0
;

proc sort data=Trans3;
    by Part;
run;

/* Update the gross requirement values of the */
/* single-level BOM data set */
data SLBOM3(drop=OldReq);
    merge SLBOM2(rename=(Req=OldReq)) Trans3(in=in2);
    by Part;

    if not in2 then Req=OldReq;
run;

```

The following code invokes PROC BOM with the new input data set. The summarized bill of material is shown in Output 1.3.1:

```

/* Generate the indented BOM and Summarized BOM */
proc bom data=SLBOM3 out=IndBOM3 summaryout=SumBOM3;
    structure / part=Part
               requirement=Req
               leadtime=LeadTime
               component=Component
               quantity=QtyPer
               id=(Desc Unit);
run;

```

```

/* Sort and display the summarized BOM data */
proc sort data=SumBOM3;
  by _Part_;
run;

proc print data=SumBOM3(where=(Net_Req NE 0)) noobs;
  title 'ABC Lamp Company';
  title3 'Summarized Bill of Material, Part LA01';
run;

```

**Output 1.3.1.** Summarized Bills of Material

| ABC Lamp Company                       |          |     |         |         |                         |        |
|----------------------------------------|----------|-----|---------|---------|-------------------------|--------|
| Summarized Bill of Material, Part LA01 |          |     |         |         |                         |        |
| _Part_                                 | Low_Code | Req | On_Hand | Net_Req | Desc                    | Unit   |
| 1100                                   | 2        | 1   | 0       | 1       | Finished shaft          | Each   |
| 1200                                   | 3        | 1   | 0       | 1       | 6-Diameter steel plate  | Each   |
| 1300                                   | 2        | 1   | 0       | 1       | Hub                     | Each   |
| 1400                                   | 3        | 6   | 0       | 6       | 1/4-20 Screw            | Each   |
| 1500                                   | 2        | 1   | 0       | 1       | Steel holder            | Each   |
| 1600                                   | 3        | 1   | 0       | 1       | One-way socket          | Each   |
| 1700                                   | 2        | 1   | 0       | 1       | Wiring assembly         | Each   |
| 2100                                   | 3        | 26  | 0       | 26      | 3/8 Steel tubing        | inches |
| 2200                                   | 3        | 12  | 0       | 12      | 16-Gauge lamp cord      | Feet   |
| 2300                                   | 3        | 1   | 0       | 1       | Standard plug terminal  | Each   |
| 4000                                   | 1        | 1   | 0       | 1       | Common parts            | Each   |
| A100                                   | 2        | 1   | 0       | 1       | Socket assembly         | Each   |
| A10X                                   | 1        | 1   | 0       | 1       | Socket assembly options | Each   |
| B100                                   | 2        | 1   | 0       | 1       | Base assembly           | Each   |
| B10X                                   | 1        | 1   | 0       | 1       | Base assembly options   | Each   |
| LAXX                                   | 0        | 1   | 0       | 1       | Lamp LA                 | Each   |
| S100                                   | 2        | 1   | 0       | 1       | Black shade             | Each   |
| S10X                                   | 1        | 1   | 0       | 1       | Shade options           | Each   |

By comparing the summarized BOM in Output 1.3.1 with the one displayed in Figure 1.6 on page 10, you can verify that the two bills of material are in agreement except for the low-level code of each part and the phantom items, such as '4000 Common parts', 'A10X Socket assembly options', 'B10X Base assembly options', and 'S10X Shade options'.

## References

- Aho, A.V., Hopcroft, J.E., and Ullman, J.D. (1983), *Data Structures and Algorithms*, Reading, MA: Addison-Wesley.
- Clement, J., Coldrick, A., and Sari, J. (1992), *Manufacturing Data Structures: Building Foundations for Excellence with Bills of Materials and Process Information*, Essex Junction, VT: Oliver Wright Limited Publications, Inc.
- Cox, J.F., III, and Blackstone, J.H., Jr., eds. (1998), *APICS Dictionary*, Ninth Edition, Alexandria, VA: APICS.

Fogarty, D.W., Blackstone, J.H., and Hoffmann, T.R. (1991), *Production and Inventory Management*, Second Edition, Cincinnati, OH: South-Western Publishing Co.

# Chapter 2

## The CPM Procedure

### Chapter Table of Contents

---

|                                                                        |    |
|------------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                              | 37 |
| <b>SYNTAX</b> . . . . .                                                | 37 |
| RESOURCE Statement . . . . .                                           | 37 |
| <b>DETAILS</b> . . . . .                                               | 38 |
| Enhancements to the Resource Data Set . . . . .                        | 38 |
| Specifying Multiple Alternates . . . . .                               | 39 |
| Resource-Driven Durations and Alternate Resources . . . . .            | 40 |
| Auxiliary Resources . . . . .                                          | 41 |
| <b>EXAMPLES</b> . . . . .                                              | 43 |
| Example 2.1 Resource-Driven Durations and Alternate Resource . . . . . | 43 |
| Example 2.2 Multiple Alternate Resources . . . . .                     | 49 |
| Example 2.3 Auxiliary Resources and Alternate Resources . . . . .      | 51 |



# Chapter 2

## The CPM Procedure

---

### Overview

Enhancements have been added to the CPM procedure as follows:

- The ADDCAL option has been added to the RESOURCE statement for including a calendar variable in the Resource Schedule data set.
- The MULTIPLEALTERNATES option has been added to the RESOURCE statement that enables multiple alternate resources to be used simultaneously for a single resource. See the “Specifying Multiple Alternates” section on page 39 for details on how you can control the use of multiple alternates.
- The CPM procedure now enables the use of alternate resources for resources that drive an activity’s duration. See the “Resource-Driven Durations and Alternate Resources” section on page 40 for details.
- Auxiliary resources can now be specified for any resource as described in the “Auxiliary Resources” section on page 41.

---

### Syntax

---

#### RESOURCE Statement

The following new or updated options are available in the RESOURCE statement:

##### **ADDCAL**

requests that a variable, `_CAL_`, be added to the Resource Schedule data set that identifies the resource calendar for each resource used by each activity. For observations that summarize the activity’s schedule, this variable identifies the activity’s calendar.

##### **MULTIPLEALTERNATES**

##### **MULTALT**

indicates that multiple alternate resources can be used to substitute for a single resource. In other words, if one of the alternate resources is not sufficient to substitute for the primary resource, the procedure will use other alternates, as needed, to fulfill the resource requirement. For example, if an activity needs 1.5 programmers and the allowed alternates are JOHN and MARY, the procedure will use JOHN (at rate 1) and MARY (at rate 0.5) to allocate a total of 1.5 programmers. See the “Specifying Multiple Alternates” section on page 39 for details.

##### **OBSTYPE=***variable*

specifies a character variable in the Resource data set that contains the type identi-

fier for each observation. Valid values for this variable are RESLEVEL, RESTYPE, RESPRTY, SUPLEVEL, ALTRATE, ALTPRTY, RESRCDUR, CALENDAR, MULTALT, MINARATE, and AUXRES. If OBSTYPE= is not specified, then all observations in the data set are assumed to denote the levels of the resources, and all resources are assumed to be replenishable and constraining. The observation types, MULTALT, MINARATE, and AUXRES, are new in Release 8.1 and are described in the “Enhancements to the Resource Data Set” section on page 38.

**RESID=variable**

specifies a variable in the RESOURCEIN= data set that indicates the name of the resource variable for which *alternate resource information* or *auxiliary resource information* is being specified in that observation. The specification of auxiliary resource information is new in Release 8.1.

Recall that observations that indicate alternate resources are identified by the values ALTRATE and ALTPRTY for the OBSTYPE variable. These values indicate whether the observation specifies a *rate of substitution* or a *priority for substitution*; the value of the RESID variable in such an observation indicates the particular resource for which that observation specifies alternate resource information. Recall also that the specification of the RESID= option triggers the use of alternate resources. See the “Specifying Alternate Resources” section in the chapter “The CPM Procedure” of the *SAS/OR User’s Guide: Project Management* for further information.

Now observations indicating auxiliary resources are identified by the value AUXRES for the OBSTYPE variable. In such observations, the value of the RESID variable specifies the name of the primary resource while the values of the other resource variables specify the rate of auxiliary resources needed for every unit of the primary resource. See the “Auxiliary Resources” section on page 41 for further information.

---

## Details

---

### Enhancements to the Resource Data Set

Recall that the RESOURCEIN data set (referred to as the Resource data set) contains all of the necessary information about the resources that are to be used by PROC CPM to schedule the project. Typically, the Resource data set contains the resource variables (numeric), a type identifier variable (character) that identifies the type of information in each observation, a period variable (numeric and usually a SAS time, date, or datetime variable), and a RESID variable that is used to specify *alternate resources*.

The value of the type identifier variable in each observation tells CPM how to interpret that observation. Valid values for this variable are RESLEVEL, RESTYPE, RESPRTY, SUPLEVEL, ALTPRTY, ALTRATE, RESRCDUR, CALENDAR, MULTALT, MINARATE, and AUXRES. The last three keywords are new in Release 8.1 and are interpreted as described below.

If the value of the type identifier variable is MULTALT, that observation indicates which resources can have multiple alternate resources. The value 1 for a resource variable in the observation indicates that multiple alternates are allowed for that re-

source, and a value 0 indicates that multiple alternates are not allowed. See the “Specifying Multiple Alternates” section on page 39 for details.

If the value of the type identifier variable is MINARATE, that observation indicates the minimum rate of substitution for each resource, whenever multiple alternates are used. Note that the MINARATE values specified in this observation are used only if the MULTIPLEALTERNATES option is specified or if the Resource data set has an observation with the type identifier value of MULTALT.

If the type identifier variable is specified as AUXRES, the Resource data set must also have a RESID variable identifying the resource for which the current observation lists the required auxiliary resources. See the “Auxiliary Resources” section on page 41 for details.

---

## Specifying Multiple Alternates

As described in the “Specifying Alternate Resources” section in the chapter “The CPM Procedure” of the *SAS/OR User’s Guide: Project Management*, you can use the Resource data set to specify alternate resources for any given resource. You can specify a rate of substitution and a priority for substitution. However, the CPM procedure will not use multiple alternate resources to substitute for a given resource. For example, suppose that an activity needs two programmers and the available programmers (alternate resources) are John and Mary. By default, the CPM procedure cannot assign both John and Mary to the activity to fulfill the resource requirement of two programmers.

However, this type of substitution is useful to effectively model group resources or skill pools. To enable substitution of multiple alternates for a single resource, use the MULTIPLEALTERNATES option in the RESOURCE statement. This option enables all resources that have alternate specifications (via observations of the type ALTRATE or ALTPRTY in the Resource data set) to use multiple alternates.

You can refine this feature to selectively enable multiple substitution or set a minimum rate of substitution, by adding special observations to the Resource data set. As with other aspects of the Resource data set, the specifications related to multiple alternates are identified by observations with special keywords. The keywords are: MULTALT and MINARATE.

Let OBSTYPE = ‘MULTALT’ for the observation that identifies which resources can have multiple alternates. Valid values for such an observation are ‘0’ and ‘1’: ‘0’ indicates that the resource cannot be substituted by multiple resources, and ‘1’ indicates that it can be substituted by multiple resources. If the Resource data set contains such an observation, the MULTIPLEALTERNATES option is ignored and the values specified in the observation are used to enable multiple substitutions for only selected resources.

Let OBSTYPE = ‘MINARATE’ for the observation that indicates the minimum rate of substitution for each resource. For example, you may not want a primary resource requirement of 1.5 programmers, to be satisfied by 5 different alternate programmers at a rate of 0.3 each. To ensure that the minimum rate of substitution is 0.5, specify the value for the resource variable, PROGRAMMER, as ‘0.5’ in the observation with



OBSTYPE = 'MINARATE'. In other words, use this observation if you do not wish to split an activity's resource requirement across several alternate resources with a very small rate of utilization per resource.

Consider the following Resource data:

| OBS | OBSTYPE  | RES_NAME | RES_DATE | JOHN | DAVID | ROBERT |
|-----|----------|----------|----------|------|-------|--------|
| 1   | RESTYPE  |          | .        | 1    | 1     | 1      |
| 2   | ALTRATE  | JOHN     | .        | 1    | 2     | 2      |
| 3   | MULTALT  | .        | .        | 1    | .     | .      |
| 4   | MINARATE | .        | .        | 0.5  | .     | .      |
| 5   | RESLEVEL |          | 15FEB91  | 0    | 1.0   | 1.0    |

In these Resource data, observations 3 and 4 control the use of multiple alternates. They specify that a requirement for John can be substituted with multiple alternates. Further, if multiple alternates are used instead of John, do not allocate them in units less than 0.5. Note also that observation 2 indicates that David and Robert require twice the effort to accomplish John's tasks. Thus, if an activity requires 1 unit of John, and he is unavailable, the CPM procedure will require 2 units of David (or Robert) to substitute for John. However, only 1 unit each of David and Robert is available. If multiple alternates are *not* allowed, the resource allocation algorithm will fail. However, since the resource John *does* allow multiple substitution, the activity can be scheduled with 1 unit of David and 1 unit of Robert (each substituting for 1/2 of the requirement for John).

Allowing multiple alternates for a single resource raises an interesting question: When distributing the resource requirements across multiple alternatives, should the primary resource be included in the list of multiple alternates? For instance, in the above example, if the resource level for John is '0.5' (in observation 5), should the activity use John at rate 0.5 and assign the remainder to one (or more) of the alternate resources? Or, should the primary resource be excluded from the list of possible alternates? You can choose either behavior for the primary resource by specifying '1' (for inclusion) or '0' (for exclusion) in the observation with OBSTYPE = 'ALTRATE' that corresponds to the primary resource (with RES\_NAME = 'JOHN'). Thus, in the above example, John can be one of the multiple alternates when substituting for himself. To exclude John from the list, set the value of the variable JOHN to '0' in observation 2. Note that you will also need to set the value of JOHN to '0' in any observation with OBSTYPE = 'ALTPRTY' and RES\_NAME = 'JOHN'.

---

## Resource-Driven Durations and Alternate Resources

The "Specifying Alternate Resources" section in the chapter "The CPM Procedure" of the *SAS/OR User's Guide: Project Management* describes the use of the RESID= option and the observations of type 'ALTRATE' and 'ALTPRTY' in the Resource data set to control the use of alternate resources during resource allocation. The behavior described in that section refers to the substitution of resources for resources that have a fixed duration. Alternate resources can also be specified for resources that drive an activity's duration. However, the specification of the alternate rate is interpreted differently: it is used as a multiplier of the resource-driven duration.

For example, consider the following Resource data:

| OBS | OBSTYPE  | RES_NAME | RES_DATE | JOHN | DAVID | ROBERT |
|-----|----------|----------|----------|------|-------|--------|
| 1   | RESTYPE  | .        | .        | 1    | 1     | 1      |
| 2   | RESRCDUR | .        | .        | 1    | 1     | 1      |
| 3   | ALTRATE  | JOHN     | .        | 1    | 2     | 2      |
| 4   | ALTPRTY  | JOHN     | .        | 1    | 2     | 3      |
| 5   | RESLEVEL | .        | 15FEB91  | .    | 1.0   | 1.0    |

In these Resource data, the second observation indicates that all the resources are driving resources. The third observation indicates that John can be substituted by David or Robert; however, either David or Robert will require twice as long to accomplish John's tasks for resource-driven activities. Thus, in contrast to the fixed-duration activities, the ALTRATE specification changes the *duration* of the alternate resource, not the *rate of use*.

For instance, consider the following activity with the specified values for the DURATION and WORK variables and the resource requirement for John:

| OBS | ACTIVITY | DURATION | WORK | JOHN | DAVID | ROBERT |
|-----|----------|----------|------|------|-------|--------|
| 1   | Act1     | 3        | 10   | 1    | .     | .      |

Activity 'Act1' requires 10 days of work from John, indicating that the resource-driven duration for Act1 is 10 days. However, in the Resource data above, John is not available, but can be substituted by David or Robert, who will require twice as long to accomplish the work. So, if Act1 is scheduled using either one of the alternate resources, its resource-driven duration will be 20 days.

---

## Auxiliary Resources

Sometimes, the use of a certain resource may require simultaneous use of other resources. For example, the use of a crane necessitates the use of a crane operator. In other words, if an activity needs the resource, CRANE, it will also need a corresponding resource, CRANEOP. Such requirements can be easily modeled by adding both CRANE and CRANEOP to the list of resources required by the activity.

However, when alternate resources are used, the problem becomes more complex. For example, suppose an activity requires a CRANE and there are two possible cranes that can be used, CRANE1 and CRANE2. You can specify CRANE1 and CRANE2 as the alternate resources for CRANE. Each of the two cranes has a specific operator, CRANEOP1 and CRANEOP2, respectively. Specifying CRANEOP1 and CRANEOP2 separately as alternates for CRANEOP will not necessarily guarantee that CRANEOP1 (or CRANEOP2) is used as the alternate for CRANEOP in conjunction with the use of the corresponding CRANE1 (or CRANE2).

You can model such a situation by the use of Auxiliary resource specification: specify CRANEOP1 and CRANEOP2 as auxiliary resources for CRANE1 and CRANE2,

respectively. Auxiliary resources are specified via the Resource data set, using observations identified by the keyword **AUXRES** for the value of the **OBSTYPE** variable. For an observation of this type, the **RESID** variable specifies the name of the primary resource. (This is similar to the specification of **ALTRATE** and **ALTPRTY**.)

Once auxiliary resources are specified in the Resource data set, it is sufficient to specify only the primary resource requirements in the Activity data set. In this situation, for example, it is sufficient to require a **CRANE** for the activity in the Activity data set.

In the Resource data set, add a new observation type, 'AUXRES', which will specify the auxiliary resources that are needed for each primary resource. For an observation of this type, the **RESID** variable specifies the name of the primary resource. The value for each auxiliary resource indicates the rate at which it is required whenever the primary resource is used. You will also need to specify **CRANE1** and **CRANE2** as the alternate resources for **CRANE** in the Resource data set.

When scheduling the activity, PROC CPM will schedule **CRANE1** (or **CRANE2**) as the alternate only if both **CRANE1** and **CRANEOP1** (or **CRANE2** and **CRANEOP2**) are available.

For instance, the above example will have the following Resource data set:

| <b>OBSTYPE</b>  | <b>RESID</b>  | <b>PER</b>     | <b>CRANE</b> | <b>CRANE1</b> | <b>CRANE2</b> | <b>CRANEOP1</b> | <b>CRANEOP2</b> |
|-----------------|---------------|----------------|--------------|---------------|---------------|-----------------|-----------------|
| <b>AUXRES</b>   | <b>CRANE1</b> | .              | .            | .             | .             | <b>1</b>        | .               |
| <b>AUXRES</b>   | <b>CRANE2</b> | .              | .            | .             | .             | .               | <b>1</b>        |
| <b>ALTRATE</b>  | <b>CRANE</b>  | .              | .            | <b>1</b>      | <b>1</b>      | .               | .               |
| <b>RESLEVEL</b> | .             | <b>10DEC99</b> | .            | <b>1</b>      | <b>1</b>      | <b>1</b>        | <b>1</b>        |

## Examples

Let's use a simple software development project to illustrate some of the new resource allocation control options.

### Example 2.1. Resource-Driven Durations and Alternate Resource

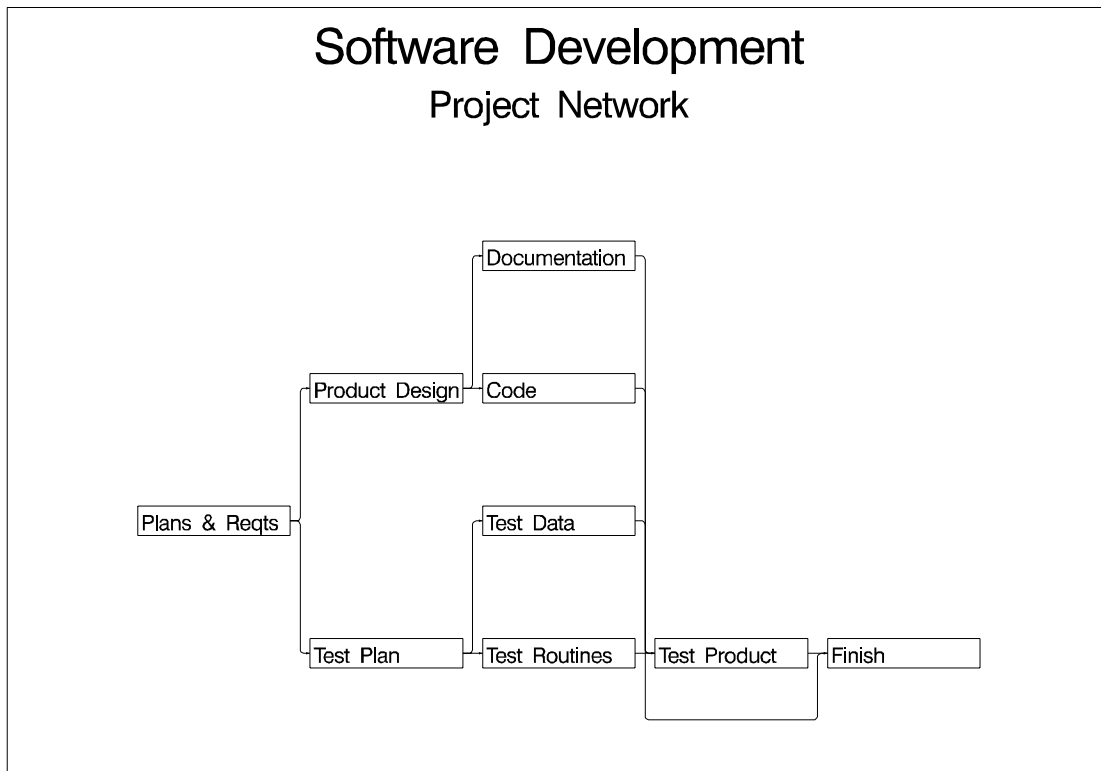


**Figure 2.1.** Software Project Network

Consider a software project requiring a single resource: a programmer. A network diagram displaying the activities and their precedence relationships is shown in Figure 2.1.

Some of the activities in this project have a fixed duration, requiring a fixed length of time from a programmer. Other activities specify the amount of work required in terms of man-days; for these activities, the length of the task will depend on the number of programmers (or rate) that is assigned to the task. The activities in the project, their durations (if fixed) or the total work required (if resource-driven) in days, the precedence constraints, and the resource requirements are displayed in Output 2.1.1.

Suppose that you have only one programmer assigned to the project. You can determine a resource-constrained schedule using PROC CPM by specifying a resource availability data set, `resin` (also in Output 2.1.1). Note that the Resource data set indicates that the resource `progrmr` is a driving resource whenever the `WORK` variable has a valid value.

Output 2.1.1. Project Data

| Software Development<br>Activity Data Set SOFTWARE |     |    |    |     |         |         |
|----------------------------------------------------|-----|----|----|-----|---------|---------|
| Activity                                           | act | s1 | s2 | dur | mandays | progrmr |
| Plans & Reqts                                      | 1   | 2  | 3  | 2   | .       | 1       |
| Product Design                                     | 2   | 4  | 5  | .   | 3       | 1       |
| Test Plan                                          | 3   | 6  | 7  | 3   | .       | .       |
| Documentation                                      | 4   | 9  | .  | 1   | 2       | 1       |
| Code                                               | 5   | 8  | .  | 1   | 10      | 1       |
| Test Data                                          | 6   | 8  | .  | 5   | .       | .       |
| Test Routines                                      | 7   | 8  | .  | 5   | .       | .       |
| Test Product                                       | 8   | 9  | .  | 6   | .       | 1       |
| Finish                                             | 9   | .  | .  | 0   | .       | .       |

| Software Development<br>Resource Availability Data Set |         |          |         |
|--------------------------------------------------------|---------|----------|---------|
| Obs                                                    | per     | otype    | progrmr |
| 1                                                      | .       | resrcdur | 1       |
| 2                                                      | 11APR00 | reslevel | 1       |

The following statements invoke PROC CPM with a WORK= specification on the RESOURCE statement, which identifies (in number of man-days, in this case) the amount of work required from the resource `progrmr` for each activity. If the WORK variable has a missing value, the activity in that observation is assumed to have a fixed duration. The project is scheduled to start on April 11, 2000, and the activities are assumed to follow a five-day work week. The resulting schedule is displayed in Output 2.1.2. For each activity in the project, the value of the variable `DUR_TYPE` indicates whether the resource drives the activity's duration ('RDRIVEN') or not ('FIXED').

```
proc cpm data=software
    out=sftout1 resout=rout1
    rsched=rsftout1
    resin=resin
    date='11apr00'd interval=weekday;
act act;
succ s1 s2;
dur dur;
res progrmr / work=mandays
               obstype=otype
               period=per
               rschedid=Activity;
id Activity;
run;

title 'Software Development';
title2 'Resource Constrained Schedule: Single Programmer';
proc print data=rsftout1;
    id Activity;
run;
```

Output 2.1.2. Resource Schedule

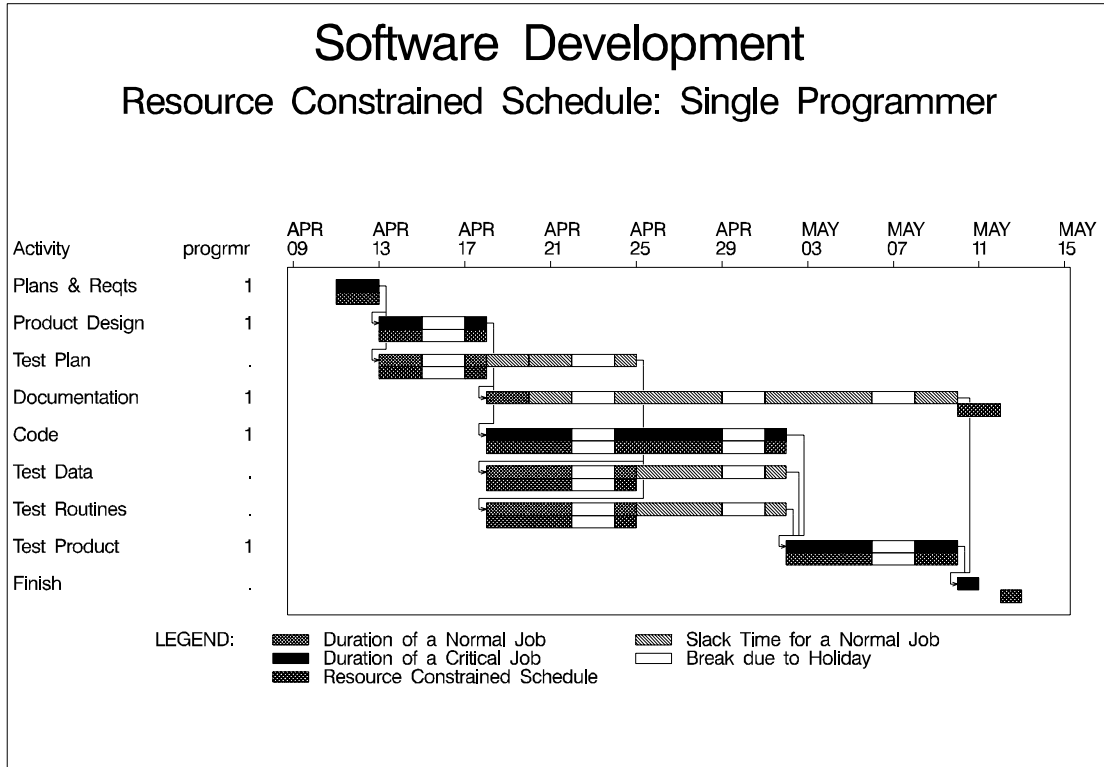
| Software Development<br>Resource Constrained Schedule: Single Programmer |          |          |          |         |          |        |         |
|--------------------------------------------------------------------------|----------|----------|----------|---------|----------|--------|---------|
| Activity                                                                 | act      | RESOURCE | DUR_TYPE | dur     | mandays  | R_RATE | S_START |
| Plans & Reqts                                                            | 1        |          |          | 2       | .        | .      | 11APR00 |
| Plans & Reqts                                                            | 1        | progrmr  | FIXED    | 2       | .        | 1      | 11APR00 |
| Product Design                                                           | 2        |          |          | 3       | .        | .      | 13APR00 |
| Product Design                                                           | 2        | progrmr  | RDRIVEN  | 3       | 3        | 1      | 13APR00 |
| Test Plan                                                                | 3        |          |          | 3       | .        | .      | 13APR00 |
| Documentation                                                            | 4        |          |          | 1       | .        | .      | 10MAY00 |
| Documentation                                                            | 4        | progrmr  | RDRIVEN  | 2       | 2        | 1      | 10MAY00 |
| Code                                                                     | 5        |          |          | 1       | .        | .      | 18APR00 |
| Code                                                                     | 5        | progrmr  | RDRIVEN  | 10      | 10       | 1      | 18APR00 |
| Test Data                                                                | 6        |          |          | 5       | .        | .      | 18APR00 |
| Test Routines                                                            | 7        |          |          | 5       | .        | .      | 18APR00 |
| Test Product                                                             | 8        |          |          | 6       | .        | .      | 02MAY00 |
| Test Product                                                             | 8        | progrmr  | FIXED    | 6       | .        | 1      | 02MAY00 |
| Finish                                                                   | 9        |          |          | 0       | .        | .      | 12MAY00 |
| Activity                                                                 | S_FINISH | E_START  | E_FINISH | L_START | L_FINISH |        |         |
| Plans & Reqts                                                            | 12APR00  | 11APR00  | 12APR00  | 11APR00 | 12APR00  |        |         |
| Plans & Reqts                                                            | 12APR00  | 11APR00  | 12APR00  | 11APR00 | 12APR00  |        |         |
| Product Design                                                           | 17APR00  | 13APR00  | 17APR00  | 13APR00 | 17APR00  |        |         |
| Product Design                                                           | 17APR00  | 13APR00  | 17APR00  | 13APR00 | 17APR00  |        |         |
| Test Plan                                                                | 17APR00  | 13APR00  | 17APR00  | 20APR00 | 24APR00  |        |         |
| Documentation                                                            | 11MAY00  | 18APR00  | 19APR00  | 08MAY00 | 09MAY00  |        |         |
| Documentation                                                            | 11MAY00  | 18APR00  | 19APR00  | 08MAY00 | 09MAY00  |        |         |
| Code                                                                     | 01MAY00  | 18APR00  | 01MAY00  | 18APR00 | 01MAY00  |        |         |
| Code                                                                     | 01MAY00  | 18APR00  | 01MAY00  | 18APR00 | 01MAY00  |        |         |
| Test Data                                                                | 24APR00  | 18APR00  | 24APR00  | 25APR00 | 01MAY00  |        |         |
| Test Routines                                                            | 24APR00  | 18APR00  | 24APR00  | 25APR00 | 01MAY00  |        |         |
| Test Product                                                             | 09MAY00  | 02MAY00  | 09MAY00  | 02MAY00 | 09MAY00  |        |         |
| Test Product                                                             | 09MAY00  | 02MAY00  | 09MAY00  | 02MAY00 | 09MAY00  |        |         |
| Finish                                                                   | 12MAY00  | 10MAY00  | 10MAY00  | 10MAY00 | 10MAY00  |        |         |

The following statements invoke PROC GANTT to display a Gantt chart of the schedule in Output 2.1.3. Note that the activity, 'Documentation', is delayed until May 10, 2000, because there is only one programmer available to the project:

```

title h=2 'Software Development';
title2 h=1.5 'Resource Constrained Schedule: Single Programmer';
proc gantt graphics data=sftout1;
  id Activity progrmr;
  chart / pcompress scale=3 increment=4 interval=weekday
         height=2.5 nojobnum nolegend between=5
         act=act succ=(s1 s2)
  ;
run;

```

**Output 2.1.3.** Resource-Constrained Schedule: Single Programmer

Next, suppose that you have two programmers assigned to your project and you can use either one of them for a given task, depending on their availability. To model this scenario, specify Chris and John as alternate resources that can be substituted for the resource `progrmr`. The Resource data set, `resin2`, printed in Output 2.1.4, indicates that Chris and John are alternates for `progrmr`. Specifying an availability of '0' for the resource `progrmr` ensures that the procedure will assign one of the two programmers, Chris or John, to each task.

The second observation in the data set `resin2` indicates two different rates of substitution for the alternate resources. A value less than 1 indicates that the alternate resource is more efficient than the primary resource, while a value greater than 1 indicates that the alternate resource is less efficient. For fixed-duration activities, the use of the alternate resource changes the *rate* of utilization of the resource, while for a resource-driven activity, it changes the *duration* of the resource.

**Output 2.1.4.** Alternate Programmers

| Resource Data Set RESIN2 |         |          |         |         |       |      |
|--------------------------|---------|----------|---------|---------|-------|------|
| Obs                      | per     | otype    | resid   | progrmr | Chris | John |
| 1                        | .       | resrcdur |         | 1       | 1.0   | 1.0  |
| 2                        | .       | altrate  | progrmr | .       | 1.3   | 0.5  |
| 3                        | 11APR00 | reslevel |         | .       | 1.0   | 1.0  |

The following statements invoke PROC CPM with the new Resource data set and a modified Activity data set that includes the newly added resource variables, Chris and John. You can see the effects of the alternate resource specifications in the Resource Schedule data set, printed in Output 2.1.5. Note that the activity 'Product Design' that takes 3 days of time from a generic programmer actually takes 3.9 days because the programmer used is Chris, who is substituted at a rate of 1.3. On the other hand, the programmer John efficiently completes the task, 'Documentation', in only 1 day:

```
data software2;
  set software;
  Chris = .;
  John = .;
run;

proc cpm data=software2 out=sftout2 rsched=rsftout2
        resin=resin2
        date='11apr00'd interval=weekday resout=rout2;
  act act;
  succ s1 s2;
  dur dur;
  res progrmr Chris John / work=mandays
                           obstype=otype
                           period=per
                           resid=resid
                           rschedid=Activity;
  id Activity;
run;
```



Output 2.1.5. Resource Schedule with Alternate Programmers

| Software Development<br>Resource Constrained Schedule<br>Alternate Resources at Varying Rates |          |          |          |         |          |        |         |
|-----------------------------------------------------------------------------------------------|----------|----------|----------|---------|----------|--------|---------|
| Activity                                                                                      | act      | RESOURCE | DUR_TYPE | dur     | mandays  | R_RATE | S_START |
| Plans & Reqts                                                                                 | 1        |          |          | 2.0     | .        | .      | 11APR00 |
| Plans & Reqts                                                                                 | 1        | progrmr  | FIXED    | 2.0     | .        | 1.0    | .       |
| Plans & Reqts                                                                                 | 1        | John     | FIXED    | 2.0     | .        | 0.5    | 11APR00 |
| Product Design                                                                                | 2        |          |          | 3.0     | .        | .      | 13APR00 |
| Product Design                                                                                | 2        | progrmr  | RDRIVEN  | 3.0     | 3.0      | 1.0    | .       |
| Product Design                                                                                | 2        | Chris    | RDRIVEN  | 3.9     | 3.9      | 1.0    | 13APR00 |
| Test Plan                                                                                     | 3        |          |          | 3.0     | .        | .      | 13APR00 |
| Documentation                                                                                 | 4        |          |          | 1.0     | .        | .      | 18APR00 |
| Documentation                                                                                 | 4        | progrmr  | RDRIVEN  | 2.0     | 2.0      | 1.0    | .       |
| Documentation                                                                                 | 4        | John     | RDRIVEN  | 1.0     | 1.0      | 1.0    | 18APR00 |
| Code                                                                                          | 5        |          |          | 1.0     | .        | .      | 18APR00 |
| Code                                                                                          | 5        | progrmr  | RDRIVEN  | 10.0    | 10.0     | 1.0    | .       |
| Code                                                                                          | 5        | Chris    | RDRIVEN  | 13.0    | 13.0     | 1.0    | 18APR00 |
| Test Data                                                                                     | 6        |          |          | 5.0     | .        | .      | 18APR00 |
| Test Routines                                                                                 | 7        |          |          | 5.0     | .        | .      | 18APR00 |
| Test Product                                                                                  | 8        |          |          | 6.0     | .        | .      | 05MAY00 |
| Test Product                                                                                  | 8        | progrmr  | FIXED    | 6.0     | .        | 1.0    | .       |
| Test Product                                                                                  | 8        | John     | FIXED    | 6.0     | .        | 0.5    | 05MAY00 |
| Finish                                                                                        | 9        |          |          | 0.0     | .        | .      | 15MAY00 |
| Activity                                                                                      | S_FINISH | E_START  | E_FINISH | L_START | L_FINISH |        |         |
| Plans & Reqts                                                                                 | 12APR00  | 11APR00  | 12APR00  | 11APR00 | 12APR00  |        |         |
| Plans & Reqts                                                                                 | .        | 11APR00  | 12APR00  | 11APR00 | 12APR00  |        |         |
| Plans & Reqts                                                                                 | 12APR00  | .        | .        | .       | .        |        |         |
| Product Design                                                                                | 17APR00  | 13APR00  | 17APR00  | 13APR00 | 17APR00  |        |         |
| Product Design                                                                                | .        | 13APR00  | 17APR00  | 13APR00 | 17APR00  |        |         |
| Product Design                                                                                | 17APR00  | .        | .        | .       | .        |        |         |
| Test Plan                                                                                     | 17APR00  | 13APR00  | 17APR00  | 20APR00 | 24APR00  |        |         |
| Documentation                                                                                 | 18APR00  | 18APR00  | 19APR00  | 08MAY00 | 09MAY00  |        |         |
| Documentation                                                                                 | .        | 18APR00  | 19APR00  | 08MAY00 | 09MAY00  |        |         |
| Documentation                                                                                 | 18APR00  | .        | .        | .       | .        |        |         |
| Code                                                                                          | 04MAY00  | 18APR00  | 01MAY00  | 18APR00 | 01MAY00  |        |         |
| Code                                                                                          | .        | 18APR00  | 01MAY00  | 18APR00 | 01MAY00  |        |         |
| Code                                                                                          | 04MAY00  | .        | .        | .       | .        |        |         |
| Test Data                                                                                     | 24APR00  | 18APR00  | 24APR00  | 25APR00 | 01MAY00  |        |         |
| Test Routines                                                                                 | 24APR00  | 18APR00  | 24APR00  | 25APR00 | 01MAY00  |        |         |
| Test Product                                                                                  | 12MAY00  | 02MAY00  | 09MAY00  | 02MAY00 | 09MAY00  |        |         |
| Test Product                                                                                  | .        | 02MAY00  | 09MAY00  | 02MAY00 | 09MAY00  |        |         |
| Test Product                                                                                  | 12MAY00  | .        | .        | .       | .        |        |         |
| Finish                                                                                        | 15MAY00  | 10MAY00  | 10MAY00  | 10MAY00 | 10MAY00  |        |         |

## Example 2.2. Multiple Alternate Resources

### Output 2.2.1. Multiple Alternates

| Software Development<br>Use of Multiple Alternate Resources<br>Activity Data Set |                |     |         |     |    |    |         |       |      |
|----------------------------------------------------------------------------------|----------------|-----|---------|-----|----|----|---------|-------|------|
| Obs                                                                              | Activity       | dur | mandays | act | s1 | s2 | progrmr | Chris | John |
| 1                                                                                | Plans & Reqts  | 2   | .       | 1   | 2  | 3  | 2       | .     | .    |
| 2                                                                                | Product Design | .   | 3       | 2   | 4  | 5  | 1       | .     | .    |
| 3                                                                                | Test Plan      | 3   | .       | 3   | 6  | 7  | .       | .     | .    |
| 4                                                                                | Documentation  | 1   | 2       | 4   | 9  | .  | 1       | .     | .    |
| 5                                                                                | Code           | 1   | 10      | 5   | 8  | .  | 1       | .     | .    |
| 6                                                                                | Test Data      | 5   | .       | 6   | 8  | .  | .       | .     | .    |
| 7                                                                                | Test Routines  | 5   | .       | 7   | 8  | .  | .       | .     | .    |
| 8                                                                                | Test Product   | 6   | .       | 8   | 9  | .  | 1       | .     | .    |
| 9                                                                                | Finish         | 0   | .       | 9   | .  | .  | .       | .     | .    |

| Software Development<br>Use of Multiple Alternate Resources<br>Resource Data Set |         |          |         |         |       |      |
|----------------------------------------------------------------------------------|---------|----------|---------|---------|-------|------|
| Obs                                                                              | per     | otype    | resid   | progrmr | Chris | John |
| 1                                                                                | .       | resrcdur |         | 1       | 1     | 1    |
| 2                                                                                | .       | altrate  | progrmr | .       | 1     | 1    |
| 3                                                                                | 11APR00 | reslevel |         | .       | 1     | 1    |

This example illustrates the use of the MULTIPLEALTERNATES option. The Activity data set printed in Output 2.2.1 is a slightly modified version of the data set in Example 2.1. The difference is in the resource requirement for the first activity in the project. The 'Plans and Requirements' task requires 2 programmers. By default, when alternate resources are used, the CPM procedures cannot use multiple alternate resources to substitute for any given resource. In this example, however, you would like the procedure to use both Chris and John for the first task. The Resource data set resmult is also printed in Output 2.2.1, showing that both Chris and John are alternates that can be substituted at the same rate as the primary resource.

To enable PROC CPM to use multiple alternates, use the MULTIPLEALTERNATES option, as shown in the following invocation:

```
proc cpm data=softmult out=sftmult rsched=rsftmult
      resin=resmult
      date='11apr00'd interval=weekday resout=routmult;
  act act;
  succ s1 s2;
  dur dur;
  res progrmr Chris John / work=mandays
      obstype=otype
      period=per resid=resid
      multiplealternates
      rschedid=Activity;
  id Activity;
run;
```

The resulting schedule is printed in Output 2.2.2. Note that both programmers are used for the activity 'Plans and Reqs'.

**Output 2.2.2.** Multiple Alternates: Resource Schedule Data Set

| Software Development<br>Use of Multiple Alternate Resources<br>Resource Constrained Schedule |          |          |          |         |          |        |         |
|----------------------------------------------------------------------------------------------|----------|----------|----------|---------|----------|--------|---------|
| Activity                                                                                     | act      | RESOURCE | DUR_TYPE | dur     | mandays  | R_RATE | S_START |
| Plans & Reqs                                                                                 | 1        |          |          | 2       | .        | .      | 11APR00 |
| Plans & Reqs                                                                                 | 1        | progrmr  | FIXED    | 2       | .        | 2      | .       |
| Plans & Reqs                                                                                 | 1        | John     | FIXED    | 2       | .        | 1      | 11APR00 |
| Plans & Reqs                                                                                 | 1        | Chris    | FIXED    | 2       | .        | 1      | 11APR00 |
| Product Design                                                                               | 2        |          |          | 3       | .        | .      | 13APR00 |
| Product Design                                                                               | 2        | progrmr  | RDRIVEN  | 3       | 3        | 1      | .       |
| Product Design                                                                               | 2        | Chris    | RDRIVEN  | 3       | 3        | 1      | 13APR00 |
| Test Plan                                                                                    | 3        |          |          | 3       | .        | .      | 13APR00 |
| Documentation                                                                                | 4        |          |          | 1       | .        | .      | 18APR00 |
| Documentation                                                                                | 4        | progrmr  | RDRIVEN  | 2       | 2        | 1      | .       |
| Documentation                                                                                | 4        | John     | RDRIVEN  | 2       | 2        | 1      | 18APR00 |
| Code                                                                                         | 5        |          |          | 1       | .        | .      | 18APR00 |
| Code                                                                                         | 5        | progrmr  | RDRIVEN  | 10      | 10       | 1      | .       |
| Code                                                                                         | 5        | Chris    | RDRIVEN  | 10      | 10       | 1      | 18APR00 |
| Test Data                                                                                    | 6        |          |          | 5       | .        | .      | 18APR00 |
| Test Routines                                                                                | 7        |          |          | 5       | .        | .      | 18APR00 |
| Test Product                                                                                 | 8        |          |          | 6       | .        | .      | 02MAY00 |
| Test Product                                                                                 | 8        | progrmr  | FIXED    | 6       | .        | 1      | .       |
| Test Product                                                                                 | 8        | Chris    | FIXED    | 6       | .        | 1      | 02MAY00 |
| Finish                                                                                       | 9        |          |          | 0       | .        | .      | 10MAY00 |
| Activity                                                                                     | S_FINISH | E_START  | E_FINISH | L_START | L_FINISH |        |         |
| Plans & Reqs                                                                                 | 12APR00  | 11APR00  | 12APR00  | 11APR00 | 12APR00  |        |         |
| Plans & Reqs                                                                                 | .        | 11APR00  | 12APR00  | 11APR00 | 12APR00  |        |         |
| Plans & Reqs                                                                                 | 12APR00  | .        | .        | .       | .        |        |         |
| Plans & Reqs                                                                                 | 12APR00  | .        | .        | .       | .        |        |         |
| Product Design                                                                               | 17APR00  | 13APR00  | 17APR00  | 13APR00 | 17APR00  |        |         |
| Product Design                                                                               | .        | 13APR00  | 17APR00  | 13APR00 | 17APR00  |        |         |
| Product Design                                                                               | 17APR00  | .        | .        | .       | .        |        |         |
| Test Plan                                                                                    | 17APR00  | 13APR00  | 17APR00  | 20APR00 | 24APR00  |        |         |
| Documentation                                                                                | 19APR00  | 18APR00  | 19APR00  | 08MAY00 | 09MAY00  |        |         |
| Documentation                                                                                | .        | 18APR00  | 19APR00  | 08MAY00 | 09MAY00  |        |         |
| Documentation                                                                                | 19APR00  | .        | .        | .       | .        |        |         |
| Code                                                                                         | 01MAY00  | 18APR00  | 01MAY00  | 18APR00 | 01MAY00  |        |         |
| Code                                                                                         | .        | 18APR00  | 01MAY00  | 18APR00 | 01MAY00  |        |         |
| Code                                                                                         | 01MAY00  | .        | .        | .       | .        |        |         |
| Test Data                                                                                    | 24APR00  | 18APR00  | 24APR00  | 25APR00 | 01MAY00  |        |         |
| Test Routines                                                                                | 24APR00  | 18APR00  | 24APR00  | 25APR00 | 01MAY00  |        |         |
| Test Product                                                                                 | 09MAY00  | 02MAY00  | 09MAY00  | 02MAY00 | 09MAY00  |        |         |
| Test Product                                                                                 | .        | 02MAY00  | 09MAY00  | 02MAY00 | 09MAY00  |        |         |
| Test Product                                                                                 | 09MAY00  | .        | .        | .       | .        |        |         |
| Finish                                                                                       | 10MAY00  | 10MAY00  | 10MAY00  | 10MAY00 | 10MAY00  |        |         |

## Example 2.3. Auxiliary Resources and Alternate Resources

This example illustrates the use of Auxiliary resources. In the earlier examples, the use of alternate resources enabled the allocation of either John or Chris to the programming tasks. Now, suppose that each of the programmers has a different tester, and whenever a particular programmer is scheduled for a given task, his tester also needs to allocate some part of his or her time, say 50 percent, to the same task. To model such a scenario, specify Tester1 and Tester2 as auxiliary resources for Chris and John, respectively. The Activity and Resource data sets are printed in Output 2.3.1. Unlike the earlier examples, all the activities are of fixed-duration.

**Output 2.3.1.** Auxiliary Resources: Input Data Sets

| Software Development<br>Alternate and Auxiliary Resources<br>Activity Data Set |                |     |     |    |    |         |       |      |         |         |
|--------------------------------------------------------------------------------|----------------|-----|-----|----|----|---------|-------|------|---------|---------|
| Obs                                                                            | Activity       | dur | act | s1 | s2 | progrmr | chris | john | Tester1 | Tester2 |
| 1                                                                              | Plans & Reqts  | 2   | 1   | 2  | 3  | 1       | .     | .    | .       | .       |
| 2                                                                              | Product Design | 3   | 2   | 4  | 5  | 1       | .     | .    | .       | .       |
| 3                                                                              | Test Plan      | 3   | 3   | 6  | 7  | .       | .     | .    | .       | .       |
| 4                                                                              | Documentation  | 3   | 4   | 9  | .  | 1       | .     | .    | .       | .       |
| 5                                                                              | Code           | 10  | 5   | 8  | .  | 1       | .     | .    | .       | .       |
| 6                                                                              | Test Data      | 5   | 6   | 8  | .  | .       | .     | .    | .       | .       |
| 7                                                                              | Test Routines  | 5   | 7   | 8  | .  | .       | .     | .    | .       | .       |
| 8                                                                              | Test Product   | 6   | 8   | 9  | .  | 1       | .     | .    | .       | .       |
| 9                                                                              | Finish         | 0   | 9   | .  | .  | .       | .     | .    | .       | .       |

| Software Development<br>Alternate and Auxiliary Resources<br>Resource Data Set |         |          |         |         |       |      |         |         |
|--------------------------------------------------------------------------------|---------|----------|---------|---------|-------|------|---------|---------|
| Obs                                                                            | per     | otype    | resid   | progrmr | Chris | John | Tester1 | Tester2 |
| 1                                                                              | .       | altrate  | progrmr | .       | 1     | 1    | .       | .       |
| 2                                                                              | .       | auxres   | Chris   | .       | .     | .    | 0.5     | .       |
| 3                                                                              | .       | auxres   | John    | .       | .     | .    | .       | 0.5     |
| 4                                                                              | 11APR00 | reslevel | .       | .       | 1     | 1    | 1.0     | 1.0     |

The following statements invoke PROC CPM with the appropriate data sets and resource variables. The resulting schedule is printed in Output 2.3.2. Note the auxiliary resources that have been included in the schedule corresponding to each primary resource: Tester1 whenever Chris is used, and Tester2 whenever John is allocated:

```
proc cpm data=softaux out=sftaux rsched=rsftaux resin=resaux
      date='11apr00'd interval=weekday resout=raux;
  act act;
  succ s1 s2;
  dur dur;
  res progrmr Chris John Tester1 Tester2 /
      obstype=otype
      period=per resid=resid
      multalt rschedid=Activity;
  id Activity;
run;
```

Output 2.3.2. Auxiliary Resources: Resource Schedule Data Set

| Software Development: Alternate and Auxiliary Resources |          |          |          |         |          |        |         |
|---------------------------------------------------------|----------|----------|----------|---------|----------|--------|---------|
| Resource Schedule Data Set                              |          |          |          |         |          |        |         |
| Activity                                                | act      | RESOURCE | DUR_TYPE | dur     | _WORK_   | R_RATE | S_START |
| Plans & Reqts                                           | 1        |          |          | 2       | .        | .      | 11APR00 |
| Plans & Reqts                                           | 1        | progrmr  | FIXED    | 2       | .        | 1.0    | .       |
| Plans & Reqts                                           | 1        | Tester1  | FIXED    | 2       | .        | 0.5    | 11APR00 |
| Plans & Reqts                                           | 1        | chris    | FIXED    | 2       | .        | 1.0    | 11APR00 |
| Product Design                                          | 2        |          |          | 3       | .        | .      | 13APR00 |
| Product Design                                          | 2        | progrmr  | FIXED    | 3       | .        | 1.0    | .       |
| Product Design                                          | 2        | Tester1  | FIXED    | 3       | .        | 0.5    | 13APR00 |
| Product Design                                          | 2        | chris    | FIXED    | 3       | .        | 1.0    | 13APR00 |
| Test Plan                                               | 3        |          |          | 3       | .        | .      | 13APR00 |
| Documentation                                           | 4        |          |          | 3       | .        | .      | 18APR00 |
| Documentation                                           | 4        | progrmr  | FIXED    | 3       | .        | 1.0    | .       |
| Documentation                                           | 4        | Tester2  | FIXED    | 3       | .        | 0.5    | 18APR00 |
| Documentation                                           | 4        | john     | FIXED    | 3       | .        | 1.0    | 18APR00 |
| Code                                                    | 5        |          |          | 10      | .        | .      | 18APR00 |
| Code                                                    | 5        | progrmr  | FIXED    | 10      | .        | 1.0    | .       |
| Code                                                    | 5        | Tester1  | FIXED    | 10      | .        | 0.5    | 18APR00 |
| Code                                                    | 5        | chris    | FIXED    | 10      | .        | 1.0    | 18APR00 |
| Test Data                                               | 6        |          |          | 5       | .        | .      | 18APR00 |
| Test Routines                                           | 7        |          |          | 5       | .        | .      | 18APR00 |
| Test Product                                            | 8        |          |          | 6       | .        | .      | 02MAY00 |
| Test Product                                            | 8        | progrmr  | FIXED    | 6       | .        | 1.0    | .       |
| Test Product                                            | 8        | Tester1  | FIXED    | 6       | .        | 0.5    | 02MAY00 |
| Test Product                                            | 8        | chris    | FIXED    | 6       | .        | 1.0    | 02MAY00 |
| Finish                                                  | 9        |          |          | 0       | .        | .      | 10MAY00 |
| Activity                                                | S_FINISH | E_START  | E_FINISH | L_START | L_FINISH |        |         |
| Plans & Reqts                                           | 12APR00  | 11APR00  | 12APR00  | 11APR00 | 12APR00  |        |         |
| Plans & Reqts                                           | .        | 11APR00  | 12APR00  | 11APR00 | 12APR00  |        |         |
| Plans & Reqts                                           | 12APR00  | .        | .        | .       | .        |        |         |
| Plans & Reqts                                           | 12APR00  | .        | .        | .       | .        |        |         |
| Product Design                                          | 17APR00  | 13APR00  | 17APR00  | 13APR00 | 17APR00  |        |         |
| Product Design                                          | .        | 13APR00  | 17APR00  | 13APR00 | 17APR00  |        |         |
| Product Design                                          | 17APR00  | .        | .        | .       | .        |        |         |
| Product Design                                          | 17APR00  | .        | .        | .       | .        |        |         |
| Test Plan                                               | 17APR00  | 13APR00  | 17APR00  | 20APR00 | 24APR00  |        |         |
| Documentation                                           | 20APR00  | 18APR00  | 20APR00  | 05MAY00 | 09MAY00  |        |         |
| Documentation                                           | .        | 18APR00  | 20APR00  | 05MAY00 | 09MAY00  |        |         |
| Documentation                                           | 20APR00  | .        | .        | .       | .        |        |         |
| Documentation                                           | 20APR00  | .        | .        | .       | .        |        |         |
| Code                                                    | 01MAY00  | 18APR00  | 01MAY00  | 18APR00 | 01MAY00  |        |         |
| Code                                                    | .        | 18APR00  | 01MAY00  | 18APR00 | 01MAY00  |        |         |
| Code                                                    | 01MAY00  | .        | .        | .       | .        |        |         |
| Code                                                    | 01MAY00  | .        | .        | .       | .        |        |         |
| Test Data                                               | 24APR00  | 18APR00  | 24APR00  | 25APR00 | 01MAY00  |        |         |
| Test Routines                                           | 24APR00  | 18APR00  | 24APR00  | 25APR00 | 01MAY00  |        |         |
| Test Product                                            | 09MAY00  | 02MAY00  | 09MAY00  | 02MAY00 | 09MAY00  |        |         |
| Test Product                                            | .        | 02MAY00  | 09MAY00  | 02MAY00 | 09MAY00  |        |         |
| Test Product                                            | 09MAY00  | .        | .        | .       | .        |        |         |
| Test Product                                            | 09MAY00  | .        | .        | .       | .        |        |         |
| Finish                                                  | 10MAY00  | 10MAY00  | 10MAY00  | 10MAY00 | 10MAY00  |        |         |

# Chapter 3

## The DTREE Procedure

### Chapter Table of Contents

---

|                                |    |
|--------------------------------|----|
| <b>OVERVIEW</b> . . . . .      | 55 |
| <b>SYNTAX</b> . . . . .        | 55 |
| PROC DTREE Statement . . . . . | 55 |



## Chapter 3

# The DTREE Procedure

---

### Overview

Two new options have been added to the DTREE procedure. These options enable you to specify whether or not to display the page number in the top right corner of each page of a multipage decision tree diagram.

---

### Syntax

#### PROC DTREE Statement

The following new options are available in the PROC DTREE statement. These options are valid only for graphics-quality decision tree diagrams and can also be specified in the TREEPLOT statement.

**PAGENUM | NOPAGENUM**

**PAGENUMBER | NOPAGENUMBER**

specifies whether the page numbers should be displayed in the top right corner of each page of a multipage decision tree diagram. If the NOPAGENUM is not specified, the pages are ordered from top to bottom, left to right.

The default is PAGENUM.





# Chapter 4

## The GANTT Procedure

### Chapter Table of Contents

---

|                           |    |
|---------------------------|----|
| <b>OVERVIEW</b> . . . . . | 59 |
| <b>SYNTAX</b> . . . . .   | 59 |
| CHART Statement . . . . . | 59 |



## Chapter 4

# The GANTT Procedure

---

### Overview

A new option to suppress the numbering of pages of a Gantt chart has been added to the GANTT procedure.

---

### Syntax

---

#### CHART Statement

##### **NOPAGENUM**

suppresses numbering the pages of a multipage Gantt chart. This is the default behavior. To number the pages of a multipage Gantt chart on the upper right hand corner of each page, use the PAGENUM option.



# Chapter 5

## The INTPOINT Procedure

### Chapter Table of Contents

---

|                                  |     |
|----------------------------------|-----|
| <b>OVERVIEW</b>                  | 63  |
| Mathematical Description of NPSC | 64  |
| Mathematical Description of LP   | 66  |
| The Interior Point Algorithm     | 66  |
| Network Models                   | 74  |
| <b>INTRODUCTION</b>              | 82  |
| Getting Started: NPSC Problems   | 82  |
| Getting Started: LP Problems     | 89  |
| Typical PROC INTPOINT Run        | 97  |
| Functional Summary               | 99  |
| <b>SYNTAX</b>                    | 102 |
| PROC INTPOINT Statement          | 104 |
| CAPACITY Statement               | 123 |
| COEF Statement                   | 123 |
| COLUMN Statement                 | 124 |
| COST Statement                   | 124 |
| DEMAND Statement                 | 125 |
| HEADNODE Statement               | 125 |
| ID Statement                     | 125 |
| LO Statement                     | 126 |
| NAME Statement                   | 126 |
| NODE Statement                   | 127 |
| QUIT Statement                   | 127 |
| RHS Statement                    | 127 |
| ROW Statement                    | 127 |
| RUN Statement                    | 128 |
| SUPDEM Statement                 | 128 |
| SUPPLY Statement                 | 128 |
| TAILNODE Statement               | 129 |
| TYPE Statement                   | 129 |
| VAR Statement                    | 131 |
| <b>DETAILS</b>                   | 132 |
| Input Data Sets                  | 132 |

|                                                                         |            |
|-------------------------------------------------------------------------|------------|
| Output Data Set . . . . .                                               | 142        |
| Case Sensitivity . . . . .                                              | 143        |
| Loop Arcs . . . . .                                                     | 144        |
| Multiple Arcs . . . . .                                                 | 144        |
| Flow and Value Bounds . . . . .                                         | 144        |
| Tightening Bounds and Side Constraints . . . . .                        | 145        |
| Reasons for Infeasibility . . . . .                                     | 145        |
| Missing S Supply and Missing D Demand Values . . . . .                  | 146        |
| Balancing Total Supply and Total Demand . . . . .                       | 151        |
| How to Make the Data Read of PROC INTPOINT More Efficient . . . . .     | 153        |
| Stopping Criteria . . . . .                                             | 158        |
| <b>EXAMPLES . . . . .</b>                                               | <b>162</b> |
| Example 5.1 Production, Inventory, Distribution Problem . . . . .       | 163        |
| Example 5.2 Altering Arc Data . . . . .                                 | 167        |
| Example 5.3 Adding Side Constraints . . . . .                           | 172        |
| Example 5.4 Using Constraints and More Alteration to Arc Data . . . . . | 177        |
| Example 5.5 Nonarc Variables in the Side Constraints . . . . .          | 181        |
| Example 5.6 Solving an LP Problem with Data in MPS Format . . . . .     | 186        |
| <b>REFERENCES . . . . .</b>                                             | <b>188</b> |

# Chapter 5

## The INTPOINT Procedure

---

### Overview

The INTPOINT procedure solves the Network Program with Side Constraints (NPSC) problem (defined in the “Mathematical Description of NPSC” section on page 64) and the more general Linear Programming (LP) problem (defined in the “Mathematical Description of LP” section on page 66).

NPSC and LP models can be used to describe a wide variety of real-world applications ranging from production, inventory, and distribution problems to financial applications.

Whether your problem is NPSC or LP, PROC INTPOINT uses the same optimization algorithm, the Interior Point algorithm. This algorithm is outlined in the section “The Interior Point Algorithm” on page 66.

While many of your problems may best be formulated as LP problems, there may be other instances when your problems are better formulated as NPSC problems. The “Network Models” section on page 74 describes typical models that have a network component and suggests reasons why NPSC may be preferable to LP. The “Getting Started: NPSC Problems” section on page 82 outlines how you supply data of any NPSC problem to the PROC INTPOINT and call the procedure. After it reads the NPSC data, PROC INTPOINT converts the problem into an equivalent LP problem, performs Interior Point optimization, then converts the solution it finds back into a form you can use as the optimum to the original NPSC model.

If your model is an LP problem, the way you supply the data to PROC INTPOINT and run the procedure is described in the “Getting Started: LP Problems” section on page 89.

The remainder of this chapter is organized as follows:

- The “Typical PROC INTPOINT Run” section on page 97 describes how to use this procedure.
- The “Functional Summary” section on page 99 lists the statements and options that can be used to control PROC INTPOINT.
- The “Syntax” section on page 102 describes all the statements and options of PROC INTPOINT.
- The “Details” section on page 132 contains fuller explanations, descriptions, and advice on the use and behavior of the procedure.
- PROC INTPOINT is demonstrated by solving several examples in the “Examples” section on page 162.
- The “References” section on page 188 concludes the chapter.



## Mathematical Description of NPSC

A network consists of a collection of nodes joined by a collection of arcs. The arcs connect nodes and convey flow of one or more commodities that are supplied at supply nodes and demanded at demand nodes in the network. Each arc has a cost per unit of flow, a flow capacity, and a lower flow bound associated with it. An important concept in network modeling is *conservation of flow*. Conservation of flow means that the total flow in arcs directed toward a node, plus the supply at the node, minus the demand at the node, equals the total flow in arcs directed away from the node.

Often all the details of a problem cannot be specified in a network model alone. In many of these cases, these details can be represented by the addition of side constraints to the model. Side constraints are linear functions of arc variables (variables containing flow through an arc) and nonarc variables (variables that are not part of the network). The data for a side constraint consist of coefficients of arcs and coefficients of nonarc variables, a constraint type (that is,  $\leq$ ,  $=$ , or  $\geq$ ) and a right-hand-side value (rhs). A nonarc variable has a name, an objective function coefficient analogous to an arc cost, an upper bound analogous to an arc capacity, and a lower bound analogous to an arc lower flow bound.

If a network component of NPSC is removed by merging arcs and nonarc variables into a single set of variables, and if the flow conservation constraints and side constraints are merged into a single set of constraints, the result is an LP problem. PROC INTPOINT will automatically transform an NPSC problem into an equivalent LP problem, perform the optimization, then transform the problem back into its original form. By doing this, PROC INTPOINT finds the flow through the network and the values of any nonarc variables that minimize the total cost of the solution. Flow conservation is met, flow through each arc is on or between the arc's lower flow bound and capacity, the value of each nonarc variable is on or between the nonarc's lower and upper bounds, and the side constraints are satisfied.

Note that, since many LPs have large embedded networks, PROC INTPOINT is an attractive alternative to the LP procedure in many cases. Rather than formulating all problems as LPs, network models remain conceptually easy since they are based on network diagrams that represent the problem pictorially. PROC INTPOINT accepts the network specification in a format that is particularly suited to networks. This not only simplifies problem description but also aids in the interpretation of the solution. The conversion to and from the equivalent LP is done "behind the scenes" by the procedure.

If a network programming problem with side constraints has  $n$  nodes,  $a$  arcs,  $g$  nonarc variables, and  $k$  side constraints, then the formal statement of the problem solved by PROC INTPOINT is

$$\min\{c^T x + d^T z\}$$

$$\begin{aligned}
\text{subject to} \quad & Fx = b \\
& Hx + Qz \geq, =, \leq r \\
& l \leq x \leq u \\
& m \leq z \leq v
\end{aligned}$$

where

$c$  is the  $a \times 1$  objective function coefficient of the arc variables vector (the cost vector)

$x$  is the  $a \times 1$  arc variable value vector (the flow vector)

$d$  is the  $g \times 1$  objective function coefficient of the nonarc variables vector

$z$  is the  $g \times 1$  nonarc variable value vector

$F$  is the  $n \times a$  node-arc incidence matrix of the network, where

$$\begin{aligned}
F_{i,j} &= -1 && \text{if arc } j \text{ is directed from node } i \\
F_{i,j} &= 1 && \text{if arc } j \text{ is directed toward node } i \\
F_{i,j} &= 0 && \text{otherwise}
\end{aligned}$$

$b$  is the  $n \times 1$  node supply/demand vector, where

$$\begin{aligned}
b_i &= s && \text{if node } i \text{ has supply capability of } s \text{ units of flow} \\
b_i &= -d && \text{if node } i \text{ has demand } d \text{ of units of flow} \\
b_i &= 0 && \text{if node } i \text{ is a transshipment node}
\end{aligned}$$

$H$  is the  $k \times a$  side constraint coefficient matrix for arc variables, where  $H_{i,j}$  is the coefficient of arc  $j$  in the  $i$ th side constraint

$Q$  is the  $k \times g$  side constraint coefficient matrix for nonarc variables, where  $Q_{i,j}$  is the coefficient of nonarc  $j$  in the  $i$ th side constraint

$r$  is the  $k \times 1$  side constraint right-hand-side vector

$l$  is the  $a \times 1$  arc lower flow bound vector

$u$  is the  $a \times 1$  arc capacity vector

$m$  is the  $g \times 1$  nonarc variable value lower bound vector

$v$  is the  $g \times 1$  nonarc variable value upper bound vector

The INTPOINT procedure can also be used to solve an unconstrained network problem, that is, one in which  $H$ ,  $Q$ ,  $d$ ,  $r$ , and  $z$  do not exist.

The INTPOINT procedure can also be used to solve a network problem with side constraints but no nonarc variables, in which case  $Q$ ,  $d$ , and  $z$  do not exist.

---

## Mathematical Description of LP

PROC INTPOINT solves LP problems. These have a linear objective function and a collection of linear constraints. PROC INTPOINT finds the values of variables that minimize the total cost of the solution. The value of each variable is on or between the variable's lower and upper bounds, and the constraints are satisfied.

If an LP has  $g$  variables and  $k$  constraints, then the formal statement of the problem solved by PROC INTPOINT is

$$\begin{array}{ll} & \min\{d^T z\} \\ \text{subject to} & Qz \geq, =, \leq r \\ & m \leq z \leq v \end{array}$$

where

$d$  is the  $g \times 1$  objective function coefficient of the variables vector

$z$  is the  $g \times 1$  variable value vector

$Q$  is the  $k \times g$  constraint coefficient matrix for the variables, where  $Q_{i,j}$  is the coefficient of variable  $j$  in the  $i$ th constraint

$r$  is the  $k \times 1$  side constraint right-hand-side vector

$m$  is the  $g \times 1$  variable value lower bound vector

$v$  is the  $g \times 1$  variable value upper bound vector

---

## The Interior Point Algorithm

The Simplex algorithm, developed shortly after World War II, was for many years the main method used to solve Linear Programming problems. Over the last fifteen years, however, the Interior Point algorithm has been developed. This algorithm also solves Linear Programming problems. From the start it showed great theoretical promise, and considerable research in the area resulted in practical implementations that performed competitively with the Simplex algorithm. More recently, Interior Point algorithms have evolved to become superior to the Simplex algorithm, in general, especially when the problems are large.

There are many variations of Interior Point algorithms. PROC INTPOINT uses the Primal-Dual with Predictor-Corrector algorithm. More information on this particular algorithm and related theory can be found in the texts by Roos, Terlaky, and Vial (1997), Wright (1996), and Ye (1996).

### Interior Point Algorithmic Details

After preprocessing, the Linear Program to be solved is

$$\min\{c^T x\}$$

$$\begin{aligned} \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

This is the *primal* problem. The matrices of  $d$ ,  $z$ , and  $Q$  of NPSC have been renamed  $c$ ,  $x$ , and  $A$ , respectively, as these symbols are by convention used more, the problem to be solved is different from the original because of preprocessing, and there has been a change of primal variable to transform the LP into one whose variables have zero lower bounds. To simplify the algebra here, assume that variables have infinite upper bounds, and constraints are equalities. (Interior Point algorithms do efficiently handle finite upper bounds, and it is easy to introduce primal slack variables to change inequalities into equalities.) The problem has  $n$  variables.  $i$  is a variable number.  $k$  is an iteration number, and if used as a subscript or superscript it denotes “of iteration  $k$ ”.

There exists an equivalent problem, the *dual* problem, stated as

$$\begin{aligned} & \max \{b^T y\} \\ \text{subject to} \quad & A^T y + s = c \\ & s \geq 0 \\ \text{where} \quad & y \text{ are dual variables, and } s \text{ are dual constraint slacks} \end{aligned}$$

What the Interior Point has to do is to solve the system of equations to satisfy the Karush-Kuhn-Tucker (KKT) conditions for optimality:

$$\begin{aligned} & Ax = b \\ & A^T y + s = c \\ & XSe = 0 \\ & x \geq 0 \\ & s \geq 0 \\ \text{where} \quad & S = \text{diag}(s), \text{ (that is, } S_{i,j} = s_i \text{ if } i = j, S_{i,j} = 0 \text{ otherwise)} \\ & X = \text{diag}(x), \text{ and} \\ & e_i = 1 \forall i \end{aligned}$$

These are the conditions for feasibility, with the *complementarity* condition  $XSe = 0$  added.  $c^T x = b^T y$  must occur at the optimum. Complementarity forces the optimal objectives of the primal and dual to be equal,  $c^T x_{opt} = b^T y_{opt}$ , as

$$\begin{aligned} 0 &= x_{opt}^T s_{opt} = s_{opt}^T x_{opt} = (c - A^T y_{opt})^T x_{opt} = \\ &= c^T x_{opt} - y_{opt}^T (Ax_{opt}) = c^T x_{opt} - b^T y_{opt} \end{aligned}$$

therefore  $0 = c^T x_{opt} - b^T y_{opt}$

Before the optimum is reached, a solution  $(x, y, s)$  may not satisfy the KKT conditions:

- Primal constraints can be broken,  $infeas_c = b - Ax \neq 0$ .
- Dual constraints can be broken,  $infeas_d = c - A^T y - s \neq 0$ .
- Complementarity is unsatisfied,  $x^T s = c^T x - b^T y \neq 0$ . This is called the *duality gap*.

The Interior Point algorithm works by using Newton's method to find a direction to move  $(\Delta x^k, \Delta y^k, \Delta s^k)$  from the current solution  $(x^k, y^k, s^k)$  toward a better solution:

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

$\alpha$  is the *step length* and is assigned a value as large as possible and not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  is “too close” to zero. The direction in which to move is found using

$$\begin{aligned} A\Delta x^k &= infeas_c \\ A^T \Delta y^k + \Delta s^k &= infeas_d \\ S^k \Delta x^k + X^k \Delta s^k &= -X^k S^k e \end{aligned}$$

To greatly improve performance, the third equation is changed to

$$S^k \Delta x^k + X^k \Delta s^k = -X^k S^k e + \sigma_k \mu_k e$$

where  $\mu_k = (x^k)^T s^k / n$ , the average complementarity, and  $0 \leq \sigma_k \leq 1$

The effect now is to find a direction in which to move to reduce infeasibilities and to reduce the complementarity toward zero, but if any  $x_i^k s_i^k$  is too close to zero, it is “nudged out” to  $\mu$ , and any  $x_i^k s_i^k$  that is larger than  $\mu$  is “nudged into”  $\mu$ . A  $\sigma_k$  close to or equal to 0.0 biases a direction toward the optimum, and a value for  $\sigma_k$  close to or equal to 1.0 “centers” the direction toward a point where all pairwise products  $x_i^k s_i^k = \mu$ . Such points make up the *Central Path* in the interior. Although centering directions make little, if any, progress in reducing  $\mu$  and moving the solution closer to the optimum, substantial progress toward the optimum can usually be made in the next iteration.

The Central Path is crucial to why the Interior Point algorithm is so efficient. As  $\mu$  is decreased, this path “guides” the algorithm to the optimum through the interior of feasible space. Without centering, the algorithm would find a series of solutions near each other close to the boundary of feasible space. Step lengths along the direction would be small and many more iterations would probably be required to reach the optimum.

That in a nutshell is the Primal-Dual Interior Point algorithm. Varieties of the algorithm differ in the way  $\alpha$  and  $\sigma_k$  are chosen and the direction adjusted during each iteration. A wealth of information can be found in the texts by Roos, Terlaky, and Vial (1997), Wright (1996), and Ye (1996).

The calculation of the direction is the most time-consuming step of the Interior Point algorithm. Assume the  $k$ th iteration is being performed, so the subscript and superscript  $k$  can be dropped from the algebra:

$$\begin{aligned} A\Delta x &= \text{infeas}_c \\ A^T \Delta y + \Delta s &= \text{infeas}_d \\ S\Delta x + X\Delta s &= -XSe + \sigma\mu e \end{aligned}$$

Rearranging the second equation

$$\Delta s = \text{infeas}_d - A^T \Delta y$$

Rearranging the third equation

$$\begin{aligned} \Delta s &= X^{-1}(-S\Delta x - XSe + \sigma\mu e) \\ \Delta s &= -\Theta\Delta x - Se + X^{-1}\sigma\mu e \end{aligned}$$

where

$$\Theta = SX^{-1}$$

Equating these two expressions for  $\Delta s$  and rearranging

$$\begin{aligned} -\Theta\Delta x - Se + X^{-1}\sigma\mu e &= \text{infeas}_d - A^T \Delta y \\ -\Theta\Delta x &= Se - X^{-1}\sigma\mu e + \text{infeas}_d - A^T \Delta y \\ \Delta x &= \Theta^{-1}(-Se + X^{-1}\sigma\mu e - \text{infeas}_d + A^T \Delta y) \\ \Delta x &= \rho + \Theta^{-1}A^T \Delta y \end{aligned}$$

where

$$\rho = \Theta^{-1}(-Se + X^{-1}\sigma\mu e - \text{infeas}_d)$$

Substituting into the first direction equation

$$\begin{aligned} A\Delta x &= \text{infeas}_c \\ A(\rho + \Theta^{-1}A^T \Delta y) &= \text{infeas}_c \\ A\Theta^{-1}A^T \Delta y &= \text{infeas}_c - A\rho \\ \Delta y &= (A\Theta^{-1}A^T)^{-1}(\text{infeas}_c - A\rho) \end{aligned}$$

$\Theta$ ,  $\rho$ ,  $\Delta y$ ,  $\Delta x$ , and  $\Delta s$  are calculated in that order. The hardest term is the factorization of the  $(A\Theta^{-1}A^T)$  matrix to determine  $\Delta y$ . Fortunately, although the *values* of  $(A\Theta^{-1}A^T)$  are different for each iteration, the *locations* of the nonzeros in this matrix remain fixed; the nonzero locations are the same as those in the matrix  $(AA^T)$ . This is because  $\Theta^{-1} = XS^{-1}$  is a diagonal matrix that has the effect of merely scaling the columns of  $(AA^T)$ .

The fact that the nonzeros in  $A\Theta^{-1}A^T$  have a constant pattern is exploited by all Interior Point algorithms and is a major reason for their excellent performance. Before iterations begin,  $AA^T$  is examined and its rows and columns are symmetrically permuted so that during Cholesky factorization, the number of *fillins* created is smaller. A list of arithmetic operations to perform the factorization is saved in concise computer data structures (working with memory locations rather than actual numerical values). This is called *symbolic factorization*. During iterations, when memory has been initialized with numerical values, the operations list is performed sequentially. Determining how the factorization should be performed again and again is unnecessary.

### The Primal-Dual Predictor-Corrector Interior Point Algorithm

The variant of the Interior Point algorithm implemented in PROC INTPOINT is a Primal-Dual Predictor-Corrector Interior Point algorithm. At first, Newton's method is used to find a direction  $(\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k)$  to move, but calculated as if  $\mu$  is zero, that is, as a step with no centering, known as an *affine* step:

$$\begin{aligned} A\Delta x_{aff}^k &= infeas_c \\ A^T\Delta y_{aff}^k + \Delta s_{aff}^k &= infeas_d \\ S^k\Delta x_{aff}^k + X^k\Delta s_{aff}^k &= -X^kS^ke \end{aligned}$$

$$(x_{aff}^k, y_{aff}^k, s_{aff}^k) = (x^k, y^k, s^k) + \alpha(\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k)$$

$\alpha$  is the *step length* as before.

Complementarity  $x^Ts$  is calculated at  $(x_{aff}^k, y_{aff}^k, s_{aff}^k)$  and compared with the complementarity at the starting point  $(x^k, y^k, s^k)$ , and the success of the affine step is gauged. If the affine step was successful in reducing the complementarity by a substantial amount, the need for centering is not great, and  $\sigma_k$  in the following linear system is assigned a value close to zero. If, however, the affine step was unsuccessful, centering would be beneficial, and  $\sigma_k$  in the following linear system is assigned a value closer to 1.0. The value of  $\sigma_k$  is therefore adaptively altered depending on the progress made toward the optimum.

A second linear system is solved to determine a centering vector  $(\Delta x_c^k, \Delta y_c^k, \Delta s_c^k)$  from  $(x_{aff}^k, y_{aff}^k, s_{aff}^k)$ :

$$\begin{aligned} A\Delta x_c^k &= 0 \\ A^T\Delta y_c^k + \Delta s_c^k &= 0 \\ S^k\Delta x_c^k + X^k\Delta s_c^k &= -X_{aff}^kS_{aff}^ke + \sigma_k\mu_k e \end{aligned}$$

then

$$(\Delta x^k, \Delta y^k, \Delta s^k) = (\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k) + (\Delta x_c^k, \Delta y_c^k, \Delta s_c^k)$$

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

where, as before,  $\alpha$  is the *step length* assigned a value as large as possible but not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  is “too close” to zero.

Although the Predictor-Corrector variant entails solving two linear systems instead of one, fewer iterations are usually required to reach the optimum. The additional overhead of calculating the second linear system is small, as the factorization of the  $(A\Theta^{-1}A^T)$  matrix has already been performed to solve the first linear system.

### **Interior Point: Upper Bounds**

If the LP had upper bounds ( $0 \leq x \leq u$  where  $u$  is the upper bound vector), then the primal and dual problems, the duality gap, and the KKT conditions would have to be expanded.

The primal Linear Program to be solved is

$$\begin{aligned} & \min \{c^T x\} \\ \text{subject to} \quad & Ax = b \\ & 0 \leq x \leq u \end{aligned}$$

$0 \leq x \leq u$  is split into  $x \geq 0$  and  $x \leq u$ . Let  $z$  be primal slack so that  $x + z = u$ , and associate dual variables  $w$  with these constraints. The Interior Point solves the system of equations to satisfy the Karush-Kuhn-Tucker (KKT) conditions for optimality:

$$\begin{aligned} Ax &= b \\ x + z &= u \\ A^T y + s - w &= c \\ XSe &= 0 \\ ZWe &= 0 \\ x, s, z, w &\geq 0 \end{aligned}$$

These are the conditions for feasibility, with the *complementarity* conditions  $XSe = 0$  and  $ZWe = 0$  added.  $c^T x = b^T y - u^T w$  must occur at the optimum. Complementarity forces the optimal objectives of the primal and dual to be equal,  $c^T x_{opt} = b^T y_{opt} - u^T w_{opt}$ , as



$$0 = z_{opt}^T w_{opt} = (u - x_{opt})^T w_{opt} = u^T w_{opt} - x_{opt}^T w_{opt}$$

$$0 = x_{opt}^T s_{opt} = s_{opt}^T x_{opt} = (c - A^T y_{opt} + w_{opt})^T x_{opt} =$$

$$c^T x_{opt} - y_{opt}^T (A x_{opt}) + w_{opt}^T x_{opt} = c^T x_{opt} - b^T y_{opt} + u^T w_{opt}$$

therefore  $0 = c^T x_{opt} - b^T y_{opt} + u^T w_{opt}$

Before the optimum is reached, a solution  $(x, y, s, z, w)$  might not satisfy the KKT conditions:

- Primal bound constraints can be broken,  $infeas_b = u - x - z \neq 0$ .
- Primal constraints can be broken,  $infeas_c = b - Ax \neq 0$ .
- Dual constraints can be broken,  $infeas_d = c - A^T y - s + w \neq 0$ .
- Complementarity conditions are unsatisfied,  $x^T s \neq 0$  and  $z^T w \neq 0$ .

The calculations of the Interior point algorithm can easily be derived in a fashion similar to calculations for when an LP has no upper bounds. See the paper by Lustig, Marsten, and Shanno (1992).

In some iteration  $k$ , the *affine* step system that must be solved is:

$$\begin{aligned}\Delta x_{aff} + \Delta z_{aff} &= infeas_b \\ A\Delta x_{aff} &= infeas_c \\ A^T \Delta y_{aff} + \Delta s_{aff} - \Delta w_{aff} &= infeas_d \\ S\Delta x_{aff} + X\Delta s_{aff} &= -XSe \\ Z\Delta w_{aff} + W\Delta z_{aff} &= -ZWe\end{aligned}$$

Therefore, the computations involved in solving the affine step are:

$$\begin{aligned}\Theta &= SX^{-1} + WZ^{-1} \\ \rho &= \Theta^{-1}(infeas_d + (S - W)e - Z^{-1}Winfeas_b) \\ \Delta y_{aff} &= (A\Theta^{-1}A^T)^{-1}(infeas_c + A\rho) \\ \Delta x_{aff} &= \Theta^{-1}A^T \Delta y_{aff} - \rho \\ \Delta z_{aff} &= infeas_b - \Delta x_{aff} \\ \Delta w_{aff} &= -We - Z^{-1}W\Delta z_{aff} \\ \Delta s_{aff} &= -Se - X^{-1}S\Delta x_{aff} \\ (x_{aff}, y_{aff}, s_{aff}, z_{aff}, w_{aff}) &= (x, y, s, z, w) + \\ &\alpha(\Delta x_{aff}, \Delta y_{aff}, \Delta s_{aff}, \Delta z_{aff}, \Delta w_{aff})\end{aligned}$$

$\alpha$  is the *step length* as before.

A second linear system is solved to determine a centering vector:  $(\Delta x_c, \Delta y_c, \Delta s_c, \Delta z_c, \Delta w_c)$  from  $(x_{aff}, y_{aff}, s_{aff}, z_{aff}, w_{aff})$ :

$$\begin{aligned}\Delta x_c + \Delta z_c &= 0 \\ A\Delta x_c &= 0 \\ A^T \Delta y_c + \Delta s_c - \Delta w_c &= 0 \\ S\Delta x_c + X\Delta s_c &= -X_{aff}S_{aff}e + \sigma\mu e \\ Z\Delta w_c + W\Delta z_c &= -Z_{aff}W_{aff}e + \sigma\mu e\end{aligned}$$

where

$$\begin{aligned}\zeta_{start} &= x^T s + z^T w, \text{ complementarity at the start of the iteration} \\ \zeta_{aff} &= x_{aff}^T s_{aff} + z_{aff}^T w_{aff}, \text{ the affine complementarity} \\ \mu &= \zeta_{aff}/2n, \text{ the average complementarity} \\ \sigma &= (\zeta_{aff}/\zeta_{start})^3\end{aligned}$$

Therefore, the computations involved in solving the centering step are:

$$\begin{aligned}\rho &= \Theta^{-1}(\sigma\mu(X^{-1} - Z^{-1})e - X^{-1}X_{aff}S_{aff}e + Z^{-1}Z_{aff}W_{aff}e) \\ \Delta y_c &= (A\Theta^{-1}A^T)^{-1}A\rho \\ \Delta x_c &= \Theta^{-1}A^T\Delta y_c - \rho \\ \Delta z_c &= -\Delta x_c \\ \Delta w_c &= \sigma\mu Z^{-1}e - Z^{-1}Z_{aff}W_{aff}e - Z^{-1}W_{aff}\Delta z_c \\ \Delta s_c &= \sigma\mu X^{-1}e - X^{-1}X_{aff}S_{aff}e - X^{-1}S_{aff}\Delta x_c\end{aligned}$$

Then

$$\begin{aligned}(\Delta x, \Delta y, \Delta s, \Delta z, \Delta w) &= \\ (\Delta x_{aff}, \Delta y_{aff}, \Delta s_{aff}, \Delta z_{aff}, \Delta w_{aff}) &+ \\ (\Delta x_c, \Delta y_c, \Delta s_c, \Delta z_c, \Delta w_c) & \\ (x^{k+1}, y^{k+1}, s^{k+1}, z^{k+1}, w^{k+1}) &= \\ (x^k, y^k, s^k, z^k, w^k) &+ \\ \alpha(\Delta x, \Delta y, \Delta s, \Delta z, \Delta w) &\end{aligned}$$

where, as before,  $\alpha$  is the *step length* assigned a value as large as possible but not so large that a  $x_i^{k+1}$ ,  $s_i^{k+1}$ ,  $z_i^{k+1}$ , or  $w_i^{k+1}$  is “too close” to zero.

The algebra in this section has been simplified by assuming that *all* variables have finite upper bounds. If the number of variables with finite upper bounds  $n_u < n$ , you need to change the algebra to reflect that the “z” and “w” arrays has dimension  $n_u \times 1$  or  $n_u \times n_u$ . Other computations need slight modification. For example, the average complementarity

$$\mu = x_{aff}^T s_{aff} / n + z_{aff}^T w_{aff} / n_u$$

An important point is that any upper bounds can be handled by specializing the algorithm and *not* by generating the constraints  $x \leq u$  and adding these to the main primal constraints  $Ax = b$ .

## Network Models

The following are descriptions of some typical NPSC models.

### **Production, Inventory, and Distribution (Supply Chain) Problems**

One common class of network model is the production-inventory-distribution or supply-chain problem. The diagram in Figure 5.1 illustrates this problem. The subscript on the Production, Inventory, and Sales nodes indicates the time period. By replicating sections of the model, the notion of time can be included.

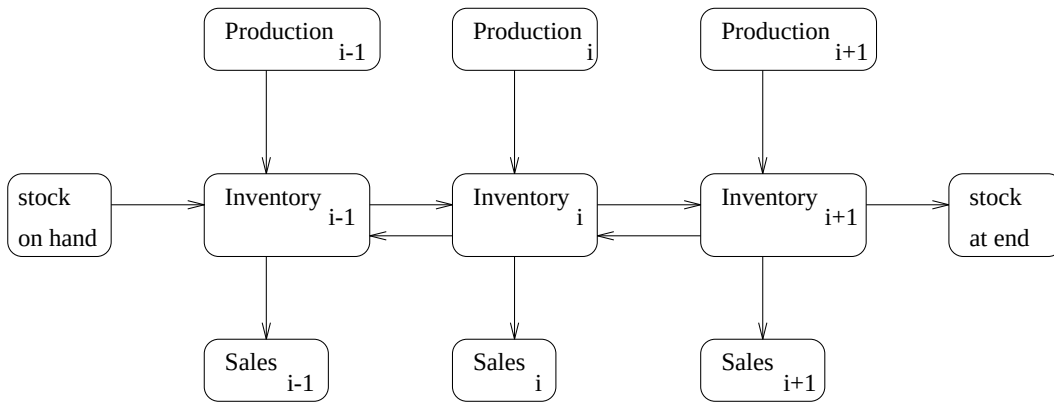


**Figure 5.1.** Production-Inventory-Distribution Problem

In this type of model, the nodes can represent a wide variety of facilities. Several examples are suppliers, spot markets, importers, farmers, manufacturers, factories, parts of a plant, production lines, waste disposal facilities, workstations, warehouses, coolstores, depots, wholesalers, export markets, ports, rail junctions, airports, road intersections, cities, regions, shops, customers, and consumers. The diversity of this selection demonstrates how rich the potential applications of this model are.

Depending upon the interpretation of the nodes, the objectives of the modeling exercise can vary widely. Some common types of objectives are

- to reduce collection or purchase costs of raw materials
- to reduce inventory holding or backorder costs. Warehouses and other storage facilities sometimes have capacities, and there can be limits on the amount of goods that can be placed on backorder.
- to decide where facilities should be located and what the capacity of these should be. Network models have been used to help decide where factories, hospitals, ambulance and fire stations, oil and water wells, and schools should be sited.

- to determine the assignment of resources (machines, production capability, workforce) to tasks, schedules, classes, or files
- to determine the optimal distribution of goods or services. This usually means minimizing transportation costs and reducing transit time or distances covered.
- to find the shortest path from one location to another
- to ensure that demands (for example, production requirements, market demands, contractual obligations) are met
- to maximize profits from the sale of products or the charge for services
- to maximize production by identifying bottlenecks

Some specific applications are

- car distribution models. These help determine which models and numbers of cars should be manufactured in which factories and where to distribute cars from these factories to zones in the United States in order to meet customer demand at least cost.
- models in the timber industry. These help determine when to plant and mill forests, schedule production of pulp, paper, and wood products, and distribute products for sale or export.
- military applications. The nodes can be theaters, bases, ammunition dumps, logistical suppliers, or radar installations. Some models are used to find the best ways to mobilize personnel and supplies and to evacuate the wounded in the least amount of time.
- communications applications. The nodes can be telephone exchanges, transmission lines, satellite links, and consumers. In a model of an electrical grid, the nodes can be transformers, powerstations, watersheds, reservoirs, dams, and consumers. The effect of high loads or outages might be of concern.

### ***Proportional Constraints***

In many models, you have the characteristic that a flow through an arc must be proportional to the flow through another arc. Side constraints are often necessary to model that situation. Such constraints are called *proportional constraints* and are useful in models where production is subject to refining or modification into different materials. The amount of each output, or any waste, evaporation, or reduction can be specified as a proportion of input.

Typically, the arcs near the supply nodes carry raw materials and the arcs near the demand nodes carry refined products. For example, in a model of the milling industry, the flow through some arcs may represent quantities of wheat. After the wheat is processed, the flow through other arcs might be flour. For others it might be bran. The side constraints model the relationship between the amount of flour or bran produced as a proportion of the amount of wheat milled. Some of the wheat can end up as neither flour, bran, nor any useful product, so this waste is drained away via arcs to a waste node.

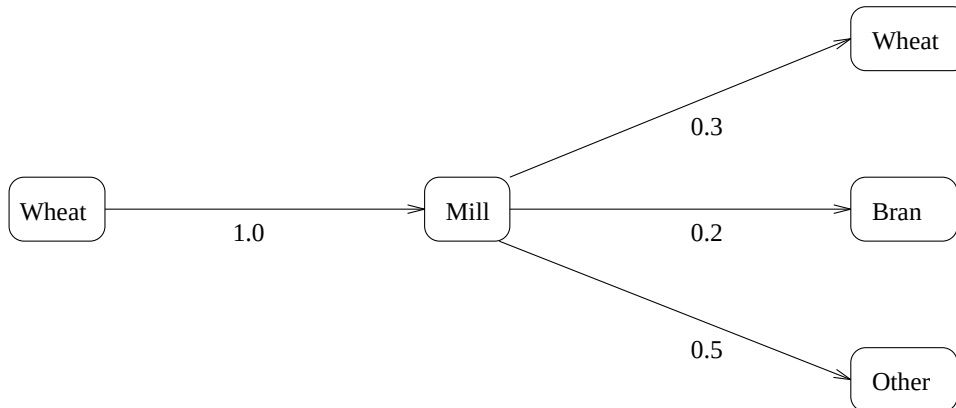


**Figure 5.2.** Proportional Constraints

In order for arcs to be specified in side constraints, they must be named. By default, PROC INTPOINT names arcs using the names of the nodes at the head and tail of the arc. An arc is named with its tail node name followed by an “\_” followed by the name of its head node name. For example, an arc from node *from* to node *to* is called *from\_to*.

Consider the network fragment in Figure 5.2. The arc *Wheat\_Mill* conveys the wheat milled. The cost of flow on this arc is the milling cost. The capacity of this arc is the capacity of the mill. The lower flow bound on this arc is the minimum quantity that must be milled for the mill to operate economically. The constraints

$$\begin{aligned} 0.3 \text{ Wheat\_Mill} - \text{Mill\_Flour} &= 0.0 \\ 0.2 \text{ Wheat\_Mill} - \text{Mill\_Bran} &= 0.0 \end{aligned}$$

force every unit of wheat that is milled to produce 0.3 units of flour and 0.2 units of bran. Note that it is not necessary to specify the constraint

$$0.5 \text{ Wheat\_Mill} - \text{Mill\_Other} = 0.0$$

since flow conservation implies that any flow that does not traverse through *Mill\_Flour* or *Mill\_Bran* must be conveyed through *Mill\_Other*. And, computationally, it is better if this constraint is not specified, since there is one less side constraint and fewer problems with numerical precision. Notice that the sum of the proportions must equal 1.0 exactly; otherwise, flow conservation is violated.

### Blending Constraints

Blending or quality constraints can also influence the recipes or proportions of ingredients that are mixed. For example, different raw materials can have different properties. In an application of the oil industry, the amount of products that are obtained could be different for each type of crude oil. Furthermore, fuel might have a minimum octane requirement or limited sulphur or lead content, so that a blending of crudes is needed to produce the product.

The network fragment in Figure 5.3 shows an example of this.

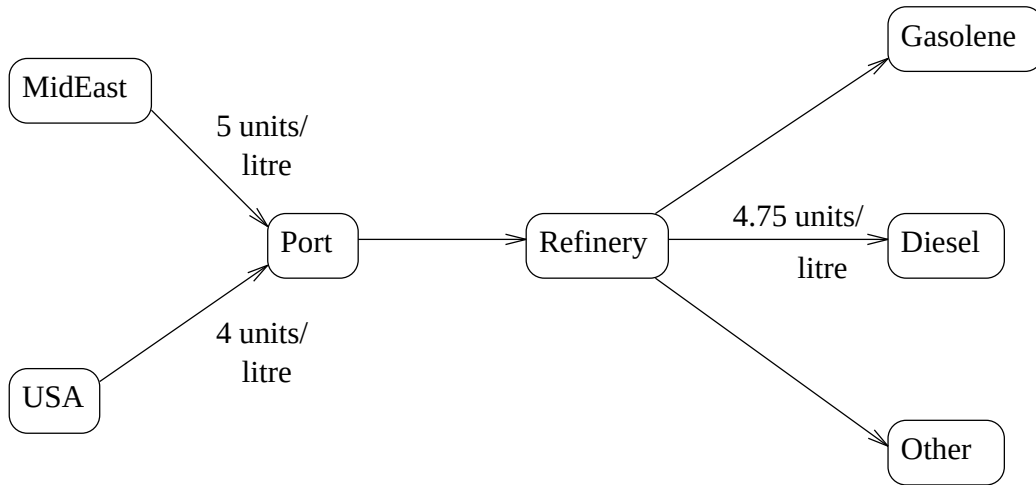


**Figure 5.3.** Blending Constraints

The arcs `MidEast_Port` and `USA_Port` convey crude oil from the two sources. The arc `Port_Refinery` represents refining while the arcs `Refinery_Gasolene` and `Refinery_Diesel` carry the gas and diesel produced. The proportional constraints

$$\begin{aligned} 0.4 \text{ Port\_Refinery} - \text{Refinery\_Gasolene} &= 0.0 \\ 0.2 \text{ Port\_Refinery} - \text{Refinery\_Diesel} &= 0.0 \end{aligned}$$

capture the restrictions for producing gasolene and diesel from crude. Suppose that only crude from the Middle East is used, then the resulting diesel would contain 5 units of sulphur per litre. If only crude from the U.S.A. is used, the resulting diesel would contain 4 units of sulphur per litre. Diesel can have at most 4.75 units of sulphur per litre. Some crude from the U.S.A. must be used if Middle East crude is used in order to meet the 4.75 sulphur per litre limit. The side constraint to model this requirement is

$$5 \text{ MidEast\_Port} + 4 \text{ USA\_Port} - 4.75 \text{ Port\_Refinery} \leq 0.0$$

Since  $\text{Port\_Refinery} = \text{MidEast\_Port} + \text{USA\_Port}$ , flow conservation allows this constraint to be simplified to

$$1 \text{ MidEast\_Port} - 3 \text{ USA\_Port} \leq 0.0$$

If, for example, 120 units of crude from the Middle East is used, then at least 40 units of crude from the U.S.A. must be used. The preceding constraint is simplified

because you assume that the sulphur concentration of diesel is proportional to the sulphur concentration of the crude mix. If this is not the case, the relation

$$0.2 \text{ Port\_Refinery} = \text{Refinery\_Diesel}$$

is used to obtain

$$5 \text{ MidEast\_Port} + 4 \text{ USA\_Port} - 4.75 (1.0/0.2 \text{ Refinery\_Diesel}) \leq 0.0$$

which equals

$$5 \text{ MidEast\_Port} + 4 \text{ USA\_Port} - 23.75 \text{ Refinery\_Diesel} \leq 0.0$$

An example similar to this oil industry problem is solved in the “Introductory NPSC Example” section on page 83.

### Multicommodity Problems

Side constraints are also used in models in which there are capacities on transportation or some other shared resource, or there are limits on overall production or demand in multicommodity, multidivisional, or multiperiod problems. Each commodity, division, or period can have a separate network coupled to one main system by the side constraints. Side constraints are used to combine the outputs of subdivisions of a problem (either commodities, outputs in distinct time periods, or different process streams) to meet overall demands or to limit overall production or expenditures. This method is more desirable than doing separate *local* optimizations for individual commodity, process, or time networks and then trying to establish relationships between each when determining an overall policy if the *global* constraint is not satisfied. Of course, to make models more realistic, side constraints may be necessary in the local problems.

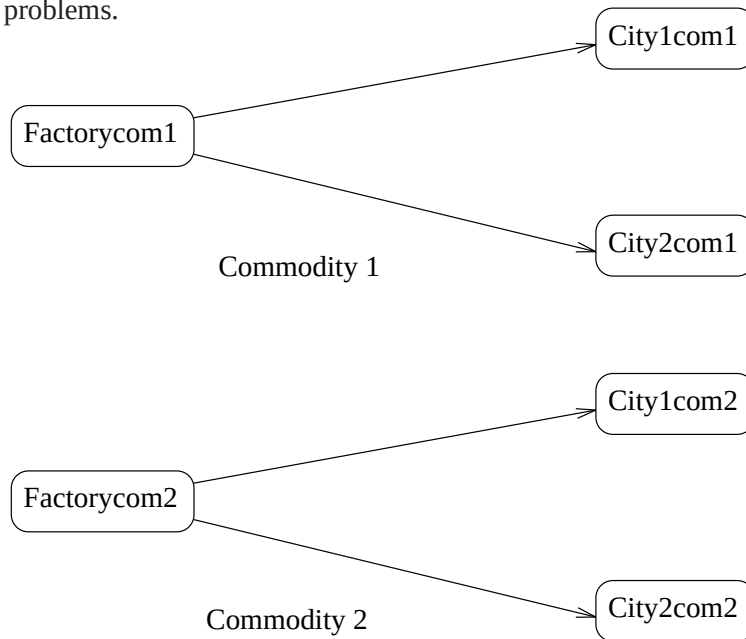


Figure 5.4. Multicommodity Problem

Figure 5.4 shows two network fragments. They represent identical production and distribution sites but of two different commodities. Suffix *com1* represents commodity 1 and suffix *com2* represents commodity 2. The nodes *Factorycom1* and *Factorycom2* model the same factory, and nodes *City1com1* and *City1com2* model the same location, city1. Similarly, *City2com1* and *City2com2* are the same location, city2. Suppose that commodity 1 occupies 2 cubic meters, commodity 2 occupies 3 cubic meters, the truck dispatched to city1 has a capacity of 200 cubic meters, and the truck dispatched to city2 has a capacity of 250 cubic meters. How much of each commodity can be loaded onto each truck? The side constraints for this case are

$$\begin{aligned} 2 \text{ Factorycom1\_City1com1} + 3 \text{ Factorycom2\_City1com2} &\leq 200 \\ 2 \text{ Factorycom1\_City2com1} + 3 \text{ Factorycom2\_City2com2} &\leq 250 \end{aligned}$$

### Large Modeling Strategy

In many cases, the flow through an arc might actually represent the flow or movement of a commodity from place to place or from time period to time period. However, sometimes an arc is included in the network as a method of capturing some aspect of the problem that you would not normally think of as part of a network model. There is no commodity movement associated with that arc. For example, in a multiprocess, multiproduct model (Figure 5.5), there might be subnetworks for each process and each product. The subnetworks can be joined together by a set of arcs that have flows that represent the amount of product *j* produced by process *i*. To model an upper-limit constraint on the total amount of product *j* that can be produced, direct all arcs carrying product *j* to a single node and from there through a single arc. The capacity of this arc is the upper limit of product *j* production. It is preferable to model this structure in the network rather than to include it in the side constraints because the efficiency of the optimizer may be less affected by a reasonable increase in the size of the network rather than increasing the number or complicating side constraints.

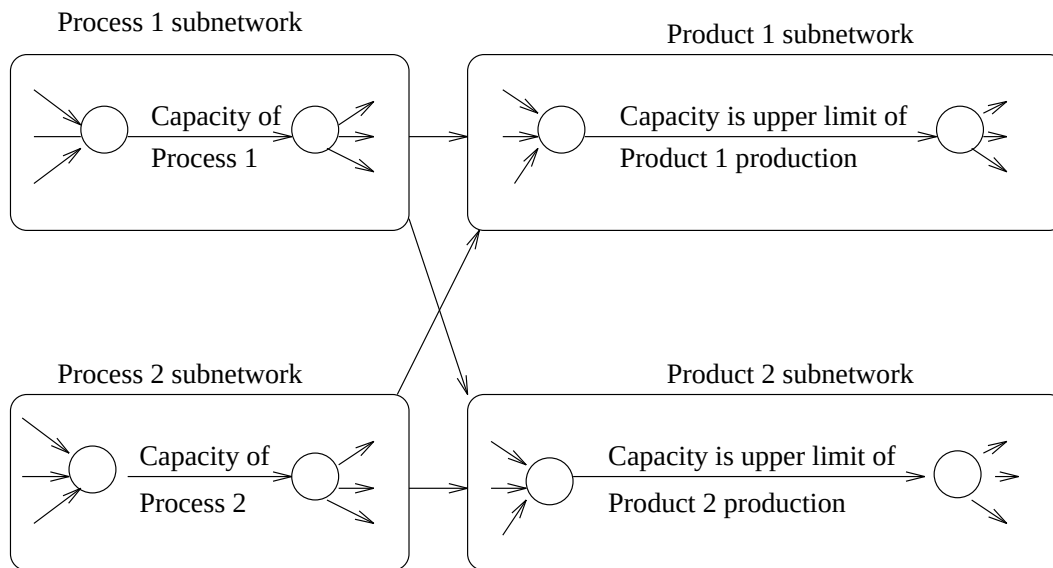


**Figure 5.5.** Multiprocess, Multiproduct Example



When starting a project, it is often a good strategy to use a small network formulation and then to use that model as a framework upon which to add detail. For example, in the multiprocess multiproduct model, you might start with the network depicted in Figure 5.5. Then, for example, the process subnetwork can be enhanced to include the distribution of products. Other phases of the operation could be included by adding more subnetworks. Initially, these subnetworks can be single nodes, but in subsequent studies they can be expanded to include greater detail.

### **Advantages of Network Models over LP Models**

Many linear programming problems have large embedded network structures. Such problems often result when modeling manufacturing processes, transportation or distribution networks, or resource allocation, or when deciding where to locate facilities. Often, some commodity is to be moved from place to place, so the more natural formulation in many applications is that of a constrained network rather than a linear program.

Using a network diagram to visualize a problem makes it possible to capture the important relationships in an easily understood picture form. The network diagram aids the communication between model builder and model user, making it easier to comprehend how the model is structured, how it can be changed, and how results can be interpreted.

If a network structure is embedded in a linear program, the problem is an NPSC (see the “Mathematical Description of NPSC” section on page 64). When the network part of the problem is large compared to the nonnetwork part, especially if the number of side constraints is small, it is worthwhile to exploit this structure to describe the model. Rather than generating the data for the flow conservation constraints, generate instead the data for the nodes and arcs of the network.

### **Flow Conservation Constraints**

The constraints  $Fx = b$  in NPSC (see the “Mathematical Description of NPSC” section on page 64) are referred to as the nodal flow conservation constraints. These constraints algebraically state that the sum of the flow through arcs directed toward a node plus that node’s supply, if any, equals the sum of the flow through arcs directed away from that node plus that node’s demand, if any. The flow conservation constraints are implicit in the network model and should not be specified explicitly in side constraint data when using PROC INTPOINT to solve NPSC problems.

### **Nonarc Variables**

Nonarc variables can be used to simplify side constraints. For example, if a sum of flows appears in many constraints, it may be worthwhile to equate this expression with a nonarc variable and use this in the other constraints. This keeps the constraint coefficient matrix sparse. By assigning a nonarc variable a nonzero objective function, it is then possible to incur a cost for using resources above some lowest feasible limit. Similarly, a profit (a negative objective function coefficient value) can be made if all available resources are not used.

In some models, nonarc variables are used in constraints to absorb excess resources or supply needed resources. Then, either the excess resource can be used or the needed resource can be supplied to another component of the model.

For example, consider a multicommodity problem of making television sets that have either 19- or 25-inch screens. In their manufacture, three and four chips, respectively, are used. Production occurs at two factories during March and April. The supplier of chips can supply only 2,600 chips to factory1 and 3,750 chips to factory2 each month. The names of arcs are in the form  $\text{Prod}n\_s\_m$ , where  $n$  is the factory number,  $s$  is the screen size, and  $m$  is the month. For example,  $\text{Prod1\_25\_Apr}$  is the arc that conveys the number of 25-inch TVs produced in factory 1 during April. You might have to determine similar systematic naming schemes for your application.

As described, the constraints are

$$\begin{aligned} 3 \text{ Prod1\_19\_Mar} + 4 \text{ Prod1\_25\_Mar} &\leq 2600 \\ 3 \text{ Prod2\_19\_Mar} + 4 \text{ Prod2\_25\_Mar} &\leq 3750 \\ 3 \text{ Prod1\_19\_Apr} + 4 \text{ Prod1\_25\_Apr} &\leq 2600 \\ 3 \text{ Prod2\_19\_Apr} + 4 \text{ Prod2\_25\_Apr} &\leq 3750 \end{aligned}$$

If there are chips that could be obtained for use in March but not used for production in March, why not keep these unused chips until April? Furthermore, if the March excess chips at factory 1 could be used either at factory 1 or factory 2 in April, the model becomes

$$\begin{aligned} 3 \text{ Prod1\_19\_Mar} + 4 \text{ Prod1\_25\_Mar} + \text{F1\_Unused\_Mar} &= 2600 \\ 3 \text{ Prod2\_19\_Mar} + 4 \text{ Prod2\_25\_Mar} + \text{F2\_Unused\_Mar} &= 3750 \\ 3 \text{ Prod1\_19\_Apr} + 4 \text{ Prod1\_25\_Apr} - \text{F1\_Kept\_Since\_Mar} &= 2600 \\ 3 \text{ Prod2\_19\_Apr} + 4 \text{ Prod2\_25\_Apr} - \text{F2\_Kept\_Since\_Mar} &= 3750 \\ \text{F1\_Unused\_Mar} + \text{F2\_Unused\_Mar} &\text{ (continued)} \\ - \text{F1\_Kept\_Since\_Mar} - \text{F2\_Kept\_Since\_Mar} &\geq 0.0 \end{aligned}$$

where  $\text{F1\_Kept\_Since\_Mar}$  is the number of chips used during April at factory1 that were obtained in March at either factory1 or factory2, and  $\text{F2\_Kept\_Since\_Mar}$  is the number of chips used during April at factory2 that were obtained in March. The last constraint ensures that the number of chips used during April that were obtained in March does not exceed the number of chips not used in March. There may be a cost to hold chips in inventory. This can be modeled having a positive objective function coefficient for the nonarc variables  $\text{F1\_Kept\_Since\_Mar}$  and  $\text{F2\_Kept\_Since\_Mar}$ . Moreover, nonarc variable upper bounds represent an upper limit on the number of chips that can be held in inventory between March and April.

See Example 5.1 through Example 5.5, which use this TV problem. The use of nonarc variables as described previously is illustrated.

---

## Introduction

---

### Getting Started: NPSC Problems

---

To solve NPSC problems using PROC INTPOINT, you save a representation of the network and the side constraints in three SAS data sets. These data sets are then passed to PROC INTPOINT for solution. There are various forms that a problem's data can take. You can use any one or a combination of several of these forms.

The NODEDATA= data set contains the names of the supply and demand nodes and the supply or demand associated with each. These are the elements in the column vector  $b$  in the NPSC problem (see the “Mathematical Description of NPSC” section on page 64).

The ARCDATA= data set contains information about the variables of the problem. Usually these are arcs, but there can be data related to nonarc variables in the ARCDATA= data set as well.

An arc is identified by the names of its tail node (where it originates) and head node (where it is directed). Each observation can be used to identify an arc in the network and, optionally, the cost per flow unit across the arc, the arc's capacity, lower flow bound, and name. These data are associated with the matrix  $F$  and the vectors  $c$ ,  $l$ , and  $u$  in the NPSC problem (see the “Mathematical Description of NPSC” section on page 64).

**Note:** although  $F$  is a node-arc incidence matrix, it is specified in the ARCDATA= data set by arc definitions. Do not explicitly specify these flow conservation constraints as constraints of the problem.

In addition, the ARCDATA= data set can be used to specify information about nonarc variables, including objective function coefficients, lower and upper value bounds, and names. These data are the elements of the vectors  $d$ ,  $m$ , and  $v$  in the NPSC problem (see the “Mathematical Description of NPSC” section on page 64). Data for an arc or nonarc variable can be given in more than one observation.

Supply and demand data also can be specified in the ARCDATA= data set. In such a case, the NODEDATA= data set may not be needed.

The CONDATA= data set describes the side constraints and their right-hand-sides. These data are elements of the matrices  $H$  and  $Q$  and the vector  $r$ . Constraint types are also specified in the CONDATA= data set. You can include in this data set upper bound values or capacities, lower flow or value bounds, and costs or objective function coefficients. It is possible to give all information about some or all nonarc variables in the CONDATA= data set.

An arc is identified in this data set by its name. If you specify an arc's name in the ARCDATA= data set, then this name is used to associate data in the CONDATA= data set with that arc. Each arc also has a default name that is the name of the tail and head node of the arc concatenated together and separated by an underscore character; tail\_head, for example.

If you use the dense side constraint input format (described in the “CONDATA= Data Set” section on page 133), and want to use these default arc names, these arc names are names of SAS variables in the VAR list of the CONDATA= data set.

If you use the sparse side constraint input format (see the “CONDATA= Data Set” section on page 133) and want to use these default arc names, these arc names are values of the COLUMN list variable of the CONDATA= data set.

PROC INTPOINT reads the data from the NODEDATA= data set, the ARCDATA= data set, and the CONDATA= data set. Error checking is performed, and the model is converted into an equivalent LP. This LP is preprocessed. Preprocessing is optional but highly recommended. Preprocessing analyses the model and tries to determine before optimization whether variables can be “fixed” to their optimal values. Knowing that, the model can be modified and these variables dropped out. It can be determined that some constraints are redundant. Sometimes, preprocessing succeeds in reducing the size of the problem, thereby making the subsequent optimization easier and faster.

The optimal solution to the equivalent LP is then found. This LP is converted back to the original NPSC problem, and the optimum for this is derived from the optimum of the equivalent LP. If the problem was preprocessed, the model is now post-processed, where fixed variables are reintroduced. The solution can be saved in the CONOUT= data set.

### ***Introductory NPSC Example***

Consider the following transshipment problem for an oil company. Crude oil is shipped to refineries where it is processed into gasoline and diesel fuel. The gasoline and diesel fuel are then distributed to service stations. At each stage, there are shipping, processing, and distribution costs. Also, there are lower flow bounds and capacities.

In addition, there are two sets of side constraints. The first set is that two times the crude from the Middle East cannot exceed the throughput of a refinery plus 15 units. (The words “plus 15 units” that finishes the last sentence is used to enable some side constraints in this example to have a nonzero rhs.) The second set of constraints are necessary to model the situation that one unit of crude mix processed at a refinery yields three-fourths of a unit of gasoline and one-fourth of a unit of diesel fuel.

Because there are two products that are not independent in the way in which they flow through the network, an NPSC is an appropriate model for this example (see Figure 5.6). The side constraints are used to model the limitations on the amount of Middle Eastern crude that can be processed by each refinery and the conversion proportions of crude to gasoline and diesel fuel.

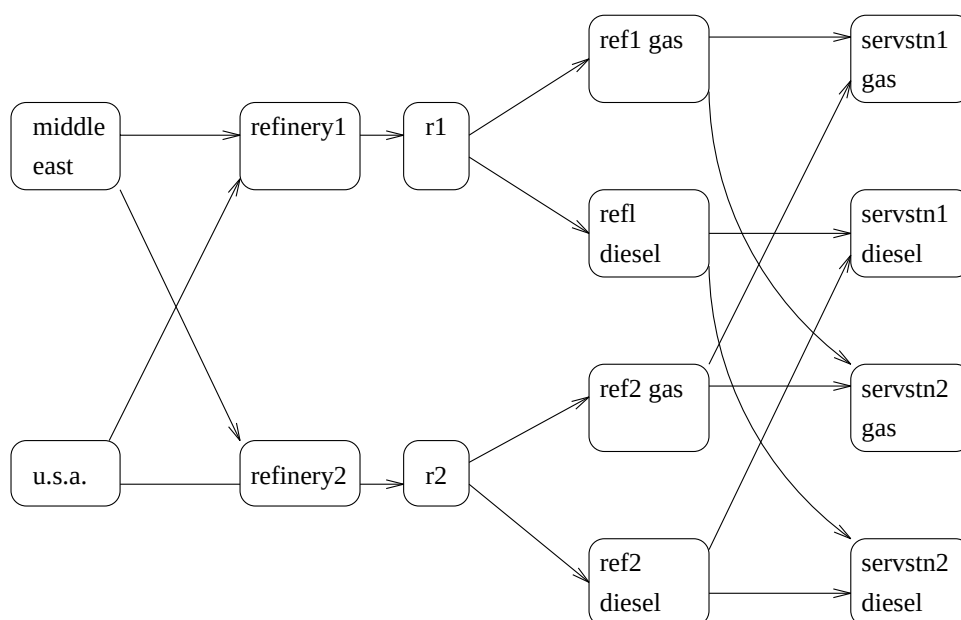


**Figure 5.6.** Oil Industry Example

To solve this problem with PROC INTPOINT, save a representation of the model in three SAS data sets. In the NODEDATA= data set, you name the supply and demand nodes and give the associated supplies and demands. To distinguish demand nodes from supply nodes, specify demands as negative quantities. For the oil example, the NODEDATA= data set can be saved as follows:

```

title 'Oil Industry Example';
title3 'Setting Up Nodedata = Noded For PROC INTPOINT';
data noded;
  input _node_&$15. _sd_;
  datalines;
middle east      100
u.s.a.           80
servstn1 gas     -95
servstn1 diesel -30
servstn2 gas     -40
servstn2 diesel -15
;

```

The ARCDATA= data set contains the rest of the information about the network. Each observation in the data set identifies an arc in the network and gives the cost per flow unit across the arc, the capacities of the arc, the lower bound on flow across the arc, and the name of the arc.

```

title3 'Setting Up Arcdata = Arcd1 For PROC INTPOINT';
data arcd1;
  input _from_&$11. _to_&$15. _cost_ _capac_ _lo_ _name_ $;
  datalines;
middle east      refinery 1      63      95      20      m_e_ref1
middle east      refinery 2      81      80      10      m_e_ref2
u.s.a.           refinery 1      55      .      .      .
u.s.a.           refinery 2      49      .      .      .
refinery 1       r1             200     175     50      thrupt1
refinery 2       r2             220     100     35      thrupt2
r1               ref1 gas        .      140     .      r1_gas
r1               ref1 diesel    .      75     .      .
r2               ref2 gas        .      100     .      r2_gas
r2               ref2 diesel    .      75     .      .
ref1 gas         servstn1 gas    15      70     .      .
ref1 gas         servstn2 gas    22      60     .      .
ref1 diesel      servstn1 diesel 18      .      .      .
ref1 diesel      servstn2 diesel 17      .      .      .
ref2 gas         servstn1 gas    17      35     5      .
ref2 gas         servstn2 gas    31      .      .      .
ref2 diesel      servstn1 diesel 36      .      .      .
ref2 diesel      servstn2 diesel 23      .      .      .
;

```

Finally, the CONDATA= data set contains the side constraints for the model:

```

title3 'Setting Up Condata = Cond1 For PROC INTPOINT';
data cond1;
  input m_e_ref1 m_e_ref2 thrupt1 r1_gas thrupt2 r2_gas
        _type_ $ _rhs_;
  datalines;
-2   .   1   .   .   .   >=   -15
.   -2   .   .   1   .   GE   -15
.   .   -3  4   .   .   EQ    0
.   .   .   .   -3  4   =    0
;

```

Note that the SAS variable names in the CONDATA= data set are the names of arcs given in the ARCDATA= data set. These are the arcs that have nonzero constraint coefficients in side constraints. For example, the proportionality constraint that specifies that one unit of crude at each refinery yields three-fourths of a unit of gasoline and one-fourth of a unit of diesel fuel is given for refinery 1 in the third observation and for refinery 2 in the last observation. The third observation requires that each unit of flow on the arc thrupt1 equals three-fourths of a unit of flow on the arc r1\_gas. Because all crude processed at refinery 1 flows through thrupt1 and all gasoline produced at refinery 1 flows through r1\_gas, the constraint models the situation. It proceeds similarly for refinery 2 in the last observation.

To find the minimum cost flow through the network that satisfies the supplies, demands, and side constraints, invoke PROC INTPOINT as follows:

```
proc intpoint
  bytes=1000000
  nodedata=noded          /* the supply and demand data */
  arcdata=arcd1           /* the arc descriptions      */
  condata=cond1           /* the side constraints      */
  conout=solution;        /* the solution data set    */
run;
```

The following messages, that appear on the SAS log, summarize the model as read by PROC INTPOINT and note the progress toward a solution:

```
NOTE: Number of nodes= 14 .
NOTE: Number of supply nodes= 2 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 180 , total demand= 180 .
NOTE: Number of arcs= 18 .
NOTE: Number of <= side constraints= 0 .
NOTE: Number of == side constraints= 2 .
NOTE: Number of >= side constraints= 2 .
NOTE: Number of side constraint coefficients= 8 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 16 .
NOTE: Number of >= constraints= 2 .
NOTE: Number of constraint coefficients= 44 .
NOTE: Number of variables= 18 .
NOTE: After preprocessing, number of <= constraints= 0.
NOTE: After preprocessing, number of == constraints= 6.
NOTE: After preprocessing, number of >= constraints= 2.
NOTE: The preprocessor eliminated 10 constraints from the
      problem.
NOTE: The preprocessor eliminated 22 constraint
      coefficients from the problem.
NOTE: After preprocessing, number of variables= 8.
NOTE: The preprocessor eliminated 10 variables from the
      problem.
NOTE: 2 columns, 0 rows and 2 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 18 nonzero elements in A * A transpose.
NOTE: Of the 8 rows and columns, 2 are sparse.
NOTE: There are 8 nonzero superdiagonal elements in the
      sparse rows of the factored A * A transpose. This
      includes fill-in.
NOTE: There are 3 operations of the form
      u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q] to factorize the
      sparse rows of A * A transpose.
NOTE: Bound feasibility attained by iteration 1.
```

NOTE: Dual feasibility attained by iteration 1.  
 NOTE: Constraint feasibility attained by iteration 2.  
 NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 12 iterations.  
 NOTE: The rate of complementarity reduction is slow. The optimum has probably been reached even though standard stopping conditions have not been met.  
 NOTE: Objective = 50875.006369.  
 NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.  
 NOTE: There were 18 observations read from the data set WORK.ARC1.  
 NOTE: There were 6 observations read from the data set WORK.NODED.  
 NOTE: There were 4 observations read from the data set WORK.COND1.  
 NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.

The first set of messages shows the size of the problem. The next set of messages provides statistics on the size of the equivalent LP problem. The number of variables may not equal the number of arcs if the problem has nonarc variables. This example has none. To convert a network to the equivalent LP problem, a flow conservation constraint must be created for each node (including an excess or bypass node, if required). This explains why the number of equality constraints and the number of constraint coefficients differ from the number of equality side constraints and the number of coefficients in all side constraints.

If the preprocessor was successful in decreasing the problem size, some messages will report how well it did. In this example, the model size was cut approximately in half!

The next set of messages describes aspects of the Interior Point algorithm. Of particular interest are those concerned with the Cholesky factorization of  $AA^T$  where  $A$  is the coefficient matrix of the final LP. It is crucial to preorder the rows and columns of this matrix to prevent *fill-in* and reduce the number of row operations to undertake the factorization. See the “Interior Point Algorithmic Details” section on page 66 for a more extensive explanation.

Unlike PROC LP, which displays the solution and other information as output, PROC INTPOINT saves the optimum in the output SAS data set that you specify. For this example, the solution is saved in the SOLUTION data set. It can be displayed with the PRINT procedure as:

```
title3 'Optimum';
proc print data=solution;
  var _from_ _to_ _cost_ _capac_ _lo_ _name_
      _supply_ _demand_ _flow_ _fcost_;
  sum _fcost_;
run;
```



| Oil Industry Example           |             |                 |       |          |          |          |       |          |       |
|--------------------------------|-------------|-----------------|-------|----------|----------|----------|-------|----------|-------|
| Optimum                        |             |                 |       |          |          |          |       |          |       |
| Flow                           | Cost        | Flow            | Cost  | Flow     | Cost     | Flow     | Cost  | Flow     | Cost  |
| Variable                       | Value       | Variable        | Value | Variable | Value    | Variable | Value | Variable | Value |
| 1 refinery 1 r1                | 200         | 175 50 thruput1 | . .   | 145.000  | 28999.99 |          |       |          |       |
| 2 refinery 2 r2                | 220         | 100 35 thruput2 | . .   | 35.000   | 7700.01  |          |       |          |       |
| 3 r1 ref1 diesel               | 0           | 75 0            | . .   | 36.250   | 0.00     |          |       |          |       |
| 4 r1 ref1 gas                  | 0           | 140 0 r1_gas    | . .   | 108.750  | 0.00     |          |       |          |       |
| 5 r2 ref2 diesel               | 0           | 75 0            | . .   | 8.750    | 0.00     |          |       |          |       |
| 6 r2 ref2 gas                  | 0           | 100 0 r2_gas    | . .   | 26.250   | 0.00     |          |       |          |       |
| 7 middle east refinery 1       | 63          | 95 20 m_e_ref1  | 100 . | 80.000   | 5039.99  |          |       |          |       |
| 8 u.s.a. refinery 1            | 55 99999999 | 0               | 80 .  | 65.000   | 3575.00  |          |       |          |       |
| 9 middle east refinery 2       | 81          | 80 10 m_e_ref2  | 100 . | 20.000   | 1620.01  |          |       |          |       |
| 10 u.s.a. refinery 2           | 49 99999999 | 0               | 80 .  | 15.000   | 735.00   |          |       |          |       |
| 11 ref1 diesel servstn1 diesel | 18 99999999 | 0               | . 30  | 30.000   | 540.00   |          |       |          |       |
| 12 ref2 diesel servstn1 diesel | 36 99999999 | 0               | . 30  | 0.000    | 0.00     |          |       |          |       |
| 13 ref1 gas servstn1 gas       | 15          | 70 0            | . 95  | 68.750   | 1031.25  |          |       |          |       |
| 14 ref2 gas servstn1 gas       | 17          | 35 5            | . 95  | 26.250   | 446.25   |          |       |          |       |
| 15 ref1 diesel servstn2 diesel | 17 99999999 | 0               | . 15  | 6.250    | 106.25   |          |       |          |       |
| 16 ref2 diesel servstn2 diesel | 23 99999999 | 0               | . 15  | 8.750    | 201.25   |          |       |          |       |
| 17 ref1 gas servstn2 gas       | 22          | 60 0            | . 40  | 40.000   | 879.99   |          |       |          |       |
| 18 ref2 gas servstn2 gas       | 31 99999999 | 0               | . 40  | 0.000    | 0.01     |          |       |          |       |
| =====                          |             |                 |       |          |          |          |       |          |       |
| 50875.01                       |             |                 |       |          |          |          |       |          |       |

Figure 5.7. CONOUT=SOLUTION

Notice that, in CONOUT=SOLUTION (Figure 5.7), the optimal flow through each arc in the network is given in the variable named `_FLOW_`, and the cost of flow through each arc is given in the variable `_FCOST_`.

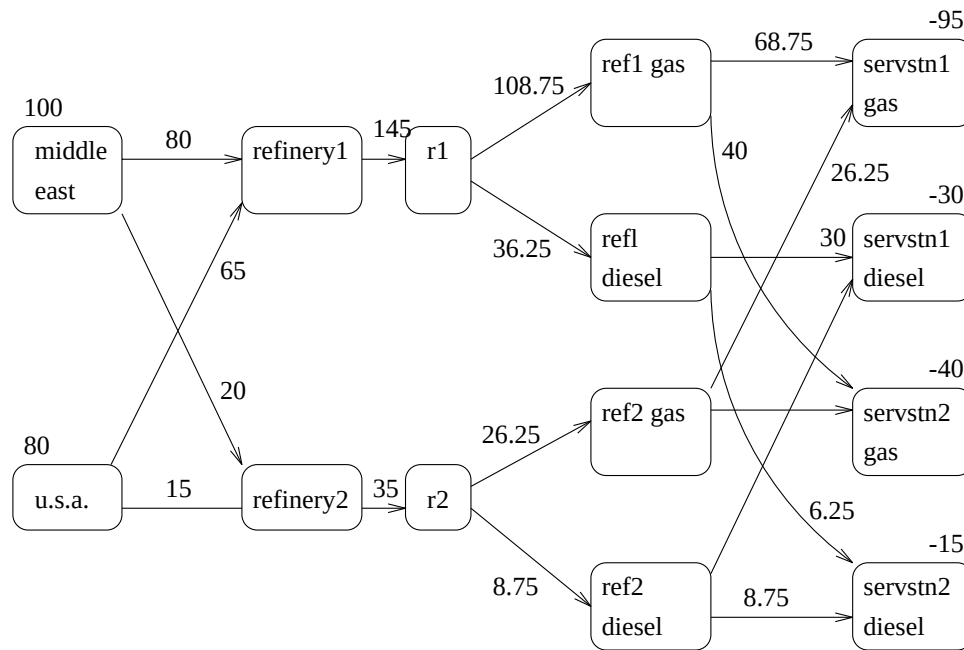


Figure 5.8. Oil Industry Solution

## Getting Started: LP Problems

Data for an LP problem resembles the data for side constraints and nonarc variables supplied to PROC INTPOINT when solving an NPSC problem. It is also very similar to the data required by the LP procedure.

To solve LP problems using PROC INTPOINT, you save a representation of the LP variables and the constraints in one or two SAS data sets. These data sets are then passed to PROC INTPOINT for solution. There are various forms that a problem's data can take. You can use any one or a combination of several of these forms.

The ARCDATA= data set contains information about the LP variables of the problem. Although this data set is called ARCDATA, it contains data for no arcs. Instead, all data in this data set are related to LP variables. This data set has no SAS variables containing values that are node names.

The ARCDATA= data set can be used to specify information about LP variables, including objective function coefficients, lower and upper value bounds, and names. These data are the elements of the vectors  $d$ ,  $m$ , and  $v$  in problem (NP). Data for an LP variable can be given in more than one observation.

The CONDATA= data set describes the constraints and their right-hand-sides. These data are elements of the matrix  $Q$  and the vector  $r$ .

Constraint types are also specified in the CONDATA= data set. You can include in this data set LP variable data such as upper bound values, lower value bounds, and

objective function coefficients. It is possible to give all information about some or all LP variables in the CONDATA= data set.

Because PROC INTPOINT evolved from PROC NETFLOW, another procedure in SAS/OR software that was originally designed to solve models with networks, the ARCDATA= data set is always expected. If the ARCDATA= data set is not specified, by default the last data set created before PROC INTPOINT is invoked is assumed to be the ARCDATA= data set. However, these characteristics of PROC INTPOINT are not helpful when an LP problem is being solved and all data is provided in a single data set specified by the CONDATA= data set, and that data set is not the last data set created before PROC INTPOINT starts. In this case, you must specify that the ARCDATA= data set and the CONDATA= data set are both equal to the input data set. PROC INTPOINT then knows that an LP problem is to be solved and that the data reside in one data set.

An LP variable is identified in this data set by its name. If you specify an LP variable's name in the ARCDATA= data set, then this name is used to associate data in the CONDATA= data set with that LP variable.

If you use the dense constraint input format (described in the “CONDATA= Data Set” section on page 133), these LP variable names are names of SAS variables in the VAR list of the CONDATA= data set.

If you use the sparse constraint input format (described in the “CONDATA= Data Set” section on page 133), these LP variable names are values of the SAS variables in the COLUMN list of the CONDATA= data set.

PROC INTPOINT reads the data from the ARCDATA= data set (if there is one) and the CONDATA= data set. Error checking is performed, and the LP is preprocessed. Preprocessing is optional but highly recommended. The Preprocessor analyses the model and tries to determine before optimization whether LP variables can be “fixed” to their optimal values. Knowing that, the model can be modified and these LP variables dropped out. Some constraints may be found to be redundant. Sometimes, preprocessing succeeds in reducing the size of the problem, thereby making the subsequent optimization easier and faster.

The optimal solution is then found for the resulting LP. If the problem was preprocessed, the model is now post-processed, where fixed LP variables are reintroduced. The solution can be saved in the CONOUT= data set.

**Introductory LP Example**

Consider the Linear Programming problem in the chapter on the LP procedure. The SAS data set in that section is created the same way here:

```

title 'Linear Programming Example';
title3 'Setting Up Condata = dcon1 For PROC INTPOINT';
data dcon1;
  input _id_ $14.
        a_light a_heavy brega naphthal naphthai
        heatingo jet_1 jet_2
        _type_ $ _rhs_;
  datalines;
profit      -175 -165 -205  0  0  0 300 300 max      .
naphtha_l_conv .035 .030 .045 -1  0  0  0  0 eq      0
naphtha_i_conv .100 .075 .135  0 -1  0  0  0 eq      0
heating_o_conv .390 .300 .430  0  0 -1  0  0 eq      0
recipe_1       0    0    0  0 .3 .7 -1  0 eq      0
recipe_2       0    0    0 .2  0 .8  0 -1 eq      0
available      110  165  80  .  .  .  .  . upperbd .
;

```

To solve this problem, use:

```

proc intpoint
  bytes=1000000
  condata=dcon1
  conout=solutn1;
run;

```

Note how it is possible to use an input SAS data set of PROC LP and, without requiring any changes to be made to the data set, to use that as an input data set for PROC INTPOINT.

The following messages that appear on the SAS log summarize the model as read by PROC INTPOINT and note the progress toward a solution:

```

NOTE: Number of variables= 8 .
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 5 .
NOTE: Number of >= constraints= 0 .
NOTE: Number of constraint coefficients= 18 .
NOTE: After preprocessing, number of <= constraints= 0.
NOTE: After preprocessing, number of == constraints= 0.
NOTE: After preprocessing, number of >= constraints= 0.
NOTE: The preprocessor eliminated 5 constraints from the
      problem.
NOTE: The preprocessor eliminated 18 constraint
      coefficients from the problem.
NOTE: After preprocessing, number of variables= 0.
NOTE: The preprocessor eliminated 8 variables from the
      problem.
WARNING: Optimization is unnecessary as the problem has 0
         variables and 0 rows.

```

NOTE: Preprocessing could have caused that.  
 NOTE: Objective = 1544.  
 NOTE: The data set WORK.SOLUTN1 has 8 observations and 6 variables.  
 NOTE: There were 7 observations read from the data set WORK.DCON1.  
 NOTE: The data set WORK.SOLUTN1 has 8 observations and 6 variables.

Notice that the preprocessor succeeded in fixing *all* LP variables to their optimal values, eliminating the need to do any actual optimization.

Unlike PROC LP, which displays the solution and other information as output, PROC INTPOINT saves the optimum in the output SAS data set you specify. For this example, the solution is saved in the SOLUTION data set. It can be displayed with PROC PRINT as:

```
title3 'LP Optimum';
proc print data=solutn1;
  var _name_ _cost_ _capac_ _lo_ _flow_ _fcost_;
  sum _fcost_;
run;
```

Notice that in the CONOUT=SOLUTION (Figure 5.9) the optimal value through each variable in the LP is given in the variable named `_FLOW_`, and that the cost of value for each variable is given in the variable `_FCOST_`.

| Linear Programming Example |          |        |          |      |        |         |
|----------------------------|----------|--------|----------|------|--------|---------|
| LP Optimum                 |          |        |          |      |        |         |
| Obs                        | _NAME_   | _COST_ | _CAPAC_  | _LO_ | _FLOW_ | _FCOST_ |
| 1                          | a_heavy  | -165   | 165      | 0    | 0.00   | 0       |
| 2                          | a_light  | -175   | 110      | 0    | 110.00 | -19250  |
| 3                          | brega    | -205   | 80       | 0    | 80.00  | -16400  |
| 4                          | heatingo | 0      | 99999999 | 0    | 77.30  | 0       |
| 5                          | jet_1    | 300    | 99999999 | 0    | 60.65  | 18195   |
| 6                          | jet_2    | 300    | 99999999 | 0    | 63.33  | 18999   |
| 7                          | naphthai | 0      | 99999999 | 0    | 21.80  | 0       |
| 8                          | naphthal | 0      | 99999999 | 0    | 7.45   | 0       |
|                            |          |        |          |      |        | =====   |
|                            |          |        |          |      |        | 1544    |

Figure 5.9. CONOUT=SOLUTN1

The same model can be specified in the sparse format as in the following scon2 dataset. This format enables you to omit the zero coefficients:

```

title3 'Setting Up Condata = scon2 For PROC INTPOINT';
data scon2;
    input _type_ $ @10 _col_ $13. @24 _row_ $16. _coef_;
    datalines;
max          .          profit          .
eq           .          napha_l_conv     .
eq           .          napha_i_conv     .
eq           .          heating_oil_conv .
eq           .          recipe_1         .
eq           .          recipe_2         .
upperbd      .          available        .
.            a_light     profit          -175
.            a_light     napha_l_conv     .035
.            a_light     napha_i_conv     .100
.            a_light     heating_oil_conv .390
.            a_light     available        110
.            a_heavy     profit          -165
.            a_heavy     napha_l_conv     .030
.            a_heavy     napha_i_conv     .075
.            a_heavy     heating_oil_conv .300
.            a_heavy     available        165
.            brega       profit          -205
.            brega       napha_l_conv     .045
.            brega       napha_i_conv     .135
.            brega       heating_oil_conv .430
.            brega       available        80
.            naphthal     napha_l_conv     -1
.            naphthal     recipe_2         .2
.            naphthai     napha_i_conv     -1
.            naphthai     recipe_1         .3
.            heatingo     heating_oil_conv -1
.            heatingo     recipe_1         .7
.            heatingo     recipe_2         .8
.            jet_1        profit          300
.            jet_1        recipe_1         -1
.            jet_2        profit          300
.            jet_2        recipe_2         -1
;

```

To find the minimum cost solution, invoke PROC INTPOINT (note the SPARSECONDATA option which must be specified) as follows:

```

proc intpoint
    bytes=1000000
    sparseconddata
    condata=scon2
    conout=solutn2;
run;

```

A data set that can be used as the ARCDATA= data set can be initialized as follows:

```
data vars3;
  input _name_ $ profit available;
  datalines;
a_heavy -165 165
a_light -175 110
brega -205 80
heatingo 0 .
jet_1 300 .
jet_2 300 .
naphthai 0 .
naphthal 0 .
;
```

The following CONDATA= data set is the original dense format CONDATA= dcon1 data set after the LP variable's nonconstraint information has been removed. (You could have left some or all of that information in CONDATA as PROC INTPOINT "merges" data, but doing that and checking for consistency takes time.)

```
data dcon3;
  input _id_ $14.
  a_light a_heavy brega naphthal naphthai
  heatingo jet_1 jet_2
  _type_ $ _rhs_;
  datalines;
naphtha_l_conv .035 .030 .045 -1 0 0 0 0 eq 0
naphtha_i_conv .100 .075 .135 0 -1 0 0 0 eq 0
heating_o_conv .390 .300 .430 0 0 -1 0 0 eq 0
recipe_1 0 0 0 0 .3 .7 -1 0 eq 0
recipe_2 0 0 0 .2 0 .8 0 -1 eq 0
;
```

**Note:** You must now specify the MAXIMIZE option; otherwise, PROC INTPOINT will optimize to the minimum (which, incidentally, has a total objective = -3539.25). You must indicate that the SAS variable profit in the ARCDATA= vars3 data set has values that are objective function coefficients, by specifying the OBJFN statement. The UPPERBD must be specified as the SAS variable available that has as values upper bounds:

```
proc intpoint
  maximize /* ***** necessary ***** */
  bytes=1000000
  arcdata=vars3
  condata=dcon3
  conout=solutn3;
objfn profit;
upperbd available;
run;
```

The ARCDATA=vars3 data set can become more concise by noting that the model variables heatingo, naphthai, and naphthal have zero objective function coeffi-

cients (the default) and default upper bounds, so those observations need not be present:

```
data vars4;
    input _name_ $ profit available;
    datalines;
a_heavy -165 165
a_light -175 110
brega -205 80
jet_1 300 .
jet_2 300 .
;
```

The CONDATA=dcon3 data set can become more concise by noting that all the constraints have the same type (eq) and zero (the default) rhs values. This model is a good candidate for using the DEFCONTYPE= options.

The DEFCONTYPE= option can be useful not only when *all* constraints have the same type as is the case here, but also when *most* constraints have the same type and you want to change the default type from  $\leq$  to  $=$  or  $\geq$ . The essential constraint type data in the CONDATA= data set is that which overrides the DEFCONTYPE= type you specified:

```
data dcon4;
    input _id_ $14.
           a_light a_heavy brega naphthal naphthai
           heatingo jet_1 jet_2;
    datalines;
naphtha_l_conv .035 .030 .045 -1 0 0 0 0
naphtha_i_conv .100 .075 .135 0 -1 0 0 0
heating_o_conv .390 .300 .430 0 0 -1 0 0
recipe_1      0 0 0 0 .3 .7 -1 0
recipe_2      0 0 0 .2 0 .8 0 -1
;

proc intpoint
    maximize defcontype=eq
    arcdata=vars3
    condata=dcon3
    conout=solutn3;
objfn profit;
upperbd available;
run;
```

Here are several different ways of using the ARCDATA= data set and a sparse format CONDATA= data set for this LP. The following CONDATA= data set is the result of removing the profit and available data from the original sparse format CONDATA=scon2 data set.



```

data scon5;
  input _type_ $ @10 _col_ $13. @24 _row_ $16. _coef_;
  datalines;
eq      .          napha_l_conv      .
eq      .          napha_i_conv      .
eq      .          heating_oil_conv   .
eq      .          recipe_1           .
eq      .          recipe_2           .
.      a_light     napha_l_conv      .035
.      a_light     napha_i_conv      .100
.      a_light     heating_oil_conv   .390
.      a_heavy     napha_l_conv      .030
.      a_heavy     napha_i_conv      .075
.      a_heavy     heating_oil_conv   .300
.      brega       napha_l_conv      .045
.      brega       napha_i_conv      .135
.      brega       heating_oil_conv   .430
.      naphthal    napha_l_conv      -1
.      naphthal    recipe_2           .2
.      naphthai    napha_i_conv      -1
.      naphthai    recipe_1           .3
.      heatingo    heating_oil_conv   -1
.      heatingo    recipe_1           .7
.      heatingo    recipe_2           .8
.      jet_1       recipe_1           -1
.      jet_2       recipe_2           -1
;

proc intpoint
  maximize
  sparsesecondata
  arcdata=vars3      /* or arcdata=vars4 */
  condata=scon5
  conout=solutn5;
objfn profit;
upperbd available;
run;

```

The CONDATA=scon5 data set can become more concise by noting that all the constraints have the same type (eq) and zero (the default) rhs values. Use the DEFCON-  
TYPE= option again. Once the first five observations of the CONDATA=scon5 data  
set are removed, the \_type\_ SAS variable has values that are missing in all of the  
remaining observations. Therefore, this SAS variable can be removed.

```

data scon6;
  input _col_ $ _row_&$16. _coef_;
  datalines;
a_light  napha_l_conv          .035
a_light  napha_i_conv          .100
a_light  heating_oil_conv      .390
a_heavy  napha_l_conv          .030
a_heavy  napha_i_conv          .075
a_heavy  heating_oil_conv      .300
brega    napha_l_conv          .045
brega    napha_i_conv          .135
brega    heating_oil_conv      .430
naphthal napha_l_conv          -1
naphthal recipe_2             .2
naphthai napha_i_conv          -1
naphthai recipe_1             .3
heatingo heating_oil_conv      -1
heatingo recipe_1             .7
heatingo recipe_2             .8
jet_1    recipe_1             -1
jet_2    recipe_2             -1
;

proc intpoint
  maximize
  defcontype=eq
  sparsesecondata
  arcdata=vars4
  condata=scon6
  conout=solutn6;
objfn profit;
upperbd available;
run;

```

---

## Typical PROC INTPOINT Run

You start PROC INTPOINT by giving the PROC INTPOINT statement. You can specify many options that control the procedure in the PROC INTPOINT statement, or you can rely on default settings and specify no options at all. However, there are some options you must specify, or will probably have to specify:

- You must specify the BYTES= parameter indicating the size of the working memory that the procedure is allowed to use. This option has no default.
- In many instances (and certainly when solving NPSC problems), you need to specify the ARCDATA= data set. This option has a default (which is the SAS data set that was created last before PROC INTPOINT began running), but that may need to be overridden.
- The CONDATA= data set must also be specified if the problem is NPSC and has side constraints, or if it is an LP problem.
- When solving a network problem, you have to specify the NODEDATA= data set, if some model data is given in such a data set.

Some options, while optional, are frequently required. To have the optimal solution output to a SAS data set, you have to specify the CONOUT= data set. You may want to indicate reasons why optimization should stop (for example, you can indicate the maximum number of iterations that can be performed), or you might want to alter stopping criteria so that optimization does not stop prematurely. Some options enable you to control other aspects of the Interior Point algorithm. Specifying certain values for these options can reduce the time it takes to solve a problem.

The SAS variable lists should be given next. If you have SAS variables in the input data sets that have special names (for example, a SAS variable in the ARCDATA= data set named \_TAIL\_ that has tail nodes of arcs as values), it may not be necessary to have many or any variable lists. If you do not specify a TAIL variable list, PROC INTPOINT will search the ARCDATA= data set for a SAS variable named \_TAIL\_.

What usually follows is a RUN statement, which indicates that all information that you, the user, need to supply to PROC INTPOINT has been given, and the procedure is to start running. This also happens if you specify a statement in your SAS program that PROC INTPOINT does not recognize as one of its own, the next DATA step or procedure.

The QUIT statement indicates that PROC INTPOINT must immediately finish.

For example, a PROC INTPOINT run might look something like this:

```
proc intpoint
    bytes=      /* working memory size */
    arcdata=    /* data set */
    condata=    /* data set */
    /* other options */
;
variable list specifications; /* if necessary */
run;           /* start running, read data, */
               /* and do the optimization. */
```

## Functional Summary

The following tables outline the options that can be specified in the PROC INTPOINT statement. This statement invokes the procedure.

**Table 5.1.** Input Data Set Options

| Description               | Statement | Option    |
|---------------------------|-----------|-----------|
| arcs input data set       | INTPOINT  | ARCDATA=  |
| nodes input data set      | INTPOINT  | NODEDATA= |
| constraint input data set | INTPOINT  | CONDATA=  |

**Table 5.2.** Options for Networks

| Description                                         | Statement | Option       |
|-----------------------------------------------------|-----------|--------------|
| default arc cost and                                | INTPOINT  | DEFCOST=     |
| default variable objective function coefficient     |           |              |
| default arc capacity                                | INTPOINT  | DEFCAPACITY= |
| default variable upper bound                        |           |              |
| default arc lower flow bound                        | INTPOINT  | DEFMINFLOW=  |
| default variable lower bound                        |           |              |
| network's only supply node                          | INTPOINT  | SOURCE=      |
| SOURCE's supply capability                          | INTPOINT  | SUPPLY=      |
| network's only demand node                          | INTPOINT  | SINK=        |
| SINK's demand                                       | INTPOINT  | DEMAND=      |
| excess supply or demand is conveyed through network | INTPOINT  | THRUNET      |
| find maximal flow between SOURCE and SINK           | INTPOINT  | MAXFLOW      |
| cost of bypass arc when solving MAXFLOW problem     | INTPOINT  | BYPASSDIV=   |
| find shortest path from SOURCE to SINK              | INTPOINT  | SHORTPATH    |

**Table 5.3.** Miscellaneous Options

| Description                                                           | Statement | Option    |
|-----------------------------------------------------------------------|-----------|-----------|
| infinity value                                                        | INTPOINT  | INFINITY= |
| scale constraint row, or nonarc variable column coefficients, or both | INTPOINT  | SCALE=    |
| maximization instead of minimization                                  | INTPOINT  | MAXIMIZE  |

**Table 5.4.** Data Set Read Options

| Description                                                    | Statement | Option            |
|----------------------------------------------------------------|-----------|-------------------|
| CONDATA has sparse data format                                 | INTPOINT  | SPARSECONDATA     |
| default constraint type                                        | INTPOINT  | DEFCONTYPE=       |
| special COLUMN variable value                                  | INTPOINT  | TYPEOBS=          |
| special COLUMN variable value                                  | INTPOINT  | RHSOBS=           |
| used to interpret arc and nonarc variable names in the CONDATA | INTPOINT  | NAMECTRL=         |
| no nonarc data in the ARCDATA                                  | INTPOINT  | ARCS_ONLY_ARCDATA |
| data for an arc found in only one obs of ARCDATA               | INTPOINT  | ARC_SINGLE_OBS    |
| data for an constraint found in only one obs of the CONDATA    | INTPOINT  | CON_SINGLE_OBS    |
| data for a coefficient found once in the CONDATA               | INTPOINT  | NON_REPLIC=       |
| data is grouped, exploited during data read                    | INTPOINT  | GROUPED=          |

**Table 5.5.** Problem Size (approx.) Options

| Description                             | Statement | Option  |
|-----------------------------------------|-----------|---------|
| number of nodes                         | INTPOINT  | NNODES= |
| number of arcs                          | INTPOINT  | NARCS=  |
| number of nonarc variables or variables | INTPOINT  | NNAS=   |
| number of coefficients                  | INTPOINT  | NCOEFS= |
| number of constraints                   | INTPOINT  | NCONS=  |

**Table 5.6.** Memory Control Options

| Description                            | Statement | Option |
|----------------------------------------|-----------|--------|
| issue memory usage messages to SASLOG  | INTPOINT  | MEMREP |
| number of bytes to use for main memory | INTPOINT  | BYTES= |

**Table 5.7.** Output Data Set Option

| Description                   | Statement | Option  |
|-------------------------------|-----------|---------|
| constrained solution data set | INTPOINT  | CONOUT= |

**Table 5.8.** Miscellaneous Options

| Description                                                        | Statement | Option   |
|--------------------------------------------------------------------|-----------|----------|
| zero tolerance - optimization                                      | INTPOINT  | ZERO2=   |
| zero tolerance - real number comparisons                           | INTPOINT  | ZEROTOL= |
| display this number of similar SAS log messages, suppress the rest | INTPOINT  | VERBOSE= |

**Table 5.9.** Interior Point Algorithm Options

| Description                                  | Statement | Option       |
|----------------------------------------------|-----------|--------------|
| allowed amount of dual infeasibility         | INTPOINT  | TOLDINF=     |
| allowed amount of primal infeasibility       | INTPOINT  | TOLPINF=     |
| allowed total amount of dual infeasibility   | INTPOINT  | TOLTOTDINF=  |
| allowed total amount of primal infeasibility | INTPOINT  | TOLTOTPINF=  |
| cut-off tolerance for Cholesky factorization | INTPOINT  | CHOLTINYTOL= |
| density threshold for Cholesky processing    | INTPOINT  | DENSETHR=    |
| step-length multiplier                       | INTPOINT  | PDSTEPMULT=  |
| preprocessing type                           | INTPOINT  | PRSLTYPE=    |
| print optimization progress on SAS log       | INTPOINT  | PRINTLEVEL2= |

**Table 5.10.** Interior Point Algorithm Options: Stopping Criteria

| Description                                           | Statement | Option             |
|-------------------------------------------------------|-----------|--------------------|
| maximum number of Interior Point algorithm iterations | INTPOINT  | MAXITERB=          |
| Primal-Dual (Duality) gap tolerance                   | INTPOINT  | PDGAPTOL=          |
| stop because of complementarity                       | INTPOINT  | STOP_C=            |
| stop because of duality gap                           | INTPOINT  | STOP_DG=           |
| stop because of <i>infeas<sub>b</sub></i>             | INTPOINT  | STOP_IB=           |
| stop because of <i>infeas<sub>c</sub></i>             | INTPOINT  | STOP_IC=           |
| stop because of <i>infeas<sub>d</sub></i>             | INTPOINT  | STOP_ID=           |
| stop because of complementarity                       | INTPOINT  | AND_STOP_C=        |
| stop because of duality gap                           | INTPOINT  | AND_STOP_DG=       |
| stop because of <i>infeas<sub>b</sub></i>             | INTPOINT  | AND_STOP_IB=       |
| stop because of <i>infeas<sub>c</sub></i>             | INTPOINT  | AND_STOP_IC=       |
| stop because of <i>infeas<sub>d</sub></i>             | INTPOINT  | AND_STOP_ID=       |
| stop because of complementarity                       | INTPOINT  | KEEPGOING_C=       |
| stop because of duality gap                           | INTPOINT  | KEEPGOING_DG=      |
| stop because of <i>infeas<sub>b</sub></i>             | INTPOINT  | KEEPGOING_IB=      |
| stop because of <i>infeas<sub>c</sub></i>             | INTPOINT  | KEEPGOING_IC=      |
| stop because of <i>infeas<sub>d</sub></i>             | INTPOINT  | KEEPGOING_ID=      |
| stop because of complementarity                       | INTPOINT  | AND_KEEPPGOING_C=  |
| stop because of duality gap                           | INTPOINT  | AND_KEEPPGOING_DG= |
| stop because of <i>infeas<sub>b</sub></i>             | INTPOINT  | AND_KEEPPGOING_IB= |
| stop because of <i>infeas<sub>c</sub></i>             | INTPOINT  | AND_KEEPPGOING_IC= |
| stop because of <i>infeas<sub>d</sub></i>             | INTPOINT  | AND_KEEPPGOING_ID= |

---

## Syntax

Below are statements used in PROC INTPOINT, listed in alphabetical order as they appear in the text that follows.

```
PROC INTPOINT options ;  
  CAPACITY variable ;  
  COEF variables ;  
  COLUMN variable ;  
  COST variable ;  
  DEMAND variable ;  
  HEADNODE variable ;  
  ID variables ;  
  LO variable ;  
  NAME variable ;  
  NODE variable ;  
  QUIT ;  
  RHS variable ;  
  ROW variables ;  
  RUN ;  
  SUPDEM variable ;  
  SUPPLY variable ;  
  TAILNODE variable ;  
  TYPE variable ;  
  VAR variables ;
```

Most of the statements of the INTPOINT procedure are variable lists. The variables specified in the variable lists must be present in one of three input SAS data sets:

|                                                                                                                                                                                                                                            |                         |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| <i>PROC INTPOINT options;</i>                                                                                                                                                                                                              | required statement      |
| <i>CAPACITY variable;</i><br><i>COST variable;</i><br><i>DEMAND variable;</i><br><i>HEADNODE variable;</i><br><i>ID variables;</i><br><i>LO variable;</i><br><i>NAME variable;</i><br><i>SUPPLY variable;</i><br><i>TAILNODE variable;</i> | optional ARCDATA lists  |
| <i>NODE variable;</i><br><i>SUPDEM variable;</i>                                                                                                                                                                                           | optional NODEDATA lists |
| <i>COEF variables;</i><br><i>COLUMN variable;</i><br><i>RHS variable;</i><br><i>ROW variables;</i><br><i>TYPE variable;</i><br><i>VAR variables;</i>                                                                                       | optional CONDATA lists  |
| <b>RUN;</b><br><b>QUIT;</b>                                                                                                                                                                                                                | optional statements     |



---

## PROC INTPOINT Statement

**PROC INTPOINT** *options* ;

This statement invokes the procedure.

### Overview of PROC INTPOINT Options

The following options can be specified in the PROC INTPOINT statement.

#### Data Set Options

This section briefly describes all the input and output data sets used by PROC INTPOINT. The ARCDATA= data set, the NODEDATA= data set, and the CONDATA= data set can contain SAS variables that have special names, for instance `_CAPAC_`, `_COST_`, and `_HEAD_`. PROC INTPOINT looks for such variables if you do not give explicit variable list specifications. If a SAS variable with a special name is found and that SAS variable is not in another variable list specification, PROC INTPOINT determines that values of the SAS variable are to be interpreted in a special way. By using SAS variables that have special names, you may not need to have any variable list specifications.

#### ARCDATA=SAS-data-set

names the data set that contains arc and, optionally, nonarc variable information and nodal supply/demand data. The ARCDATA= data set must be specified in all PROC INTPOINT statements when solving NPSC problems.

If your problem is an LP, the ARCDATA= data set is optional. You can specify LP variable information such as objective function coefficients, and lower and upper bounds.

#### CONDATA=SAS-data-set

names the data set that contains the side constraint data. The data set can also contain other data such as arc costs, capacities, lower flow bounds, nonarc variable upper and lower bounds, and objective function coefficients. PROC INTPOINT needs a CONDATA= data set to solve a constrained problem. See the “CONDATA= Data Set” section on page 133 for more information.

If your problem is an LP, this data set contains the constraint data, and can also contain other data such as objective function coefficients, and lower and upper bounds. PROC INTPOINT needs a CONDATA= data set to solve an LP.

#### CONOUT=SAS-data-set

#### CONOUT=SAS-data-set

names the output data set that receives an optimal solution. See the “CONOUT= Data Set” section on page 142 for more information.

If PROC INTPOINT is outputting observations to the output data set and you want this to stop, press the keys used to stop SAS procedures.

#### NODEDATA=SAS-data-set

names the data set that contains the node supply and demand specifications. You do not need observations in the NODEDATA= data set for transshipment nodes. (Trans-

shipment nodes neither supply nor demand flow.) All nodes are assumed to be transshipment nodes until supply or demand data indicate otherwise. It is acceptable for some arcs to be directed toward supply nodes or away from demand nodes.

This data set is used only when you are solving network problems (not when solving LP problems), in which case the use of the NODEDATA= data set is optional provided that, if the NODEDATA= data set is not used, supply and demand details are specified by other means. Other means include using the MAXFLOW or SHORT-PATH option, SUPPLY or DEMAND variable list (or both) in the ARCDATA= data set, and the SOURCE=, SUPPLY=, SINK=, or DEMAND= option in the PROC INTPOINT statement.

### **General Options**

The following is a list of options you can use with PROC INTPOINT. The options are listed in alphabetical order.

#### **ARCS\_ONLY\_ARCDATA**

indicates that data for arcs only are in the ARCDATA= data set. When PROC INTPOINT reads the data in the ARCDATA= data set, memory would not be wasted to receive data for nonarc variables. The read might then be performed faster. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 153.

#### **ARC\_SINGLE\_OBS**

indicates that for all arcs and nonarc variables, data for each arc or nonarc variable is found in only one observation of the ARCDATA= data set. When reading the data in the ARCDATA= data set, PROC INTPOINT knows that the data in an observation is for an arc or a nonarc variable that has not had data previously read and that needs to be checked for consistency. The read might then be performed faster.

When solving an LP, specifying the ARC\_SINGLE\_OBS option indicates that for all LP variables, data for each LP variable is found in only one observation of the ARCDATA= data set. When reading the data in the ARCDATA= data set, PROC INTPOINT knows that the data in an observation is for an LP variable that has not had data previously read and that needs to be checked for consistency. The read might then be performed faster.

If you specify ARC\_SINGLE\_OBS, PROC INTPOINT automatically works as if GROUPED=ARCDATA is also specified.

See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 153.

#### **BYPASSDIVIDE=*b***

#### **BYPASSDIV=*b***

#### **BPD=*b***

should be used only when the MAXFLOW option has been specified; that is, PROC INTPOINT is solving a maximal flow problem. PROC INTPOINT prepares to solve maximal flow problems by setting up a bypass arc. This arc is directed from the SOURCE= to the SINK= and will eventually convey flow equal to INFINITY minus the maximal flow through the network. The cost of the bypass arc must be great

enough to drive flow through the network, rather than through the bypass arc. Also, the cost of the bypass arc must be greater than the eventual total cost of the maximal flow, which can be nonzero if some network arcs have nonzero costs. The cost of the bypass is set to the value of the INFINITY= option. Valid values for the BYPASS-DIV= option must be greater than or equal to 1.1.

If there are no nonzero costs of arcs in the MAXFLOW problem, the cost of the bypass arc is set to 1.0 (-1.0 if maximizing) if you do not specify the BYPASSDIV= option. The default value for the BYPASSDIV= option (in the presence of nonzero arc costs) is 100.0.

#### **BYTES=b**

indicates the size of the main working memory (in bytes) that PROC INTPOINT will allocate. Specifying this option is mandatory. The working memory is used to store all the arrays and buffers used by PROC INTPOINT. If this memory has a size smaller than what is required to store all arrays and buffers, PROC INTPOINT uses various schemes that page information between auxiliary memory (often your machine's disk) and RAM.

For small problems, specify BYTES=100000. For large problems (those with hundreds of thousands or millions of variables), BYTES=1000000 might do. For solving problems of that size, if you are running on a machine with an inadequate amount of RAM, PROC INTPOINT's performance will suffer since it will be forced to page or to rely on virtual memory.

If you specify the MEMREP option, PROC INTPOINT will issue messages on the SAS log informing you of its memory usage; that is, how much memory is required to prevent paging, and details about the amount of paging that must be performed, if applicable.

#### **CON\_SINGLE\_OBS**

improves how the CONDATA= data set is read. How it works depends on whether the CONDATA has a dense or sparse format.

If the CONDATA= data set has the dense format, specifying CON\_SINGLE\_OBS indicates that, for each constraint, data for each can be found in only one observation of the CONDATA= data set.

If the CONDATA= data set has a sparse format, and data for each arc, nonarc variable, or LP variable can be found in only one observation of the CONDATA, then specify the CON\_SINGLE\_OBS option. If there are  $n$  SAS variables in the ROW and COEF list, then each arc or nonarc can have at most  $n$  constraint coefficients in the model. See the "How to Make the Data Read of PROC INTPOINT More Efficient" section on page 153.

#### **DEFCAPACITY=c**

##### **DC=c**

requests that the default arc capacity and the default nonarc variable value upper bound (or for LP problems, the default LP variable value upper bound) be  $c$ . If this option is not specified, then DEFCAPACITY= INFINITY.

**DEFCTYPE=*c*****DEFTYPE=*c*****DCT=*c***

specifies the default constraint type. This default constraint type is either *less than or equal to* or is the type indicated by DEFCTYPE=*c*. Valid values for this option are

LE, le, or <=     for *less than or equal to*

EQ, eq, or =       for *equal to*

GE, ge, or >=     for *greater than or equal to*

The values do not need to be enclosed in quotes.

**DEFCOST=*c***

requests that the default arc cost and the default nonarc variable objective function coefficient (or for an LP, the default LP variable objective function coefficient) be *c*. If this option is not specified, then DEFCOST=0.0.

**DEFMINFLOW=*m*****DMF=*m***

requests that the default lower flow bound through arcs and the default lower value bound of nonarc variables (or for an LP, the default lower value bound of LP variables) be *m*. If a value is not specified, then DEFMINFLOW=0.0.

**DEMAND=*d***

specifies the demand at the SINK node specified by the SINK= option. The DEMAND= option should be used only if the SINK= option is given in the PROC INTPOINT statement and neither the SHORTPATH option nor the MAXFLOW option is specified. If you are solving a minimum cost network problem and the SINK= option is used to identify the sink node, and the DEMAND= option is not specified, then the demand at the sink node is made equal to the network's total supply.

**GROUPED=*c***

PROC INTPOINT can take a much shorter time to read data if the data have been grouped prior to the PROC INTPOINT call. This enables PROC INTPOINT to conclude that, for instance, a new NAME list variable value seen in the ARCDATA= data set grouped by the values of the NAME list variable before PROC INTPOINT was called is new. PROC INTPOINT does not need to check that the NAME has been read in a previous observation. See the "How to Make the Data Read of PROC INTPOINT More Efficient" section on page 153.

- GROUPED=ARCDATA indicates that the ARCDATA= data set has been grouped by values of the NAME list variable. If `_NAME_` is the name of the NAME list variable, you could use:

```
proc sort data=arcdata;by _name_;
```

prior to calling PROC INTPOINT. Technically, you do not have to sort the data, only to ensure that all similar values of the NAME list variable are grouped together. If you specify the ARCS\_ONLY\_ARCDATA option, PROC INTPOINT automatically works as if GROUPED=ARCDATA is also specified.

- GROUPED=CONDATA indicates that the CONDATA= data set has been grouped.

If the CONDATA= data set has a dense format, GROUPED=CONDATA indicates that the CONDATA= data set has been grouped by values of the ROW list variable. If `_ROW_` is the name of the ROW list variable, you could use:

```
proc sort data=condata;by _row_;
```

prior to calling PROC INTPOINT. Technically, you do not have to sort the data, only to ensure that all similar values of the ROW list variable are grouped together. If you specify the CON\_SINGLE\_OBS option, or if there is no ROW list variable, PROC INTPOINT automatically works as if GROUPED=CONDATA has been specified.

If the CONDATA has the sparse format, GROUPED=CONDATA indicates that the CONDATA has been grouped by values of the COLUMN list variable. If `_COL_` is the name of the COLUMN list variable, you could use:

```
proc sort data=condata;by _col_;
```

prior to calling PROC INTPOINT. Technically, you do not have to sort the data, only to ensure that all similar values of the COLUMN list variable are grouped together.

- GROUPED=BOTH indicates that both GROUPED=ARCDATA and GROUPED=CONDATA are TRUE.
- GROUPED=NONE indicates that the data sets have not been grouped, that is, neither GROUPED=ARCDATA nor GROUPED=CONDATA is TRUE. This is the default, but it is much better if GROUPED=ARCDATA, or GROUPED=CONDATA, or GROUPED=BOTH.

A data set like

```
... _XXXXX_ ....
    bbb
    bbb
    aaa
    ccc
    ccc
```

is a candidate for the GROUPED= option. Similar values are grouped together. When PROC INTPOINT is reading the  $i$ th observation, either the value of the `_XXXXX_` variable is the same as the  $(i - 1)$ th (that is, the previous observation's) `_XXXXX_` value, or it is a new `_XXXXX_` value not seen in any previous observation. This also means that if the  $i$ th `_XXXXX_` value is different from the  $(i - 1)$ th `_XXXXX_` value, the value of the  $(i - 1)$ th `_XXXXX_` variable will not be seen in any observations  $i$ ,  $i + 1$ , ... .

**INFINITY=*i*****INF=*i***

is the largest number used by PROC INTPOINT in computations. A number too small can adversely affect the solution process. You should avoid specifying an enormous value for the INFINITY= option because numerical roundoff errors can result. If a value is not specified, then INFINITY=999999. The INFINITY= option cannot be assigned a value less than 9999.

**MAXFLOW****MF**

specifies that PROC INTPOINT solve a maximum flow problem. In this case, the PROC INTPOINT procedure finds the maximum flow from the node specified by the SOURCE= option to the node specified by the SINK= option. PROC INTPOINT automatically assigns an INFINITY= option supply to the SOURCE= option node and the SINK= option is assigned the INFINITY= option demand. In this way, the MAXFLOW option sets up a maximum flow problem as an equivalent minimum cost problem.

You can use the MAXFLOW option when solving any flow problem (not necessarily a maximum flow problem) when the network has one supply node (with infinite supply) and one demand node (with infinite demand). The MAXFLOW option can be used in conjunction with all other options (except SHORTPATH, SUPPLY=, and DEMAND=) and capabilities of PROC INTPOINT.

**MAXIMIZE****MAX**

specifies that PROC INTPOINT find the maximum cost flow through the network. If both the MAXIMIZE and the SHORTPATH options are specified, the solution obtained is the longest path between the SOURCE= and SINK= nodes. Similarly, MAXIMIZE and MAXFLOW together cause PROC INTPOINT to find the minimum flow between these two nodes; this is zero if there are no nonzero lower flow bounds. If solving an LP, specifying the MAXIMIZE option is necessary if you want the maximal optimal solution found instead of the minimal optimum.

**MEMREP**

indicates that information on the memory usage and paging schemes (if necessary) is reported by PROC INTPOINT on the SAS log.

**NAMECTRL=*i***

is used to interpret arc and nonarc variable names in the CONDATA= data set. In the ARCDATA= data set, an arc is identified by its tail and head node. In the CONDATA= data set, arcs are identified by names. You can give a name to an arc by having a NAME list specification that indicates a SAS variable in the ARCDATA= data set that has names of arcs as values.

PROC INTPOINT requires that arcs that have information about them in the CONDATA= data set have names, but arcs that do not have information about them in the CONDATA= data set can also have names. Unlike a nonarc variable whose name uniquely identifies it, an arc can have several different names. An arc has a default name in the form *tail\_head*, that is, the name of the arc's tail node followed by an underscore and the name of the arc's head node.

In the CONDATA= data set, if the dense data format is used (described in the “CONDATA= Data Set” section on page 133), a name of an arc or a nonarc variable is the *name* of a SAS variable listed in the VAR list specification. If the sparse data format of the CONDATA= data set is used, a name of an arc or a nonarc variable is a *value* of the SAS variable listed in the COLUMN list specification.

The NAMECTRL= option is used when a name of an arc or a nonarc variable in the CONDATA= data set (either a VAR list variable name or a value of the COLUMN list variable) is in the form *tail\_head* and there exists an arc with these end nodes. If *tail\_head* has not already been tagged as belonging to an arc or nonarc variable in the ARCDATA= data set, PROC INTPOINT needs to know whether *tail\_head* is the name of the arc or the name of a nonarc variable.

If you specify NAMECTRL=1, a name that is not defined in the ARCDATA= data set is assumed to be the name of a nonarc variable. NAMECTRL=2 treats *tail\_head* as the name of the arc with these endnodes, provided no other name is used to associate data in the CONDATA= data set with this arc. If the arc does have other names that appear in the CONDATA= data set, *tail\_head* is assumed to be the name of a nonarc variable. If you specify NAMECTRL=3, *tail\_head* is assumed to be a name of the arc with these end nodes, whether the arc has other names or not. The default value of NAMECTRL is 3.

If the dense format is used for the CONDATA= data set, there are two circumstances that affect how this data set is read:

1. if you are running SAS version 6, or a previous version to that, or if you are running SAS version 7 onward and you specify

**options validvarname=v6;**

in your SAS session. Let’s refer to this as *case 1*.

2. if you are running SAS version 7 onward and you do not specify

**options validvarname=v6;**

in your SAS session. Let’s refer to this as *case 2*.

If *case 1* (and the text in this and the five paragraphs that follow are relevant only for *case 1*), the SAS System converts SAS variable names in a SAS program to uppercase. The VAR list variable names are uppercased. Because of this, PROC INTPOINT automatically uppercases names of arcs and nonarc variables or LP variables (the values of the NAME list variable) in the ARCDATA= data set. The names of arcs and nonarc variables or LP variables (the values of the NAME list variable) appear uppercased in the CONOUT= data set.

Also, if the dense format is used for the CONDATA= data set, be careful with default arc names (names in the form *tailnode\_headnode*). Node names (values in the TAILNODE and HEADNODE list variables) in the ARCDATA= data set are not automatically uppercased by PROC INTPOINT. Consider the following code:



```

data arcdata;
    input _from_ $ _to_ $ _name $ ;
    datalines;
from to1 .
from to2 arc2
TAIL TO3 .
;
data densecon;
    input from_to1 from_to2 arc2 tail_to3;
    datalines;
2 3 5
;
proc intpoint
    arcdata=arcdata condata=densecon;
run;

```

The SAS System does not uppercase character string values within SAS data sets. PROC INTPOINT never uppercases node names, so the arcs in observations 1, 2, and 3 in the preceeding ARCDATA= data set have the default names `from_to1`, `from_to2`, and `TAIL_TO3`, respectively. When the dense format of the CONDATA= data set is used, PROC INTPOINT does uppercase values of the NAME list variable, so the name of the arc in the second observation of the ARCDATA= data set is `ARC2`. Thus, the second arc has two names; its default `from_to2` and the other that was specified `ARC2`.

As the SAS System uppercases program code, you must think of the input statement

```
input from_to1 from_to2 arc2 tail_to3;
```

as really being

```
INPUT FROM_TO1 FROM_TO2 ARC2 TAIL_TO3;
```

The SAS variables named `FROM_TO1` and `FROM_TO2` are *not* associated with any of the arcs in the preceeding ARCDATA= data set. The values `FROM_TO1` and `FROM_TO2` are different from all of the arc names `from_to1`, `from_to2`, `TAIL_TO3`, and `ARC2`. `FROM_TO1` and `FROM_TO2` could end up being the names of two nonarc variables.

The SAS variable named `ARC2` is the name of the second arc in the ARCDATA= data set, even though the name specified in the ARCDATA= data set looks like `arc2`. The SAS variable named `TAIL_TO3` is the default name of the third arc in the ARCDATA= data set.

If case 2, the SAS System does not convert SAS variable names in a SAS program to uppercase. The VAR list variable names are not uppercased. PROC INTPOINT does not automatically uppercase names of arcs and nonarc variables or LP variables (the values of the NAME list variable) in the ARCDATA= data set. PROC INTPOINT does not uppercase any SAS variable names, data set values, or indeed anything. Therefore, PROC INTPOINT respects case, and characters in the data if compared must have the right case if you mean them to be the same. Note how the input



statement in the data step that initialized the data set densecon below is specified in the following code:

```
data arcdata;
  input _from_ $ _to_ $ _name $ ;
  datalines;
from to1 .
from to2 arc2
TAIL T03 .
;
data densecon;
  input from_to1 from_to2 arc2 TAIL_T03;
  datalines;
2 3 5
;
proc intpoint
  arcdata=arcdata condata=densecon;
run;
```

**NARCS=*n***

specifies the approximate number of arcs. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 153.

**NCOEFS=*n***

specifies the approximate number of constraint coefficients. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 153.

**NCONS=*n***

specifies the approximate number of constraints. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 153.

**NNAS=*n***

specifies the approximate number of nonarc variables. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 153.

**NNODES=*n***

specifies the approximate number of nodes. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 153.

**NON\_REPLIC=*c***

prevents PROC INTPOINT from doing unnecessary checks of data previously read.

- NON\_REPLIC=COEFS indicates that each constraint coefficient is specified *once* in the CONDATA= data set.
- NON\_REPLIC=NONE indicates that constraint coefficients can be specified more than once in the CONDATA= data set. NON\_REPLIC=NONE is the default.

See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 153.

**RHSOBS=***charstr*

specifies the keyword that identifies a right-hand-side observation when using the sparse format for data in the CONDATA= data set. The keyword is expected as a value of the SAS variable in the CONDATA= data set named in the COLUMN list specification. The default value of the RHSOBS= option is `_RHS_` or `_rhs_`. If *charstr* is not a valid SAS variable name, enclose it in quotes.

**SCALE=***s*

indicates that the NPSC side constraints or the LP constraints are to be scaled. Scaling is useful when some coefficients are either much larger or much smaller than other coefficients. Scaling might make all coefficients have values that have a smaller range, and this can make computations more stable numerically. Try the SCALE= option if PROC INTPOINT is unable to solve a problem because of numerical instability. Specify

- SCALE=ROW, SCALE=CON, or SCALE=CONSTRAINT if you want the largest absolute value of coefficients in each constraint to be about 1.0
- SCALE=COL, SCALE=COLUMN, or SCALE=NONARC if you want NPSC nonarc variable columns or LP variable columns to be scaled so that the absolute value of the largest constraint coefficient of that variable is near to 1
- SCALE=BOTH if you want the largest absolute value of coefficients in each constraint, and the absolute value of the largest constraint coefficient of an NPSC nonarc variable or LP variable to be near to 1. This is the default.
- SCALE=NONE if no scaling should be done

**SHORTPATH****SP**

specifies that PROC INTPOINT solve a shortest path problem. The INTPOINT procedure finds the shortest path between the nodes specified in the SOURCE= option and the SINK= option. The costs of arcs are their *lengths*. PROC INTPOINT automatically assigns a supply of one flow unit to the SOURCE= node, and the SINK= node is assigned to have a one flow unit demand. In this way, the SHORTPATH option sets up a shortest path problem as an equivalent minimum cost problem.

If a network has one supply node (with supply of one unit) and one demand node (with demand of one unit), you could specify the SHORTPATH option, with the SOURCE= and SINK= nodes, even if the problem is not a shortest path problem. You then should not provide any supply or demand data in the NODEDATA= data set or the ARCDATA= data set.

**SINK=***sinkname***SINKNODE=***sinkname*

identifies the demand node. The SINK= option is useful when you specify the MAXFLOW option or the SHORTPATH option and you need to specify toward which node the shortest path or maximum flow is directed. The SINK= option also can be used when a minimum cost problem has only one demand node. Rather than having this information in the ARCDATA= data set or the NODEDATA= data set, use the SINK= option with an accompanying DEMAND= specification for this node.

The SINK= option must be the name of a head node of at least one arc; thus, it must have a character value. If the value of the SINK= option is not a valid SAS character variable name (if, for example, it contains embedded blanks), it must be enclosed in quotes.

**SOURCE=***sourcename*

**SOURCENODE=***sourcename*

identifies a supply node. The SOURCE= option is useful when you specify the MAXFLOW or the SHORTPATH option and need to specify from which node the shortest path or maximum flow originates. The SOURCE= option also can be used when a minimum cost problem has only one supply node. Rather than having this information in the ARCDATA= data set or the NODEDATA= data set, use the SOURCE= option with an accompanying SUPPLY= amount of supply at this node. The SOURCE= option must be the name of a tail node of at least one arc; thus, it must have a character value. If the value of the SOURCE= option is not a valid SAS character variable name (if, for example, it contains embedded blanks), it must be enclosed in quotes.

**SPARSECONDATA**

**SCDATA**

indicates that the CONDATA= data set has data in the sparse data format. Otherwise, it is assumed that the data are in the dense format.

**Note:** If the SPARSECONDATA option is not specified, and you are running SAS software Version 6 or you have specified

**options validvarname=v6;**

all NAME list variable values in the ARCDATA= data set are uppercased. See the “Case Sensitivity” section on page 143.

**SUPPLY=s**

specifies the supply at the source node specified by the SOURCE= option. The SUPPLY= option should be used only if the SOURCE= option is given in the PROC INTPOINT statement and neither the SHORTPATH option nor the MAXFLOW option is specified. If you are solving a minimum cost network problem and the SOURCE= option is used to identify the source node and the SUPPLY= option is not specified, then by default the supply at the source node is made equal to the network’s total demand.

**THRUNET**

tells PROC INTPOINT to force through the network any excess supply (the amount by which total supply exceeds total demand) or any excess demand (the amount by which total demand exceeds total supply) as is required. If a network problem has unequal total supply and total demand and the THRUNET option is not specified, PROC INTPOINT drains away the excess supply or excess demand in an optimal manner. The consequences of specifying or not specifying THRUNET are discussed in the “Balancing Total Supply and Total Demand” section on page 151.

**TYPEOBS=charstr**

specifies the keyword that identifies a type observation when using the sparse format for data in the CONDATA= data set. The keyword is expected as a value of the SAS variable in the CONDATA= data set named in the COLUMN list specification. The default value of the TYPEOBS= option is `_TYPE_` or `_type_`. If *charstr* is not a valid SAS variable name, enclose it in quotes.

**Options to Halt Optimization****VERBOSE=v**

limits the number of similar messages that are displayed on the SAS log.

For example, when reading the ARCDATA= data set, PROC INTPOINT might have cause to issue the following message many times:

**ERROR: The HEAD list variable value in obs i in the ARCDATA is missing, - the TAIL list variable value of this obs is nonmissing. This is an incomplete arc specification.**

If there are lots of observations that have this fault, messages that are similar are issued for only the first VERBOSE= such observations. After the ARCDATA= data set has been read, PROC INTPOINT will issue the message

**NOTE: More messages similar to the ones immediately above could have been issued but were suppressed as VERBOSE= v.**

If observations in the ARCDATA= data set have this error, PROC INTPOINT stops and you have to fix the data. Imagine that this error is only a warning and PROC INTPOINT proceeded to other operations such as reading the CONDATA= data set. If PROC INTPOINT finds there are numerous errors when reading that data set, the number of messages issued to the SAS log are also limited by the VERBOSE= option.

When PROC INTPOINT finishes and messages have been suppressed, the message

**NOTE: To see all messages, specify VERBOSE=v.**

is issued. The value of *v* is the smallest value that should be specified for the VERBOSE= option so that *all* messages are displayed if PROC INTPOINT is run again with the same data and everything else.

The default value for the VERBOSE= option is 12.

**ZERO2=z****Z2=z**

specifies the zero tolerance level used when determining whether the final solution has been reached. ZERO2= is also used when outputting the solution to the CONOUT= data set. Values within *z* of zero are set to 0.0, where *z* is the value of the ZERO2= option. Flows close to the lower flow bound or capacity of arcs are re-assigned those exact values. If there are nonarc variables, values close to the lower or upper value bound of nonarc variables are re-assigned those exact values. When solving an LP problem, values close to the lower or upper value bound of LP variables are re-assigned those exact values.

The ZERO2= option works when determining whether optimality has been reached or whether an element in the vector  $(\Delta x^k, \Delta y^k, \Delta s^k)$  is less than or greater than zero. It is crucial to know that when determining the maximal value for the step length  $\alpha$  in the formula

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

See the description of the PDSTEPMULT= option for more details on this computation.

Two values are deemed to be close if one is within  $z$  of the other. The default value for the ZERO2= option is 0.000001. Any value specified for the ZERO2= option that is  $< 0.0$  or  $> 0.0001$  is not valid.

#### **ZEROTOL=z**

specifies the zero tolerance used when PROC INTPOINT must compare any real number with another real number, or zero. For example, if  $x$  and  $y$  are real numbers, then for  $x$  to be considered greater than  $y$ ,  $x$  must be at least  $y + \text{ZEROTOL}$ . The ZEROTOL= option is used throughout any PROC INTPOINT run.

ZEROTOL=z controls the way PROC INTPOINT performs all double precision comparisons; that is, whether a double precision number is equal to, not equal to, greater than (or equal to), or less than (or equal to) zero or some other double precision number. A double precision number is deemed to be the same as another such value if the absolute differences between them is less than or equal to the value of the ZEROTOL= option.

The default value for the ZEROTOL= option is 1.0E-14. You can specify the ZEROTOL= option in the INTPOINT statement. Valid values for the ZEROTOL= option must be  $> 0.0$  and  $< 0.0001$ . Do not specify a value too close to zero as this defeats the purpose of the ZEROTOL= option. Neither should the value be too large, as comparisons might be incorrectly performed.

#### **TOLDINF=t**

##### **RTOLDINF=t**

specifies the allowed amount of dual infeasibility. In the “Interior Point Algorithmic Details” section on page 66, the vector  $infeas_d$  is defined. If all elements of this vector are  $\leq t$ , the solution is considered dual feasible.  $infeas_d$  is replaced by a zero vector, which makes computations faster. This option is the dual equivalent to the TOLPINF= option. Increasing the value of the TOLDINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are between  $1.0E - 12$  and  $1.0E - 1$ . The default is  $1.0E - 7$ .

#### **TOLPINF=t**

##### **RTOLPINF=t**

specifies the allowed amount of primal infeasibility. This option is the dual equivalent to the TOLDINF= option. In the “Interior Point: Upper Bounds” section on page 71, the vector  $infeas_b$  is defined. In the “Interior Point Algorithmic Details” section on page 66, the vector  $infeas_c$  is defined. If all elements in these vectors are  $\leq t$ , the solution is considered primal feasible.  $infeas_b$  and  $infeas_c$  is replaced by a zero

vectors, which makes computations faster. Increasing the value of the TOLPINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are between  $1.0E - 12$  and  $1.0E - 1$ . The default is  $1.0E - 7$ .

**TOLTOTDINF= $t$**

**RTOLTOTDINF= $t$**

specifies the allowed total amount of dual infeasibility. In the “Interior Point Algorithmic Details” section on page 66, the vector  $infeas_d$  is defined. If  $\sum_{i=1}^n infeas_{di} \leq t$ , the solution is considered dual feasible.  $infeas_d$  is replaced by a zero vector, which makes computations faster. This option is the dual equivalent to the TOLTOTPINF= option. Increasing the value of the TOLTOTDINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are between  $1.0E - 12$  and  $1.0E - 1$ . The default is  $1.0E - 7$ .

**TOLTOTPINF= $t$**

**RTOLTOTPINF= $t$**

specifies the allowed total amount of primal infeasibility. This option is the dual equivalent to the TOLTOTDINF= option. In the “Interior Point: Upper Bounds” section on page 71, the vector  $infeas_b$  is defined. In the “Interior Point Algorithmic Details” section on page 66, the vector  $infeas_c$  is defined. If  $\sum_{i=1}^n infeas_{bi} \leq t$  and  $\sum_{i=1}^m infeas_{ci} \leq t$ , the solution is considered primal feasible.  $infeas_b$  and  $infeas_c$  are replaced by a zero vector, which makes computations faster. Increasing the value of the TOLTOTPINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are between  $1.0E - 12$  and  $1.0E - 1$ . The default is  $1.0E - 7$ .

**CHOLTINYTOL= $c$**

**RCHOLTINYTOL= $c$**

specifies the cut-off tolerance for Cholesky factorization of the  $A\Theta A^{-1}$ . If a diagonal value drops below  $c$ , the row is essentially treated as dependent and is ignored in the factorization. Valid values for  $c$  are between  $1.0E - 30$  and  $1.0E - 6$ . The default value is  $1.0E - 8$ .

**DENSETHR= $d$**

**RDENSETHR= $d$**

specifies the density threshold for Cholesky factorization. When the symbolic factorization encounters a column of  $L$  (where  $L$  is the remaining unfactorized submatrix) that has DENSETHR= proportion of nonzeros and the remaining part of  $L$  is at least  $12 \times 12$ , the remainder of  $L$  is treated as dense. In practice, the lower right part of the Cholesky triangular factor  $L$  is quite dense and it can be computationally more efficient to treat it as 100% dense. The default value for  $d$  is 0.7. A specification of  $d \leq 0.0$  causes all dense processing;  $d \geq 1.0$  causes all sparse processing.

**PDSTEPMULT= $p$** **RPDSTEPMULT= $p$** 

specifies the step-length multiplier. The maximum feasible step-length chosen by the Interior point algorithm is multiplied by the value of the PDSTEPMULT= option. This number must be less than 1 to avoid moving beyond the barrier. An actual step-length greater than 1 indicates numerical difficulties. Valid values for  $p$  are between 0.01 and 0.999999. The default value is 0.99995.

In the “Interior Point Algorithmic Details” section on page 66, the solution of the next iteration is obtained by moving along a direction from the current iterations solution.

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

Let  $\alpha = \text{Min}_i \{\alpha : x_i^k + \alpha \Delta x = 0 \mid s_i^k + \alpha \Delta s = 0\}$ . If  $\alpha \leq 1$ , then  $\alpha = p\alpha$ .  $\alpha$  is a value as large as possible but  $\leq 1.0$  and not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  of some variable  $i$  is not “too close” to zero.

**PRSLTYPE= $p$** **IPRSLTYPE= $p$** 

Preprocessing the Linear Programming problem often succeeds in allowing some variables and constraints to be temporarily eliminated from the resulting LP that must be solved. This reduces the solution time and possibly also the chance that the optimizer will run into numerical difficulties. The task of preprocessing is inexpensive to do.

You control how much preprocessing to do by specifying the PRSLTYPE=  $p$ , where  $p$  can be -1, 0, 1, 2, or 3:

- 1 Do not perform preprocessing. For most problems, specifying PRSLTYPE= -1 is *not* recommended.
- 0 Given upper and lower bounds on each variable, the greatest and least contribution to the row activity of each variable is computed. If these are within the limits set by the upper and lower bounds on the row activity, then the row is redundant and can be discarded. Otherwise, whenever possible, the bounds on any of the variables are tightened. For example, if all coefficients in a constraint are positive and all variables have zero lower bounds, then the row’s smallest contribution is zero. If the rhs value of this constraint is zero, then if the constraint type is = or  $\leq$ , all the variables in that constraint are fixed to zero. These variables and the constraint are removed. If the constraint type is  $\geq$ , the constraint is redundant. If the rhs is negative and the constraint is  $\leq$ , the problem is infeasible. If just one variable in a row is not fixed, the row is used to impose an implicit upper or lower bound on the variable and then this row is eliminated. The preprocessor also tries to tighten the bounds on constraint right hand sides. PRSLTYPE=0 is the default.
- 1 When there are exactly two unfixed variables with coefficients in an equality constraint, one variable is solved in terms of the other. The problem will have one less variable. The new matrix will have

at least two fewer coefficients and one less constraint. In other constraints where both variables appear, two coefficients are combined into one. PRSLTYPE=0 reductions are also done.

- 2 It may be possible to determine that an equality constraint is not constraining a variable. That is, if all variables are nonnegative, then  $x - \sum_i y_i = 0$  does not constrain  $x$ , since it must be nonnegative if all the  $y_i$ 's are nonnegative. In this case,  $x$  is eliminated by subtracting this equation from all others containing  $x$ . This is useful when the only other entry for  $x$  is in the objective function. This reduction is performed if there is at most one other nonobjective coefficient. PRSLTYPE=0 reductions are also done.
- 3 All possible reductions are performed.

Preprocessing is iterative. As variables are fixed and eliminated, and constraints are found to be redundant and they too are eliminated, and as variable bounds and constraint right hand sides are tightened, the LP to be optimized is modified to reflect these changes. Another iteration of preprocessing of the modified LP may reveal more variables and constraints that are eliminated, or tightened.

#### PRINTLEVEL2= $p$

is used when you want to see PROC INTPOINT's progress to the optimum. PROC INTPOINT will produce a table on the SAS log. A row of the table is generated during each iteration and may consist of values of

- the affine step complementarity
- the complementarity of the solution for the next iteration
- the total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the "Interior Point: Upper Bounds" section on page 71)
- the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the "Interior Point Algorithmic Details" section on page 66)
- the total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the "Interior Point Algorithmic Details" section on page 66)

As optimization progresses, the values in all columns should converge to zero. If you specify PRINTLEVEL2=2, all columns will appear in the table. If PRINTLEVEL2=1 is specified, only the affine step complementarity, the complementarity of the solution for the next iteration, will appear. Some time is saved by not calculating the infeasibility values.

PRINTLEVEL2=2 is specified in all PROC INTPOINT runs in the "Examples" section on page 162.



**Interior Point Algorithm Options: Stopping Criteria****MAXITERB= $m$** **IMAXITERB= $m$** 

specifies the maximum number of iterations that the Interior point algorithm can perform. The default value for  $m$  is 100. One of the most remarkable aspects of the Interior Point algorithm is that for most problems, it usually needs to do a small number of iterations, *no matter the size of the problem*.

**PDGAPTOL= $p$** **RPDGAPTOL= $p$** 

specifies the Primal-Dual gap or Duality gap tolerance. Duality gap is defined in the “Interior Point Algorithmic Details” section on page 66. If the relative gap ( $dualitygap/(c^T x)$ ) between the primal and dual objectives is smaller than the value of the PDGAPTOL= option and both the primal and dual problems are feasible, then PROC INTPOINT stops optimization with a solution that is deemed optimal. Valid values for  $p$  are between  $1.0E - 12$  and  $1.0E - 1$ . the default is  $1.0E - 7$ .

**STOP\_C= $s$** 

is used to determine whether optimization should stop. At the beginning of each iteration, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

**STOP\_DG= $s$** 

is used to determine whether optimization should stop. At the beginning of each iteration, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

**STOP\_IB= $s$** 

is used to determine whether optimization should stop. At the beginning of each iteration, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 71; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

**STOP\_IC= $s$** 

is used to determine whether optimization should stop. At the beginning of each iteration, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 66; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

**STOP\_ID= $s$** 

is used to determine whether optimization should stop. At the beginning of each iteration, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 66; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , opti-

mization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

#### **AND\_STOP\_C=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

#### **AND\_STOP\_DG=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

#### **AND\_STOP\_IB=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 71; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

#### **AND\_STOP\_IC=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 66; this value appears in the Tot\_infeasc column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

#### **AND\_STOP\_ID=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 66; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 158.

#### **KEEPGOING\_C=s**

is used to determine whether optimization should stop. If a stopping condition is met, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**KEEPPGOING\_DG=s**

is used to determine whether optimization should stop. If a stopping condition is met, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**KEEPPGOING\_IB=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 71; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**KEEPPGOING\_IC=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 66; this value appears in the Tot\_infeasc column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**KEEPPGOING\_ID=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 66; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**AND\_KEEPPGOING\_C=s**

is used to determine whether optimization should stop. If a stopping condition is met, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , *and* the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**AND\_KEEPPGOING\_DG=s**

is used to determine whether optimization should stop. If a stopping condition is met, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , *and* the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**AND\_KEEPPGOING\_IB=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 71; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , *and* the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**AND\_KEEPPGOING\_IC=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 66; this value appears in the Tot\_infeasc column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

**AND\_KEEPPGOING\_ID=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 66; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 158.

---

**CAPACITY Statement**

**CAPACITY** *variable* ;

**CAPAC** *variable* ;

**UPPERBD** *variable* ;

The CAPACITY statement identifies the SAS variable in the ARCDATA= data set that contains the maximum feasible flow or capacity of the network arcs. If an observation contains nonarc variable information, the CAPACITY list variable is the upper value bound for the nonarc variable named in the NAME list variable in that observation.

When solving an LP, the CAPACITY statement identifies the SAS variable in the ARCDATA= data set that contains the maximum feasible value of the LP variables.

The CAPACITY list variable must have numeric values. It is not necessary to have a CAPACITY statement if the name of the SAS variable is \_CAPAC\_, \_UPPER\_, \_UPPERBD, or \_HI\_.

---

**COEF Statement**

**COEF** *variables* ;

The COEF list is used with the sparse input format of the CONDATA= data set. The COEF list can contain more than one SAS variable, each of which must have numeric values. If the COEF statement is not specified, the CONDATA= data set is searched and SAS variables with names beginning with \_COE are used. The number of SAS

variables in the COEF list must be no greater than the number of SAS variables in the ROW list.

The values of the COEF list variables in an observation can be interpreted differently than these variables' values in other observations. The values can be coefficients in the side constraints, costs and objective function coefficients, bound data, constraint type data, or rhs data. If the COLUMN list variable has a value that is a name of an arc or a nonarc variable, the *i*th COEF list variable is associated with the constraint or special row name named in the *i*th ROW list variable. Otherwise, the COEF list variables indicate type values, rhs values, or missing values.

When solving an LP, the values of the COEF list variables in an observation can be interpreted differently than these variables' values in other observations. The values can be coefficients in the constraints, objective function coefficients, bound data, constraint type data, or rhs data. If the COLUMN list variable has a value that is a name of an LP variable, the *i*th COEF list variable is associated with the constraint or special row name named in the *i*th ROW list variable. Otherwise, the COEF list variables indicate type values, rhs values, or missing values.

---

## COLUMN Statement

**COLUMN** *variable* ;

The COLUMN list is used with the sparse input format of the CONDATA= data set.

This list consists of one SAS variable in the CONDATA= data set that has as values the names of arc variables, nonarc variables, or missing values. When solving an LP, this list consists of one SAS variable in the CONDATA= data set that has as values the names of LP variables, or missing values. Some, if not all, of these values also can be values of the NAME list variables of the ARCDATA= data set. The COLUMN list variable can have other special values (Refer to the TYPEOBS= and RHSOBS= options). If the COLUMN list is not specified after the PROC INTPOINT statement, the CONDATA= data set is searched and a SAS variable named `_COLUMN_` is used. The COLUMN list variable must have character values.

---

## COST Statement

**COST** *variable* ;  
**OBJFN** *variable* ;

The COST statement identifies the SAS variable in the ARCDATA= data set that contains the per unit flow cost through an arc. If an observation contains nonarc variable information, the value of the COST list variable is the objective function coefficient of the nonarc variable named in the NAME list variable in that observation.

If solving an LP, the COST statement identifies the SAS variable in the ARCDATA= data set that contains the per unit objective function coefficient of an LP variable named in the NAME list variable in that observation.

The COST list variable must have numeric values. It is not necessary to specify a COST statement if the name of the SAS variable is `_COST_` or `_LENGTH_`.

---

## DEMAND Statement

**DEMAND** *variable* ;

The DEMAND statement identifies the SAS variable in the ARCDATA= data set that contains the demand at the node named in the corresponding HEADNODE list variable. The DEMAND list variable must have numeric values. It is not necessary to have a DEMAND statement if the name of this SAS variable is `_DEMAND_`. See the “Missing S Supply and Missing D Demand Values” section on page 146 for cases when the SUPDEM list variable values can have other values. There should be no DEMAND statement if you are solving an LP.

---

## HEADNODE Statement

**HEADNODE** *variable* ;

**HEAD** *variable* ;

**TONODE** *variable* ;

**TO** *variable* ;

The HEADNODE statement specifies the SAS variable that must be present in the ARCDATA= data set that contains the names of nodes toward which arcs are directed. It is not necessary to have a HEADNODE statement if the name of the SAS variable is `_HEAD_` or `_TO_`. The HEADNODE variable must have character values.

There should be no HEAD statement if you are solving an LP.

---

## ID Statement

**ID** *variables* ;

The ID statement specifies SAS variables containing values for pre- and post-optimal processing and analysis. These variables are not processed by PROC INTPOINT but are read by the procedure and written in the CONOUT= data set. For example, imagine a network used to model a distribution system. The SAS variables listed on the ID statement can contain information on the type of vehicle, the transportation mode, the condition of the road, the time to complete the journey, the name of the

driver, or other ancillary information useful for report writing or describing facets of the operation that do not have bearing on the optimization. The ID variables can be character, numeric, or both.

If no ID list is specified, the procedure forms an ID list of all SAS variables not included in any other implicit or explicit list specification. If the ID list is specified, any SAS variables in the ARCDATA= data set not in any list are dropped and do not appear in the CONOUT= data set.

---

## LO Statement

```
LO variable ;
LOWERBD variable ;

MINFLOW variable ;
```

The LO statement identifies the SAS variable in the ARCDATA= data set that contains the minimum feasible flow or lower flow bound for arcs in the network. If an observation contains nonarc variable information, the LO list variable has the value of the lower bound for the nonarc variable named in the NAME list variable. If solving an LP, the LO statement identifies the SAS variable in the ARCDATA= data set that contains the lower value bound for LP variables. The LO list variables must have numeric values. It is not necessary to have a LO statement if the name of this SAS variable is \_LOWER\_, \_LO\_, \_LOWERBD, or \_MINFLOW.

---

## NAME Statement

```
NAME variable ;
ARCNAME variable ;

VARNAME variable ;
```

Each arc and nonarc variable in an NPSC, or each variable in an LP, that has data in the CONDATA= data set must have a unique name. This name is a value of the NAME list variable. The NAME list variable must have character values (see the NAMECTRL= option in the PROC INTPOINT statement for more information). It is not necessary to have a NAME statement if the name of this SAS variable is \_NAME\_.

---

## NODE Statement

**NODE** *variable* ;

The NODE list variable, which must be present in the NODEDATA= data set, has names of nodes as values. These values must also be values of the TAILNODE list variable, the HEADNODE list variable, or both. If this list is not explicitly specified, the NODEDATA= data set is searched for a SAS variable with the name `_NODE_`. The NODE list variable must have character values.

---

## QUIT Statement

**QUIT**;

The QUIT statement indicates that PROC INTPOINT is to stop immediately. The solution is not saved in the CONOUT= data set. The QUIT statement has no options.

---

## RHS Statement

**RHS** *variable* ;

The RHS variable list is used when the dense format of the CONDATA= data set is used. The values of the SAS variable specified in the RHS list are constraint right-hand-side values. If the RHS list is not specified, the CONDATA= data set is searched and a SAS variable with the name `_RHS_` is used. The RHS list variable must have numeric values. If there is no RHS list and no SAS variable named `_RHS_`, all constraints are assumed to have zero right-hand-side values.

---

## ROW Statement

**ROW** *variables* ;

The ROW list is used when either the sparse or dense format of the CONDATA= data set is being used. SAS variables in the ROW list have values that are constraint or special row names. The SAS variables in the ROW list must have character values.

If the dense data format is used, there must be only one SAS variable in this list. In this case, if a ROW list is not specified, the CONDATA= data set is searched and the SAS variable with the name `_ROW_` or `_CON_` is used. If that search failed to find a suitable SAS variable, data for each constraint must reside in only one observation.

If the sparse data format is used and the ROW statement is not specified, the CONDATA= data set is searched and SAS variables with names beginning with `_ROW` or `_CON` are used. The number of SAS variables in the ROW list must not be less than



the number of SAS variables in the COEF list. The  $i$ th ROW list variable is paired with the  $i$ th COEF list variable. If the number of ROW list variables is greater than the number of COEF list variables, the last ROW list variables have no COEF partner. These ROW list variables that have no corresponding COEF list variable are used in observations that have a TYPE list variable value. All ROW list variable values are tagged as having the type indicated. If there is no TYPE list variable, all ROW list variable values are constraint names.

---

## RUN Statement

**RUN;**

The RUN statement causes optimization to be started. The RUN statement has no options. If PROC INTPOINT is called and is not terminated because of an error or a QUIT statement, and you have not used a RUN statement, a RUN statement is assumed implicitly as the last statement of PROC INTPOINT. Therefore, PROC INTPOINT reads that data, performs optimization, and saves the optimal solution in the CONOUT= data set.

---

## SUPDEM Statement

**SUPDEM** *variable* ;

The SAS variable in this list, which must be present in the NODEDATA= data set, contains supply and demand information for the nodes in the NODE list. A positive SUPDEM list variable value  $s$  ( $s > 0$ ) denotes that the node named in the NODE list variable can supply  $s$  units of flow. A negative SUPDEM list variable value  $-d$  ( $d > 0$ ) means that this node demands  $d$  units of flow. If a SAS variable is not explicitly specified, a SAS variable with the name `_SUPDEM_` or `_SD_` in the NODEDATA= data set is used as the SUPDEM variable. If a node is a transshipment node (neither a supply nor a demand node), an observation associated with this node need not be present in the NODEDATA= data set. If present, the SUPDEM list variable value must be zero or a missing value. See the “Missing S Supply and Missing D Demand Values” section on page 146 for cases when the SUPDEM list variable values can have other values.

---

## SUPPLY Statement

**SUPPLY** *variable* ;

The SUPPLY statement identifies the SAS variable in the ARCDATA= data set that contains the supply at the node named in that observation’s TAILNODE list variable. If a tail node does not supply flow, use zero or a missing value for the observation’s SUPPLY list variable value. If a tail node has supply capability, a missing value

indicates that the supply quantity is given in another observation. It is not necessary to have a SUPPLY statement if the name of this SAS variable is `_SUPPLY_`. See the “Missing S Supply and Missing D Demand Values” section on page 146 for cases when the SUPDEM list variable values can have other values. There should be no SUPPLY statement if you are solving an LP.

---

## TAILNODE Statement

```

TAILNODE variable ;
TAIL variable ;

FROMNODE variable ;

FROM variable ;

```

The TAILNODE statement specifies the SAS variable that must (when solving an NPSC problem) be present in the ARCDATA= data set that has as values the names of tail nodes of arcs. The TAILNODE variable must have character values. It is not necessary to have a TAILNODE statement if the name of the SAS variable is `_TAIL_` or `_FROM_`. If the TAILNODE list variable value is missing, it is assumed that the observation of the ARCDATA= data set contains information concerning a nonarc variable. There should be no TAILNODE statement if you are solving an LP.

---

## TYPE Statement

```

TYPE variable ;
CONTYPE variable ;

```

The TYPE list, which is optional, names the SAS variable that has as values keywords that indicate either the constraint type for each constraint or the type of special rows in the CONDATA= data set. The values of the TYPE list variable also indicate, in each observation of the CONDATA= data set, how values of the VAR or COEF list variables are to be interpreted and how the type of each constraint or special row name is determined. If the TYPE list is not specified, the CONDATA= data set is searched and a SAS variable with the name `_TYPE_` is used. Valid keywords for the TYPE variable are given below. If there is no TYPE statement and no other method is used to furnish type information (see the DEFCONTYPE= option), all constraints are assumed to be of the type “less than or equal to” and no special rows are used. The TYPE list variable must have character values and can be used when the data in the CONDATA= data set is in either the sparse or dense format. If the TYPE list variable value has a \* as its first character, the observation is ignored because it is a comment observation.

**TYPE List Variable Values**

The following are valid TYPE list variable values. The letters in boldface denote the characters that PROC INTPOINT uses to determine what type the value suggests. You need to have at least these characters. Below, the minimal TYPE list variable values have additional characters to aid you in remembering these values.

The valid TYPE list variable values are

|                         |                                                                                |
|-------------------------|--------------------------------------------------------------------------------|
| <                       | less than or equal to ( $\leq$ )                                               |
| =                       | equal to (=)                                                                   |
| >                       | greater than or equal to ( $\geq$ )                                            |
| <b>CAPAC</b>            | capacity                                                                       |
| <b>COST</b>             | cost                                                                           |
| <b>EQ</b>               | equal to                                                                       |
| <b>FREE</b>             | free row (used only for Linear Programs solved by Interior Point)              |
| <b>GE</b>               | greater than or equal to                                                       |
| <b>LE</b>               | less than or equal to                                                          |
| <b>LOWERBD</b>          | lower flow or value bound                                                      |
| <b>LOW</b> <i>blank</i> | lower flow or value bound                                                      |
| <b>MAXIMIZE</b>         | maximize (opposite of cost)                                                    |
| <b>MINIMIZE</b>         | minimize (same as cost)                                                        |
| <b>OBJECTIVE</b>        | objective function (same as cost)                                              |
| <b>RHS</b>              | rhs of constraint                                                              |
| <b>TYPE</b>             | type of constraint                                                             |
| <b>UPPCOST</b>          | reserved for future use                                                        |
| <b>UNREST</b>           | unrestricted variable (used only for Linear Programs solved by Interior Point) |
| <b>UPPER</b>            | upper value bound or capacity; second letter must not be N                     |

The valid TYPE list variable values in function order are

- **LE** less than or equal to ( $\leq$ )
- **EQ** Equal to (=)
- **GE** Greater than or equal to ( $\geq$ )
- **COST**  
**MINIMIZE**  
**MAXIMIZE**  
**OBJECTIVE**  
cost or objective function coefficient
- **CAPAC**  
**UPPER** capacity or upper value bound

- **LOWERBD**  
**LOW***blank* Lower flow or value bound
- **RHS** rhs of constraint
- **TYPE** type of constraint

A TYPE list variable value that has the first character \* causes the observation to be treated as a comment. If the first character is a negative sign, then  $\leq$  is the type. If the first character is a zero, then = is the type. If the first character is a positive number, then  $\geq$  is the type.

---

## VAR Statement

**VAR** *variables* ;

The VAR variable list is used when the dense data format is used for the CONDATA= data set. The names of these SAS variables are also names of the arc and nonarc variables that have data in the CONDATA= data set. If solving an LP, the names of these SAS variables are also names of the LP variables. If no explicit VAR list is specified, all numeric SAS variables in the CONDATA= data set that are not in other SAS variable lists are put onto the VAR list. The VAR list variables must have numeric values. The values of the VAR list variables in some observations can be interpreted differently than in other observations. The values can be coefficients in the side constraints, costs and objective function coefficients, or bound data. When solving an LP, the values of the SAS variables in the VAR list can be constraint coefficients, objective function coefficients, or bound data. How these numeric values are interpreted depends on the value of each observation's TYPE or ROW list variable value. If there are no TYPE list variables, the VAR list variable values are all assumed to be side constraint coefficients.

## Details

### Input Data Sets

PROC INTPOINT is designed so that there are as few rules as possible that you must obey when inputting a problem's data. Raw data are acceptable. This should cut the amount of processing required to groom the data before it is input to PROC INTPOINT. Data formats are so flexible that, due to space restrictions, all possible forms for a problem's data are not shown here. Try any reasonable form for your problem's data; it should be acceptable. PROC INTPOINT will outline its objections.

You can supply the same piece of data several ways. You do not have to restrict yourself to using any particular one. If you use several ways, PROC INTPOINT checks that the data are consistent each time that the data are encountered. After all input data sets have been read, data are merged so that the problem is described completely. The observations can be in any order.

#### **ARCDATA= Data Set**

See the “Getting Started: NPSC Problems” section on page 82 and the “Introductory NPSC Example” section on page 83 for a description of this input data set.

**Note:** Information for an arc or nonarc variable can be specified in more than one observation. For example, consider an arc directed from node A toward node B that has a cost of 50, capacity of 100, and lower flow bound of 10 flow units. Some possible observations in the ARCDATA= data set are

| <b>_tail_</b> | <b>_head_</b> | <b>_cost_</b> | <b>_capac_</b> | <b>_lo_</b> |
|---------------|---------------|---------------|----------------|-------------|
| <b>A</b>      | <b>B</b>      | <b>50</b>     | <b>.</b>       | <b>.</b>    |
| <b>A</b>      | <b>B</b>      | <b>.</b>      | <b>100</b>     | <b>.</b>    |
| <b>A</b>      | <b>B</b>      | <b>.</b>      | <b>.</b>       | <b>10</b>   |
| <b>A</b>      | <b>B</b>      | <b>50</b>     | <b>100</b>     | <b>.</b>    |
| <b>A</b>      | <b>B</b>      | <b>.</b>      | <b>100</b>     | <b>10</b>   |
| <b>A</b>      | <b>B</b>      | <b>50</b>     | <b>.</b>       | <b>10</b>   |
| <b>A</b>      | <b>B</b>      | <b>50</b>     | <b>100</b>     | <b>10</b>   |

Similarly, for a nonarc variable that has an upper bound of 100, a lower bound of 10, and an objective function coefficient of 50, the **\_TAIL\_** and **\_HEAD\_** values are missing.

When solving an LP that has an LP variable named **my\_var** with an upper bound of 100, a lower bound of 10, and an objective function coefficient of 50, some possible observations in the ARCDATA= data set are

| <b>_name_</b> | <b>_cost_</b> | <b>_capac_</b> | <b>_lo_</b> |
|---------------|---------------|----------------|-------------|
| <b>my_var</b> | <b>50</b>     | <b>.</b>       | <b>.</b>    |
| <b>my_var</b> | <b>.</b>      | <b>100</b>     | <b>.</b>    |
| <b>my_var</b> | <b>.</b>      | <b>.</b>       | <b>10</b>   |
| <b>my_var</b> | <b>50</b>     | <b>100</b>     | <b>.</b>    |
| <b>my_var</b> | <b>.</b>      | <b>100</b>     | <b>10</b>   |
| <b>my_var</b> | <b>50</b>     | <b>.</b>       | <b>10</b>   |
| <b>my_var</b> | <b>50</b>     | <b>100</b>     | <b>10</b>   |

**CONDATA= Data Set**

Regardless of whether the data in the CONDATA= data set is in the sparse or dense format, you will receive a warning if PROC INTPOINT finds a constraint row that has no coefficients. You will also be warned if any nonarc or LP variable has no constraint coefficients.

**Dense Input Format**

If the dense format is used, most SAS variables in the CONDATA= data set belong to the VAR list. The names of the SAS variables belonging to this list have names of arc and nonarc variables or, if solving an LP, names of the LP variables. These names can be values of the SAS variables in the ARCDATA= data set that belong to the NAME list, or names of nonarc variables, or names in the form *tail\_head*, or any combination of these three forms. Names in the form *tail\_head* are default arc names, and if you use them, you must specify node names in the ARCDATA= data set (values of the TAILNODE and HEADNODE list variables).

The CONDATA= data set can have three other SAS variables belonging, respectively, to the ROW, the TYPE, and the RHS lists. The CONDATA= data set of the oil industry example in the “Introductory NPSC Example” section on page 83 uses the dense data format.

Consider the SAS code that creates a dense format CONDATA= data set that has data for three constraints. This data set was used in the “Introductory NPSC Example” section on page 83:

```
data cond1;
  input m_e_ref1 m_e_ref2 thrupt1 r1_gas thrupt2 r2_gas
        _type_ $ _rhs_;
  datalines;
-2 . 1 . . . >= -15
. -2 . . 1 . GE -15
. . -3 4 . . EQ 0
. . . . -3 4 = 0
;
```

You can use nonconstraint type values to furnish data on costs, capacities, lower flow bounds (and, if there are nonarc or LP variables, objective function coefficients and upper and lower bounds). You need not have such (or as much) data in the ARCDATA= data set. The first three observations in the following data set are examples of observations that provide cost, capacity, and lower bound data:

```

data cond1b;
  input m_e_ref1 m_e_ref2 thrupt1 r1_gas thrupt2 r2_gas
        _type_ $ _rhs_;
  datalines;
63 81 200 . 220 . cost .
95 80 175 140 100 100 capac .
20 10 50 . 35 . lo .
-2 . 1 . . . >= -15
. -2 . . 1 . GE -15
. . -3 4 . . EQ 0
. . . . -3 4 = 0
;

```

If a ROW list variable is used, the data for a constraint can be spread over more than one observation. To illustrate, the data for the first constraint (which is called `con1`) and the cost and capacity data (in special rows called `costrow` and `caprow`, respectively) will be spread over more than one observation in the following data set:

```

data cond1b;
  input _row_ $
        m_e_ref1 m_e_ref2 thrupt1 r1_gas thrupt2 r2_gas
        _type_ $ _rhs_;
  datalines;
costrow 63 . . . . . .
costrow . 81 200 . . . cost .
. . . . 220 . cost .
caprow . . . . . . capac .
caprow 95 . 175 . 100 100 . .
caprow . 80 175 140 . . . .
lorow 20 10 50 . 35 . lo .
con1 -2 . 1 . . . .
con1 . . . . . >= -15
con2 . -2 . . 1 . GE -15
con3 . . -3 4 . . EQ 0
con4 . . . . -3 4 = 0
;

```

Using both ROW and TYPE lists, you can use special row names. Examples of these are `costrow` and `caprow` in the last data set. It should be restated that in any of the input data sets of PROC INTPOINT, the order of the observation does not matter. However, the `CONDATA=` data set can be read more quickly if PROC INTPOINT knows what type of constraint or special row a ROW list variable value is. For example, when the first observation is read, PROC INTPOINT does not know whether `costrow` is a constraint or special row and how to interpret the value 63 for the arc with the name `m_e_ref1`. When PROC INTPOINT reads the second observation, it learns that `costrow` has cost type and that the values 81 and 200 are costs. When the entire `CONDATA=` data set has been read, PROC INTPOINT knows the type of all special rows and constraints. Data that PROC INTPOINT had to set aside (such as the first observation 63 value and the `costrow` ROW list variable value, which at the time had unknown type, but is subsequently known to be a cost special row) is reprocessed. During this second pass, if a ROW list variable value has unassigned

constraint or special row type, it is treated as a constraint with DEFCONTYPE= (or DEFCONTYPE= default) type. Associated VAR list variable values are coefficients of that constraint.

### Sparse Input Format

The side constraints usually become sparse as the problem size increases. When the sparse data format of the CONDATA= data set is used, only nonzero constraint coefficients must be specified. Remember to specify the SPARSECONDATA option in the PROC INTPOINT statement. With the sparse method of specifying constraint information, the names of arc and nonarc variables or, if solving an LP, the names of LP variables do not have to be valid SAS variable names.

A sparse format CONDATA= data set for the oil industry example in the “Introductory NPSC Example” section on page 83 is displayed below.

```

title 'Setting Up Condata = Cond2 for PROC INTPOINT';
data cond2;
    input _column_ $ _row1 $ _coef1 _row2 $ _coef2 ;
    datalines;
m_e_ref1  con1  -2      .      .
m_e_ref2  con2  -2      .      .
thruput1  con1   1  con3  -3
r1_gas    .      .  con3   4
thruput2  con2   1  con4  -3
r2_gas    .      .  con4   4
_type_    con1   1  con2   1
_type_    con3   0  con4   0
_rhs_     con1 -15  con2 -15
;

```

Recall that the COLUMN list variable values `_type_` and `_rhs_` are the default values of the TYPEOBS= and RHSOBS= options. Also, the default rhs value of constraints (con3 and con4) is zero. The third to last observation has the value `_type_` for the COLUMN list variable. The `_ROW1` variable value is con1, and the `_COEF1_` variable has the value 1. This indicates that the constraint con1 is *greater than* or equal to type (because the value 1 is *greater than* zero). Similarly, the data in the second to last observation's `_ROW2` and `_COEF2` variables indicate that con2 is an *equality* constraint (0 *equals* zero).



An alternative, using a TYPE list variable, is

```

title 'Setting Up Condata = Cond3 for PROC INTPOINT';
data cond3;
  input _column_ $ _row1 $ _coef1 _row2 $ _coef2 _type_ $ ;
  datalines;
m_e_ref1  con1  -2      .      .  >=
m_e_ref2  con2  -2      .      .  .
thruput1  con1   1  con3  -3      .
r1_gas    .      .  con3   4      .
thruput2  con2   1  con4  -3      .
r2_gas    .      .  con4   4      .
.          con3   .  con4   .  eq
.          con1 -15  con2 -15 ge
;

```

If the COLUMN list variable is missing in a particular observation (the last 2 observations in the data set `cond3`, for instance), the constraints named in the ROW list variables all have the constraint type indicated by the value in the TYPE list variable. It is for this type of observation that you are allowed more ROW list variables than COEF list variables. If corresponding COEF list variables are not missing (for example, the last observation in the data set `cond3`), these values are the rhs values of those constraints. Therefore, you can specify both constraint type and rhs in the same observation.

As in the previous CONDATA= data set, if the COLUMN list variable is an arc or nonarc variable, the COEF list variable values are coefficient values for that arc or nonarc variable in the constraints indicated in the corresponding ROW list variables. If in this same observation the TYPE list variable contains a constraint type, all constraints named in the ROW list variables in that observation have this constraint type (for example, the first observation in the data set `cond3`). Therefore, you can specify both constraint type and coefficient information in the same observation.

Also note that DEFCONTYPE=EQ could have been specified, saving you from having to include in the data that `con3` and `con4` are of this type.

In the oil industry example, arc costs, capacities, and lower flow bounds are presented in the ARCDATA= data set. Alternatively, you could have used the following input data sets. The `arcd2` data set has only two SAS variables. For each arc, there is an observation in which the arc's tail and head node is specified:

```

title3 'Setting Up Arcdata = Arcd2 for PROC INTPOINT';
data arcd2;
    input _from_&$11. _to_&$15. ;
    datalines;
middle east refinery 1
middle east refinery 2
u.s.a. refinery 1
u.s.a. refinery 2
refinery 1 r1
refinery 2 r2
r1 ref1 gas
r1 ref1 diesel
r2 ref2 gas
r2 ref2 diesel
ref1 gas servstn1 gas
ref1 gas servstn2 gas
ref1 diesel servstn1 diesel
ref1 diesel servstn2 diesel
ref2 gas servstn1 gas
ref2 gas servstn2 gas
ref2 diesel servstn1 diesel
ref2 diesel servstn2 diesel
;

title 'Setting Up Condata = Cond4 for PROC INTPOINT';
data cond4;
    input _column_&$27. _row1 $ _coef1 _row2 $ _coef2 _type_ $ ;
    datalines;
. con1 -15 con2 -15 ge
. costrow . . cost
. . caprow . capac
middle east_refinery 1 con1 -2 . .
middle east_refinery 2 con2 -2 . .
refinery 1_r1 con1 1 con3 -3
r1_ref1 gas . . con3 4 =
refinery 2_r2 con2 1 con4 -3
r2_ref2 gas . . con4 4 eq
middle east_refinery 1 costrow 63 caprow 95
middle east_refinery 2 costrow 81 caprow 80
u.s.a._refinery 1 costrow 55 .
u.s.a._refinery 2 costrow 49 .
refinery 1_r1 costrow 200 caprow 175
refinery 2_r2 costrow 220 caprow 100
r1_ref1 gas . . caprow 140
r1_ref1 diesel . . caprow 75
r2_ref2 gas . . caprow 100
r2_ref2 diesel . . caprow 75
ref1 gas_servstn1 gas costrow 15 caprow 70
ref1 gas_servstn2 gas costrow 22 caprow 60
ref1 diesel_servstn1 diesel costrow 18 .
ref1 diesel_servstn2 diesel costrow 17 .
ref2 gas_servstn1 gas costrow 17 caprow 35
ref2 gas_servstn2 gas costrow 31 .
ref2 diesel_servstn1 diesel costrow 36 .

```

```

ref2 diesel_servstn2 diesel costrow 23      .      .      .
middle east_refinery 1      . 20      .      .      lo
middle east_refinery 2      . 10      .      .      lo
refinery 1_r1                . 50      .      .      lo
refinery 2_r2                . 35      .      .      lo
ref2 gas_servstn1 gas       . 5       .      .      lo
;

```

The first observation in the `cond4` data set defines `con1` and `con2` as *greater than or equal to* ( $\geq$ ) constraints that both (by coincidence) have rhs values of -15. The second observation defines the special row `costrow` as a cost row. When `costrow` is a ROW list variable value, the associated COEF list variable value is interpreted as a cost or objective function coefficient. PROC INTPOINT has to do less work if constraint names and special rows are defined in observations near the top of a data set, but this is not a strict requirement. The fourth to ninth observations contain constraint coefficient data. Observations seven and nine have TYPE list variable values that indicate that constraints `con3` and `con4` are equality constraints. The last five observations contain lower flow bound data. Observations that have an arc or nonarc variable name in the COLUMN list variable, a nonconstraint type TYPE list variable value, and a value in (one of) the COEF list variables are valid.

The following data set is equivalent to the `cond4` data set:

```

title 'Setting Up Condata = Cond5 for PROC INTPOINT';
data cond5;
  input _column_&$27. _row1 $ _coef1 _row2 $ _coef2 _type_ $ ;
  datalines;
middle east_refinery 1      con1 -2 costrow 63      .
middle east_refinery 2      con2 -2  lorow 10      .
refinery 1_r1                .      .      con3 -3      =
r1_ref1 gas                  caprow 140      con3 4      .
refinery 2_r2                con2 1      con4 -3      .
r2_ref2 gas                  .      .      con4 4      eq
.                             CON1 -15      CON2 -15      GE
ref2 diesel_servstn1 diesel . 36 costrow . cost
.                             .      .      caprow . capac
.                             lorow .      .      .      lo
middle east_refinery 1      caprow 95      lorow 20      .
middle east_refinery 2      caprow 80 costrow 81      .
u.s.a._refinery 1          .      .      .      55      cost
u.s.a._refinery 2          costrow 49      .      .      .
refinery 1_r1                con1 1      caprow 175      .
refinery 1_r1                lorow 50 costrow 200      .
refinery 2_r2                costrow 220 caprow 100      .
refinery 2_r2                . 35      .      .      lo
r1_ref1 diesel              caprow2 75      .      .      capac
r2_ref2 gas                  .      .      caprow 100      .
r2_ref2 diesel              caprow2 75      .      .      .
ref1 gas_servstn1 gas       costrow 15      caprow 70      .
ref1 gas_servstn2 gas       caprow2 60 costrow 22      .
ref1 diesel_servstn1 diesel .      .      costrow 18      .
ref1 diesel_servstn2 diesel costrow 17      .      .      .

```

```

ref2 gas_servstn1 gas          costrow 17   lorow  5   .
ref2 gas_servstn1 gas          .   . caprow2 35   .
ref2 gas_servstn2 gas          . 31   .   . cost
ref2 diesel_servstn2 diesel    .   . costrow 23   .
;

```

### Converting from an NPSC to an LP Problem

If you have data for a linear programming program that has an embedded network, the steps required to change that data into a form that is acceptable by PROC INTPOINT are

1. Identify the nodal flow conservation constraints. The coefficient matrix of these constraints (a submatrix of the LP's constraint coefficient matrix) has only two nonzero elements in each column, -1 and 1.
2. Assign a node to each nodal flow conservation constraint.
3. The rhs values of conservation constraints are the corresponding node's supplies and demands. Use this information to create the NODEDATA= data set.
4. Assign an arc to each column of the flow conservation constraint coefficient matrix. The arc is directed from the node associated with the row that has the 1 element in it and directed toward to the node associated with the row that has the -1 element in it. Set up the ARCDATA= data set that has two SAS variables. This data set could resemble ARCDATA=arcd2. These will eventually be the TAILNODE and HEADNODE list variables when PROC INTPOINT is used. Each observation consists of the tail and head node of each arc.
5. Remove from the data of the linear program all data concerning the nodal flow conservation constraints.
6. Put the remaining data into a CONDATA= data set. This data set will probably resemble CONDATA=cond4 or CONDATA=cond5.

### The Sparse Format Summary

The following list illustrates possible CONDATA= data set observation sparse formats. a1, b1, b2, b3 and c1 have as a `_COLUMN_` variable value either the name of an arc (possibly in the form *tail\_head*) or the name of a nonarc variable (if you are solving an NPSC), or the name of the LP variable (if you are solving an LP). These are collectively referred to as *variable* in the tables that follow.

- If there is no TYPE list variable in the CONDATA= data set, the problem must be constrained and there is no nonconstraint data in the CONDATA= data set:

|    | <code>_COLUMN_</code>                                       | <code>_ROWx_</code> | <code>_COEFx_</code> | <code>_ROWy_</code><br>(no <code>_COEFy_</code> )<br>(may not be<br>in CONDATA) |
|----|-------------------------------------------------------------|---------------------|----------------------|---------------------------------------------------------------------------------|
| a1 | variable                                                    | constraint          | lhs coef             | +-----+                                                                         |
| a2 | <code>_TYPE_</code> or<br><code>TYPEOBS=</code>             | constraint          | -1 0 1               |                                                                                 |
| a3 | <code>_RHS_</code> or<br><code>RHSOBS=</code> or<br>missing | constraint          | rhs value            | constraint<br>or<br>missing                                                     |
| a4 | <code>_TYPE_</code> or<br><code>TYPEOBS=</code>             | constraint          | missing              |                                                                                 |
| a5 | <code>_RHS_</code> or<br><code>RHSOBS=</code> or<br>missing | constraint          | missing              |                                                                                 |
|    |                                                             |                     |                      | +-----+                                                                         |

Observations of the form a4 and a5 serve no useful purpose but are still allowed to make problem generation easier.

- If there are no ROW list variables in the data set, the problem has no constraints and the information is nonconstraint data. There must be a TYPE list variable and only one COEF list variable in this case. The COLUMN list variable has as values the names of arcs or nonarc variables and must not have missing values or special row names as values:

|    | <code>_COLUMN_</code> | <code>_TYPE_</code> | <code>_COEFx_</code> |
|----|-----------------------|---------------------|----------------------|
| b1 | variable              | UPPERBD             | capacity             |
| b2 | variable              | LOWERBD             | lower flow           |
| b3 | variable              | COST                | cost                 |

- Using a TYPE list variable for constraint data implies the following:

|    | <u>_COLUMN_</u>                          | <u>_TYPE_</u> | <u>_ROWx_</u>       | <u>_COEFx_</u> | <u>_ROWy_</u><br>(no <u>_COEFy_</u> )<br>(may not be<br>in CONDATA) |
|----|------------------------------------------|---------------|---------------------|----------------|---------------------------------------------------------------------|
| c1 | variable                                 | missing       | +-----+             | lhs coef       | +-----+                                                             |
| c2 | <u>_TYPE_</u> or<br>TYPEOBS=             | missing       | c                   | -1 0 1         |                                                                     |
| c3 | <u>_RHS_</u> or<br>missing<br>or RHSOBS= | missing       | n  <br>  s  <br>  t | rhs value      | constraint<br>or<br>missing                                         |
| c4 | variable                                 | con type      | r                   | lhs coef       |                                                                     |
| c5 | <u>_RHS_</u> or<br>missing<br>or RHSOBS= | con type      | a  <br>  i  <br>  n | rhs value      |                                                                     |
| c6 | missing                                  | TYPE          | t                   | -1 0 1         |                                                                     |
| c7 | missing                                  | RHS           | +-----+             | rhs value      | +-----+                                                             |

If the observation is in form c4 or c5, and the \_COEFx\_ values are missing, the constraint is assigned the type data specified in the \_TYPE\_ variable.

- Using a TYPE list variable for arc and nonarc variable data implies the following:

|    | <u>_COLUMN_</u> | <u>_TYPE_</u> | <u>_ROWx_</u>          | <u>_COEFx_</u>                                                | <u>_ROWy_</u><br>(no <u>_COEFy_</u> )<br>(may not be<br>in CONDATA) |
|----|-----------------|---------------|------------------------|---------------------------------------------------------------|---------------------------------------------------------------------|
| d1 | variable        | UPPERBD       | missing                | capacity                                                      | missing                                                             |
| d2 | variable        | LOWERBD       | or                     | lowerflow                                                     | or                                                                  |
| d3 | variable        | COST          | special<br>row<br>name | cost                                                          | special<br>row<br>name                                              |
| d4 | missing         |               | special<br>row<br>name |                                                               |                                                                     |
| d5 | variable        | missing       | +-----+                | value that<br>is interpreted<br>according to<br><u>_ROWx_</u> | +-----+<br>missing                                                  |

The observation with form d1 to d5 can have ROW list variable values. Observation d4 must have ROW list variable values. The ROW value is put into the ROW name tree so that when dealing with observation d4 or d5, the COEF list variable value is interpreted according to the type of ROW list variable value. For example, the following three observations define the \_ROWx\_ variable values up\_row, lo\_row and co\_row as being an upper value bound row, lower value bound row, and cost row, respectively:

| <b>_COLUMN_</b> | <b>_TYPE_</b> | <b>_ROWx_</b> | <b>_COEFx_</b> |
|-----------------|---------------|---------------|----------------|
| .               | UPPERBD       | up_row        | .              |
| variable_a      | LOWERBD       | lo_row        | lower flow     |
| variable_b      | COST          | co_row        | cost           |

PROC INTPOINT is now able to correctly interpret the following observation:

| <b>_COLUMN_</b> | <b>_TYPE_</b> | <b>_ROW1_</b> | <b>_COEF1_</b> | <b>_ROW2_</b> | <b>_COEF2_</b> | <b>_ROW3_</b> | <b>_COEF3_</b> |
|-----------------|---------------|---------------|----------------|---------------|----------------|---------------|----------------|
| var_c           | .             | up_row        | upval          | lo_row        | loval          | co_row        | cost           |

If the TYPE list variable value is a constraint type and the value of the COLUMN list variable equals the value of the TYPEOBS= option or the default value **\_TYPE\_**, the TYPE list variable value is ignored.

### **NODEDATA= Data Set**

See the “Getting Started: NPSC Problems” section on page 82 and the “Introductory NPSC Example” section on page 83 for a description of this input data set.

---

## **Output Data Set**

For NPSC problems, the procedure determines the flow that should pass through each arc as well as the value that should be assigned to each nonarc variable. The goal is that the minimum flow bounds, capacities, lower and upper value bounds, and side constraints are not violated. This goal is reached when total cost incurred by such a flow pattern and value assignment is feasible and optimal. The solution found must also conserve flow at each node.

For LP problems, the procedure determines the value that should be assigned to each variable. The goal is that the lower and upper value bounds and the constraints are not violated. This goal is reached when the total cost incurred by such a value assignment is feasible and optimal.

The CONOUT= data set can be produced and contains a solution obtained after performing optimization.

### **CONOUT= Data Set**

The CONOUT= data set contains the following variables, and their possible values in an observation are

|                |                                                                                                           |
|----------------|-----------------------------------------------------------------------------------------------------------|
| <b>_FROM_</b>  | a tail node of an arc. This is a missing value if an observation has information about a nonarc variable. |
| <b>_TO_</b>    | a head node of an arc. This is a missing value if an observation has information about a nonarc variable. |
| <b>_COST_</b>  | the cost of an arc or the objective function coefficient of a nonarc variable.                            |
| <b>_CAPAC_</b> | the capacity of an arc or upper value bound of a nonarc variable                                          |
| <b>_LO_</b>    | the lower flow bound of an arc or lower value bound of a nonarc variable                                  |
| <b>_NAME_</b>  | a name of an arc or nonarc variable                                                                       |

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_SUPPLY_</code> | the supply of the tail node of the arc in the observation. This is a missing value if an observation has information about a nonarc variable. |
| <code>_DEMAND_</code> | the demand of the head node of the arc in the observation. This is a missing value if an observation has information about a nonarc variable. |
| <code>_FLOW_</code>   | the flow through the arc or value of the nonarc variable                                                                                      |
| <code>_FCOST_</code>  | flow cost, the product of <code>_COST_</code> and <code>_FLOW_</code>                                                                         |
| <code>_RCOST_</code>  | the reduced cost of the arc or nonarc variable                                                                                                |
| <code>_STATUS_</code> | the status of the arc or nonarc variable                                                                                                      |

The variables present in the `ARCDATA=` data set are present in a `CONOUT=` data set. For example, if there is a variable called `tail` in the `ARCDATA=` data set and you specified the SAS variable list:

```
from tail;
```

then `tail` is a variable in the `CONOUT=` data sets instead of `_FROM_`. Any ID list variables also appear in the `CONOUT=` data sets.

---

## Case Sensitivity

Whenever the `INTPOINT` procedure has to compare character strings, whether they are node names, arc names, nonarc names, LP variable names, or constraint names, if the two strings have different lengths, or on a character by character basis the character is different *or has different cases*, `PROC INTPOINT` judges the character strings to be different.

Not only is this rule enforced when one or both character strings are obtained as values of SAS variables in `PROC INTPOINT`'s input data sets, it also should be obeyed if one or both character strings were originally SAS variable names, or were obtained as the values of options or statements parsed to `PROC INTPOINT`. For example, if the network has only one node that has supply capability, or if you are solving a `MAXFLOW` or `SHORTPATH` problem, you can indicate that node using the `SOURCE` option. If you specify:

```
proc intpoint source=NotableNode
```

then `PROC INTPOINT` looks for a value of the `TAILNODE` list variable that is `NotableNode`.

Version 6 of the SAS System converts text that makes up statements into uppercase. The name of the node searched for would be `NOTABLENODE`, even if this was your SAS code:

```
proc intpoint source=NotableNode
```



If you want PROC INTPOINT to behave as it did in Version 6, specify:

```
options validvarname=v6;
```

If the SPARSECONDATA option is not specified, and you are running SAS software Version 6, or you are running SAS software Version 7 onward and have specified:

```
options validvarname=v6;
```

all values of the SAS variables that belong to the NAME list are uppercased. This is because the SAS System has uppercased all SAS variable names, particularly those in the VAR list of the CONDATA= data set.

Entities that contain blanks must be enclosed in quotes.

---

## Loop Arcs

Loop arcs (which are arcs directed toward nodes from which they originate) are prohibited. Rather, introduce a dummy intermediate node in loop arcs. For example, replace arc (A,A) with (A,B) and (B,A). B is the name of a new node, and it must be distinct for each loop arc.

---

## Multiple Arcs

Multiple arcs with the same tail and head nodes are prohibited. PROC INTPOINT checks to ensure there are no such arcs before proceeding with the optimization. Introduce a new dummy intermediate node in multiple arcs. This node must be distinct for each multiple arc. For example, if some network has three arcs directed from node A toward node B, then replace one of these three with arcs (A,C) and (C,B) and replace another one with (A,D) and (D,B). C and D are new nodes added to the network.

---

## Flow and Value Bounds

The capacity and lower flow bound of an arc can be equal. Negative arc capacities and lower flow bounds are permitted. If both arc capacities and lower flow bounds are negative, the lower flow bound must be at least as negative as the capacity. An arc  $(A, B)$  that has a negative flow of  $-f$  units can be interpreted as an arc that conveys  $f$  units of flow from node  $B$  to node  $A$ .

The upper and lower value bound of a nonarc variable can be equal. Negative upper and lower bounds are permitted. If both are negative, the lower bound must be at least as negative as the upper bound.

When solving an LP, the upper and lower value bound of an LP variable can be equal. Negative upper and lower bounds are permitted. If both are negative, the lower bound must be at least as negative as the upper bound.

In short, for any problem to be feasible, a lower bound must be  $\leq$  the associated upper bound.

---

## Tightening Bounds and Side Constraints

If any piece of data is furnished to PROC INTPOINT more than once, PROC INTPOINT checks for consistency so that no conflict exists concerning the data values. For example, if the cost of some arc is seen to be one value and as more data are read, the cost of the same arc is seen to be another value, PROC INTPOINT issues an error message on the SAS log and stops. There are two exceptions to this:

- The bounds of arcs and nonarc variables, or the bounds of LP variables, are made as tight as possible. If several different values are given for the lower flow bound of an arc, the greatest value is used. If several different values are given for the lower bound of a nonarc or LP variable, the greatest value is used. If several different values are given for the capacity of an arc, the smallest value is used. If several different values are given for the upper bound of a nonarc or LP variable, the smallest value is used.
- Several values can be given for inequality constraint right-hand-sides. For a particular constraint, the lowest rhs value is used for the rhs if the constraint is of *less than or equal to* type. For a particular constraint, the greatest rhs value is used for the rhs if the constraint is of *greater than or equal to* type.

---

## Reasons for Infeasibility

Before optimization commences, PROC INTPOINT tests to ensure that the problem is not infeasible by ensuring that, with respect to supplies, demands, and arc flow bounds, flow conservation can be obeyed at each node:

- Let  $IN$  be the sum of lower flow bounds of arcs directed toward a node plus the node's supply. Let  $OUT$  be the sum of capacities of arcs directed from that node plus the node's demand. If  $IN$  exceeds  $OUT$ , not enough flow can leave the node.
- Let  $OUT$  be the sum of lower flow bounds of arcs directed from a node plus the node's demand. Let  $IN$  be the total capacity of arcs directed toward the node plus the node's supply. If  $OUT$  exceeds  $IN$ , not enough flow can arrive at the node.

Reasons why a network problem can be infeasible are similar to those previously mentioned but apply to a set of nodes rather than for an individual node.

Consider the network illustrated in Figure 5.10.

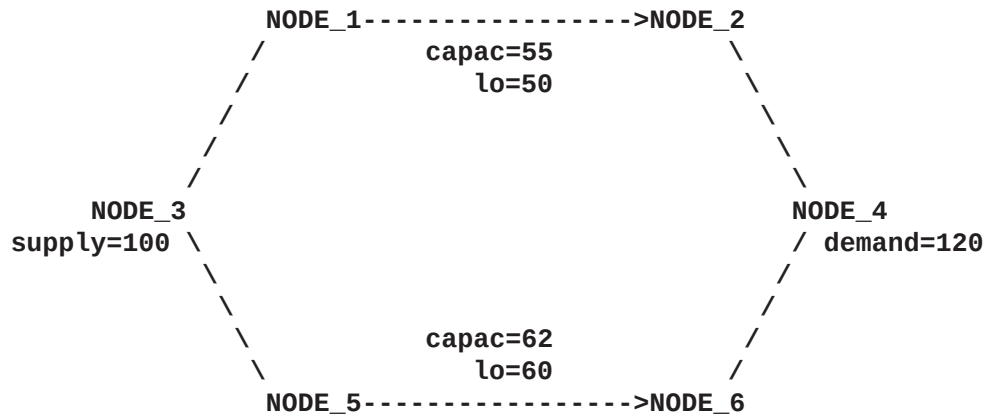


**Figure 5.10.** An Infeasible Network

The demand of NODE\_4 is 120. That can never be satisfied because the maximal flow through arcs (NODE\_1, NODE\_2) and (NODE\_5, NODE\_6) is 117. More specifically, the implicit supply of NODE\_2 and NODE\_6 is only 117, which is insufficient to satisfy the demand of other nodes (real or implicit) in the network.

Furthermore, the lower flow bounds of arcs (NODE\_1, NODE\_2) and (NODE\_5, NODE\_6) are greater than the flow that can reach the tail nodes of these arcs, that, by coincidence, is the total supply of the network. The implicit demand of nodes NODE\_1 and NODE\_5 is 110, which is greater than the amount of flow that can reach these nodes.

---

## Missing S Supply and Missing D Demand Values

In some models, you may want a node to be either a supply or demand node but you want the node to supply or demand the optimal number of flow units. To indicate that a node is such a supply node, use a missing S value in the SUPPLY list variable in the ARCDATA= data set or the SUPDEM list variable in the NODEDATA= data set. To indicate that a node is such a demand node, use a missing D value in the DEMAND list variable in the ARCDATA= data set or the SUPDEM list variable in the NODEDATA= data set.

Suppose the oil example in the “Introductory NPSC Example” section on page 83 is changed so that crude oil can be obtained from either the Middle East or U.S.A. in any amounts. You should specify that the node `middle east` is a supply node, but you do not want to stipulate that it supplies 100 units, as before. The node `u.s.a.` should also remain a supply node, but you do not want to stipulate that it supplies 80 units. You must specify that these nodes have missing S supply capabilities:

```

title 'Oil Industry Example';
title3 'Crude Oil can come from anywhere';
data miss_s;
    missing S;
    input _node_&$15. _sd_;
    datalines;
middle east          S
u.s.a.               S
servstn1 gas         -95
servstn1 diesel      -30
servstn2 gas         -40
servstn2 diesel      -15
;

```

The following PROC INTPOINT run uses the same ARCDATA= and CONDATA= data sets used in the “Introductory NPSC Example” section on page 83:

```

proc intpoint
    bytes=100000
    nodedata=miss_s          /* the supply (missing S) and */
                             /* demand data                      */
    arcdata=arcd1            /* the arc descriptions      */
    condata=cond1            /* the side constraints      */
    conout=solution;         /* the solution data set    */
run;
proc print;
    var _from_ _to_ _cost_ _capac_ _lo_ _flow_ _fcost_;
    sum _fcost_;
run;

```

The following messages appear on the SAS log:

```

NOTE: Number of nodes= 14 .
NOTE: Number of supply nodes= 2 .
NOTE: Of these, 2 have unspecified (.S) supply capability.
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 0 , total demand= 180 .
NOTE: Number of arcs= 18 .
NOTE: Number of <= side constraints= 0 .
NOTE: Number of == side constraints= 2 .
NOTE: Number of >= side constraints= 2 .
NOTE: Number of side constraint coefficients= 8 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 17 .
NOTE: Number of >= constraints= 2 .
NOTE: Number of constraint coefficients= 48 .
NOTE: Number of variables= 20 .
NOTE: After preprocessing, number of <= constraints= 0.
NOTE: After preprocessing, number of == constraints= 8.

```

NOTE: After preprocessing, number of  $\geq$  constraints= 2.  
 NOTE: The preprocessor eliminated 9 constraints from the problem.  
 NOTE: The preprocessor eliminated 20 constraint coefficients from the problem.  
 NOTE: After preprocessing, number of variables= 11.  
 NOTE: The preprocessor eliminated 9 variables from the problem.  
 NOTE: 2 columns, 0 rows and 2 coefficients were added to the problem to handle unrestricted variables, variables that are split, and constraint slack or surplus variables.  
 NOTE: There are 21 nonzero elements in  $A * A$  transpose.  
 NOTE: Of the 10 rows and columns, 4 are sparse.  
 NOTE: There are 15 nonzero superdiagonal elements in the sparse rows of the factored  $A * A$  transpose. This includes fill-in.  
 NOTE: There are 5 operations of the form  $u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$  to factorize the sparse rows of  $A * A$  transpose.  
 NOTE: Bound feasibility attained by iteration 1.  
 NOTE: Dual feasibility attained by iteration 1.  
 NOTE: Constraint feasibility attained by iteration 2.  
 NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 7 iterations.  
 NOTE: Objective = 50075.  
 NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.  
 NOTE: There were 18 observations read from the data set WORK.ARC1.  
 NOTE: There were 6 observations read from the data set WORK.MISS\_S.  
 NOTE: There were 4 observations read from the data set WORK.COND1.  
 NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.

The CONOUT= data set is shown in Figure 5.11.

| Oil Industry Example             |             |                 |        |          |      |          |
|----------------------------------|-------------|-----------------|--------|----------|------|----------|
| Crude Oil can come from anywhere |             |                 |        |          |      |          |
| Obs                              | _from_      | _to_            | _cost_ | _capac_  | _lo_ | _FLOW_   |
| 1                                | refinery 1  | r1              | 200    | 175      | 50   | 145.00   |
| 2                                | refinery 2  | r2              | 220    | 100      | 35   | 35.00    |
| 3                                | r1          | ref1 diesel     | 0      | 75       | 0    | 36.25    |
| 4                                | r1          | ref1 gas        | 0      | 140      | 0    | 108.75   |
| 5                                | r2          | ref2 diesel     | 0      | 75       | 0    | 8.75     |
| 6                                | r2          | ref2 gas        | 0      | 100      | 0    | 26.25    |
| 7                                | middle east | refinery 1      | 63     | 95       | 20   | 20.00    |
| 8                                | u.s.a.      | refinery 1      | 55     | 99999999 | 0    | 125.00   |
| 9                                | middle east | refinery 2      | 81     | 80       | 10   | 10.00    |
| 10                               | u.s.a.      | refinery 2      | 49     | 99999999 | 0    | 25.00    |
| 11                               | ref1 diesel | servstn1 diesel | 18     | 99999999 | 0    | 30.00    |
| 12                               | ref2 diesel | servstn1 diesel | 36     | 99999999 | 0    | 0.00     |
| 13                               | ref1 gas    | servstn1 gas    | 15     | 70       | 0    | 68.75    |
| 14                               | ref2 gas    | servstn1 gas    | 17     | 35       | 5    | 26.25    |
| 15                               | ref1 diesel | servstn2 diesel | 17     | 99999999 | 0    | 6.25     |
| 16                               | ref2 diesel | servstn2 diesel | 23     | 99999999 | 0    | 8.75     |
| 17                               | ref1 gas    | servstn2 gas    | 22     | 60       | 0    | 40.00    |
| 18                               | ref2 gas    | servstn2 gas    | 31     | 99999999 | 0    | 0.00     |
|                                  |             |                 |        |          |      | =====    |
|                                  |             |                 |        |          |      | 50075.00 |

**Figure 5.11.** Missing S SUPDEM values in NODEDATA

The optimal supplies of nodes middle east and u.s.a. are 145 and 35 units, respectively. For this example, the same optimal solution is obtained if these nodes had supplies less than these values (each supplies 1 unit, for example) and the THRUNET option was specified in the PROC INTPOINT statement. With the THRUNET option active, when total supply exceeds total demand, the specified nonmissing demand values are the lowest number of flow units that must be absorbed by the corresponding node. This is demonstrated in the following PROC INTPOINT run. The missing S is most useful when nodes are to supply optimal numbers of flow units and it turns out that for some nodes, the optimal supply is zero:

```

data miss_s_x;
  missing S;
  input _node_&$15. _sd_;
  datalines;
middle east      1
u.s.a.           1
servstn1 gas     -95
servstn1 diesel -30
servstn2 gas     -40
servstn2 diesel -15
;
proc intpoint
  bytes=100000
  thrunet
  nodedata=miss_s_x /* No supply (missing S) */
  arcdata=arcd1     /* the arc descriptions */
  condata=cond1     /* the side constraints */
  conout=solution;  /* the solution data set */
run;
proc print;
  var _from_ _to_ _cost_ _capac_ _lo_ _flow_ _fcost_;
  sum _fcost_;
run;

```

The following messages appear on the SAS log. Note that the Total supply= 2, not zero as in the last run:

```

NOTE: Number of nodes= 14 .
NOTE: Number of supply nodes= 2 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 2 , total demand= 180 .
NOTE: Number of arcs= 18 .
NOTE: Number of <= side constraints= 0 .
NOTE: Number of == side constraints= 2 .
NOTE: Number of >= side constraints= 2 .
NOTE: Number of side constraint coefficients= 8 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 17 .
NOTE: Number of >= constraints= 2 .
NOTE: Number of constraint coefficients= 48 .
NOTE: Number of variables= 20 .
NOTE: After preprocessing, number of <= constraints= 0.
NOTE: After preprocessing, number of == constraints= 8.
NOTE: After preprocessing, number of >= constraints= 2.
NOTE: The preprocessor eliminated 9 constraints from the
      problem.
NOTE: The preprocessor eliminated 20 constraint
      coefficients from the problem.
NOTE: After preprocessing, number of variables= 11.
NOTE: The preprocessor eliminated 9 variables from the

```

problem.

NOTE: 2 columns, 0 rows and 2 coefficients were added to the problem to handle unrestricted variables, variables that are split, and constraint slack or surplus variables.

NOTE: There are 21 nonzero elements in  $A * A$  transpose.

NOTE: Of the 10 rows and columns, 4 are sparse.

NOTE: There are 15 nonzero superdiagonal elements in the sparse rows of the factored  $A * A$  transpose. This includes fill-in.

NOTE: There are 5 operations of the form  $u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$  to factorize the sparse rows of  $A * A$  transpose.

NOTE: Bound feasibility attained by iteration 1.

NOTE: Dual feasibility attained by iteration 1.

NOTE: Constraint feasibility attained by iteration 2.

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 7 iterations.

NOTE: Objective = 50075.

NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.

NOTE: There were 18 observations read from the data set WORK.ARC1.

NOTE: There were 6 observations read from the data set WORK.MISS\_S\_X.

NOTE: There were 4 observations read from the data set WORK.COND1.

NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.

If total supply exceeds total demand, any missing S values are ignored. If total demand exceeds total supply, any missing D values are ignored.

---

## Balancing Total Supply and Total Demand

### ***When Total Supply Exceeds Total Demand***

When total supply of a network problem exceeds total demand, PROC INTPOINT adds an extra node (called the *excess node*) to the problem and sets the demand at that node equal to the difference between total supply and total demand. There are three ways that this excess node can be joined to the network. All three ways entail PROC INTPOINT generating a set of arcs (henceforth referred to as the *generated arcs*) that are directed toward the excess node. The total amount of flow in generated arcs equals the demand of the excess node. The generated arcs originate from one of three sets of nodes.

When you specify the THRUNET option, the set of nodes that generated arcs originate from are all demand nodes, even those demand nodes with unspecified demand capability. You indicate that a node has unspecified demand capability by using a missing D value instead of an actual value for demand data (discussed in the “Missing S Supply and Missing D Demand Values” section on page 146). The value specified as the demand of a demand node is in effect a lower bound of the number of flow units that node can actually demand. For missing D demand nodes, this lower bound is zero.



If you do not specify the THRUNET option, the way in which the excess node is joined to the network depends on whether there are demand nodes with unspecified demand capability (nodes with missing D demand) or not.

If there are missing D demand nodes, these nodes are the set of nodes that generated arcs originate from. The value specified as the demand of a demand node, if not missing D, is the number of flow units that node can actually demand. For a missing D demand node, the actual demand of that node may be zero or greater.

If there are no missing D demand nodes, the set of nodes that generated arcs originate from are the set of supply nodes. The value specified as the supply of a supply node is in effect an upper bound of the number of flow units that node can actually supply. For missing S supply nodes (discussed in the “Missing S Supply and Missing D Demand Values” section on page 146), this upper bound is zero, so missing S nodes when total supply exceeds total demand are transshipment nodes, that is, nodes that neither supply nor demand flow.

### **When Total Supply Is Less Than Total Demand**

When total supply of a network problem is less than total demand, PROC INTPOINT adds an extra node (called the *excess node*) to the problem and sets the supply at that node equal to the difference between total demand and total supply. There are three ways that this excess node can be joined to the network. All three ways entail PROC INTPOINT generating a set of arcs (henceforth referred to as the *generated arcs*) that originate from the excess node. The total amount of flow in generated arcs equals the supply of the excess node. The generated arcs are directed toward one of three sets of nodes.

When you specify the THRUNET option, the set of nodes that generated arcs are directed toward are all supply nodes, even those supply nodes with unspecified supply capability. You indicate that a node has unspecified supply capability by using a missing S value instead of an actual value for supply data (discussed in the “Missing S Supply and Missing D Demand Values” section on page 146). The value specified as the supply of a supply node is in effect a lower bound of the number of flow units that the node can actually supply. For missing S supply nodes, this lower bound is zero.

If you do not specify the THRUNET option, the way in which the excess node is joined to the network depends on whether there are supply nodes with unspecified supply capability (nodes with missing S supply) or not.

If there are missing S supply nodes, these nodes are the set of nodes that generated arcs are directed toward. The value specified as the supply of a supply node, if not missing S, is the number of flow units that the node can actually supply. For a missing S supply node, the actual supply of that node may be zero or greater.

If there are no missing S supply nodes, the set of nodes that generated arcs are directed toward are the set of demand nodes. The value specified as the demand of a demand node is in effect an upper bound of the number of flow units that node can actually demand. For missing D demand nodes (discussed in the “Missing S Supply and Missing D Demand Values” section on page 146), this upper bound is zero, so missing D nodes when total supply is less than total demand are transshipment nodes, that is, nodes that neither supply nor demand flow.

## How to Make the Data Read of PROC INTPOINT More Efficient

This section contains information that is useful when you want to solve large constrained network problems. However, much of this information is also useful if you have a large linear programming problem. All of the options described in this section that are not directly applicable to networks (options such as ARCS\_ONLY\_ARCDATA ARC\_SINGLE\_OBS, NNODES, and NARCS) can be specified to improve the speed at which LP data is read.

### **Large Constrained Network Problems**

Many of the models presented to PROC INTPOINT are enormous. They can be considered large by linear programming standards; problems with thousands, even millions, of variables and constraints. When dealing with side constrained network programming problems, models can have not only a linear programming component of that magnitude, but also a larger, possibly *much* larger, network component.

The majority of network problem's decision variables are arcs. Like an LP decision variable, an arc has an objective function coefficient, upper and lower value bounds, and a name. Arcs can have coefficients in constraints. Therefore, an arc is quite similar to an LP variable and places the same memory demands on optimization software as an LP variable. But a typical network model has many more arcs and nonarc variables than the typical LP model has variables. And arcs have tail and head nodes. Storing and processing node names require huge amounts of memory. To make matters worse, node names occupy memory at times when a lot of other data should reside in memory as well.

While memory requirements are lower for a model with embedded network component compared with the equivalent LP *once optimization starts*, the same is usually not true *during the data read*. Even though nodal flow conservation constraints in the LP should not be specified in the constrained network formulation, the memory requirements to read the latter are greater because each arc (unlike an LP variable) originates at one node and is directed toward another.

### **Paging**

PROC INTPOINT has facilities to read data when the available memory is insufficient to store all the data at once. PROC INTPOINT does this by allocating memory for different purposes; for example, to store an array or receive data read from an input SAS data set. After that memory has filled, the information is written to disk and PROC INTPOINT can resume filling that memory with new information. Often, information must be retrieved from disk so that data previously read can be examined or checked for consistency. Sometimes, to prevent any data from being lost, or to retain any changes made to the information in memory, the contents of the memory must be sent to disk before other information can take its place. This process of swapping information to and from disk is called paging. Paging can be very time consuming, so it is crucial to minimize the amount of paging performed.

There are several steps you can take to make PROC INTPOINT read the data of network and linear programming models more efficiently, particularly when memory is scarce and the amount of paging must be reduced. PROC INTPOINT will then be able to tackle large problems in what can be considered reasonable amounts of time.

### **The Order of Observations**

PROC INTPOINT is quite flexible in the ways data can be supplied to it. Data can be given by any reasonable means. PROC INTPOINT has convenient defaults that can save you work when generating the data. There can be several ways to supply the same piece of data, and some pieces of data can be given more than once. PROC INTPOINT reads everything, then merges it all together. However, this flexibility and convenience come at a price; PROC INTPOINT may not assume the data has a characteristic that, if possessed by the data, could save time and memory during the data read. Several options can indicate that the data has some exploitable characteristic.

For example, an arc cost can be specified once or several times in the ARCDATA= data set or the CONDATA= data set, or both. Every time it is given in the ARCDATA= data set, a check is made to ensure that the new value is the same as any corresponding value read in a previous observation of the ARCDATA= data set. Every time it is given in the CONDATA= data set, a check is made to ensure that the new value is the same as the value read in a previous observation of the CONDATA= data set, or previously in the ARCDATA= data set. PROC INTPOINT would save time if it knew that arc cost data would be encountered only once while reading the ARCDATA= data set, so performing the time-consuming check for consistency would not be necessary. Also, if you indicate that the CONDATA= data set contains data for constraints only, PROC INTPOINT will not expect any arc information, so memory will not be allocated to receive such data while reading the CONDATA= data set. This memory is used for other purposes and this might lead to a reduction in paging. If applicable, use the ARC\_SINGLE\_OBS or the CON\_SINGLE\_OBS option, or both, and the NON\_REPLIC=COEFS specification to improve how the ARCDATA= data set and the CONDATA= data set are read.

PROC INTPOINT allows the observations in input data sets to be in any order. However, major time savings can result if you are prepared to order observations in particular ways. Time spent by the SORT procedure to sort the input data sets, particularly the CONDATA= data set, may be more than made up for when PROC INTPOINT reads them, because PROC INTPOINT has in memory information possibly used when the previous observation was read. PROC INTPOINT can assume a piece of data is either similar to that of the last observation read or is new. In the first case, valuable information such as an arc or a nonarc variable number or a constraint number is retained from the previous observation. In the last case, checking the data with what has been read previously is not necessary.

Even if you do not sort the CONDATA= data set, grouping observations that contain data for the same arc or nonarc variable or the same row pays off. PROC INTPOINT establishes whether an observation being read is similar to the observation just read.

In practice, many input data sets for PROC INTPOINT have this characteristic, because it is natural for data for each constraint to be grouped together (when using the dense format of the CONDATA= data set) or data for each column to be grouped together (when using the sparse format of the CONDATA= data set). If data for each arc or nonarc is spread over more than one observation of the ARCDATA= data set, it is natural to group these observation together.

Use the GROUPED= parameter to indicate whether observations of the ARCDATA= data set, the CONDATA= data set, or both, are grouped in a way that can be exploited during data read.

You can save time if the type data for each row appears near the top of the CONDATA= data set, especially if it has the sparse format. Otherwise, when reading an observation, if PROC INTPOINT does not know if a row is a constraint or special row, the data is set aside. Once the data set has been completely read, PROC INTPOINT must reprocess the data it set aside. By then, it knows the type of each constraint or row or, if its type was not provided, it is assumed to have a default type.

### **Better Memory Utilization**

In order for PROC INTPOINT to make better utilization of available memory, you can now specify options that indicate the approximate size of the model. PROC INTPOINT then knows what to expect. For example, if you indicate that the problem has no nonarc variables, PROC INTPOINT will not allocate memory to store nonarc data. That memory is better utilized for other purposes. Memory is often allocated to receive or store data of some type. If you indicate that the model does not have much data of a particular type, the memory that would otherwise have been allocated to receive or store that data can be used to receive or store data of another type.

These are the problem size options:

- NNODES= approximate number of nodes
- NARCS= approximate number of arcs
- NNAS= approximate number of nonarc variables or LP variables
- NCONS= approximate number of NPSC side constraints or LP constraints
- NCOEFS= approximate number of NPSC side constraint coefficients or LP constraint coefficients

These options will sometimes be referred to as Nxxxx= options.

You do not need to specify all these options for the model, but the more you do, the better. If you do not specify some or all of these options, PROC INTPOINT guesses the size of the problem by using what it already knows about the model. Sometimes PROC INTPOINT guesses the size of the model by looking at the number of observations in the ARCDATA= and the CONDATA= data sets. However, PROC INTPOINT uses rough rules of thumb, that typical models are proportioned in certain ways (for example, if there are constraints, arcs, nonarc variables, or LP variables usually have about five constraint coefficients). If your model has an unusual shape or a disproportionate number of something, you are encouraged to use these options.

If you do use the options, if you do not know the exact values to specify, *overestimate* the values. For example, if you specify NARCS=10000 but the model has 10100 arcs, when dealing with the last 100 arcs, PROC INTPOINT might have to page out data for 10000 arcs each time one of the last arcs must be dealt with. Memory could have been allocated for all 10100 arcs without affecting (much) the rest of the data read, so NARCS=10000 could be more of a hindrance than a help.

The point of these Nxxxx= options is to indicate the model size when PROC INTPOINT does not know it. When PROC INTPOINT knows the “real” value, that value is used instead of Nxxxx=.

ARCS\_ONLY\_ARCDATA indicates that data for only arcs are in the ARCDATA= data set. Memory would not be wasted to receive data for nonarc variables.

Use the memory usage parameters:

- BYTES= size of PROC INTPOINT main working memory in number of bytes
- MEMREP indicates that memory usage report is to be displayed on the SAS log

Specifying an appropriate value for the BYTES= parameter is particularly important. Specify as large a number as possible, but not so large a number that will cause PROC INTPOINT (that is, the SAS System running underneath PROC INTPOINT) to run out of memory.

PROC INTPOINT reports its memory requirements on the SAS log if you specify the MEMREP option.

### **Use Defaults to Reduce the Amount of Data**

Use the parameters that specify default values as much as possible. For example, if there are lots of arcs with the same cost value *c*, use DEFCOST=*c* for arcs that have that cost. Use missing values in the COST variable in the ARCDATA= data set instead of *c*. PROC INTPOINT ignores missing values, but must read, store, and process nonmissing values, even if they are equal to a default option or could have been equal to a default parameter had it been specified. Sometimes, using default parameters makes the need for some SAS variables in the ARCDATA= and the CONDATA= data sets no longer necessary, or reduces the quantity of data that must be read. The default options are:

- DEFCOST= default cost of arcs, objective function of nonarc variables or LP variables
- DEFMINFLOW= default lower flow bound of arcs, lower bound of nonarc variables or LP variables
- DEFCAPACITY= default capacity of arcs, upper bound of nonarc variables or LP variables
- DEFCTYPE= LE or DEFCTYPE= <=  
DEFCTYPE= EQ or DEFCTYPE= =  
DEFCTYPE= GE or DEFCTYPE= >=

DEFCTYPE=LE is the default.

The default options themselves have defaults. For example, you do not need to specify DEFCOST=0 in the PROC INTPOINT statement. You should still have missing values in the COST variable in the ARCDATA= data set for arcs that have zero costs.

If the network has only one supply node, one demand node, or both, use

- SOURCE= name of single node that has supply capability
- SUPPLY= the amount of supply of SOURCE

- SINK= name of single node that demands flow
- DEMAND= the amount of flow SINK demands

Do not specify that a constraint has zero right-hand-side values. That is the default. The only time it might be practical to specify a zero rhs is in observations of the CONDATA read early so that PROC INTPOINT can infer that a row is a constraint. This could prevent coefficient data from being put aside because PROC INTPOINT did not know the row was a constraint.

### **Names of Things**

To cut data read time and memory requirements, reduce the number of bytes in the longest node name, the longest arc name, the longest nonarc variable name, the longest LP variable name, and the longest constraint name to 8 bytes or less. The longer a name, the more bytes must be stored and compared with other names.

If an arc has no constraint coefficients, do not give it a name in the NAME list variable in the ARCDATA= data set. Names for such arcs serve no purpose.

PROC INTPOINT can have a default name for each arc. If an arc is directed from node *tailname* toward node *headname*, the default name for that arc is *tailname\_headname*. If you do not want PROC INTPOINT to use these default arc names, specify NAMECTRL=1. Otherwise, PROC INTPOINT must use memory for storing node names and these node names must be searched often.

If you want to use the default *tailname\_headname* name, that is, NAMECTRL=2 or NAMECTRL=3, do not use underscores in node names. If the CONDATA has a dense format and has a variable in the VAR list A\_B\_C\_D, or if the value A\_B\_C\_D is encountered as a value of the COLUMN list variable when reading the CONDATA= data set that has the sparse format, PROC INTPOINT first looks for a node named A. If it finds it, it looks for a node called B\_C\_D. It then looks for a node with the name A\_B and possibly a node with name C\_D. A search is then conducted for a node named A\_B\_C and possibly a node named D is done. Underscores could have caused PROC INTPOINT to look unnecessarily for nonexistent nodes. Searching for node names can be expensive, and the amount of memory to store node names is often large. It might be better to assign the arc name A\_B\_C\_D directly to an arc by having that value as a NAME list variable value for that arc in the ARCDATA= data set and specify NAMECTRL=1.

### **Other Ways to Speed-up Data Reads**

Arcs and nonarc variables, or LP variables, can have associated with them values or quantities that have no bearing with the optimization. This information is given in the ARCDATA= data set in the ID list variables. For example, in a distribution problem, information such as truck number and driver's name can be associated with each arc. This is useful when the optimal solution saved in the CONOUT= data set is analyzed. However, PROC INTPOINT needs to reserve memory to process this information when data is being read. For large problems when memory is scarce, it might be better to remove ancillary data from the ARCDATA. After PROC INTPOINT runs, use SAS software to merge this information into the CONOUT= data set that contains the optimal solution.



## Stopping Criteria

There are several reasons why PROC INTPOINT stops Interior Point optimization. Optimization stops when:

- the number of iteration equals MAXITERB= $m$
- the relative gap ( $dualitygap/(c^T x)$ ) between the primal and dual objectives is smaller than the value of the PDGAPTOL= option, and both the primal and dual problems are feasible. Duality gap is defined in the “Interior Point Algorithmic Details” section on page 66.

PROC INTPOINT may stop optimization when it detects that the rate at which the complementarity or duality gap is being reduced is too slow; that is, that there are consecutive iterations when the complementarity or duality gap has stopped getting smaller and the infeasibilities, if nonzero, have also stalled. Sometimes this indicates that the problem is infeasible.

The reasons to stop optimization outlined in the previous paragraph will be termed the *usual* stopping conditions in the following explanation.

However, when solving some problems, especially if the problems are large, the usual stopping criteria are inappropriate. PROC INTPOINT might stop optimizing prematurely. If it were allowed to perform additional optimization, a better solution would be found. On other occasions, PROC INTPOINT might do too much work. A sufficiently good solution might be reached several iterations before PROC INTPOINT eventually stops.

You can see PROC INTPOINT’s progress to the optimum by specifying PRINTLEVEL2=2. PROC INTPOINT will produce a table on the SAS log. A row of the table is generated during each iteration and consists of values of the affine step complementarity, the complementarity of the solution for the next iteration, the total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 71), and the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 66), the total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 66). As optimization progresses, the values in all columns should converge to zero.

To tailor stopping criteria to your problem, you can use two sets of parameters: the STOP\_x and the KEEPGOING\_x parameters. The STOP\_x parameters (STOP\_C, STOP\_DG, STOP\_IB, STOP\_IC, STOP\_ID) are used to test for some condition at the beginning of each iteration and if met, to stop optimizing immediately. The KEEPGOING\_x parameters (KEEPGOING\_C, KEEPGOING\_DG, KEEPGOING\_IB, KEEPGOING\_IC, KEEPGOING\_ID) are used when PROC INTPOINT would ordinarily stop optimizing but does not if some conditions are not met.

For the sake of conciseness, a set of options might be referred to as the part of the option name they have in common followed by the suffix x. For example, STOP\_C, STOP\_DG, STOP\_IB, STOP\_IC, and STOP\_ID will collectively be referred to as STOP\_x.

At the beginning of each iteration, PROC INTPOINT will test whether complementarity is  $\leq \text{STOP\_C}$  (provided you specified a `STOP_C` parameter) and if it is, PROC INTPOINT will stop optimizing. If the duality gap is  $\leq \text{STOP\_DG}$  (provided you specified a `STOP_DG` parameter), PROC INTPOINT will stop optimizing immediately. This is true as well for the other `STOP_x` parameters that are related to infeasibilities, `STOP_IB`, `STOP_IC`, and `STOP_ID`.

For example, if you want PROC INTPOINT to stop optimizing for the usual stopping conditions, plus the additional condition, complementarity  $\leq 100$  or duality gap  $\leq 0.001$ , then use:

```
proc intpoint stop_c=100 stop_dg=0.001
```

If you want PROC INTPOINT to stop optimizing for the usual stopping conditions, plus the additional condition, complementarity  $\leq 1000$  and duality gap  $\leq 0.001$  and constraint infeasibility  $\leq 0.0001$ , then use:

```
proc intpoint  
  and_stop_c=1000 and_stop_dg=0.01 and_stop_ic=0.0001
```

Unlike the `STOP_x` parameters that cause PROC INTPOINT to stop optimizing when any one of them is satisfied, the corresponding `AND_STOP_x` parameters (`AND_STOP_C`, `AND_STOP_DG`, `AND_STOP_IB`, `AND_STOP_IC`, and `AND_STOP_ID`) cause PROC INTPOINT to stop only if all (more precisely, all that are specified) options are satisfied. For example, if PROC INTPOINT should stop optimizing when

- complementarity  $\leq 100$  or duality gap  $\leq 0.001$  or
- complementarity  $\leq 1000$  and duality gap  $\leq 0.001$  and constraint infeasibility  $\leq 0.000$

then use:

```
proc intpoint  
  stop_c=100 stop_dg=0.001  
  and_stop_c=1000 and_stop_dg=0.01 and_stop_ic=0.0001
```

Just as the `STOP_x` parameters have `AND_STOP_x` partners, the `KEEPGOING_x` parameters have `AND_KEEPPGOING_x` partners. The role of the `KEEPGOING_x` and `AND_KEEPPGOING_x` parameters is to prevent optimization from stopping too early, even though a usual stopping criteria is met.

When PROC INTPOINT detects that it should stop optimizing for a usual stopping condition,

- it will test whether complementarity is  $> \text{KEEPGOING\_C}$  (provided you specified a `KEEPGOING_C` parameter), and if it is, PROC INTPOINT will perform more optimization.



- Otherwise, PROC INTPOINT will then test whether the primal-dual gap is  $>$  KEEPGOING\_DG (provided you specified a KEEPGOING\_DG parameter), and if it is, PROC INTPOINT will perform more optimization.
- Otherwise, PROC INTPOINT will then test whether the total bound infeasibility  $\sum_{i=1}^n infeas_{bi} >$  KEEPGOING\_IB (provided you specified a KEEPGOING\_IB parameter), and if it is, PROC INTPOINT will perform more optimization.
- Otherwise, PROC INTPOINT will then test whether the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci} >$  KEEPGOING\_IC (provided you specified a KEEPGOING\_IC parameter), and if it is, PROC INTPOINT will perform more optimization.
- Otherwise, PROC INTPOINT will then test whether the total dual infeasibility  $\sum_{i=1}^n infeas_{di} >$  KEEPGOING\_ID (provided you specified a KEEPGOING\_ID parameter), and if it is, PROC INTPOINT will perform more optimization.
- Otherwise it will test whether complementarity is  $>$  AND\_KEEPPGOING\_C (provided you specified a AND\_KEEPPGOING\_C parameter), *and* the primal-dual gap is  $>$  AND\_KEEPPGOING\_DG (provided you specified a AND\_KEEPPGOING\_DG parameter), *and* the total bound infeasibility  $\sum_{i=1}^n infeas_{bi} >$  AND\_KEEPPGOING\_IB (provided you specified a AND\_KEEPPGOING\_IB parameter), *and* the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci} >$  AND\_KEEPPGOING\_IC (provided you specified a AND\_KEEPPGOING\_IC parameter), *and* the total dual infeasibility  $\sum_{i=1}^n infeas_{di} >$  AND\_KEEPPGOING\_ID (provided you specified a AND\_KEEPPGOING\_ID parameter), and if it is, PROC INTPOINT will perform more optimization.

If all these tests to decide whether more optimization should be performed are false, optimization is stopped.

The following PROC INTPOINT example will be used to illustrate how several stopping criteria options can be used together:

```
proc intpoint
  stop_c=1000
  and_stop_c=2000 and_stop_dg=0.01
  and_stop_ib=1 and_stop_ic=1 and_stop_id=1
  keepgoing_c=1500
  and_keepgoing_c=2500 and_keepgoing_dg=0.05
  and_keepgoing_ib=1 and_keepgoing_ic=1 and_keepgoing_id=1
```

At the beginning of each iteration, PROC INTPOINT will stop optimizing if

- complementarity  $\leq 1000$  or
- complementarity  $\leq 2000$  and duality gap  $\leq 0.01$  and the total bound, constraint, and dual infeasibilities are each  $\leq 1$

When PROC INTPOINT determines it should stop optimizing because a usual stopping condition is met, it will stop optimizing only if

- complementarity  $\leq 1500$  or
- complementarity  $\leq 2500$  and duality gap  $\leq 0.05$  and the total bound, constraint, and dual infeasibilities are each  $\leq 1$

## Examples

The following examples illustrate some of the capabilities of PROC INTPOINT. These examples, together with the other SAS/OR examples, can be found in the SAS sample library.

In order to illustrate variations in the use of the INTPOINT procedure, Example 5.1 through Example 5.5 use data from a company that produces two sizes of televisions. The company makes televisions with a diagonal screen measurement of either 19 inches or 25 inches. These televisions are made between March and May at both of the company's two factories. Each factory has a limit on the total number of televisions of each screen dimension that can be made during those months.

The televisions are distributed to one of two shops, stored at the factory where they were made, and sold later or shipped to the other factory. Some sets can be used to fill back-orders from the previous months. Each shop demands a number of each type of TV for the months March through May. The following network in Figure 5.12 illustrates the model. Arc costs can be interpreted as production costs, storage costs, back-order penalty costs, inter-factory transportation costs, and sales profits. The arcs can have capacities and lower flow bounds.

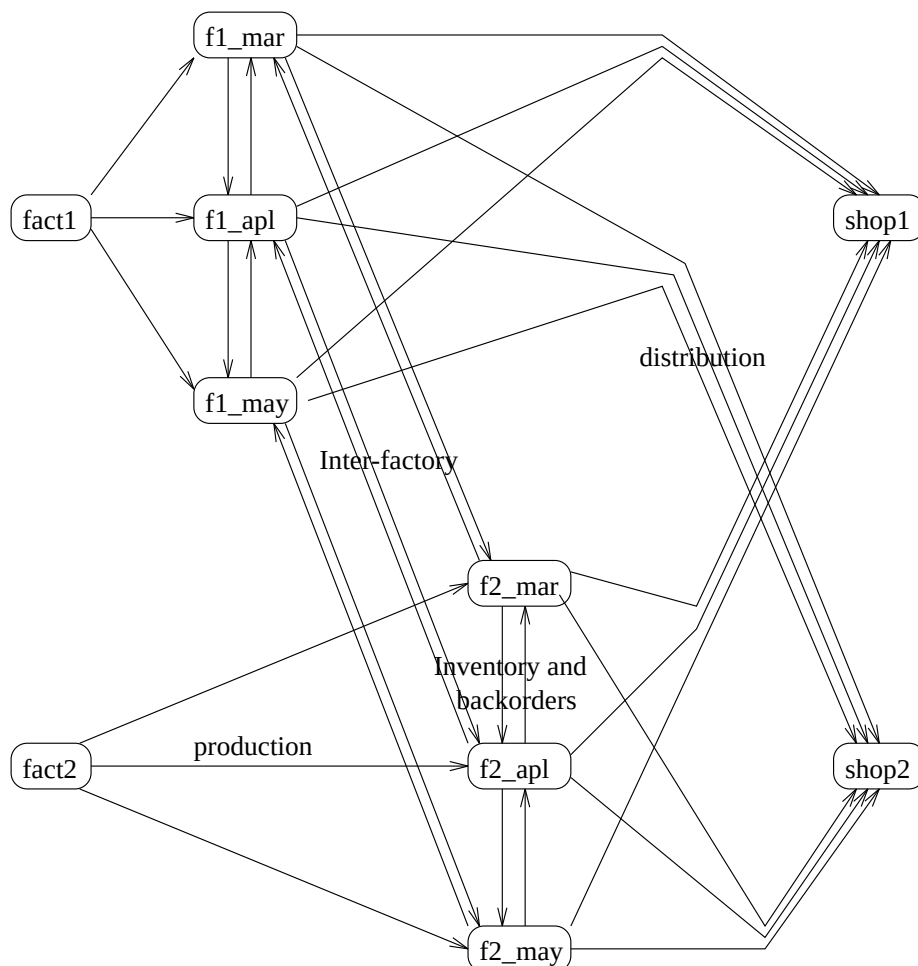


**Figure 5.12.** TV Problem

There are two similarly structured networks, one for the 19-inch televisions and the other for the 25-inch screen TVs. The minimum cost production, inventory, and distribution plan for both TV types can be determined in the same run of PROC INTPOINT. To ensure that node names are unambiguous, the names of nodes in the 19-inch network have suffix \_1, and the node names in the 25-inch network have suffix \_2.

---

### Example 5.1. Production, Inventory, Distribution Problem

The following code shows how to save a specific problem's data in data sets and solve the model with PROC INTPOINT:

```

title 'Production Planning/Inventory/Distribution';
title2 'Minimum Cost Flow problem';
title3;
data node0;
    input _node_ $ _supdem_ ;
    datalines;
fact1_1    1000
fact2_1     850
fact1_2    1000
fact2_2    1500
shop1_1    -900
shop2_1    -900
shop1_2    -900
shop2_2   -1450
;
data arc0;
    input _tail_ $ _head_ $ _cost_ _capac_ _lo_ diagonal factory
          key_id $10. mth_made $ _name_&$17. ;
    datalines;
fact1_1  f1_mar_1  127.9  500 50 19 1 production March prod f1 19 mar
fact1_1  f1_apr_1   78.6  600 50 19 1 production April prod f1 19 apl
fact1_1  f1_may_1   95.1  400 50 19 1 production May .
f1_mar_1 f1_apr_1   15     50 . 19 1 storage March .
f1_apr_1 f1_may_1   12     50 . 19 1 storage April .
f1_apr_1 f1_mar_1   28     20 . 19 1 backorder April back f1 19 apl
f1_may_1 f1_apr_1   28     20 . 19 1 backorder May back f1 19 may
f1_mar_1 f2_mar_1   11     . . 19 . f1_to_2 March .
f1_apr_1 f2_apr_1   11     . . 19 . f1_to_2 April .
f1_may_1 f2_may_1   16     . . 19 . f1_to_2 May .
f1_mar_1 shop1_1  -327.65 250 . 19 1 sales March .
f1_apr_1 shop1_1  -300    250 . 19 1 sales April .
f1_may_1 shop1_1  -285    250 . 19 1 sales May .
f1_mar_1 shop2_1  -362.74 250 . 19 1 sales March .
f1_apr_1 shop2_1  -300    250 . 19 1 sales April .
f1_may_1 shop2_1  -245    250 . 19 1 sales May .
fact2_1  f2_mar_1   88.0  450 35 19 2 production March prod f2 19 mar
fact2_1  f2_apr_1   62.4  480 35 19 2 production April prod f2 19 apl
fact2_1  f2_may_1  133.8  250 35 19 2 production May .
f2_mar_1 f2_apr_1   18     30 . 19 2 storage March .
f2_apr_1 f2_may_1   20     30 . 19 2 storage April .
f2_apr_1 f2_mar_1   17     15 . 19 2 backorder April back f2 19 apl
f2_may_1 f2_apr_1   25     15 . 19 2 backorder May back f2 19 may
f2_mar_1 f1_mar_1   10     40 . 19 . f2_to_1 March .
f2_apr_1 f1_apr_1   11     40 . 19 . f2_to_1 April .

```

```

f2_may_1 f1_may_1    13    40 . 19 . f2_to_1    May .
f2_mar_1 shop1_1 -297.4  250 . 19 2 sales    March .
f2_apr_1 shop1_1 -290    250 . 19 2 sales    April .
f2_may_1 shop1_1 -292    250 . 19 2 sales    May .
f2_mar_1 shop2_1 -272.7  250 . 19 2 sales    March .
f2_apr_1 shop2_1 -312    250 . 19 2 sales    April .
f2_may_1 shop2_1 -299    250 . 19 2 sales    May .
fact1_2  f1_mar_2  217.9  400 40 25 1 production March prod f1 25 mar
fact1_2  f1_apr_2  174.5  550 50 25 1 production April prod f1 25 apl
fact1_2  f1_may_2  133.3  350 40 25 1 production May .
f1_mar_2 f1_apr_2    20    40 . 25 1 storage    March .
f1_apr_2 f1_may_2    18    40 . 25 1 storage    April .
f1_apr_2 f1_mar_2    32    30 . 25 1 backorder April back f1 25 apl
f1_may_2 f1_apr_2    41    15 . 25 1 backorder May back f1 25 may
f1_mar_2 f2_mar_2    23 . . 25 . f1_to_2    March .
f1_apr_2 f2_apr_2    23 . . 25 . f1_to_2    April .
f1_may_2 f2_may_2    26 . . 25 . f1_to_2    May .
f1_mar_2 shop1_2 -559.76 . . 25 1 sales    March .
f1_apr_2 shop1_2 -524.28 . . 25 1 sales    April .
f1_may_2 shop1_2 -475.02 . . 25 1 sales    May .
f1_mar_2 shop2_2 -623.89 . . 25 1 sales    March .
f1_apr_2 shop2_2 -549.68 . . 25 1 sales    April .
f1_may_2 shop2_2 -460.00 . . 25 1 sales    May .
fact2_2  f2_mar_2  182.0  650 35 25 2 production March prod f2 25 mar
fact2_2  f2_apr_2  196.7  680 35 25 2 production April prod f2 25 apl
fact2_2  f2_may_2  201.4  550 35 25 2 production May .
f2_mar_2 f2_apr_2    28    50 . 25 2 storage    March .
f2_apr_2 f2_may_2    38    50 . 25 2 storage    April .
f2_apr_2 f2_mar_2    31    15 . 25 2 backorder April back f2 25 apl
f2_may_2 f2_apr_2    54    15 . 25 2 backorder May back f2 25 may
f2_mar_2 f1_mar_2    20    25 . 25 . f2_to_1    March .
f2_apr_2 f1_apr_2    21    25 . 25 . f2_to_1    April .
f2_may_2 f1_may_2    43    25 . 25 . f2_to_1    May .
f2_mar_2 shop1_2 -567.83 500 . 25 2 sales    March .
f2_apr_2 shop1_2 -542.19 500 . 25 2 sales    April .
f2_may_2 shop1_2 -461.56 500 . 25 2 sales    May .
f2_mar_2 shop2_2 -542.83 500 . 25 2 sales    March .
f2_apr_2 shop2_2 -559.19 500 . 25 2 sales    April .
f2_may_2 shop2_2 -489.06 500 . 25 2 sales    May .
;
proc intpoint
  bytes=1000000
  printlevel=2
  nodedata=node0
  arcdata=arc0
  conout=arc1;
run;

```

```

proc print data=arc1;
  var _from_ _to_ _cost_ _capac_ _lo_ _flow_ _fcost_
      diagonal factory key_id mth_made;
  sum _fcost_;
run;

```

The following notes appear on the SAS log:

```

NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 4 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 4350 , total demand= 4150 .
NOTE: Number of arcs= 64 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 21 .
NOTE: Number of >= constraints= 0 .
NOTE: Number of constraint coefficients= 136 .
NOTE: Number of variables= 68 .
NOTE: 0 columns, 0 rows and 0 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 54 nonzero elements in A * A transpose.
NOTE: Of the 21 rows and columns, 14 are sparse.
NOTE: There are 59 nonzero superdiagonal elements in the
      sparse rows of the factored A * A transpose. This
      includes fill-in.
NOTE: There are 80 operations of the form
       $u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$  to factorize the
      sparse rows of A * A transpose.

```

| Iter | Complem_aff | Complem-ity | Duality_gap  | Tot_infeasb | Tot_infeasc  | Tot_infeasd |
|------|-------------|-------------|--------------|-------------|--------------|-------------|
| 0    | -1.000000   | 167086681   | 0.835362     | 52835       | 26238        | 34922       |
| 1    | 35446536    | 17335378    | 0.912917     | 1663.290349 | 825.996205   | 0           |
| 2    | 1957002     | 878825      | 0.416636     | 0           | 1.956907E-12 | 0           |
| 3    | 550439      | 237247      | 0.155174     | 0           | 0            | 0           |
| 4    | 126652      | 53958       | 0.039021     | 0           | 0            | 0           |
| 5    | 27526       | 18386       | 0.013499     | 0           | 0            | 0           |
| 6    | 9153.811245 | 3029.529789 | 0.002242     | 0           | 0            | 0           |
| 7    | 905.846996  | 614.526357  | 0.000455     | 0           | 0            | 0           |
| 8    | 219.526873  | 73.592013   | 0.000054542  | 0           | 0            | 0           |
| 9    | 11.383058   | 2.111334    | 0.000001565  | 0           | 0            | 0           |
| 10   | 0.089558    | 0.000361    | 2.672549E-10 | 0           | 0            | 0           |

```

NOTE: Primal-Dual Predictor-Corrector Interior point
      algorithm performed 10 iterations.
NOTE: Objective = -1281110.35.
NOTE: The data set WORK.ARC1 has 64 observations and 18
      variables.
NOTE: There were 64 observations read from the data set
      WORK.ARC0.
NOTE: There were 8 observations read from the data set
      WORK.NODE0.
NOTE: The data set WORK.ARC1 has 64 observations and 18
      variables.

```

The solution is given in the CONOUT=arc1 data sets. In the CONOUT= data set, shown in Output 5.1.1, the variables diagonal, factory, key\_id, and mth\_made form an implicit ID list. The diagonal variable has one of two values, 19 or 25. factory also has one of two values, 1 or 2, to denote the factory where either production or storage occurs, from where TVs are either sold to shops or satisfy back-orders. production, storage, sales, and backorder are values of the key\_id variable.

Other values of this variable, f1\_to\_2 and f2\_to\_1, are used when flow through arcs represents the transportation of TVs between factories. The mth\_made variable has values March, April, and May, the months when TVs that are modeled as flow through an arc were made (assuming that no televisions are stored for more than one month and none manufactured in May are used to fill March back-orders).

These ID variables can be used after the PROC INTPOINT run to produce reports and perform analysis on particular parts of the company's operation. For example, reports can be generated for production numbers for each factory; optimal sales figures for each shop; and how many TVs should be stored, used to fill back-orders, sent to the other factory, or any combination of these, for TVs with a particular screen, those produced in a particular month, or both.

**Output 5.1.1.** CONOUT=ARC1

|          |          |         |                 |     |         |           |                 | d i f       |       | m t h |  |
|----------|----------|---------|-----------------|-----|---------|-----------|-----------------|-------------|-------|-------|--|
| t a i l  | h e a d  | c o s t | c a p a c i t y | l o | F L O W | F C O S T | g c o t n a r y | k e y _ i d |       |       |  |
| fact1_1  | f1_apr_1 | 78.6    | 600             | 50  | 600.000 | 47160.00  | 19 1            | production  | April |       |  |
| f1_mar_1 | f1_apr_1 | 15.0    | 50              | 0   | 0.000   | 0.00      | 19 1            | storage     | March |       |  |
| f1_may_1 | f1_apr_1 | 28.0    | 20              | 0   | 0.000   | 0.00      | 19 1            | backorder   | May   |       |  |
| f2_apr_1 | f1_apr_1 | 11.0    | 40              | 0   | 0.000   | 0.00      | 19 .            | f2_to_1     | April |       |  |
| fact1_2  | f1_apr_2 | 174.5   | 550             | 50  | 550.000 | 95975.00  | 25 1            | production  | April |       |  |
| f1_mar_2 | f1_apr_2 | 20.0    | 40              | 0   | 0.000   | 0.00      | 25 1            | storage     | March |       |  |
| f1_may_2 | f1_apr_2 | 41.0    | 15              | 0   | 15.000  | 615.00    | 25 1            | backorder   | May   |       |  |
| f2_apr_2 | f1_apr_2 | 21.0    | 25              | 0   | 0.000   | 0.00      | 25 .            | f2_to_1     | April |       |  |
| fact1_1  | f1_mar_1 | 127.9   | 500             | 50  | 345.000 | 44125.49  | 19 1            | production  | March |       |  |
| f1_apr_1 | f1_mar_1 | 28.0    | 20              | 0   | 20.000  | 560.00    | 19 1            | backorder   | April |       |  |
| f2_mar_1 | f1_mar_1 | 10.0    | 40              | 0   | 40.000  | 400.00    | 19 .            | f2_to_1     | March |       |  |
| fact1_2  | f1_mar_2 | 217.9   | 400             | 40  | 400.000 | 87160.00  | 25 1            | production  | March |       |  |
| f1_apr_2 | f1_mar_2 | 32.0    | 30              | 0   | 30.000  | 960.00    | 25 1            | backorder   | April |       |  |
| f2_mar_2 | f1_mar_2 | 20.0    | 25              | 0   | 25.000  | 500.00    | 25 .            | f2_to_1     | March |       |  |
| fact1_1  | f1_may_1 | 95.1    | 400             | 50  | 50.000  | 4755.01   | 19 1            | production  | May   |       |  |
| f1_apr_1 | f1_may_1 | 12.0    | 50              | 0   | 50.000  | 600.00    | 19 1            | storage     | April |       |  |
| f2_may_1 | f1_may_1 | 13.0    | 40              | 0   | 0.000   | 0.00      | 19 .            | f2_to_1     | May   |       |  |
| fact1_2  | f1_may_2 | 133.3   | 350             | 40  | 40.000  | 5332.00   | 25 1            | production  | May   |       |  |
| f1_apr_2 | f1_may_2 | 18.0    | 40              | 0   | 0.000   | 0.00      | 25 1            | storage     | April |       |  |
| f2_may_2 | f1_may_2 | 43.0    | 25              | 0   | 0.000   | 0.00      | 25 .            | f2_to_1     | May   |       |  |
| f1_apr_1 | f2_apr_1 | 11.0    | 99999999        | 0   | 30.000  | 330.00    | 19 .            | f1_to_2     | April |       |  |
| fact2_1  | f2_apr_1 | 62.4    | 480             | 35  | 480.000 | 29952.00  | 19 2            | production  | April |       |  |
| f2_mar_1 | f2_apr_1 | 18.0    | 30              | 0   | 0.000   | 0.00      | 19 2            | storage     | March |       |  |
| f2_may_1 | f2_apr_1 | 25.0    | 15              | 0   | 0.000   | 0.00      | 19 2            | backorder   | May   |       |  |
| f1_apr_2 | f2_apr_2 | 23.0    | 99999999        | 0   | 0.000   | 0.00      | 25 .            | f1_to_2     | April |       |  |
| fact2_2  | f2_apr_2 | 196.7   | 680             | 35  | 680.000 | 133756.00 | 25 2            | production  | April |       |  |
| f2_mar_2 | f2_apr_2 | 28.0    | 50              | 0   | 0.000   | 0.00      | 25 2            | storage     | March |       |  |
| f2_may_2 | f2_apr_2 | 54.0    | 15              | 0   | 15.000  | 810.00    | 25 2            | backorder   | May   |       |  |
| f1_mar_1 | f2_mar_1 | 11.0    | 99999999        | 0   | 0.000   | 0.00      | 19 .            | f1_to_2     | March |       |  |
| fact2_1  | f2_mar_1 | 88.0    | 450             | 35  | 290.000 | 25520.00  | 19 2            | production  | March |       |  |
| f2_apr_1 | f2_mar_1 | 17.0    | 15              | 0   | 0.000   | 0.00      | 19 2            | backorder   | April |       |  |
| f1_mar_2 | f2_mar_2 | 23.0    | 99999999        | 0   | 0.000   | 0.00      | 25 .            | f1_to_2     | March |       |  |

| fact     |          | head    | cost     |      | cap     | alloc      | flow  | fact |            | month |
|----------|----------|---------|----------|------|---------|------------|-------|------|------------|-------|
| table    | id       | name    | cost     | unit | price   | quantity   | value | fact | id         | month |
| fact2_2  | f2_mar_2 | 182.00  | 650      | 35   | 645.000 | 117390.00  | 25    | 2    | production | March |
| f2_apr_2 | f2_mar_2 | 31.00   | 15       | 0    | 0.000   | 0.00       | 25    | 2    | backorder  | April |
| f1_may_1 | f2_may_1 | 16.00   | 99999999 | 0    | 100.000 | 1600.00    | 19    | 1    | f1_to_2    | May   |
| fact2_1  | f2_may_1 | 133.80  | 250      | 35   | 35.000  | 4683.00    | 19    | 2    | production | May   |
| f2_apr_1 | f2_may_1 | 20.00   | 30       | 0    | 15.000  | 300.00     | 19    | 2    | storage    | April |
| f1_may_2 | f2_may_2 | 26.00   | 99999999 | 0    | 0.000   | 0.00       | 25    | 1    | f1_to_2    | May   |
| fact2_2  | f2_may_2 | 201.40  | 550      | 35   | 35.000  | 7049.00    | 25    | 2    | production | May   |
| f2_apr_2 | f2_may_2 | 38.00   | 50       | 0    | 0.000   | 0.00       | 25    | 2    | storage    | April |
| f1_mar_1 | shop1_1  | -327.65 | 250      | 0    | 155.000 | -50785.73  | 19    | 1    | sales      | March |
| f1_apr_1 | shop1_1  | -300.00 | 250      | 0    | 250.000 | -75000.00  | 19    | 1    | sales      | April |
| f1_may_1 | shop1_1  | -285.00 | 250      | 0    | 0.000   | 0.00       | 19    | 1    | sales      | May   |
| f2_mar_1 | shop1_1  | -297.40 | 250      | 0    | 250.000 | -74350.00  | 19    | 2    | sales      | March |
| f2_apr_1 | shop1_1  | -290.00 | 250      | 0    | 245.000 | -71050.02  | 19    | 2    | sales      | April |
| f2_may_1 | shop1_1  | -292.00 | 250      | 0    | 0.000   | 0.00       | 19    | 2    | sales      | May   |
| f1_mar_2 | shop1_2  | -559.76 | 99999999 | 0    | 0.000   | 0.00       | 25    | 1    | sales      | March |
| f1_apr_2 | shop1_2  | -524.28 | 99999999 | 0    | 0.000   | 0.00       | 25    | 1    | sales      | April |
| f1_may_2 | shop1_2  | -475.02 | 99999999 | 0    | 25.000  | -11875.50  | 25    | 1    | sales      | May   |
| f2_mar_2 | shop1_2  | -567.83 | 500      | 0    | 500.000 | -283915.00 | 25    | 2    | sales      | March |
| f2_apr_2 | shop1_2  | -542.19 | 500      | 0    | 375.000 | -203321.25 | 25    | 2    | sales      | April |
| f2_may_2 | shop1_2  | -461.56 | 500      | 0    | 0.000   | 0.00       | 25    | 2    | sales      | May   |
| f1_mar_1 | shop2_1  | -362.74 | 250      | 0    | 250.000 | -90685.00  | 19    | 1    | sales      | March |
| f1_apr_1 | shop2_1  | -300.00 | 250      | 0    | 250.000 | -75000.00  | 19    | 1    | sales      | April |
| f1_may_1 | shop2_1  | -245.00 | 250      | 0    | 0.000   | 0.00       | 19    | 1    | sales      | May   |
| f2_mar_1 | shop2_1  | -272.70 | 250      | 0    | 0.000   | 0.00       | 19    | 2    | sales      | March |
| f2_apr_1 | shop2_1  | -312.00 | 250      | 0    | 250.000 | -78000.00  | 19    | 2    | sales      | April |
| f2_may_1 | shop2_1  | -299.00 | 250      | 0    | 150.000 | -44850.00  | 19    | 2    | sales      | May   |
| f1_mar_2 | shop2_2  | -623.89 | 99999999 | 0    | 455.000 | -283869.95 | 25    | 1    | sales      | March |
| f1_apr_2 | shop2_2  | -549.68 | 99999999 | 0    | 535.000 | -294078.80 | 25    | 1    | sales      | April |
| f1_may_2 | shop2_2  | -460.00 | 99999999 | 0    | 0.000   | 0.00       | 25    | 1    | sales      | May   |
| f2_mar_2 | shop2_2  | -542.83 | 500      | 0    | 120.000 | -65139.60  | 25    | 2    | sales      | March |
| f2_apr_2 | shop2_2  | -559.19 | 500      | 0    | 320.000 | -178940.80 | 25    | 2    | sales      | April |
| f2_may_2 | shop2_2  | -489.06 | 500      | 0    | 20.000  | -9781.20   | 25    | 2    | sales      | May   |

### Example 5.2. Altering Arc Data

This example examines the effect of changing some of the arc costs. The backorder penalty costs are increased by 20 percent. The sales profit of 25-inch TVs sent to the shops in May is increased by 30 units. The backorder penalty costs of 25-inch TVs manufactured in May for April consumption is decreased by 30 units. The production cost of 19-inch and 25-inch TVs made in May are decreased by 5 units and 20 units, respectively. How does the optimal solution of the network after these arc cost alterations compare with the optimum of the original network?

These SAS statements produce the new NODEDATA= and ARCDATA= data sets:

```

title2 'Minimum Cost Flow problem- Altered Arc Data';
data arc2;
    set arc1;
        oldcost=_cost_;
        oldfc=_fcost_;
        oldflow=_flow_;
        if key_id='backorder'
            then _cost_=_cost_*1.2;
            else if _tail_='f2_may_2' then _cost_=_cost_-30;
        if key_id='production' & mth_made='May' then
            if diagonal=19 then _cost_=_cost_-5;
                else cost = cost -20;

```



```

proc intpoint
  bytes=100000
  printlevel2=2
  nodedata=node0
  arcdata=arc2
  conout=arc3;
run;

proc print data=arc3;
  var _tail_ _head_ _capac_ _lo_ _supply_ _demand_ _name_
      _cost_ _flow_ _fcost_ oldcost oldflow oldfc
      diagonal factory key_id mth_made;
  /* to get this variable order */
  sum oldfc _fcost_;
run;

```

The following notes appear on the SAS log:

```

NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 4 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 4350 , total demand= 4150 .
NOTE: Number of arcs= 64 .
NOTE: The following messages relate to the equivalent Linear
      Program solved by the Interior Point algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 21 .
NOTE: Number of >= constraints= 0 .
NOTE: Number of constraint coefficients= 136 .
NOTE: Number of variables= 68 .
NOTE: 0 columns, 0 rows and 0 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 54 nonzero elements in A * A transpose.
NOTE: Of the 21 rows and columns, 14 are sparse.
NOTE: There are 59 nonzero superdiagonal elements in the
      sparse rows of the factored A * A transpose. This
      includes fill-in.
NOTE: There are 80 operations of the form
      u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q] to factorize the
      sparse rows of A * A transpose.

```

| Iter | Complem_aff | Complem-ity | Duality_gap  | Tot_infeasb | Tot_infeasc  | Tot_infeasd |
|------|-------------|-------------|--------------|-------------|--------------|-------------|
| 0    | -1.000000   | 167898972   | 0.834344     | 52835       | 26238        | 35092       |
| 1    | 35575010    | 17420416    | 0.911474     | 1664.001693 | 826.349461   | 0           |
| 2    | 1969763     | 739523      | 0.369979     | 0           | 1.267542E-12 | 0           |
| 3    | 373828      | 189905      | 0.127240     | 0           | 0            | 0           |
| 4    | 117843      | 45042       | 0.032721     | 0           | 0            | 0           |
| 5    | 26090       | 15668       | 0.011493     | 0           | 0            | 0           |
| 6    | 8608.105391 | 2994.589912 | 0.002212     | 0           | 0            | 0           |
| 7    | 1477.385197 | 411.268625  | 0.000304     | 0           | 0            | 0           |
| 8    | 103.724839  | 24.733676   | 0.000018304  | 0           | 0            | 0           |
| 9    | 2.140932    | 0.002805    | 2.0758339E-9 | 0           | 0            | 0           |

```

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm
      performed 9 iterations.
NOTE: Objective = -1285086.45.
NOTE: The data set WORK.ARC3 has 64 observations and 21
      variables.

```

**NOTE:** There were 64 observations read from the data set  
WORK.ARC2.

**NOTE:** There were 8 observations read from the data set  
WORK.NODE0.

**NOTE:** The data set WORK.ARC3 has 64 observations and 21  
variables.

The solution is displayed in Output 5.2.1.

Output 5.2.1. CONOUT=ARC3

| _tail_    | _head_   | _capac_  | _lo_      | _SUPPLY_ | _DEMAND_ | _name_         | _cost_   | _FLOW_  |
|-----------|----------|----------|-----------|----------|----------|----------------|----------|---------|
| fact1_1   | f1_apr_1 | 600      | 50        | 1000     | .        | prod f1 19 apl | 78.6     | 540.000 |
| f1_mar_1  | f1_apr_1 | 50       | 0         | .        | .        |                | 15.0     | 0.000   |
| f1_may_1  | f1_apr_1 | 20       | 0         | .        | .        | back f1 19 may | 33.6     | 0.000   |
| f2_apr_1  | f1_apr_1 | 40       | 0         | .        | .        |                | 11.0     | 0.000   |
| fact1_2   | f1_apr_2 | 550      | 50        | 1000     | .        | prod f1 25 apl | 174.5    | 250.000 |
| f1_mar_2  | f1_apr_2 | 40       | 0         | .        | .        |                | 20.0     | 0.000   |
| f1_may_2  | f1_apr_2 | 15       | 0         | .        | .        | back f1 25 may | 49.2     | 15.000  |
| f2_apr_2  | f1_apr_2 | 25       | 0         | .        | .        |                | 21.0     | 0.000   |
| fact1_1   | f1_mar_1 | 500      | 50        | 1000     | .        | prod f1 19 mar | 127.9    | 340.000 |
| f1_apr_1  | f1_mar_1 | 20       | 0         | .        | .        | back f1 19 apl | 33.6     | 20.000  |
| f2_mar_1  | f1_mar_1 | 40       | 0         | .        | .        |                | 10.0     | 40.000  |
| fact1_2   | f1_mar_2 | 400      | 40        | 1000     | .        | prod f1 25 mar | 217.9    | 400.000 |
| f1_apr_2  | f1_mar_2 | 30       | 0         | .        | .        | back f1 25 apl | 38.4     | 30.000  |
| f2_mar_2  | f1_mar_2 | 25       | 0         | .        | .        |                | 20.0     | 25.000  |
| fact1_1   | f1_may_1 | 400      | 50        | 1000     | .        |                | 90.1     | 115.000 |
| f1_apr_1  | f1_may_1 | 50       | 0         | .        | .        |                | 12.0     | 0.000   |
| f2_may_1  | f1_may_1 | 40       | 0         | .        | .        |                | 13.0     | 0.000   |
| fact1_2   | f1_may_2 | 350      | 40        | 1000     | .        |                | 113.3    | 350.000 |
| f1_apr_2  | f1_may_2 | 40       | 0         | .        | .        |                | 18.0     | 0.000   |
| f2_may_2  | f1_may_2 | 25       | 0         | .        | .        |                | 13.0     | 0.000   |
| f1_apr_1  | f2_apr_1 | 99999999 | 0         | .        | .        |                | 11.0     | 20.000  |
| fact2_1   | f2_apr_1 | 480      | 35        | 850      | .        | prod f2 19 apl | 62.4     | 480.000 |
| f2_mar_1  | f2_apr_1 | 30       | 0         | .        | .        |                | 18.0     | 0.000   |
| f2_may_1  | f2_apr_1 | 15       | 0         | .        | .        | back f2 19 may | 30.0     | 0.000   |
| f1_apr_2  | f2_apr_2 | 99999999 | 0         | .        | .        |                | 23.0     | 0.000   |
| fact2_2   | f2_apr_2 | 680      | 35        | 1500     | .        | prod f2 25 apl | 196.7    | 680.000 |
| f2_mar_2  | f2_apr_2 | 50       | 0         | .        | .        |                | 28.0     | 0.000   |
| _FCOST_   | oldcost  | oldflow  | oldfc     | diagonal | factory  | key_id         | mth_made |         |
| 42443.99  | 78.6     | 600.000  | 47160.00  | 19       | 1        | production     | April    |         |
| 0.00      | 15.0     | 0.000    | 0.00      | 19       | 1        | storage        | March    |         |
| 0.00      | 28.0     | 0.000    | 0.00      | 19       | 1        | backorder      | May      |         |
| 0.00      | 11.0     | 0.000    | 0.00      | 19       | .        | f2_to_1        | April    |         |
| 43625.00  | 174.5    | 550.000  | 95975.00  | 25       | 1        | production     | April    |         |
| 0.00      | 20.0     | 0.000    | 0.00      | 25       | 1        | storage        | March    |         |
| 738.00    | 41.0     | 15.000   | 615.00    | 25       | 1        | backorder      | May      |         |
| 0.00      | 21.0     | 0.000    | 0.00      | 25       | .        | f2_to_1        | April    |         |
| 43486.01  | 127.9    | 345.000  | 44125.49  | 19       | 1        | production     | March    |         |
| 672.00    | 28.0     | 20.000   | 560.00    | 19       | 1        | backorder      | April    |         |
| 400.00    | 10.0     | 40.000   | 400.00    | 19       | .        | f2_to_1        | March    |         |
| 87160.00  | 217.9    | 400.000  | 87160.00  | 25       | 1        | production     | March    |         |
| 1152.00   | 32.0     | 30.000   | 960.00    | 25       | 1        | backorder      | April    |         |
| 500.00    | 20.0     | 25.000   | 500.00    | 25       | .        | f2_to_1        | March    |         |
| 10361.50  | 95.1     | 50.000   | 4755.01   | 19       | 1        | production     | May      |         |
| 0.00      | 12.0     | 50.000   | 600.00    | 19       | 1        | storage        | April    |         |
| 0.00      | 13.0     | 0.000    | 0.00      | 19       | .        | f2_to_1        | May      |         |
| 39655.00  | 133.3    | 40.000   | 5332.00   | 25       | 1        | production     | May      |         |
| 0.00      | 18.0     | 0.000    | 0.00      | 25       | 1        | storage        | April    |         |
| 0.00      | 43.0     | 0.000    | 0.00      | 25       | .        | f2_to_1        | May      |         |
| 220.00    | 11.0     | 30.000   | 330.00    | 19       | .        | f1_to_2        | April    |         |
| 29952.00  | 62.4     | 480.000  | 29952.00  | 19       | 2        | production     | April    |         |
| 0.00      | 18.0     | 0.000    | 0.00      | 19       | 2        | storage        | March    |         |
| 0.00      | 25.0     | 0.000    | 0.00      | 19       | 2        | backorder      | May      |         |
| 0.00      | 23.0     | 0.000    | 0.00      | 25       | .        | f1_to_2        | April    |         |
| 133756.00 | 196.7    | 680.000  | 133756.00 | 25       | 2        | production     | April    |         |
| 0.00      | 28.0     | 0.000    | 0.00      | 25       | 2        | storage        | March    |         |
| _tail_    | _head_   | _capac_  | _lo_      | _SUPPLY_ | _DEMAND_ | _name_         | _cost_   | _FLOW_  |
| f2_may_2  | f2_apr_2 | 15       | 0         | .        | .        | back f2 25 may | 64.8     | 0.000   |
| f1_mar_1  | f2_mar_1 | 99999999 | 0         | .        | .        |                | 11.0     | 0.000   |
| fact2_1   | f2_mar_1 | 450      | 35        | 850      | .        | prod f2 19 mar | 88.0     | 290.000 |
| f2_apr_1  | f2_mar_1 | 15       | 0         | .        | .        | back f2 19 apl | 20.4     | 0.000   |
| f1_mar_2  | f2_mar_2 | 99999999 | 0         | .        | .        |                | 23.0     | 0.000   |
| _FCOST_   | oldcost  | oldflow  | oldfc     | diagonal | factory  | key_id         | mth_made |         |
| 0.00      | 54.0     | 15.000   | 810.00    | 25       | 2        | backorder      | May      |         |
| 0.00      | 11.0     | 0.000    | 0.00      | 19       | .        | f1_to_2        | March    |         |
| 25520.00  | 88.0     | 290.000  | 25520.00  | 19       | 2        | production     | March    |         |
| 0.00      | 17.0     | 0.000    | 0.00      | 19       | 2        | backorder      | April    |         |
| 0.00      | 23.0     | 0.000    | 0.00      | 25       | .        | f1_to_2        | March    |         |

| _tail_     | _head_   | _capac_  | _lo_           | _SUPPLY_ | _DEMAND_ | _name_         | _cost_   | _FLOW_  |
|------------|----------|----------|----------------|----------|----------|----------------|----------|---------|
| fact2_2    | f2_mar_2 | 650      | 35             | 1500     | .        | prod f2 25 mar | 182.00   | 635.000 |
| f2_apr_2   | f2_mar_2 | 15       | 0              | .        | .        | back f2 25 apl | 37.20    | 0.000   |
| f1_may_1   | f2_may_1 | 99999999 | 0              | .        | .        |                | 16.00    | 115.000 |
| fact2_1    | f2_may_1 | 250      | 35             | 850      | .        |                | 128.80   | 35.000  |
| f2_apr_1   | f2_may_1 | 30       | 0              | .        | .        |                | 20.00    | 0.000   |
| f1_may_2   | f2_may_2 | 99999999 | 0              | .        | .        |                | 26.00    | 335.000 |
| fact2_2    | f2_may_2 | 550      | 35             | 1500     | .        |                | 181.40   | 35.000  |
| f2_apr_2   | f2_may_2 | 50       | 0              | .        | .        |                | 38.00    | 0.000   |
| f1_mar_1   | shop1_1  | 250      | 0              | .        | 900      |                | -327.65  | 150.000 |
| f1_apr_1   | shop1_1  | 250      | 0              | .        | 900      |                | -300.00  | 250.000 |
| f1_may_1   | shop1_1  | 250      | 0              | .        | 900      |                | -285.00  | 0.000   |
| f2_mar_1   | shop1_1  | 250      | 0              | .        | 900      |                | -297.40  | 250.000 |
| f2_apr_1   | shop1_1  | 250      | 0              | .        | 900      |                | -290.00  | 250.000 |
| f2_may_1   | shop1_1  | 250      | 0              | .        | 900      |                | -292.00  | 0.000   |
| f1_mar_2   | shop1_2  | 99999999 | 0              | .        | 900      |                | -559.76  | 0.000   |
| f1_apr_2   | shop1_2  | 99999999 | 0              | .        | 900      |                | -524.28  | 0.000   |
| f1_may_2   | shop1_2  | 99999999 | 0              | .        | 900      |                | -475.02  | 0.000   |
| f2_mar_2   | shop1_2  | 500      | 0              | .        | 900      |                | -567.83  | 500.000 |
| f2_apr_2   | shop1_2  | 500      | 0              | .        | 900      |                | -542.19  | 400.000 |
| f2_may_2   | shop1_2  | 500      | 0              | .        | 900      |                | -491.56  | 0.000   |
| f1_mar_1   | shop2_1  | 250      | 0              | .        | 900      |                | -362.74  | 250.000 |
| f1_apr_1   | shop2_1  | 250      | 0              | .        | 900      |                | -300.00  | 250.000 |
| f1_may_1   | shop2_1  | 250      | 0              | .        | 900      |                | -245.00  | 0.000   |
| f2_mar_1   | shop2_1  | 250      | 0              | .        | 900      |                | -272.70  | 0.000   |
| f2_apr_1   | shop2_1  | 250      | 0              | .        | 900      |                | -312.00  | 250.000 |
| f2_may_1   | shop2_1  | 250      | 0              | .        | 900      |                | -299.00  | 150.000 |
| f1_mar_2   | shop2_2  | 99999999 | 0              | .        | 1450     |                | -623.89  | 455.000 |
|            |          |          |                |          |          |                |          |         |
| _FCOST_    | oldcost  | oldflow  | oldfc diagonal |          |          | factory key_id | mth_made |         |
| 115570.00  | 182.00   | 645.000  | 117390.00      | 25       | 2        | production     | March    |         |
| 0.00       | 31.00    | 0.000    | 0.00           | 25       | 2        | backorder      | April    |         |
| 1840.00    | 16.00    | 100.000  | 1600.00        | 19       | .        | f1_to_2        | May      |         |
| 4508.00    | 133.80   | 35.000   | 4683.00        | 19       | 2        | production     | May      |         |
| 0.00       | 20.00    | 15.000   | 300.00         | 19       | 2        | storage        | April    |         |
| 8710.00    | 26.00    | 0.000    | 0.00           | 25       | .        | f1_to_2        | May      |         |
| 6349.00    | 201.40   | 35.000   | 7049.00        | 25       | 2        | production     | May      |         |
| 0.00       | 38.00    | 0.000    | 0.00           | 25       | 2        | storage        | April    |         |
| -49147.54  | -327.65  | 155.000  | -50785.73      | 19       | 1        | sales          | March    |         |
| -75000.00  | -300.00  | 250.000  | -75000.00      | 19       | 1        | sales          | April    |         |
| 0.00       | -285.00  | 0.000    | 0.00           | 19       | 1        | sales          | May      |         |
| -74350.00  | -297.40  | 250.000  | -74350.00      | 19       | 2        | sales          | March    |         |
| -72499.97  | -290.00  | 245.000  | -71050.02      | 19       | 2        | sales          | April    |         |
| 0.00       | -292.00  | 0.000    | 0.00           | 19       | 2        | sales          | May      |         |
| 0.00       | -559.76  | 0.000    | 0.00           | 25       | 1        | sales          | March    |         |
| 0.00       | -524.28  | 0.000    | 0.00           | 25       | 1        | sales          | April    |         |
| -0.02      | -475.02  | 25.000   | -11875.50      | 25       | 1        | sales          | May      |         |
| -283915.00 | -567.83  | 500.000  | -283915.00     | 25       | 2        | sales          | March    |         |
| -216875.98 | -542.19  | 375.000  | -203321.25     | 25       | 2        | sales          | April    |         |
| 0.00       | -461.56  | 0.000    | 0.00           | 25       | 2        | sales          | May      |         |
| -90685.00  | -362.74  | 250.000  | -90685.00      | 19       | 1        | sales          | March    |         |
| -75000.00  | -300.00  | 250.000  | -75000.00      | 19       | 1        | sales          | April    |         |
| 0.00       | -245.00  | 0.000    | 0.00           | 19       | 1        | sales          | May      |         |
| 0.00       | -272.70  | 0.000    | 0.00           | 19       | 2        | sales          | March    |         |
| -78000.00  | -312.00  | 250.000  | -78000.00      | 19       | 2        | sales          | April    |         |
| -44850.00  | -299.00  | 150.000  | -44850.00      | 19       | 2        | sales          | May      |         |
| -283869.95 | -623.89  | 455.000  | -283869.95     | 25       | 1        | sales          | March    |         |
|            |          |          |                |          |          |                |          |         |
| _tail_     | _head_   | _capac_  | _lo_           | _SUPPLY_ | _DEMAND_ | _name_         | _cost_   | _FLOW_  |
| f1_apr_2   | shop2_2  | 99999999 | 0              | .        | 1450     |                | -549.68  | 235.000 |
| f1_may_2   | shop2_2  | 99999999 | 0              | .        | 1450     |                | -460.00  | 0.000   |
| f2_mar_2   | shop2_2  | 500      | 0              | .        | 1450     |                | -542.83  | 110.000 |
| f2_apr_2   | shop2_2  | 500      | 0              | .        | 1450     |                | -559.19  | 280.000 |
| f2_may_2   | shop2_2  | 500      | 0              | .        | 1450     |                | -519.06  | 370.000 |
|            |          |          |                |          |          |                |          |         |
| _FCOST_    | oldcost  | oldflow  | oldfc diagonal |          |          | factory key_id | mth_made |         |
| -129174.80 | -549.68  | 535.000  | -294078.80     | 25       | 1        | sales          | April    |         |
| 0.00       | -460.00  | 0.000    | 0.00           | 25       | 1        | sales          | May      |         |
| -59711.31  | -542.83  | 120.000  | -65139.60      | 25       | 2        | sales          | March    |         |
| -156573.22 | -559.19  | 320.000  | -178940.80     | 25       | 2        | sales          | April    |         |
| -192052.18 | -489.06  | 20.000   | -9781.20       | 25       | 2        | sales          | May      |         |

### Example 5.3. Adding Side Constraints

The manufacturer of Gizmo chips, which are parts needed to make televisions, can supply only 2,600 chips to factory 1 and 3,750 chips to factory 2 in time for production in each of the months March and April. However, Gizmo chips will not be in short supply in May. Three chips are required to make each 19-inch TV while the 25-inch TVs require four chips each. To limit the production of televisions produced at factory 1 in March so that the TVs have the correct number of chips, a side constraint called FACT1 MAR GIZMO is used. The form of this constraint is

```
3 * prod f1 19 mar + 4 * prod f1 25 mar <= 2600
```

prod f1 19 mar is the name of the arc directed from the node fact1\_1 toward node f1\_mar\_1 and, in the previous constraint, designates the flow assigned to this arc. The ARCDATA= and CONOUT= data sets have arc names in a variable called \_name\_.

The other side constraints (shown below) are called FACT2 MAR GIZMO, FACT1 APL GIZMO, and FACT2 APL GIZMO.

```
3 * prod f2 19 mar + 4 * prod f2 25 mar <= 3750
3 * prod f1 19 apl + 4 * prod f1 25 apl <= 2600
3 * prod f2 19 apl + 4 * prod f2 25 apl <= 3750
```

To maintain customer goodwill, the total number of backorders is not to exceed 50 sets. The side constraint TOTAL BACKORDER that models this restriction is

```
back f1 19 apl + back f1 25 apl +
back f2 19 apl + back f2 25 apl +
back f1 19 may + back f1 25 may +
back f2 19 may + back f2 25 may <= 50
```

The sparse CONDATA= data set format is used. All side constraints are of less than or equal type. Because this is the default type value for the DEFCONTYPE= option, type information is not necessary in the following CONDATA=con3. Also, DEFCONTYPE='<=' does not have to be specified in the PROC INTPOINT statement that follows. Notice that the \_column\_ variable value CHIP/BO LIMIT indicates that an observation of the con3 data set contains rhs information. Therefore, specify RHSOBS='CHIP/BO LIMIT':

```
title2 'Adding Side Constraints';
data con3;
  input _column_ &$14. _row_ &$15. _coef_ ;
  datalines;
prod f1 19 mar FACT1 MAR GIZMO 3
prod f1 25 mar FACT1 MAR GIZMO 4
CHIP/BO LIMIT FACT1 MAR GIZMO 2600
prod f2 19 mar FACT2 MAR GIZMO 3
prod f2 25 mar FACT2 MAR GIZMO 4
CHIP/BO LIMIT FACT2 MAR GIZMO 3750
prod f1 19 apl FACT1 APL GIZMO 3
```

```

prod f1 25 apl FACT1 APL GIZMO 4
CHIP/B0 LIMIT FACT1 APL GIZMO 2600
prod f2 19 apl FACT2 APL GIZMO 3
prod f2 25 apl FACT2 APL GIZMO 4
CHIP/B0 LIMIT FACT2 APL GIZMO 3750
back f1 19 apl TOTAL BACKORDER 1
back f1 25 apl TOTAL BACKORDER 1
back f2 19 apl TOTAL BACKORDER 1
back f2 25 apl TOTAL BACKORDER 1
back f1 19 may TOTAL BACKORDER 1
back f1 25 may TOTAL BACKORDER 1
back f2 19 may TOTAL BACKORDER 1
back f2 25 may TOTAL BACKORDER 1
CHIP/B0 LIMIT TOTAL BACKORDER 50
;

```

The four pairs of data sets that follow can be used as ARCDATA= and NODEDATA= data sets in the following PROC INTPOINT run. The set used depends on which cost information the arcs are to have.

```

ARCDATA=arc0      NODEDATA=node0
ARCDATA=arc1      NODEDATA=node0
ARCDATA=arc2      NODEDATA=node0
ARCDATA=arc3      NODEDATA=node0

```

arc0, node0, and arc1 were created in Example 5.1. The first two data sets are the original input data sets.

In the previous example, arc2 was created by modifying arc1 to reflect different arc costs. arc2 and node0 can also be used as the ARCDATA= and NODEDATA= data sets in a PROC INTPOINT run.

If you are going to continue optimization using the changed arc costs, it is probably best to use arc3 and node0 as the ARCDATA= and NODEDATA= data sets.

PROC INTPOINT is used to find the changed cost network solution that obeys the chip limit and backorder side constraints. An explicit ID list has also been specified so that the variables oldcost, oldfc, and oldflow do not appear in the subsequent output data sets:

```

proc intpoint
  bytes=1000000
  print level2=2
  nodedata=node0 arcdata=arc3
  condata=con3 sparsesecondata rhsobs='CHIP/B0 LIMIT'
  conout=arc4;
  id diagonal factory key_id mth_made;
run;

```

```

proc print data=arc4;
  var _tail_ _head_ _cost_ _capac_ _lo_ _flow_ _fcost_;
  /* to get this variable order */
  sum _fcost_;
run;

```

The following messages appear on the SAS log:

NOTE: The following variables in ARCDATA do not belong to any SAS variable list. These will be ignored.

```

_FLOW_
_FCOST_
_RCost_
_ANUMB_
_TNUMB_
_STATUS_
oldcost
oldfc
oldflow

```

NOTE: Number of nodes= 20 .

NOTE: Number of supply nodes= 4 .

NOTE: Number of demand nodes= 4 .

NOTE: Total supply= 4350 , total demand= 4150 .

NOTE: Number of arcs= 64 .

NOTE: Number of <= side constraints= 5 .

NOTE: Number of == side constraints= 0 .

NOTE: Number of >= side constraints= 0 .

NOTE: Number of side constraint coefficients= 16 .

NOTE: The following messages relate to the equivalent Linear Program solved by the Interior Point algorithm.

NOTE: Number of <= constraints= 5 .

NOTE: Number of == constraints= 21 .

NOTE: Number of >= constraints= 0 .

NOTE: Number of constraint coefficients= 152 .

NOTE: Number of variables= 68 .

NOTE: 5 columns, 0 rows and 5 coefficients were added to the problem to handle unrestricted variables, variables that are split, and constraint slack or surplus variables.

NOTE: There are 82 nonzero elements in A \* A transpose.

NOTE: Of the 26 rows and columns, 14 are sparse.

NOTE: There are 80 nonzero superdiagonal elements in the sparse rows of the factored A \* A transpose. This includes fill-in.

NOTE: There are 68 operations of the form

$u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$  to factorize the sparse rows of A \* A transpose.

| Iter | Complem_    | aff         | Complem-ity | Duality_gap | Tot_infeasb | Tot_infeasc  | Tot_infeasd |
|------|-------------|-------------|-------------|-------------|-------------|--------------|-------------|
| 0    | -1.000000   |             | 174740627   | 0.834344    | 52835       | 40217        | 46058       |
| 1    | 48947018    |             | 21567675    | 0.910848    | 2958.654286 | 2252.108767  | 2470.152448 |
| 2    | 4201633     |             | 1377647     | 0.520587    | 0           | 1.331557E-11 | 46.321197   |
| 3    | 377792      |             | 253074      | 0.163432    | 0           | 0            | 8.455959    |
| 4    | 124457      |             | 71148       | 0.051146    | 0           | 0            | 1.425193    |
| 5    | 46811       |             | 28865       | 0.021063    | 0           | 0            | 0.532792    |
| 6    | 11689       | 6377.059370 |             | 0.004710    | 0           | 0            | 0.092989    |
| 7    | 3201.780084 | 1923.406854 |             | 0.001424    | 0           | 0            | 0.019251    |
| 8    | 468.068267  | 245.400716  |             | 0.000182    | 0           | 0            | 0.002654    |

|    |           |          |              |   |   |   |
|----|-----------|----------|--------------|---|---|---|
| 9  | 29.762950 | 5.252240 | 0.000003889  | 0 | 0 | 0 |
| 10 | 0.005544  | 0.000290 | -4.924982E-9 | 0 | 0 | 0 |

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 10 iterations.

NOTE: Objective = -1282708.625.

NOTE: The data set WORK.ARC4 has 64 observations and 14 variables.

NOTE: There were 64 observations read from the data set WORK.ARC3.

NOTE: There were 8 observations read from the data set WORK.NODE0.

NOTE: There were 21 observations read from the data set WORK.CON3.

NOTE: The data set WORK.ARC4 has 64 observations and 14 variables.



**Output 5.3.1.** CONOUT=ARC4

| Adding Side Constraints |          |          |         |          |      |         |            |
|-------------------------|----------|----------|---------|----------|------|---------|------------|
| Obs                     | _tail_   | _head_   | _cost_  | _capac_  | _lo_ | _FLOW_  | _FCOST_    |
| 1                       | fact1_1  | f1_apr_1 | 78.60   | 600      | 50   | 533.333 | 41920.00   |
| 2                       | f1_mar_1 | f1_apr_1 | 15.00   | 50       | 0    | 0.000   | 0.00       |
| 3                       | f1_may_1 | f1_apr_1 | 33.60   | 20       | 0    | 0.000   | 0.00       |
| 4                       | f2_apr_1 | f1_apr_1 | 11.00   | 40       | 0    | 0.000   | 0.00       |
| 5                       | fact1_2  | f1_apr_2 | 174.50  | 550      | 50   | 250.000 | 43625.00   |
| 6                       | f1_mar_2 | f1_apr_2 | 20.00   | 40       | 0    | 0.000   | 0.00       |
| 7                       | f1_may_2 | f1_apr_2 | 49.20   | 15       | 0    | 0.000   | 0.00       |
| 8                       | f2_apr_2 | f1_apr_2 | 21.00   | 25       | 0    | 0.000   | 0.00       |
| 9                       | fact1_1  | f1_mar_1 | 127.90  | 500      | 50   | 333.333 | 42633.33   |
| 10                      | f1_apr_1 | f1_mar_1 | 33.60   | 20       | 0    | 20.000  | 672.00     |
| 11                      | f2_mar_1 | f1_mar_1 | 10.00   | 40       | 0    | 40.000  | 400.00     |
| 12                      | fact1_2  | f1_mar_2 | 217.90  | 400      | 40   | 400.000 | 87160.00   |
| 13                      | f1_apr_2 | f1_mar_2 | 38.40   | 30       | 0    | 30.000  | 1152.00    |
| 14                      | f2_mar_2 | f1_mar_2 | 20.00   | 25       | 0    | 25.000  | 500.00     |
| 15                      | fact1_1  | f1_may_1 | 90.10   | 400      | 50   | 128.333 | 11562.83   |
| 16                      | f1_apr_1 | f1_may_1 | 12.00   | 50       | 0    | 0.000   | 0.00       |
| 17                      | f2_may_1 | f1_may_1 | 13.00   | 40       | 0    | 0.000   | 0.00       |
| 18                      | fact1_2  | f1_may_2 | 113.30  | 350      | 40   | 350.000 | 39655.00   |
| 19                      | f1_apr_2 | f1_may_2 | 18.00   | 40       | 0    | 0.000   | 0.00       |
| 20                      | f2_may_2 | f1_may_2 | 13.00   | 25       | 0    | 0.000   | 0.00       |
| 21                      | f1_apr_1 | f2_apr_1 | 11.00   | 99999999 | 0    | 13.333  | 146.67     |
| 22                      | fact2_1  | f2_apr_1 | 62.40   | 480      | 35   | 480.000 | 29952.00   |
| 23                      | f2_mar_1 | f2_apr_1 | 18.00   | 30       | 0    | 0.000   | 0.00       |
| 24                      | f2_may_1 | f2_apr_1 | 30.00   | 15       | 0    | 0.000   | 0.00       |
| 25                      | f1_apr_2 | f2_apr_2 | 23.00   | 99999999 | 0    | 0.000   | 0.00       |
| 26                      | fact2_2  | f2_apr_2 | 196.70  | 680      | 35   | 577.500 | 113594.25  |
| 27                      | f2_mar_2 | f2_apr_2 | 28.00   | 50       | 0    | 0.000   | 0.00       |
| 28                      | f2_may_2 | f2_apr_2 | 64.80   | 15       | 0    | 0.000   | 0.00       |
| 29                      | f1_mar_1 | f2_mar_1 | 11.00   | 99999999 | 0    | 0.000   | 0.00       |
| 30                      | fact2_1  | f2_mar_1 | 88.00   | 450      | 35   | 290.000 | 25520.00   |
| 31                      | f2_apr_1 | f2_mar_1 | 20.40   | 15       | 0    | 0.000   | 0.00       |
| 32                      | f1_mar_2 | f2_mar_2 | 23.00   | 99999999 | 0    | 0.000   | 0.00       |
| 33                      | fact2_2  | f2_mar_2 | 182.00  | 650      | 35   | 650.000 | 118300.00  |
| 34                      | f2_apr_2 | f2_mar_2 | 37.20   | 15       | 0    | 0.000   | 0.00       |
| 35                      | f1_may_1 | f2_may_1 | 16.00   | 99999999 | 0    | 115.000 | 1840.00    |
| 36                      | fact2_1  | f2_may_1 | 128.80  | 250      | 35   | 35.000  | 4508.00    |
| 37                      | f2_apr_1 | f2_may_1 | 20.00   | 30       | 0    | 0.000   | 0.00       |
| 38                      | f1_may_2 | f2_may_2 | 26.00   | 99999999 | 0    | 350.000 | 9100.00    |
| 39                      | fact2_2  | f2_may_2 | 181.40  | 550      | 35   | 122.500 | 22221.50   |
| 40                      | f2_apr_2 | f2_may_2 | 38.00   | 50       | 0    | 0.000   | 0.00       |
| 41                      | f1_mar_1 | shop1_1  | -327.65 | 250      | 0    | 143.333 | -46963.17  |
| 42                      | f1_apr_1 | shop1_1  | -300.00 | 250      | 0    | 250.000 | -75000.00  |
| 43                      | f1_may_1 | shop1_1  | -285.00 | 250      | 0    | 13.333  | -3800.00   |
| 44                      | f2_mar_1 | shop1_1  | -297.40 | 250      | 0    | 250.000 | -74350.00  |
| 45                      | f2_apr_1 | shop1_1  | -290.00 | 250      | 0    | 243.333 | -70566.67  |
| 46                      | f2_may_1 | shop1_1  | -292.00 | 250      | 0    | 0.000   | 0.00       |
| 47                      | f1_mar_2 | shop1_2  | -559.76 | 99999999 | 0    | 0.000   | 0.00       |
| 48                      | f1_apr_2 | shop1_2  | -524.28 | 99999999 | 0    | 0.000   | 0.00       |
| 49                      | f1_may_2 | shop1_2  | -475.02 | 99999999 | 0    | 0.000   | 0.00       |
| 50                      | f2_mar_2 | shop1_2  | -567.83 | 500      | 0    | 500.000 | -283915.00 |
| 51                      | f2_apr_2 | shop1_2  | -542.19 | 500      | 0    | 400.000 | -216876.00 |
| 52                      | f2_may_2 | shop1_2  | -491.56 | 500      | 0    | 0.000   | 0.00       |
| 53                      | f1_mar_1 | shop2_1  | -362.74 | 250      | 0    | 250.000 | -90685.00  |
| 54                      | f1_apr_1 | shop2_1  | -300.00 | 250      | 0    | 250.000 | -75000.00  |
| 55                      | f1_may_1 | shop2_1  | -245.00 | 250      | 0    | 0.000   | 0.00       |
| Adding Side Constraints |          |          |         |          |      |         |            |
| Obs                     | _tail_   | _head_   | _cost_  | _capac_  | _lo_ | _FLOW_  | _FCOST_    |
| 56                      | f2_mar_1 | shop2_1  | -272.70 | 250      | 0    | 0.0     | 0.00       |
| 57                      | f2_apr_1 | shop2_1  | -312.00 | 250      | 0    | 250.0   | -78000.00  |
| 58                      | f2_may_1 | shop2_1  | -299.00 | 250      | 0    | 150.0   | -44850.00  |
| 59                      | f1_mar_2 | shop2_2  | -623.89 | 99999999 | 0    | 455.0   | -283869.95 |
| 60                      | f1_apr_2 | shop2_2  | -549.68 | 99999999 | 0    | 220.0   | -120929.60 |
| 61                      | f1_may_2 | shop2_2  | -460.00 | 99999999 | 0    | 0.0     | 0.00       |
| 62                      | f2_mar_2 | shop2_2  | -542.83 | 500      | 0    | 125.0   | -67853.75  |
| 63                      | f2_apr_2 | shop2_2  | -559.19 | 500      | 0    | 177.5   | -99256.23  |
| 64                      | f2_may_2 | shop2_2  | -519.06 | 500      | 0    | 472.5   | -245255.85 |
| =====                   |          |          |         |          |      |         |            |
| -1282708.63             |          |          |         |          |      |         |            |

## Example 5.4. Using Constraints and More Alteration to Arc Data

Suppose the 25-inch screen TVs produced at factory 1 in May can be sold at either shop with an increased profit of 40 dollars each. What is the new optimal solution?

```

title2 'Using Constraints and Altering arc data';
data new_arc4;
  set arc4;
  oldcost=_cost_;
  oldflow=_flow_;
  oldfc=_fcost_;
  if _tail_='f1_may_2'
    & (_head_='shop1_2' | _head_='shop2_2')
    then _cost_=_cost_-40;

proc intpoint
  bytes=1000000
  printlevel=2
  arcdata=new_arc4 nodedata=node0
  condata=con3 sparseseconddata rhsobs='CHIP/B0 LIMIT'
  conout=arc5;
run;
proc print data=arc5 (drop = _status_ _rcost_);
  var _tail_ _head_ _cost_ _capac_ _lo_
      _supply_ _demand_ _name_
      _flow_ _fcost_ oldflow oldfc;
      /* to get this variable order */
  sum oldfc _fcost_;
run;

```

The following messages appear on the SAS log:

```

NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 4 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 4350 , total demand= 4150 .
NOTE: Number of arcs= 64 .
NOTE: Number of <= side constraints= 5 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of >= side constraints= 0 .
NOTE: Number of side constraint coefficients= 16 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 5 .
NOTE: Number of == constraints= 21 .
NOTE: Number of >= constraints= 0 .
NOTE: Number of constraint coefficients= 152 .
NOTE: Number of variables= 68 .
NOTE: 5 columns, 0 rows and 5 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 82 nonzero elements in A * A transpose.
NOTE: Of the 26 rows and columns, 14 are sparse.
NOTE: There are 80 nonzero superdiagonal elements in the

```

sparse rows of the factored  $A * A$  transpose. This includes fill-in.

NOTE: There are 68 operations of the form  $u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$  to factorize the sparse rows of  $A * A$  transpose.

| Iter | Complem_aff | Complem-ity | Duality_gap  | Tot_infeasb | Tot_infeasc  | Tot_infeasd |
|------|-------------|-------------|--------------|-------------|--------------|-------------|
| 0    | -1.000000   | 176072547   | 0.833846     | 52835       | 40217        | 46430       |
| 1    | 49323199    | 21763015    | 0.910646     | 2971.887954 | 2262.182150  | 2506.228441 |
| 2    | 4256602     | 1419262     | 0.526045     | 0           | 6.291145E-11 | 51.536868   |
| 3    | 371793      | 247491      | 0.159553     | 0           | 0            | 8.939364    |
| 4    | 119362      | 60394       | 0.043112     | 0           | 0            | 1.565229    |
| 5    | 27925       | 21537       | 0.015622     | 0           | 0            | 0.576834    |
| 6    | 10494       | 6617.424035 | 0.004839     | 0           | 0            | 0.124639    |
| 7    | 3367.128699 | 1357.529806 | 0.000996     | 0           | 0            | 0.001406    |
| 8    | 498.362201  | 156.077166  | 0.000115     | 0           | 0            | 0.000163    |
| 9    | 28.159576   | 1.000647    | 0.000000735  | 0           | 0            | 0           |
| 10   | 0.000549    | 0.000050153 | -8.10256E-11 | 0           | 0            | 0           |

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 10 iterations.

NOTE: Objective = -1295661.8.

NOTE: The data set WORK.ARC5 has 64 observations and 17 variables.

NOTE: There were 64 observations read from the data set WORK.NEW\_ARC4.

NOTE: There were 8 observations read from the data set WORK.NODE0.

NOTE: There were 21 observations read from the data set WORK.CON3.

NOTE: The data set WORK.ARC5 has 64 observations and 17 variables.

Output 5.4.1. CONOUT=ARC5

| _tail_         | _head_   | _cost_    | _capac_  | _lo_      | _SUPPLY_ | _DEMAND_ |
|----------------|----------|-----------|----------|-----------|----------|----------|
| fact1_1        | f1_apr_1 | 78.6      | 600      | 50        | 1000     | .        |
| f1_mar_1       | f1_apr_1 | 15.0      | 50       | 0         | .        | .        |
| f1_may_1       | f1_apr_1 | 33.6      | 20       | 0         | .        | .        |
| f2_apr_1       | f1_apr_1 | 11.0      | 40       | 0         | .        | .        |
| fact1_2        | f1_apr_2 | 174.5     | 550      | 50        | 1000     | .        |
| f1_mar_2       | f1_apr_2 | 20.0      | 40       | 0         | .        | .        |
| f1_may_2       | f1_apr_2 | 49.2      | 15       | 0         | .        | .        |
| f2_apr_2       | f1_apr_2 | 21.0      | 25       | 0         | .        | .        |
| fact1_1        | f1_mar_1 | 127.9     | 500      | 50        | 1000     | .        |
| f1_apr_1       | f1_mar_1 | 33.6      | 20       | 0         | .        | .        |
| f2_mar_1       | f1_mar_1 | 10.0      | 40       | 0         | .        | .        |
| fact1_2        | f1_mar_2 | 217.9     | 400      | 40        | 1000     | .        |
| f1_apr_2       | f1_mar_2 | 38.4      | 30       | 0         | .        | .        |
| f2_mar_2       | f1_mar_2 | 20.0      | 25       | 0         | .        | .        |
| fact1_1        | f1_may_1 | 90.1      | 400      | 50        | 1000     | .        |
| f1_apr_1       | f1_may_1 | 12.0      | 50       | 0         | .        | .        |
| f2_may_1       | f1_may_1 | 13.0      | 40       | 0         | .        | .        |
| fact1_2        | f1_may_2 | 113.3     | 350      | 40        | 1000     | .        |
| f1_apr_2       | f1_may_2 | 18.0      | 40       | 0         | .        | .        |
| f2_may_2       | f1_may_2 | 13.0      | 25       | 0         | .        | .        |
| f1_apr_1       | f2_apr_1 | 11.0      | 99999999 | 0         | .        | .        |
| fact2_1        | f2_apr_1 | 62.4      | 480      | 35        | 850      | .        |
| f2_mar_1       | f2_apr_1 | 18.0      | 30       | 0         | .        | .        |
| f2_may_1       | f2_apr_1 | 30.0      | 15       | 0         | .        | .        |
| f1_apr_2       | f2_apr_2 | 23.0      | 99999999 | 0         | .        | .        |
| fact2_2        | f2_apr_2 | 196.7     | 680      | 35        | 1500     | .        |
| f2_mar_2       | f2_apr_2 | 28.0      | 50       | 0         | .        | .        |
|                |          |           |          |           |          |          |
| _name_         | _FLOW_   | _FCOST_   | oldflow  | oldfc     |          |          |
| prod f1 19 apl | 533.333  | 41920.00  | 533.333  | 41920.00  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                |          |           |          |           |          |          |
| back f1 19 may | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                |          |           |          |           |          |          |
| prod f1 25 apl | 250.000  | 43625.00  | 250.000  | 43625.00  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                |          |           |          |           |          |          |
| back f1 25 may | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                |          |           |          |           |          |          |
| prod f1 19 mar | 333.333  | 42633.33  | 333.333  | 42633.33  |          |          |
|                | 20.000   | 672.00    | 20.000   | 672.00    |          |          |
|                | 40.000   | 400.00    | 40.000   | 400.00    |          |          |
| prod f1 25 mar | 400.000  | 87160.00  | 400.000  | 87160.00  |          |          |
|                | 30.000   | 1152.00   | 30.000   | 1152.00   |          |          |
|                | 25.000   | 500.00    | 25.000   | 500.00    |          |          |
| back f1 25 apl | 128.333  | 11562.83  | 128.333  | 11562.83  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 350.000  | 39655.00  | 350.000  | 39655.00  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 13.333   | 146.67    | 13.333   | 146.67    |          |          |
|                | 480.000  | 29952.00  | 480.000  | 29952.00  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| prod f2 19 apl | 550.000  | 108185.00 | 577.500  | 113594.25 |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                |          |           |          |           |          |          |
|                |          |           |          |           |          |          |
| _tail_         | _head_   | _cost_    | _capac_  | _lo_      | _SUPPLY_ | _DEMAND_ |
| f2_may_2       | f2_apr_2 | 64.8      | 15       | 0         | .        | .        |
| f1_mar_1       | f2_mar_1 | 11.0      | 99999999 | 0         | .        | .        |
| fact2_1        | f2_mar_1 | 88.0      | 450      | 35        | 850      | .        |
| f2_apr_1       | f2_mar_1 | 20.4      | 15       | 0         | .        | .        |
| f1_mar_2       | f2_mar_2 | 23.0      | 99999999 | 0         | .        | .        |
| fact2_2        | f2_mar_2 | 182.0     | 650      | 35        | 1500     | .        |
| f2_apr_2       | f2_mar_2 | 37.2      | 15       | 0         | .        | .        |
|                |          |           |          |           |          |          |
| _name_         | _FLOW_   | _FCOST_   | oldflow  | oldfc     |          |          |
| back f2 25 may | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| prod f2 19 mar | 290.000  | 25520.00  | 290.000  | 25520.00  |          |          |
| back f2 19 apl | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| prod f2 25 mar | 650.000  | 118300.00 | 650.000  | 118300.00 |          |          |
| back f2 25 apl | 0.000    | 0.00      | 0.000    | 0.00      |          |          |

| _tail_   | _head_   | _cost_     | _capac_  | _lo_       | _SUPPLY_ | _DEMAND_ |
|----------|----------|------------|----------|------------|----------|----------|
| f1_may_1 | f2_may_1 | 16.00      | 99999999 | 0          | .        | .        |
| fact2_1  | f2_may_1 | 128.80     | 250      | 35         | 850      | .        |
| f2_apr_1 | f2_may_1 | 20.00      | 30       | 0          | .        | .        |
| f1_may_2 | f2_may_2 | 26.00      | 99999999 | 0          | .        | .        |
| fact2_2  | f2_may_2 | 181.40     | 550      | 35         | 1500     | .        |
| f2_apr_2 | f2_may_2 | 38.00      | 50       | 0          | .        | .        |
| f1_mar_1 | shop1_1  | -327.65    | 250      | 0          | .        | 900      |
| f1_apr_1 | shop1_1  | -300.00    | 250      | 0          | .        | 900      |
| f1_may_1 | shop1_1  | -285.00    | 250      | 0          | .        | 900      |
| f2_mar_1 | shop1_1  | -297.40    | 250      | 0          | .        | 900      |
| f2_apr_1 | shop1_1  | -290.00    | 250      | 0          | .        | 900      |
| f2_may_1 | shop1_1  | -292.00    | 250      | 0          | .        | 900      |
| f1_mar_2 | shop1_2  | -559.76    | 99999999 | 0          | .        | 900      |
| f1_apr_2 | shop1_2  | -524.28    | 99999999 | 0          | .        | 900      |
| f1_may_2 | shop1_2  | -515.02    | 99999999 | 0          | .        | 900      |
| f2_mar_2 | shop1_2  | -567.83    | 500      | 0          | .        | 900      |
| f2_apr_2 | shop1_2  | -542.19    | 500      | 0          | .        | 900      |
| f2_may_2 | shop1_2  | -491.56    | 500      | 0          | .        | 900      |
| f1_mar_1 | shop2_1  | -362.74    | 250      | 0          | .        | 900      |
| f1_apr_1 | shop2_1  | -300.00    | 250      | 0          | .        | 900      |
| f1_may_1 | shop2_1  | -245.00    | 250      | 0          | .        | 900      |
| f2_mar_1 | shop2_1  | -272.70    | 250      | 0          | .        | 900      |
| f2_apr_1 | shop2_1  | -312.00    | 250      | 0          | .        | 900      |
| f2_may_1 | shop2_1  | -299.00    | 250      | 0          | .        | 900      |
| f1_mar_2 | shop2_2  | -623.89    | 99999999 | 0          | .        | 1450     |
| f1_apr_2 | shop2_2  | -549.68    | 99999999 | 0          | .        | 1450     |
| f1_may_2 | shop2_2  | -500.00    | 99999999 | 0          | .        | 1450     |
| _name_   | _FLOW_   | _FCOST_    | oldflow  | oldfc      |          |          |
|          | 115.000  | 1840.00    | 115.000  | 1840.00    |          |          |
|          | 35.000   | 4508.00    | 35.000   | 4508.00    |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 0.000    | 0.00       | 350.000  | 9100.00    |          |          |
|          | 150.000  | 27210.00   | 122.500  | 22221.50   |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 143.333  | -46963.17  | 143.333  | -46963.17  |          |          |
|          | 250.000  | -75000.00  | 250.000  | -75000.00  |          |          |
|          | 13.333   | -3800.00   | 13.333   | -3800.00   |          |          |
|          | 250.000  | -74350.00  | 250.000  | -74350.00  |          |          |
|          | 243.333  | -70566.67  | 243.333  | -70566.67  |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 350.000  | -180257.00 | 0.000    | 0.00       |          |          |
|          | 500.000  | -283915.00 | 500.000  | -283915.00 |          |          |
|          | 50.000   | -27109.50  | 400.000  | -216876.00 |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 250.000  | -90685.00  | 250.000  | -90685.00  |          |          |
|          | 250.000  | -75000.00  | 250.000  | -75000.00  |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 250.000  | -78000.00  | 250.000  | -78000.00  |          |          |
|          | 150.000  | -44850.00  | 150.000  | -44850.00  |          |          |
|          | 455.000  | -283869.95 | 455.000  | -283869.95 |          |          |
|          | 220.000  | -120929.60 | 220.000  | -120929.60 |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
| _tail_   | _head_   | _cost_     | _capac_  | _lo_       | _SUPPLY_ | _DEMAND_ |
| f2_mar_2 | shop2_2  | -542.83    | 500      | 0          | .        | 1450     |
| f2_apr_2 | shop2_2  | -559.19    | 500      | 0          | .        | 1450     |
| f2_may_2 | shop2_2  | -519.06    | 500      | 0          | .        | 1450     |
| _name_   | _FLOW_   | _FCOST_    | oldflow  | oldfc      |          |          |
|          | 125.000  | -67853.75  | 125.000  | -67853.75  |          |          |
|          | 500.000  | -279595.00 | 177.500  | -99256.23  |          |          |
|          | 150.000  | -77859.00  | 472.500  | -245255.85 |          |          |

## Example 5.5. Nonarc Variables in the Side Constraints

You can verify that the FACT2 MAR GIZMO constraint has a left-hand-side activity of 3,470, which is not equal to the `_RHS_` of this constraint. Not all of the 3,750 chips that can be supplied to factory 2 for March production are used. It is suggested that all the possible chips be obtained in March and those not used be saved for April production. Because chips must be kept in an air-controlled environment, it costs one dollar to store each chip purchased in March until April. The maximum number of chips that can be stored in this environment at each factory is 150. In addition, a search of the parts inventory at factory 1 turned up 15 chips available for their March production.

Nonarc variables are used in the side constraints that handle the limitations of supply of Gizmo chips. A nonarc variable called `f1 unused mar` has as a value the number of chips that are not used at factory 1 in March. Another nonarc variable, `f2 unused mar`, has as a value the number of chips that are not used at factory 2 in March. `f1 chips from mar` has as a value the number of chips left over from March used for production at factory 1 in April. Similarly, `f2 chips from mar` has as a value the number of chips left over from March used for April production at factory 2 in April. The last two nonarc variables have objective function coefficients of 1 and upper bounds of 150. The Gizmo side constraints are

```
3*prod f1 19 mar + 4*prod f1 25 mar + f1 unused chips = 2615
3*prod f2 19 apl + 4*prod f2 25 apl + f2 unused chips = 3750
3*prod f1 19 apl + 4*prod f1 25 apl - f1 chips from mar = 2600
3*prod f2 19 apl + 4*prod f2 25 apl - f2 chips from mar = 3750
f1 unused chips + f2 unused chips -
f1 chips from mar - f2 chips from mar >= 0
```

The last side constraint states that the number of chips not used in March is not less than the number of chips left over from March and used in April. Here, this constraint is called `CHIP LEFTOVER`.

The following SAS code creates a new data set containing constraint data. It seems that most of the constraints are now equalities, so you specify `DEFCONTYPE=EQ` in the `PROC INTPOINT` statements from now on and provide constraint type data for constraints that are not “equal to” type, using the default `TYPEOBS` value `_TYPE_` as the `_COLUMN_` variable value to indicate observations that contain constraint type data. Also, from now on, the default `RHSOBS` value is used:

```

title2 'Nonarc Variables in the Side Constraints';
data con6;
  input _column_ &$17. _row_ &$15. _coef_ ;
  datalines;
prod f1 19 mar      FACT1 MAR GIZMO  3
prod f1 25 mar      FACT1 MAR GIZMO  4
f1 unused chips    FACT1 MAR GIZMO  1
_RHS_              FACT1 MAR GIZMO 2615
prod f2 19 mar      FACT2 MAR GIZMO  3
prod f2 25 mar      FACT2 MAR GIZMO  4
f2 unused chips    FACT2 MAR GIZMO  1
_RHS_              FACT2 MAR GIZMO 3750
prod f1 19 apl      FACT1 APL GIZMO  3
prod f1 25 apl      FACT1 APL GIZMO  4
f1 chips from mar  FACT1 APL GIZMO -1
_RHS_              FACT1 APL GIZMO 2600
prod f2 19 apl      FACT2 APL GIZMO  3
prod f2 25 apl      FACT2 APL GIZMO  4
f2 chips from mar  FACT2 APL GIZMO -1
_RHS_              FACT2 APL GIZMO 3750
f1 unused chips    CHIP LEFTOVER  1
f2 unused chips    CHIP LEFTOVER  1
f1 chips from mar  CHIP LEFTOVER -1
f2 chips from mar  CHIP LEFTOVER -1
_TYPE_            CHIP LEFTOVER  1
back f1 19 apl      TOTAL BACKORDER 1
back f1 25 apl      TOTAL BACKORDER 1
back f2 19 apl      TOTAL BACKORDER 1
back f2 25 apl      TOTAL BACKORDER 1
back f1 19 may      TOTAL BACKORDER 1
back f1 25 may      TOTAL BACKORDER 1
back f2 19 may      TOTAL BACKORDER 1
back f2 25 may      TOTAL BACKORDER 1
_TYPE_            TOTAL BACKORDER -1
_RHS_            TOTAL BACKORDER 50
;

```

The nonarc variables f1 chips from mar and f2 chips from mar have objective function coefficients of 1 and upper bounds of 150. There are various ways in which this information can be furnished to PROC INTPOINT. If there were a TYPE list variable in the CONDATA= data set, observations could be in the form:

| <u>_COLUMN_</u>   | <u>_TYPE_</u> | <u>_ROW_</u> | <u>_COEF_</u> |
|-------------------|---------------|--------------|---------------|
| f1 chips from mar | objfn         | .            | 1             |
| f1 chips from mar | upperbd       | .            | 150           |
| f2 chips from mar | objfn         | .            | 1             |
| f2 chips from mar | upperbd       | .            | 150           |

It is desirable to assign ID list variable values to all the nonarc variables:

```
data arc6;
  set arc5;
  drop oldcost oldfc oldflow _flow_ _fcost_ _status_ _rcost_;
data arc6_b;
  input _name_ &$17. _cost_ _capac_ factory key_id $ ;
  datalines;
f1 unused chips      .      . 1 chips
f2 unused chips      .      . 2 chips
f1 chips from mar    1 150 1 chips
f2 chips from mar    1 150 2 chips
;
proc append force
  base=arc6 data=arc6_b;
proc intpoint
  bytes=1000000
  printlevel2=2
  nodedata=node0 arcddata=arc6
  condata=con6 defcontype=eq sparsesecondata
  conout=arc7;
run;
```

The following messages appear on the SAS log:

```
NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 4 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 4350 , total demand= 4150 .
NOTE: Number of arcs= 64 .
NOTE: Number of nonarc variables= 4 .
NOTE: Number of <= side constraints= 1 .
NOTE: Number of == side constraints= 4 .
NOTE: Number of >= side constraints= 1 .
NOTE: Number of side constraint coefficients= 24 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 1 .
NOTE: Number of == constraints= 25 .
NOTE: Number of >= constraints= 1 .
NOTE: Number of constraint coefficients= 160 .
NOTE: Number of variables= 72 .
NOTE: 2 columns, 0 rows and 2 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 86 nonzero elements in A * A transpose.
NOTE: Of the 27 rows and columns, 16 are sparse.
NOTE: There are 102 nonzero superdiagonal elements in the
      sparse rows of the factored A * A transpose. This
      includes fill-in.
NOTE: There are 160 operations of the form
      u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q] to factorize the
      sparse rows of A * A transpose.
```

| Iter | Complem_aff | Complem-ity | Duality_gap | Tot_infeasb | Tot_infeasc | Tot_infeasd |
|------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0    | -1.000000   | 180211988   | 0.837584    | 55030       | 38331       | 44219       |



```

1      54358286      27998179      0.909216 4949.974964 3447.853782 6457.752523
2      9372834      2493303      0.657738      0 8.594938E-10 196.155157
3      360076      311567      0.192309      0      0 24.426434
4      133734      91250      0.064112      0      0 5.827344
5      66155      36320      0.026230      0      0 0
6      18053 8903.419999      0.006517      0      0 0
7 3897.577387 1910.615008      0.001402      0      0 0
8 847.313077 362.303033      0.000266      0      0 0
9 146.082127 45.122412 0.000033136      0      0 0
10 6.217057 0.574649 0.000000422      0      0 0
11 0.002810 0.000029061 2.152507E-11      0      0 0

```

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 11 iterations.

NOTE: Objective = -129542.742.

NOTE: The data set WORK.ARC7 has 68 observations and 14 variables.

NOTE: There were 68 observations read from the data set WORK.ARC6.

NOTE: There were 8 observations read from the data set WORK.NODE0.

NOTE: There were 31 observations read from the data set WORK.CON6.

NOTE: The data set WORK.ARC7 has 68 observations and 14 variables.

The optimal solution data set, CONOUT=ARC7 in Output 5.5.1 follow:

```

proc print data=arc7;
  var _tail_ _head_ _name_ _cost_ _capac_ _lo_
    _flow_ _fcost_;
  sum _fcost_;
run;

```

**Output 5.5.1.** CONOUT=ARC7

| Nonarc Variables in the Side Constraints |          |          |                |         |          |      |         |           |  |
|------------------------------------------|----------|----------|----------------|---------|----------|------|---------|-----------|--|
| Obs                                      | _tail_   | _head_   | _name_         | _cost_  | _capac_  | _lo_ | _FLOW_  | _FCOST_   |  |
| 1                                        | fact1_1  | f1_apr_1 | prod f1 19 apl | 78.60   | 600      | 50   | 540.000 | 42444.00  |  |
| 2                                        | f1_mar_1 | f1_apr_1 |                | 15.00   | 50       | 0    | 0.000   | 0.00      |  |
| 3                                        | f1_may_1 | f1_apr_1 | back f1 19 may | 33.60   | 20       | 0    | 0.000   | 0.00      |  |
| 4                                        | f2_apr_1 | f1_apr_1 |                | 11.00   | 40       | 0    | 0.000   | 0.00      |  |
| 5                                        | fact1_2  | f1_apr_2 | prod f1 25 apl | 174.50  | 550      | 50   | 250.000 | 43625.00  |  |
| 6                                        | f1_mar_2 | f1_apr_2 |                | 20.00   | 40       | 0    | 0.000   | 0.00      |  |
| 7                                        | f1_may_2 | f1_apr_2 | back f1 25 may | 49.20   | 15       | 0    | 0.000   | 0.00      |  |
| 8                                        | f2_apr_2 | f1_apr_2 |                | 21.00   | 25       | 0    | 25.000  | 525.00    |  |
| 9                                        | fact1_1  | f1_mar_1 | prod f1 19 mar | 127.90  | 500      | 50   | 338.333 | 43272.83  |  |
| 10                                       | f1_apr_1 | f1_mar_1 | back f1 19 apl | 33.60   | 20       | 0    | 20.000  | 672.00    |  |
| 11                                       | f2_mar_1 | f1_mar_1 |                | 10.00   | 40       | 0    | 40.000  | 400.00    |  |
| 12                                       | fact1_2  | f1_mar_2 | prod f1 25 mar | 217.90  | 400      | 40   | 400.000 | 87160.00  |  |
| 13                                       | f1_apr_2 | f1_mar_2 | back f1 25 apl | 38.40   | 30       | 0    | 30.000  | 1152.00   |  |
| 14                                       | f2_mar_2 | f1_mar_2 |                | 20.00   | 25       | 0    | 25.000  | 500.00    |  |
| 15                                       | fact1_1  | f1_may_1 |                | 90.10   | 400      | 50   | 116.667 | 10511.67  |  |
| 16                                       | f1_apr_1 | f1_may_1 |                | 12.00   | 50       | 0    | 0.000   | 0.00      |  |
| 17                                       | f2_may_1 | f1_may_1 |                | 13.00   | 40       | 0    | 0.000   | 0.00      |  |
| 18                                       | fact1_2  | f1_may_2 |                | 113.30  | 350      | 40   | 350.000 | 39655.00  |  |
| 19                                       | f1_apr_2 | f1_may_2 |                | 18.00   | 40       | 0    | 0.000   | 0.00      |  |
| 20                                       | f2_may_2 | f1_may_2 |                | 13.00   | 25       | 0    | 0.000   | 0.00      |  |
| 21                                       | f1_apr_1 | f2_apr_1 |                | 11.00   | 99999999 | 0    | 20.000  | 220.00    |  |
| 22                                       | fact2_1  | f2_apr_1 | prod f2 19 apl | 62.40   | 480      | 35   | 480.000 | 29952.00  |  |
| 23                                       | f2_mar_1 | f2_apr_1 |                | 18.00   | 30       | 0    | 0.000   | 0.00      |  |
| 24                                       | f2_may_1 | f2_apr_1 | back f2 19 may | 30.00   | 15       | 0    | 0.000   | 0.00      |  |
| 25                                       | f1_apr_2 | f2_apr_2 |                | 23.00   | 99999999 | 0    | 0.000   | 0.00      |  |
| 26                                       | fact2_2  | f2_apr_2 | prod f2 25 apl | 196.70  | 680      | 35   | 577.500 | 113594.25 |  |
| 27                                       | f2_mar_2 | f2_apr_2 |                | 28.00   | 50       | 0    | 0.000   | 0.00      |  |
| 28                                       | f2_may_2 | f2_apr_2 | back f2 25 may | 64.80   | 15       | 0    | 0.000   | 0.00      |  |
| 29                                       | f1_mar_1 | f2_mar_1 |                | 11.00   | 99999999 | 0    | 0.000   | 0.00      |  |
| 30                                       | fact2_1  | f2_mar_1 | prod f2 19 mar | 88.00   | 450      | 35   | 290.000 | 25520.00  |  |
| 31                                       | f2_apr_1 | f2_mar_1 | back f2 19 apl | 20.40   | 15       | 0    | 0.000   | 0.00      |  |
| 32                                       | f1_mar_2 | f2_mar_2 |                | 23.00   | 99999999 | 0    | 0.000   | 0.00      |  |
| 33                                       | fact2_2  | f2_mar_2 | prod f2 25 mar | 182.00  | 650      | 35   | 650.000 | 118300.00 |  |
| 34                                       | f2_apr_2 | f2_mar_2 | back f2 25 apl | 37.20   | 15       | 0    | 0.000   | 0.00      |  |
| 35                                       | f1_may_1 | f2_may_1 |                | 16.00   | 99999999 | 0    | 115.000 | 1840.00   |  |
| 36                                       | fact2_1  | f2_may_1 |                | 128.80  | 250      | 35   | 35.000  | 4508.00   |  |
| 37                                       | f2_apr_1 | f2_may_1 |                | 20.00   | 30       | 0    | 0.000   | 0.00      |  |
| 38                                       | f1_may_2 | f2_may_2 |                | 26.00   | 99999999 | 0    | 0.000   | 0.00      |  |
| 39                                       | fact2_2  | f2_may_2 |                | 181.40  | 550      | 35   | 122.500 | 22221.50  |  |
| 40                                       | f2_apr_2 | f2_may_2 |                | 38.00   | 50       | 0    | 0.000   | 0.00      |  |
| 41                                       | f1_mar_1 | shop1_1  |                | -327.65 | 250      | 0    | 148.333 | -48601.42 |  |
| 42                                       | f1_apr_1 | shop1_1  |                | -300.00 | 250      | 0    | 250.000 | -75000.00 |  |
| 43                                       | f1_may_1 | shop1_1  |                | -285.00 | 250      | 0    | 1.667   | -475.00   |  |
| 44                                       | f2_mar_1 | shop1_1  |                | -297.40 | 250      | 0    | 250.000 | -74350.00 |  |
| 45                                       | f2_apr_1 | shop1_1  |                | -290.00 | 250      | 0    | 250.000 | -72500.00 |  |

The optimal value of the nonarc variable f2 unused chips is 280. This means that although there are 3,750 chips that can be used at factory 2 in March, only 3,470 are used. As the optimal value of f1 unused chips is zero, all chips available for production in March at factory 1 are used. The nonarc variable f2 chips from mar also has zero optimal value. This means that the April production at factory 2 does not need any chips that could have been held in inventory since March. However, the nonarc variable f1 chips from mar has value of 20. Thus, 3,490 chips should be ordered for factory 2 in March. Twenty of these chips should be held in inventory until April, then sent to factory 1.

### Example 5.6. Solving an LP Problem with Data in MPS Format

In this example, PROC INTPOINT is ultimately used to solve an LP. But prior to that, there is SAS code that is used to read a MPS format file and initialize an input SAS data set. MPS was an optimization package developed for IBM computers many years ago and the format by which data had to be supplied to that system became the industry standard for other optimization software packages, including those developed recently. The MPS format is described in Murtagh (1981). If you have an LP with has data in MPS format in a file /your-directorys/your-filename.dat, then the following SAS code should be run:

```
filename w '/your-directorys/your-filename.dat';
data raw;
  infile w lrecl=80 pad;
  input field1 $ 2-3 field2 $ 5-12 field3 $ 15-22
        field4 25-36 field5 $ 40-47 field6 50-61;
run;
%sasmpsxs;
data lp;
  set;
  if _type_="FREE" then _type_="MIN";
  if lag(_type_)="*HS" then _type_="RHS";
run;
proc sort data=lp;by _col_;run;

proc intpoint
  arcdata=lp
  condata=lp sparseseconddata rhsobs=rhs grouped=condata
  conout=solutn /* SAS data set for the optimal solution */
  bytes=20000000
  nnas=1700 ncoefs=4000 ncons=700
  printlevel2=2 memrep;
run;

proc lp
  data=lp sparsedata
  endpause time=3600 maxit1=100000 maxit2=100000;
run;
show status;
quit;
```

You will have to specify the appropriate path and file name in which your MPS format data resides.

`%sasmpsxs;` is a SAS macro provided within SAS/OR software. The MPS format resembles the sparse format of the `CONDATA=` data set for `PROC INTPOINT`. The SAS macro `%sasmpsxs;` examines the MPS data and transfers it into a SAS data set while automatically taking into account how the MPS format differs slightly from `PROC INTPOINT`'s sparse format.

The parameters `NNAS==1700`, `NCOEFS=4000`, and `NCONS=700` indicate the approximate (over-estimated) number of variables, coefficients and constraints this model has. You must change these to your problems dimensions. Knowing these, `PROC INTPOINT` is able to utilize memory better and read the data faster. These parameters are optional.

The `PROC SORT` preceeding `PROC INTPOINT` is not necessary, but sorting the SAS data set can speed up `PROC INTPOINT` when it reads the data. After the sort, data for each column is grouped together. `GROUPED=condata` can be specified.

For small problems, presorting and specifying those additional options is not going to greatly influence `PROC INTPOINT`'s run time. However, when problems are large, presorting and specifying those additional options can be very worthwhile.

If you generate the model yourself, you will be familiar enough with it to know what to specify for the `RHSOBS=` parameter. If the value of the SAS variable in the `COLUMN` list is equal to the character string specified as the `RHSOBS=` option, the data in that observation is interpreted as right-hand-side data as opposed to coefficient data. If you do not know what to specify for the `RHSOBS=rhs-charstr`, you should first run `PROC LP` and optionally set `maxit1=1` and `maxit2=1`. `PROC LP` will output a Problem Summary that includes the line

```
Rhs Variable      rhs-charstr.
```

`BYTES=20000000` is the size of working memory `PROC INTPOINT` is allowed.

The options `PRINTLEVEL2=2` and `MEMREP` indicate that you want to see an iteration log and messages about memory usage. Specifying these options are optional.

---

## References

- Lustig, I.J., Marsten, R.E., and Shanno, D.F. (1992), “On Implementing Methrotra’s Predictor-Corrector Interior-Point Method for Linear Programming,” *SIAM Journal of Optimization* 2 (3), 435-449
- Murtagh, B.A. (1981), “Advanced Linear Programming,” McGraw-Hill International Book Company.
- Reid, J.K. (1975), “A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases,” *Harwell Report CSS 20*, Atomic Energy Research Establishment, Didcot, Oxfordshire, England.
- Roos, C., Terlaky, T., and Vial, J.-Ph. (1997), “Theory and Algorithms for Linear Optimization,” Chichester, England: John Wiley & Sons.
- Wright, S.J. (1996), “Primal-Dual Interior Point Algorithms,” Philadelphia: SIAM.
- Ye, Y. (1996), “Interior Point Algorithms: Theory and Analysis,” Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons.

# Chapter 6

## The NETDRAW Procedure

### Chapter Table of Contents

---

|                            |     |
|----------------------------|-----|
| <b>OVERVIEW</b> . . . . .  | 191 |
| <b>SYNTAX</b> . . . . .    | 191 |
| ACTNET Statement . . . . . | 191 |



# Chapter 6

## The NETDRAW Procedure

---

### Overview

Two new options have been added to the NETDRAW procedure:

- the option to number pages of a network diagram
- the option to display hierarchical networks in a top-down format

---

### Syntax

---

#### ACTNET Statement

The following new options are available in the ACTNET statement after a slash (/). Both these options are valid only for graphics quality network diagrams.

##### **PAGENUMBER**

##### **PAGENUM**

numbers the pages of the network diagram on the top right-hand corner of the page if the diagram exceeds one page. The numbering scheme is from left to right, top to bottom (unless the REVERSEY option is specified).

##### **ROTATE**

rotates the network diagram to change the orientation of the network to be from top to bottom instead of from left to right. For example, you can use this option to draw a Bill of Materials diagram that is traditionally drawn from top to bottom with the Final Product drawn at the top of the tree. Note that this option is similar to the graphics option, ROTATE (GOPTIONS ROTATE). However, some device drivers ignore the global graphics option, ROTATE (for example, the SASGDDMX driver). Use the ROTATE option on the ACTNET statement for such device drivers. In addition, use the ROTATETEXT option to rotate the text within each node.





# Chapter 7

## The NETFLOW Procedure

### Chapter Table of Contents

---

|                                                               |     |
|---------------------------------------------------------------|-----|
| <b>OVERVIEW</b> . . . . .                                     | 195 |
| <b>SYNTAX</b> . . . . .                                       | 195 |
| Interior Point Algorithm Options: Stopping Criteria . . . . . | 196 |
| <b>DETAILS</b> . . . . .                                      | 199 |
| Balancing Total Supply and Total Demand . . . . .             | 199 |
| Interior Point Algorithmic Details . . . . .                  | 201 |
| <b>REFERENCES</b> . . . . .                                   | 210 |



# Chapter 7

## The NETFLOW Procedure

---

### Overview

Enhancements to the NETFLOW procedure include the following:

- The TOLTOTDINF and TOLTOTPINF options have been added to the NETFLOW and RESET statements for controlling when the Interior Point algorithm considers that the total dual and primal infeasibilities have been reduced to an amount close enough to zero so that from then on infeasibilities are no longer used in the computations.
- The PRINTLEVEL2 option has been added to the NETFLOW and RESET statements to indicate that information regarding Interior Point optimization convergence is output to the SAS log.
- There are several new options that allow you to specify when Interior Point optimization should stop, and when optimization should continue when the usual stopping criteria are met. See the “Stopping Criteria” section on page 206 for details.
- When solving a network problem (by either the Simplex or Interior Point algorithms) and the network’s total supply differs from the total demand, PROC NETFLOW attaches an \_EXCESS\_ node to the network to either supply or demand the difference. The way the \_EXCESS\_ node is attached is now fully documented. See the “Balancing Total Supply and Total Demand” section on page 199 for details.
- More details about the Interior Point algorithm used by the NETFLOW procedure are provided in the “Interior Point Algorithmic Details” section on page 201.

---

### Syntax

The following new or updated options are available in the NETFLOW and RESET statements:

**TOLTOTDINF=*t***

**RTOLTOTDINF=*t***

specifies the allowed total amount of dual infeasibility. In the “Interior Point Algorithmic Details” section on page 201, the vector  $infeas_d$  is defined. If  $\sum_{i=1}^n infeas_{di} \leq t$ , the solution is deemed feasible.  $infeas_d$  is replaced by a zero vector, which makes computations faster. This option is the dual equivalent to the TOLTOTPINF= option. Valid values for  $t$  are between  $1.0E - 12$  and  $1.0E - 1$ . The default is  $1.0E - 7$ .

**TOLTOTPINF=*t*****RTOLTOTPINF=*t***

specifies the allowed total amount of primal infeasibility. This option is the dual equivalent to the TOLTOTDINF= option. In the “Interior Point: Upper Bounds” section on page 209, the vector  $infeas_b$  is defined. In the “Interior Point Algorithmic Details” section on page 201, the vector  $infeas_c$  is defined. If  $\sum_{i=1}^n infeas_{bi} \leq t$  and  $\sum_{i=1}^m infeas_{ci} \leq t$ , the solution is deemed feasible.  $infeas_b$  and  $infeas_c$  are replaced by a zero vector, which makes computations faster. Increasing the value of the TOLTOTPINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are between  $1.0E - 12$  and  $1.0E - 1$ . The default is  $1.0E - 7$ .

**PRINTLEVEL2=*p***

is used when you want to see PROC NETFLOW’s progress to the optimum. PROC NETFLOW will produce a table on the SAS log. A row of the table is generated during each iteration and may consist of values of

- the affine step complementarity
- the complementarity of the solution for the next iteration
- the total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 209)
- the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 201)
- the total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 201)

As optimization progresses, the values in all columns should converge to zero. If you specify PRINTLEVEL2=2, all columns will appear in the table. If PRINTLEVEL2=1 is specified, only the affine step complementarity, the complementarity of the solution for the next iteration, will appear. Some time is saved by not calculating the infeasibility values.

---

## Interior Point Algorithm Options: Stopping Criteria

**STOP\_C=*s***

is used to determine whether optimization should stop. At the beginning of each iteration, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**STOP\_DG=*s***

is used to determine whether optimization should stop. At the beginning of each iteration, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**STOP\_IB=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total bound infeasibility  $\sum_{i=1}^n \text{infeas}_{bi}$  (see the  $\text{infeas}_b$  array in the “Interior Point: Upper Bounds” section on page 209; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**STOP\_IC=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total constraint infeasibility  $\sum_{i=1}^m \text{infeas}_{ci}$  (see the  $\text{infeas}_c$  array in the “Interior Point Algorithmic Details” section on page 201; this value appears in the Tot\_infeasc column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**STOP\_ID=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total dual infeasibility  $\sum_{i=1}^n \text{infeas}_{di}$  (see the  $\text{infeas}_d$  array in the “Interior Point Algorithmic Details” section on page 201; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_STOP\_C=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_STOP\_DG=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_STOP\_IB=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total bound infeasibility  $\sum_{i=1}^n \text{infeas}_{bi}$  (see the  $\text{infeas}_b$  array in the “Interior Point: Upper Bounds” section on page 209; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_STOP\_IC=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 201; this value appears in the Tot\_infeas column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_STOP\_ID=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 201; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 206.

**KEEPPGOING\_C=s**

is used to determine whether optimization should stop. If a stopping condition is met, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**KEEPPGOING\_DG=s**

is used to determine whether optimization should stop. If a stopping condition is met, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**KEEPPGOING\_IB=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 209; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**KEEPPGOING\_IC=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 201; this value appears in the Tot\_infeas column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**KEEPPGOING\_ID=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 201; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_KEEPPGOING\_C=s**

is used to determine whether optimization should stop. If a stopping condition is met, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_KEEPPGOING\_DG=s**

is used to determine whether optimization should stop. If a stopping condition is met, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_KEEPPGOING\_IB=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 209; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_KEEPPGOING\_IC=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 201; this value appears in the Tot\_infeasc column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

**AND\_KEEPPGOING\_ID=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 201; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 206.

---

## Details

---

### Balancing Total Supply and Total Demand

**When Total Supply Exceeds Total Demand**

When total supply of a network problem exceeds total demand, PROC NETFLOW can add an extra node (called the *excess node*) to the problem and set the demand at



that node equal to the difference between total supply and total demand. There are three ways that this excess node can be joined to the network. All three ways entail PROC NETFLOW generating a set of arcs (henceforth referred to as the *generated arcs*) that are directed toward the excess node. The total amount of flow in generated arcs equals the demand of the excess node. The generated arcs originate from one of three sets of nodes.

When you specify the THRUNET option, the set of nodes that generated arcs originate from are all demand nodes, even those demand nodes with unspecified demand capability. You indicate that a node has unspecified demand capability by using a missing D value instead of an actual value for demand data. The value specified as the demand of a demand node is in effect a lower bound of the number of flow units that node can actually demand. For missing D demand nodes, this lower bound is zero.

If you do not specify the THRUNET option, the way in which the excess node is joined to the network depends on whether there are demand nodes with unspecified demand capability (nodes with missing D demand) or not.

If there are missing D demand nodes, these nodes are the set of nodes that generated arcs originate from. The value specified as the demand of a demand node, if not missing D, is the number of flow units that node actually demands. For a missing D demand node, the actual demand of that node may be zero or greater.

If there are no missing D demand nodes, the set of nodes that generated arcs originate from are the set of supply nodes. The value specified as the supply of a supply node is in effect an upper bound of the number of flow units that node can actually supply. For missing S supply nodes this upper bound is zero, so missing S nodes when total supply exceeds total demand are transshipment nodes, nodes that neither supply nor demand flow.

### **When Total Supply Is Less Than Total Demand**

When total supply of a network problem is less than total demand, PROC NETFLOW can add an extra node (called the *excess node*) to the problem and set the supply at that node equal to the difference between total demand and total supply. There are three ways that this excess node can be joined to the network. All three ways entail PROC NETFLOW generating a set of arcs (henceforth referred to as the *generated arcs*) that originate from the excess node. The total amount of flow in generated arcs equals the supply of the excess node. The generated arcs are directed toward one of three sets of nodes.

When you specify the THRUNET option, the set of nodes that generated arcs are directed toward are all supply nodes, even those supply nodes with unspecified supply capability. You indicate that a node has unspecified supply capability by using a missing S value instead of an actual value for supply data. The value specified as the supply of a supply node is in effect a lower bound of the number of flow units that node can actually supply. For missing S supply nodes, this lower bound is zero.

If you do not specify the THRUNET option, the way in which the excess node is joined to the network depends on whether there are supply nodes with unspecified supply capability (nodes with missing S supply) or not.

If there are missing S supply nodes, these nodes are the set of nodes that generated arcs are directed toward. The value specified as the supply of a supply node, if not missing S, is the number of flow units that node actually supplies. For a missing S supply node, the actual supply of that node may be zero or greater.

If there are no missing S supply nodes, the set of nodes that generated arcs are directed toward are the set of demand nodes. The value specified as the demand of a demand node is in effect an upper bound of the number of flow units that node can actually demand. For missing D demand nodes, this upper bound is zero, so missing D nodes when total supply is less than total demand are transshipment nodes, nodes that neither supply nor demand flow.

---

## Interior Point Algorithmic Details

After preprocessing, the Linear Program to be solved is

$$\begin{array}{ll} \min & \{c^T x\} \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

This is the *primal* problem. The matrices of  $d$ ,  $z$ , and  $Q$  of NPSC (defined in the section “Mathematical Description of NPSC” in Chapter 5, “The INTPOINT Procedure,”) have been renamed  $c$ ,  $x$ , and  $A$ , respectively, as these symbols are by convention used more, the problem to be solved is different from the original because of preprocessing, and there has been a change of primal variable to transform the LP into one whose variables have zero lower bounds. To simplify the algebra here, assume that variables have infinite upper bounds, and constraints are equalities. (Interior Point algorithms do efficiently handle finite upper bounds, and it is easy to introduce primal slack variables to change inequalities into equalities.) The problem has  $n$  variables.  $i$  is a variable number.  $k$  is an iteration number, and if used as a subscript or superscript it denotes “of iteration  $k$ ”.

There exists an equivalent problem, the *dual* problem, stated as

$$\begin{array}{ll} \max & \{b^T y\} \\ \text{subject to} & A^T y + s = c \\ & s \geq 0 \\ \text{where} & y \text{ are dual variables, and } s \text{ are dual constraint slacks} \end{array}$$

What the Interior Point has to do is to solve the system of equations to satisfy the Karush-Kuhn-Tucker (KKT) conditions for optimality:

$$\begin{array}{l} Ax = b \\ A^T y + s = c \\ XSe = 0 \\ x \geq 0 \end{array}$$

$$\begin{aligned}
& s \geq 0 \\
\text{where } & S = \text{diag}(s), \text{ (that is, } S_{i,j} = s_i \text{ if } i = j, S_{i,j} = 0 \text{ otherwise)} \\
& X = \text{diag}(x), \text{ and} \\
& e_i = 1 \forall i
\end{aligned}$$

These are the conditions for feasibility, with the *complementarity* condition  $XSe = 0$  added.  $c^T x = b^T y$  must occur at the optimum. Complementarity forces the optimal objectives of the primal and dual to be equal,  $c^T x_{opt} = b^T y_{opt}$ , as

$$\begin{aligned}
0 &= x_{opt}^T s_{opt} = s_{opt}^T x_{opt} = (c - A^T y_{opt})^T x_{opt} = \\
& c^T x_{opt} - y_{opt}^T (Ax_{opt}) = c^T x_{opt} - b^T y_{opt}
\end{aligned}$$

therefore  $0 = c^T x_{opt} - b^T y_{opt}$

Before the optimum is reached, a solution  $(x, y, s)$  may not satisfy the KKT conditions:

- Primal constraints can be broken,  $infeas_c = b - Ax \neq 0$ .
- Dual constraints can be broken,  $infeas_d = c - A^T y - s \neq 0$ .
- Complementarity is unsatisfied,  $x^T s = c^T x - b^T y \neq 0$ . This is called the *duality gap*.

The Interior Point algorithm works by using Newton's method to find a direction to move  $(\Delta x^k, \Delta y^k, \Delta s^k)$  from the current solution  $(x^k, y^k, s^k)$  toward a better solution:

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

$\alpha$  is the *step length* and is assigned a value as large as possible and not so large that  $x_i^{k+1}$  or  $s_i^{k+1}$  is "too close" to zero. The direction in which to move is found using

$$\begin{aligned}
A\Delta x^k &= infeas_c \\
A^T \Delta y^k + \Delta s^k &= infeas_d \\
S^k \Delta x^k + X^k \Delta s^k &= -X^k S^k e
\end{aligned}$$

To greatly improve performance, the third equation is changed to

$$S^k \Delta x^k + X^k \Delta s^k = -X^k S^k e + \sigma_k \mu_k e$$

where

$$\mu_k = (x^k)^T s^k / n, \text{ the average complementarity, and}$$

$$0 \leq \sigma_k \leq 1$$

The effect now is to find a direction in which to move to reduce infeasibilities and to reduce the complementarity toward zero, but if any  $x_i^k s_i^k$  is too close to zero, it is “nudged out” to  $\mu$ , and any  $x_i^k s_i^k$  that is larger than  $\mu$  is “nudged into”  $\mu$ . A  $\sigma_k$  close to or equal to 0.0 biases a direction toward the optimum, and a value for  $\sigma_k$  close to or equal to 1.0 “centers” the direction toward a point where all pairwise products  $x_i^k s_i^k = \mu$ . Such points make up the *Central Path* in the interior. Although centering directions make little, if any, progress in reducing  $\mu$  and moving the solution closer to the optimum, substantial progress toward the optimum can usually be made in the next iteration.

The Central Path is crucial to why the Interior Point algorithm is so efficient. As  $\mu$  is decreased, this path “guides” the algorithm to the optimum through the interior of feasible space. Without centering, the algorithm would find a series of solutions near each other close to the boundary of feasible space. Step lengths along the direction would be small and many more iterations would probably be required to reach the optimum.

That in a nutshell is the Primal-Dual Interior Point algorithm. Varieties of the algorithm differ in the way  $\alpha$  and  $\sigma_k$  are chosen and the direction adjusted during each iteration. A wealth of information can be found in the texts by Roos, Terlaky, and Vial (1997), Wright (1996), and Ye (1996).

The calculation of the direction is the most time-consuming step of the Interior Point algorithm. Assume the  $k$ th iteration is being performed, so the subscript and superscript  $k$  can be dropped from the algebra:

$$A \Delta x = \text{infeas}_c$$

$$A^T \Delta y + \Delta s = \text{infeas}_d$$

$$S \Delta x + X \Delta s = -X S e + \sigma \mu e$$

Rearranging the second equation

$$\Delta s = \text{infeas}_d - A^T \Delta y$$

Rearranging the third equation

$$\Delta s = X^{-1}(-S \Delta x - X S e + \sigma \mu e)$$

$$\Delta s = -\Theta \Delta x - S e + X^{-1} \sigma \mu e$$

where

$$\Theta = S X^{-1}$$

Equating these two expressions for  $\Delta s$  and rearranging

$$\begin{aligned} -\Theta\Delta x - Se + X^{-1}\sigma\mu e &= \text{infeas}_d - A^T\Delta y \\ -\Theta\Delta x &= Se - X^{-1}\sigma\mu e + \text{infeas}_d - A^T\Delta y \\ \Delta x &= \Theta^{-1}(-Se + X^{-1}\sigma\mu e - \text{infeas}_d + A^T\Delta y) \\ \Delta x &= \rho + \Theta^{-1}A^T\Delta y \end{aligned}$$

where

$$\rho = \Theta^{-1}(-Se + X^{-1}\sigma\mu e - \text{infeas}_d)$$

Substituting into the first direction equation

$$\begin{aligned} A\Delta x &= \text{infeas}_c \\ A(\rho + \Theta^{-1}A^T\Delta y) &= \text{infeas}_c \\ A\Theta^{-1}A^T\Delta y &= \text{infeas}_c - A\rho \\ \Delta y &= (A\Theta^{-1}A^T)^{-1}(\text{infeas}_c - A\rho) \end{aligned}$$

$\Theta$ ,  $\rho$ ,  $\Delta y$ ,  $\Delta x$ , and  $\Delta s$  are calculated in that order. The hardest term is the factorization of the  $(A\Theta^{-1}A^T)$  matrix to determine  $\Delta y$ . Fortunately, although the *values* of  $(A\Theta^{-1}A^T)$  are different for each iteration, the *locations* of the nonzeros in this matrix remain fixed; the nonzero locations are the same as those in the matrix  $(AA^T)$ . This is because  $\Theta^{-1} = XS^{-1}$  is a diagonal matrix that has the effect of merely scaling the columns of  $(AA^T)$ .

The fact that the nonzeros in  $A\Theta^{-1}A^T$  have a constant pattern is exploited by all Interior Point algorithms and is a major reason for their excellent performance. Before iterations begin,  $AA^T$  is examined and its rows and columns are symmetrically permuted so that during Cholesky factorization, the number of *fillins* created is smaller. A list of arithmetic operations to perform the factorization is saved in concise computer data structures (working with memory locations rather than actual numerical values). This is called *symbolic factorization*. During iterations, when memory has been initialized with numerical values, the operations list is performed sequentially. Determining how the factorization should be performed again and again is unnecessary.

### The Primal-Dual Predictor-Corrector IntPoint Algorithm

The variant of the Interior Point algorithm implemented in PROC NETFLOW is a Primal-Dual Predictor-Corrector Interior Point algorithm. At first, Newton's method is used to find a direction  $(\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k)$  to move, but calculated as if  $\mu$  is zero, that is, as a step with no centering, known as an *affine* step:

$$\begin{aligned} A\Delta x_{aff}^k &= \text{infeas}_c \\ A^T\Delta y_{aff}^k + \Delta s_{aff}^k &= \text{infeas}_d \\ S^k\Delta x_{aff}^k + X^k\Delta s_{aff}^k &= -X^kS^ke \end{aligned}$$

$$(x_{aff}^k, y_{aff}^k, s_{aff}^k) = (x^k, y^k, s^k) + \alpha(\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k)$$

$\alpha$  is the *step length* as before.

Complementarity  $x^T s$  is calculated at  $(x_{aff}^k, y_{aff}^k, s_{aff}^k)$  and compared with the complementarity at the starting point  $(x^k, y^k, s^k)$ , and the success of the affine step is gauged. If the affine step was successful in reducing the complementarity by a substantial amount, the need for centering is not great, and  $\sigma_k$  in the following linear system is assigned a value close to zero. If, however, the affine step was unsuccessful, centering would be beneficial, and  $\sigma_k$  in the following linear system is assigned a value closer to 1.0. The value of  $\sigma_k$  is therefore adaptively altered depending on the progress made toward the optimum.

A second linear system is solved to determine a centering vector  $(\Delta x_c^k, \Delta y_c^k, \Delta s_c^k)$  from  $(x_{aff}^k, y_{aff}^k, s_{aff}^k)$ :

$$\begin{aligned} A\Delta x_c^k &= 0 \\ A^T\Delta y_c^k + \Delta s_c^k &= 0 \\ S^k\Delta x_c^k + X^k\Delta s_c^k &= -X_{aff}^k S_{aff}^k e + \sigma_k \mu_k e \end{aligned}$$

then

$$\begin{aligned} (\Delta x^k, \Delta y^k, \Delta s^k) &= (\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k) + (\Delta x_c^k, \Delta y_c^k, \Delta s_c^k) \\ (x^{k+1}, y^{k+1}, s^{k+1}) &= (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k) \end{aligned}$$

where, as before,  $\alpha$  is the *step length* assigned a value as large as possible but not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  is “too close” to zero.

Although the Predictor-Corrector variant entails solving two linear systems instead of one, fewer iterations are usually required to reach the optimum. The additional overhead of calculating the second linear system is small, as the factorization of the  $(A\Theta^{-1}A^T)$  matrix has already been performed to solve the first linear system.

The variant of the IntPoint algorithm implemented in PROC NETFLOW is a Primal-Dual Predictor-Corrector IntPoint algorithm. At first, Newton’s method is used to find a direction to move  $(\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k)$ , but calculated as if  $\mu$  is zero, that is, a step with no centering, known as an *affine step*:

$$\begin{aligned} A\Delta x_{aff}^k &= -infeas_c \\ A^T\Delta y_{aff}^k + \Delta s_{aff}^k &= -infeas_d \\ S^k\Delta x_{aff}^k + X^k\Delta s_{aff}^k &= -X^k S^k e \end{aligned}$$

$$(x_{aff}^k, y_{aff}^k, s_{aff}^k) = (x^k, y^k, s^k) + \alpha(\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k)$$

$\alpha$  is the *step length* as before.

Complementarity  $x^T s$  is calculated at  $(x_{aff}^k, y_{aff}^k, s_{aff}^k)$  and compared with the complementarity at the starting point  $(x^k, y^k, s^k)$ , and the success of the affine step is

gauged. If the affine step was successful in reducing the complementarity by a substantial amount, the need for centering is not great, and the value of  $\sigma_k$  in the following linear system is assigned a value close to zero. If, however, the affine step was unsuccessful, centering would be beneficial, and the value of  $\sigma_k$  in the following linear system is assigned a value closer to 1.0. The value of  $\sigma_k$  is therefore adaptively altered depending on the progress made toward the optimum.

A second linear system is solved to determine a centering vector  $(\Delta x_c^k, \Delta y_c^k, \Delta s_c^k)$  from  $(x_{aff}^k, y_{aff}^k, s_{aff}^k)$

$$\begin{aligned} A\Delta x_c^k &= 0 \\ A^T\Delta y_c^k + \Delta s_c^k &= 0 \\ S^k\Delta x_c^k + X^k\Delta s_c^k &= -X^kS^ke \\ S^k\Delta x_c^k + X^k\Delta s_c^k &= -X_{aff}^kS_{aff}^ke + \sigma_k\mu_k e \end{aligned}$$

then

$$\begin{aligned} (\Delta x^k, \Delta y^k, \Delta s^k) &= (\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k) + (\Delta x_c^k, \Delta y_c^k, \Delta s_c^k) \\ (x^{k+1}, y^{k+1}, s^{k+1}) &= (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k) \end{aligned}$$

where, as before,  $\alpha$  is the *step length* assigned a value as large as possible but not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  is “too close” to zero.

Although the Predictor-Corrector variant entails solving two linear systems instead of one, fewer iterations are usually required to reach the optimum. The additional overhead of calculating the second linear system is small, as the factorization of the  $(A\Theta^{-1}A^T)$  matrix has already been performed to solve the first linear system.

### Stopping Criteria

There are several reasons why PROC NETFLOW stops Interior Point optimization. Optimization stops when:

- the number of iteration equals MAXITERB= $m$
- the relative gap ( $dualitygap/(c^Tx)$ ) between the primal and dual objectives is smaller than the value of the PDGAPTOL= option, and both the primal and dual problems are feasible. Duality gap is defined in the “Interior Point Algorithmic Details” section on page 201.

PROC NETFLOW may stop optimization when it detects that the rate at which the complementarity or duality gap is being reduced is too slow, that is, there are consecutive iterations when the complementarity or duality gap has stopped getting smaller and the infeasibilities, if nonzero, have also stalled. Sometimes, this indicates the problem is infeasible.

The reasons to stop optimization outlined in the previous paragraph will be termed the *usual* stopping conditions in the following explanation.

However, when solving some problems, especially if the problems are large, the usual stopping criteria are inappropriate. PROC NETFLOW might stop prematurely. If it were allowed to perform additional optimization, a better solution would be found. On other occasions, PROC NETFLOW might do too much work. A sufficiently good solution might be reached several iterations before PROC NETFLOW eventually stops.

You can see PROC NETFLOW's progress to the optimum by specifying PRINTLEVEL=2. PROC NETFLOW will produce a table on the SAS log. A row of the table is generated during each iteration and consists of values of the affine step complementarity, the complementarity of the solution for the next iteration, the total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the "Interior Point: Upper Bounds" section on page 209), the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the "Interior Point Algorithmic Details" section on page 201), and the total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the "Interior Point Algorithmic Details" section on page 201). As optimization progresses, the values in all columns should converge to zero.

To tailor stopping criteria to your problem, you can use two sets of parameters: the STOP\_x and the KEEPGOING\_x parameters. The STOP\_x parameters (STOP\_C, STOP\_DG, STOP\_IB, STOP\_IC, and STOP\_ID) are used to test for some condition at the beginning of each iteration and if met, to stop immediately. The KEEPGOING\_x parameters (KEEPGOING\_C, KEEPGOING\_DG, KEEPGOING\_IB, KEEPGOING\_IC, and KEEPGOING\_ID) are used when PROC NETFLOW would ordinarily stop but does not if some conditions are not met.

For the sake of conciseness, a set of options might be referred to as the part of the option name they have in common followed by the suffix x. For example, STOP\_C, STOP\_DG, STOP\_IB, STOP\_IC, and STOP\_ID will collectively be referred to as STOP\_x.

At the beginning of each iteration, PROC NETFLOW will test whether complementarity is  $\leq$  STOP\_C (provided you specified a STOP\_C parameter) and if it is, PROC NETFLOW will stop. If the duality gap is  $\leq$  STOP\_DG (provided you specified a STOP\_DG parameter), PROC NETFLOW will stop immediately. This is true as well for the other STOP\_x parameters that are related to infeasibilities, STOP\_IB, STOP\_IC, and STOP\_ID.

For example, if you want PROC NETFLOW to stop optimizing for the usual stopping conditions, plus the additional condition, complementarity  $\leq 100$  or duality gap  $\leq 0.001$ , then use

```
proc netflow stop_c=100 stop_dg=0.001
```

If you want PROC NETFLOW to stop optimizing for the usual stopping conditions, plus the additional condition, complementarity  $\leq 1000$  and duality gap  $\leq 0.001$  and constraint infeasibility  $\leq 0.0001$ , then use

```
proc netflow  
  and_stop_c=1000 and_stop_dg=0.01 and_stop_ic=0.0001
```



Unlike the STOP\_x parameters that cause PROC NETFLOW to stop when any one of them is satisfied, the corresponding AND\_STOP\_x parameters (AND\_STOP\_C, AND\_STOP\_DG, AND\_STOP\_IB, AND\_STOP\_IC, and AND\_STOP\_ID) cause PROC NETFLOW to stop only if all (more precisely, all that are specified) options are satisfied. For example, if PROC NETFLOW should stop when

- complementarity  $\leq 100$  or duality gap  $\leq 0.001$  or
- complementarity  $\leq 1000$  and duality gap  $\leq 0.001$  and constraint infeasibility  $\leq 0.000$

then use

```
proc netflow
  stop_c=100 stop_dg=0.001
  and_stop_c=1000 and_stop_dg=0.01 and_stop_ic=0.0001
```

Just as the STOP\_x parameters have AND\_STOP\_x partners, the KEEPGOING\_x parameters have AND\_KEEPPGOING\_x partners. The role of the KEEPGOING\_x and AND\_KEEPPGOING\_x parameters is to prevent optimization from stopping too early, even though a usual stopping criteria is met.

When PROC NETFLOW detects that it should stop for a usual stopping condition,

- it will test whether complementarity is  $> \text{KEEPGOING\_C}$  (provided you specified a KEEPPGOING\_C parameter), and if it is, PROC NETFLOW will perform more optimization.
- Otherwise, PROC NETFLOW will then test whether the primal-dual gap is  $> \text{KEEPGOING\_DG}$  (provided you specified a KEEPPGOING\_DG parameter), and if it is, PROC NETFLOW will perform more optimization.
- Otherwise, PROC NETFLOW will then test whether the total bound infeasibility  $\sum_{i=1}^n \text{infeas}_{bi} > \text{KEEPGOING\_IB}$  (provided you specified a KEEPPGOING\_IB parameter), and if it is, PROC NETFLOW will perform more optimization.
- Otherwise, PROC NETFLOW will then test whether the total constraint infeasibility  $\sum_{i=1}^m \text{infeas}_{ci} > \text{KEEPGOING\_IC}$  (provided you specified a KEEPPGOING\_IC parameter), and if it is, PROC NETFLOW will perform more optimization.
- Otherwise, PROC NETFLOW will then test whether the total dual infeasibility  $\sum_{i=1}^n \text{infeas}_{di} > \text{KEEPGOING\_ID}$  (provided you specified a KEEPPGOING\_ID parameter), and if it is, PROC NETFLOW will perform more optimization.
- Otherwise it will test whether complementarity is  $> \text{AND\_KEEPPGOING\_C}$  (provided you specified a AND\_KEEPPGOING\_C parameter), and the primal-dual gap is  $> \text{AND\_KEEPPGOING\_DG}$  (provided you specified a AND\_KEEPPGOING\_DG parameter), and the total bound infeasibility  $\sum_{i=1}^n \text{infeas}_{bi} > \text{AND\_KEEPPGOING\_IB}$  (provided you specified a

AND\_KEEPPGOING\_IB parameter), and the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci} > \text{AND\_KEEPPGOING\_IC}$  and the total dual infeasibility  $\sum_{i=1}^n infeas_{di} > \text{AND\_KEEPPGOING\_ID}$  (provided you specified a AND\_KEEPPGOING\_ID parameter), and if it is, PROC NETFLOW will perform more optimization.

If all these tests to decide whether more optimization should be performed are false, optimization is stopped.

For example,

```
proc netflow
  stop_c=1000
  and_stop_c=2000 and_stop_dg=0.01
  and_stop_ib=1 and_stop_ic=1 and_stop_id=1
  keepgoing_c=1500
  and_keepgoing_c=2500 and_keepgoing_dg=0.05
  and_keepgoing_ib=1 and_keepgoing_ic=1 and_keepgoing_id=1
```

At the beginning of each iteration, PROC NETFLOW will stop if

- complementarity  $\leq 1000$  or
- complementarity  $\leq 2000$  and duality gap  $\leq 0.01$  and the total bound, constraint, and dual infeasibilities are each  $\leq 1$

When PROC NETFLOW determines it should stop because a usual stopping condition is met, it will stop only if

- complementarity  $\leq 1500$  or
- complementarity  $\leq 2500$  and duality gap  $\leq 0.05$  and the total bound, constraint, and dual infeasibilities are each  $\leq 1$

### **Interior Point: Upper Bounds**

If the LP model had upper bounds ( $0 \leq x \leq u$  where  $u$  is the upper bound vector), then the primal and dual problems, the duality gap, and the KKT conditions would have to be expanded.

The primal Linear Program to be solved is

$$\begin{aligned} & \min \{c^T x\} \\ \text{subject to} \quad & Ax = b \\ & 0 \leq x \leq u \end{aligned}$$

$0 \leq x \leq u$  is split into  $x \geq 0$  and  $x \leq u$ . Let  $z$  be primal slack so that  $x + z = u$ , and associate dual variables  $w$  with these constraints. The Interior Point solves the system of equations to satisfy the Karush-Kuhn-Tucker (KKT) conditions for optimality:

$$\begin{aligned}
Ax &= b \\
x + z &= u \\
A^T y + s - w &= c \\
x^T s &= 0 \\
z^T w &= 0 \\
x, s, z, w &\geq 0
\end{aligned}$$

These are the conditions for feasibility, with the *complementarity* conditions  $x^T s = 0$  and  $z^T w = 0$  added.  $c^T x = b^T y - u^T w$  must occur at the optimum. Complementarity forces the optimal objectives of the primal and dual to be equal,  $c^T x_{opt} = b^T y_{opt} - u^T w_{opt}$ , as

$$\begin{aligned}
0 &= z_{opt}^T w_{opt} = (u - x_{opt})^T w_{opt} = u^T w_{opt} - x_{opt}^T w_{opt} \\
0 &= x_{opt}^T s_{opt} = s_{opt}^T x_{opt} = (c - A^T y_{opt} + w_{opt})^T x_{opt} = \\
&= c^T x_{opt} - y_{opt}^T (Ax_{opt}) + w_{opt}^T x_{opt} = c^T x_{opt} - b^T y_{opt} + u^T w_{opt} \\
0 &= c^T x_{opt} - b^T y_{opt} + u^T w_{opt}
\end{aligned}$$

Before the optimum is reached, a solution  $(x, y, s, z, w)$  might not satisfy the KKT conditions:

- Primal bound constraints can be broken,  $infeas_b = u - x - z \neq 0$ .
- Primal constraints can be broken,  $infeas_c = b - Ax \neq 0$ .
- Dual constraints can be broken,  $infeas_d = c - A^T y - s + w \neq 0$ .
- Complementarity conditions are unsatisfied,  $x^T s \neq 0$  and  $z^T w \neq 0$ .

The calculations of the Interior point algorithm can easily be derived in a fashion similar to calculations for when an LP has no upper bounds. See the paper by Lustig, Marsten, and Shanno (1992). An important point is that upper bounds can be handled by specializing the algorithm and *not* by generating the constraints  $x + z = u$  and adding these to the main primal constraints  $Ax = b$ .

---

## References

- Lustig, I.J., Marsten, R.E., and Shanno, D.F. (1992), "On Implementing Methrotra's Predictor-Corrector Interior-Point Method for Linear Programming," *SIAM Journal of Optimization* 2 (3), 435-449.
- Roos, C., Terlaky, T., and Vial, J.-Ph. (1997), "Theory and Algorithms for Linear Optimization," Chichester, England: John Wiley & Sons.
- Wright, S.J. (1996), "Primal-Dual Interior Point Algorithms," Philadelphia: SIAM.
- Ye, Y. (1996), "Interior Point Algorithms: Theory and Analysis," Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons.

# Chapter 8

## The PM Procedure

### Chapter Table of Contents

---

|                               |     |
|-------------------------------|-----|
| <b>OVERVIEW</b>               | 213 |
| <b>PM WINDOW</b>              | 213 |
| User Interface Features       | 213 |
| Gantt View                    | 213 |
| Creating and Editing Projects | 215 |
| <b>DETAILS</b>                | 216 |
| Schedule Data Set             | 216 |



# Chapter 8

## The PM Procedure

---

### Overview

The following features are new in the PM procedure:

- A task information dialog window in the Gantt view that enables you to view activity-specific information such as job number, duration, duration units, calendar information, precedence information, and resource information. You can also edit the activity duration from the task information window.
- A set of resource usage columns in the Table View when there are alternate resources.
- A new set of observations in the Schedule data set when there are resource-driven activities.

---

### PM Window

---

#### User Interface Features

You can manipulate the durations of the tasks using the Task Information dialog box shown in Figure 8.1 or by changing the length of the Early Schedule bar in the Gantt View. You can also move the task in time by dragging the Early Schedule bar over to a new position. This affects the Target Date for the associated task.

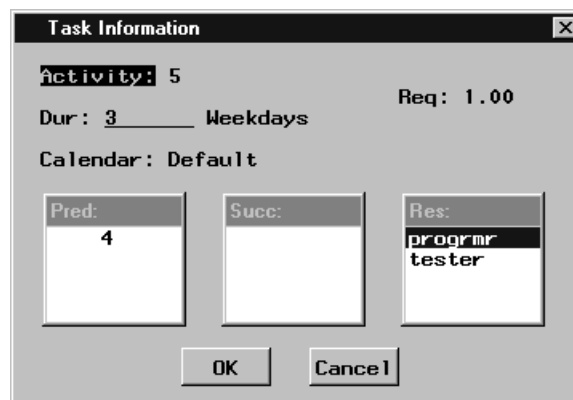


Figure 8.1. Task Information Dialog Box

---

#### Gantt View

In addition to displaying the activity schedules in an easy-to-view format, the Gantt View in the PM Window can also be used to change the durations of the activities,

add or delete precedence constraints, set activity alignment constraints, set progress information, and provide access to calendar, precedence, and resource information.

### Display Task Information

You can display detailed information for an activity by right-clicking on any of its schedule bars and selecting “Task Information” from the resulting pop-up menu (see Figure 8.2).



**Figure 8.2.** Schedule Bar Pop-up Menu

The ensuing Task Information dialog window (see Figure 8.3) displays the job number, duration, duration units, a list of predecessor activities, and a list of successor activities, as well as applicable calendar and resource information for the selected activity. You can also edit the activity duration from the task information window.

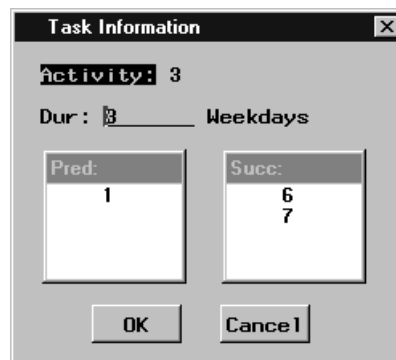


**Figure 8.3.** Task Information Dialog Window

If any calendars have been defined to the project, the activity calendar is also displayed. If the project utilizes any resources, there is a list box that lists the resources required by the activity (see Figure 8.4). Selecting a resource from this list box displays the quantity required by the activity in the “Req” field. Furthermore, if the selected resource drives the duration of the activity, then the appropriate work value is also displayed in the “Work” field.

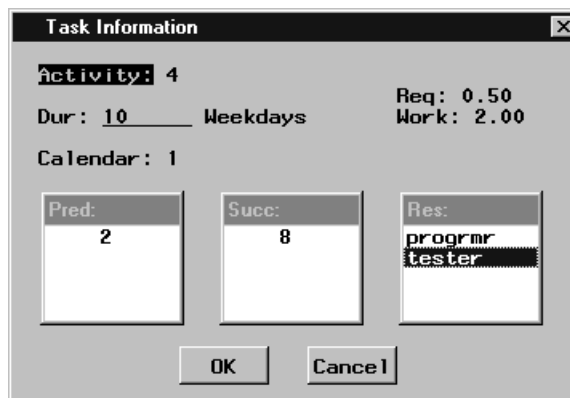


**Figure 8.4.** Task Information Dialog Window (Calendar and Resources)

**Modify Durations**

You can modify the duration of an activity in one of several ways. In the Gantt View, you can enter it directly using the task information window as described in the “Display Task Information” section on page 214 or change it indirectly by altering the width of the schedule bar. To change the duration of an activity, point to the right edge of the activity’s Early Schedule bar with the left mouse button, and drag it to the left or the right depending on whether you want to decrease or increase the duration. You can also edit activity durations in the Table View.

---

**Creating and Editing Projects**
**Add Precedence Constraints**

To add precedence constraints, point the cursor at the right edge of the predecessor activity till it changes to a cross-hair and drag it vertically up or down to the left edge of the successor activity. By starting and dropping at different ends of the activity bar, you can create nonstandard precedence relationships between the activities. You can view the predecessor and successor tasks for an activity from the task information window.

**Change Duration**

The duration of an activity can be directly changed from the Duration column of the Table View or the task information window of the Gantt View. It can also be indirectly changed by dragging the activity bar at the right edge using the left mouse button in the Gantt View.

**Delete Precedence Constraints**

To delete a precedence constraint, click anywhere on the arc with the right mouse button and choose “Delete” from the pop-up menu. You can view the predecessor and successor tasks for an activity from the task information window.

**Edit Calendar Specifications**

Calendars are defined by the CALEDATA= option in the PROC PM statement. This option is similar to the corresponding option in PROC CPM. Once calendars are defined in the Project, an activity’s calendar can be changed or set in the Table View by editing the Activity Calendar or Calendar Name columns. You can either type the values or select them from the pull-down menu displayed by pressing the right mouse button in either of the Calendar columns. You can also view the activity calendar from the task information window.

**Edit Resource Requirements**

The resource requirement information for each activity is displayed and can be edited in the Table View. A column for a resource is created in the Table View when it is specified in the RESOURCE statement of the PROC PM invocation, or it is created by the Resource Manager of PROJMAN.

Changing the resource requirement causes the project to be rescheduled using the new resources. You can also view the resource requirements for an activity from the task information window.

If alternate resources are used by the scheduling algorithm, an extra set of columns is added to the Table View. These columns (one for every resource in the project)



display the resources that were actually used. These Usage columns for the resources cannot be edited.

---

## Details

---

### Schedule Data Set

---

There is now yet another difference between the forms of the Schedule output data sets produced by the PM and CPM procedures. The Schedule output data set produced by the PM procedure creates a new set of observations when there are resource-driven durations.

The PM procedure contains three sets of observations, identified by three different values of a new variable, `OBS_TYPE`:

- The first set of observations contains one observation for every activity in the project. The value of the `OBS_TYPE` variable for these observations is 'SCHEDULE'. These observations contain all the activity information such as the duration, the start and finish times, the resource requirements, and so forth.
- The second set of observations contains one observation for every precedence constraint in the project. The value of the `OBS_TYPE` variable for these observations is 'LOGIC'. These observations contain all the precedence information such as the activity, successor, and lag information.
- The third set of observations is present only if the project has resource-driven durations. The value of the `OBS_TYPE` variable for these observations is 'WORK'. These observations specify the WORK value for each resource used by each activity in the project.

# Subject Index

## A

- affine step
  - INTPOINT procedure, 70, 72
  - NETFLOW procedure, 204
- alternate resources
  - CPM procedure, 37
- AND\_KEEPPGOING\_C= option
  - INTPOINT procedure, 122
  - NETFLOW procedure, 199
- AND\_KEEPPGOING\_DG= option
  - INTPOINT procedure, 122
  - NETFLOW procedure, 199
- AND\_KEEPPGOING\_IB= option
  - INTPOINT procedure, 122
  - NETFLOW procedure, 199
- AND\_KEEPPGOING\_IC= option
  - INTPOINT procedure, 123
  - NETFLOW procedure, 199
- AND\_KEEPPGOING\_ID= option
  - INTPOINT procedure, 123
  - NETFLOW procedure, 199
- AND\_STOP\_C= option
  - INTPOINT procedure, 121
  - NETFLOW procedure, 197
- AND\_STOP\_DG= option
  - INTPOINT procedure, 121
  - NETFLOW procedure, 197
- AND\_STOP\_IB= option
  - INTPOINT procedure, 121
  - NETFLOW procedure, 197
- AND\_STOP\_IC= option
  - INTPOINT procedure, 121
  - NETFLOW procedure, 198
- AND\_STOP\_ID= option
  - INTPOINT procedure, 121
  - NETFLOW procedure, 198
- arc names
  - INTPOINT procedure, 82, 85, 109, 124, 126, 133
- ARC\_SINGLE\_OBS option
  - INTPOINT procedure, 105
- ARCDATA= option
  - INTPOINT procedure, 82–84, 89–90, 97, 104, 109, 114, 123–126, 128–129, 132
- ARCNAME statement
  - INTPOINT procedure, 126
- ARCS\_ONLY\_ARCDATA option
  - INTPOINT procedure, 105
- auxiliary resources
  - CPM procedure, 41

## B

- bill of material (BOM), 3
  - indented, 3
  - modularized, 25
  - multilevel, 3
  - planning, 25
  - single-level, 3
  - summarized, 3
- blending constraints
  - INTPOINT procedure, 77
- BOM
  - See bill of material
- BOM data set
  - Indented, 7, 12
  - Single-level, 3, 11
  - Summarized, 9, 12
- BOM examples
  - bills of material verification, 31
  - planning bill of material, 25
  - with lead time information, 19
- BOM procedure
  - Indented BOM data set, 12
  - input data sets, 14
  - missing values, 18
  - output data sets, 15, 17
  - overview, 3
  - Single-level BOM data set, 11
  - Summarized BOM data set, 12
- BPDIV= option
  - INTPOINT procedure, 105
- BYPASSDIV= option
  - INTPOINT procedure, 105
- BYPASSDIVIDE= option
  - INTPOINT procedure, 105
- BYTES= option
  - INTPOINT procedure, 97, 106

## C

- calendar information
  - PM Window, 215
- CAPAC statement
  - INTPOINT procedure, 123
- \_CAPAC\_ variable
  - ARCDATA data set (INTPOINT), 123
  - CONOUT data set (INTPOINT), 142
- CAPACITY statement
  - INTPOINT procedure, 123
- case sensitivity
  - INTPOINT procedure, 114, 143

- centering step
    - INTPOINT procedure, 70, 73
    - NETFLOW procedure, 205
  - central path
    - INTPOINT procedure, 68
    - NETFLOW procedure, 203
  - Cholesky factorization
    - INTPOINT procedure, 70, 87
    - NETFLOW procedure, 204
  - CHOLTINYTOL= option
    - INTPOINT procedure, 117
  - \_COE, SAS variables with names that begin with
    - CONDATA data set (INTPOINT), 123
  - COEF statement
    - INTPOINT procedure, 123, 129, 136
  - COLUMN statement
    - INTPOINT procedure, 110, 113, 115, 124, 135
  - \_COLUMN\_ variable
    - CONDATA data set (INTPOINT), 124
  - complementarity
    - INTPOINT procedure, 68, 71
    - NETFLOW procedure, 202
  - Component variables
    - Single-level BOM data set, 12
  - \_CON, SAS variables with names that begin with
    - CONDATA data set (INTPOINT), 127
  - \_CON\_ variable
    - CONDATA data set (INTPOINT), 127
  - CON\_SINGLE\_OBS option
    - INTPOINT procedure, 106
  - CONDATA= option
    - INTPOINT procedure, 82–83, 85, 89–90, 97, 104, 109, 113–115, 123–124, 126, 129, 131, 133
  - CONOUT= option
    - INTPOINT procedure, 83, 90, 104, 125, 142
  - CONTYPE statement
    - INTPOINT procedure, 129
  - \_COST\_ variable
    - ARCDATA data set (INTPOINT), 125
  - COST statement
    - INTPOINT procedure, 124
  - \_COST\_ variable
    - CONOUT data set (INTPOINT), 142
  - COUT= option
    - INTPOINT procedure, 104
  - CPM procedure
    - auxiliary resources, 41
    - multiple alternates, 39
    - resource allocation, 38, 40
    - resource-driven durations, 40
- D**
- data set options
    - INTPOINT procedure, 104
  - DC= option
    - INTPOINT procedure, 106
  - DCT= option
    - INTPOINT procedure, 107
  - DEFCAPACITY= option
    - INTPOINT procedure, 106
  - DEFCONTYPE= option
    - INTPOINT procedure, 95, 107, 129, 136
  - DEFCOST= option
    - INTPOINT procedure, 107
  - DEFMINFLOW= option
    - INTPOINT procedure, 107
  - DEFTYPE= option
    - INTPOINT procedure, 107
  - delete precedence constraints
    - PM Window, 215
  - DEMAND statement
    - INTPOINT procedure, 125, 146
  - \_DEMAND\_ variable
    - ARCDATA data set (INTPOINT), 125
    - CONOUT data set (INTPOINT), 143
  - DEMAND= option
    - INTPOINT procedure, 107, 109
  - dense input format
    - INTPOINT procedure, 83, 90, 106, 108, 110, 114, 127, 129, 131, 133, 154, 157
  - DENSETHR= option
    - INTPOINT procedure, 117
  - dependent demand, 14
  - dependent demand process, 14, 17, 19
  - details
    - INTPOINT procedure, 132
  - dialog window
    - task information, 213
  - display task information
    - Gantt View, 214
  - distribution problem
    - INTPOINT procedure, 74
  - DMF= option
    - INTPOINT procedure, 107
  - DTREE procedure
    - graphics options, 55
  - dual problem
    - INTPOINT procedure, 67
    - NETFLOW procedure, 201
- E**
- edit duration
    - Gantt View, 215
    - PM Window, 215
  - edit project
    - PM procedure, 215
  - embedded networks
    - INTPOINT procedure, 80
  - examples
    - See also BOM examples
    - BOM procedure, 19
    - INTPOINT procedure, 162
  - excess node
    - INTPOINT procedure, 151
    - NETFLOW procedure, 199

**F**

\_FCOST\_ variable  
     CONOUT data set (INTPOINT), 88, 92, 143  
 flow conservation constraints  
     INTPOINT procedure, 64, 76–77, 80, 87, 139, 145, 153  
 \_FLOW\_ variable  
     CONOUT data set (INTPOINT), 88, 92, 143  
 FROM statement  
     INTPOINT procedure, 129  
 \_FROM\_ variable  
     ARCDATA data set (INTPOINT), 129  
     CONOUT data set (INTPOINT), 142  
 FROMNODE statement  
     INTPOINT procedure, 129  
 functional summary  
     INTPOINT procedure, 99

**G**

Gantt View, 213  
     display task information, 214  
     edit duration, 215  
     nonstandard precedence relationships, 215  
     PM procedure, 213  
     PM Window, 213  
 getting started  
     INTPOINT procedure, 82  
 graphics version  
     options specific to (DTREE), 55  
     options specific to (NETDRAW), 191  
 GROUPED= option  
     INTPOINT procedure, 105, 107

**H**

HEAD statement  
     INTPOINT procedure, 125  
 \_HEAD\_ variable  
     ARCDATA data set (INTPOINT), 125  
 HEADNODE statement  
     INTPOINT procedure, 125, 127  
 \_HL\_ variable  
     ARCDATA data set (INTPOINT), 123

**I**

ID statement  
     INTPOINT procedure, 125  
 ID variables  
     Single-level BOM data set, 12  
     Summarized BOM data set, 17  
 IMAXITERB= option  
     INTPOINT procedure, 120  
 indented bill of material, 3  
 Indented BOM data set, 7, 12  
     variables, 15  
 independent demand, 14  
 INF= option  
     INTPOINT procedure, 109  
 infeasibility  
     INTPOINT procedure, 68, 72, 145

NETFLOW procedure, 202  
 INFINITY= option  
     INTPOINT procedure, 105–106, 109  
 input data sets  
     See also Single-level BOM data set  
     BOM procedure, 14  
     INTPOINT procedure, 104, 132  
 Interior Point algorithm  
     INTPOINT procedure, 63, 66  
     NETFLOW procedure, 201  
 INTPOINT procedure  
     affine step, 70, 72  
     AND\_KEEPPGOING\_C= option, 122  
     AND\_KEEPPGOING\_DG= option, 122  
     AND\_KEEPPGOING\_IB= option, 122  
     AND\_KEEPPGOING\_IC= option, 123  
     AND\_KEEPPGOING\_ID= option, 123  
     AND\_STOP\_C= option, 121  
     AND\_STOP\_DG= option, 121  
     AND\_STOP\_IB= option, 121  
     AND\_STOP\_IC= option, 121  
     AND\_STOP\_ID= option, 121  
     arc names, 82, 85, 109, 124, 126, 133  
     ARC\_SINGLE\_OBS option, 105  
     ARCDATA= option, 82–84, 89–90, 97, 104, 109, 114, 123–126, 128–129, 132  
     ARCNAME statement, 126  
     ARCS\_ONLY\_ARCDATA option, 105  
     blending constraints, 77  
     BPDIV= option, 105  
     BYPASSDIV= option, 105  
     BYPASSDIVIDE= option, 105  
     BYTES= option, 97, 106  
     CAPAC statement, 123  
     CAPACITY statement, 123  
     case sensitivity, 114, 143  
     centering step, 70, 73  
     central path, 68  
     Cholesky factorization, 70, 87  
     CHOLTINYTOL= option, 117  
     COEF statement, 123, 129, 136  
     COLUMN statement, 110, 113, 115, 124, 135  
     complementarity, 68, 71  
     CON\_SINGLE\_OBS option, 106  
     CONDATA= option, 82–83, 85, 89–90, 97, 104, 109, 113–115, 123–124, 126, 129, 131, 133  
     CONOUT= option, 83, 90, 104, 125, 142  
     CONTYPE statement, 129  
     COST statement, 124  
     COUT= option, 104  
     data set options, 104  
     DC= option, 106  
     DCT= option, 107  
     DEFCAPACITY= option, 106  
     DEFCONTYPE= option, 95, 107, 129, 136  
     DEFCOST= option, 107  
     DEFMINFLOW= option, 107  
     DEFTYPE= option, 107

- DEMAND statement, 125, 146
- DEMAND= option, 107, 109
- dense input format, 83, 90, 106, 108, 110, 114, 127, 129, 131, 133, 154, 157
- DENSETHR= option, 117
- details, 132
- distribution problem, 74
- DMF= option, 107
- dual problem, 67
- embedded networks, 80
- examples, 162
- excess node, 151
- flow conservation constraints, 64, 76–77, 80, 87, 139, 145, 153
- FROM statement, 129
- FROMNODE statement, 129
- functional summary, 99
- getting started, 82
- GROUPED= option, 105, 107
- HEAD statement, 125
- HEADNODE statement, 125, 127
- ID statement, 125
- IMAXITERB= option, 120
- INF= option, 109
- infeasibility, 68, 72, 145
- INFINITY= option, 105–106, 109
- input data sets, 104, 132
- Interior Point algorithm, 63, 66
- INTPOINT statement, 97, 99, 104
- introductory example, 133, 135
- introductory LP example, 91
- introductory NPSC example, 83
- inventory problem, 74
- IPRSLTYPE= option, 118
- Karush-Kuhn-Tucker (KKT) conditions, 67, 71
- KEEPGOING\_C= option, 121
- KEEPGOING\_DG= option, 122
- KEEPGOING\_IB= option, 122
- KEEPGOING\_IC= option, 122
- KEEPGOING\_ID= option, 122
- linear programming problem, 66, 89, 139, 153
- LO statement, 126
- loop arcs, 144
- LOWERBD statement, 126
- LP, 66
- LP variable names, 90
- mathematical description of LP, 66
- mathematical description of NPSC, 64
- MAX option, 109
- MAXFLOW option, 107, 109, 113–114
- maximal flow problem, 105, 113–114
- MAXIMIZE option, 94, 109
- MAXITERB= option, 120
- MEMREP option, 109
- MF option, 109
- MINFLOW statement, 126
- minimum flow problem, 109
- missing S supply and missing D demand, 146
- multicommodity problems, 78
- multiple arcs, 144
- multiprocess, multiproduct example, 79
- NAME statement, 109, 114, 124, 126, 133
- NAMECTRL= option, 109, 126
- NARCS= option, 112
- NCOEFS= option, 112
- NCONS= option, 112
- network problems, 74
- network programming problem with side constraints, 64
- NNAS= option, 112
- NNODES= option, 112
- NODE statement, 127–128
- NODEDATA= option, 82–84, 97, 104, 114, 127–128
- NON\_REPLIC= option, 112
- nonarc variables, 80
- NPSC, 64
- OBJFN statement, 124
- options classified by function, 99
- output data sets, 104, 142
- PDGAPTOL= option, 120
- PDSTEPMULT= option, 118
- preprocessing, 67, 83, 90, 118
- Primal-Dual with Predictor-Corrector algorithm, 66, 70
- PRINTLEVEL2= option, 119
- production-inventory-distribution problem, 74
- proportional constraints, 75, 85
- PRSLTYPE= option, 118
- QUIT statement, 98, 127
- RCHOLTINYTOL= option, 117
- RDENSETHR= option, 117
- RHS statement, 127, 133
- RHSOBS= option, 113, 124, 135
- ROW statement, 124, 127, 131, 133
- RPDGAPTOL= option, 120
- RPDSTEPMULT= option, 118
- RTOLDINF= option, 116
- RTOLPINF= option, 116
- RTOLTOTDINF= option, 117
- RTOLTOTPINF= option, 117
- RUN statement, 98, 128
- SCALE= option, 113
- SCDATA option, 114
- shortest path problem, 113–114
- SHORTPATH option, 107, 109, 113–114
- side constraints, 64, 80
- SINK= option, 105, 107, 109, 113
- SOURCE= option, 105, 109, 113–114
- SP option, 113
- sparse input format, 83, 90, 95, 106, 108, 110, 113–115, 123–124, 127, 129, 135, 144, 154, 157, 172
- SPARSECONDATA option, 114, 135
- special rows, 124, 129, 134, 138
- STOP\_C= option, 120
- STOP\_DG= option, 120
- STOP\_IB= option, 120

STOP\_IC= option, 120  
 STOP\_ID= option, 120  
 stopping criteria, 120, 158  
 summary of PROC INTPOINT Options, 99  
 SUPDEM statement, 128, 146  
 SUPPLY statement, 128, 146  
 supply-chain problem, 74  
 SUPPLY= option, 109, 114  
 symbolic factorization, 70  
 syntax, 102  
 table of syntax elements, 99  
 TAIL statement, 129  
 TAILNODE statement, 127–129  
 THRUNET option, 114, 151  
 TO statement, 125  
 TOLDINF= option, 116  
 TOLPINF= option, 116  
 TOLTOTDINF= option, 117  
 TOLTOTPINF= option, 117  
 TONODE statement, 125  
 TYPE statement, 129, 133  
 TYPEOBS= option, 115, 124, 135  
 upper bounds, 67, 71  
 UPPERBD statement, 123  
 VAR statement, 110, 129, 131, 133  
 VARNAME statement, 126  
 VERBOSE= option, 115  
 Z2= option, 115  
 ZERO2= option, 115  
 ZEROTOL= option, 116  
 INTPOINT statement  
   INTPOINT procedure, 97, 99, 104  
 introductory example  
   INTPOINT procedure, 133, 135  
 introductory LP example  
   INTPOINT procedure, 91  
 introductory NPSC example  
   INTPOINT procedure, 83  
 inventory problem  
   INTPOINT procedure, 74  
 IPRSLTYPE= option  
   INTPOINT procedure, 118  
 item master file  
   See part master data file

## K

Karush-Kuhn-Tucker (KKT) conditions  
   INTPOINT procedure, 67, 71  
   NETFLOW procedure, 201, 209  
 KEEPGOING\_C= option  
   INTPOINT procedure, 121  
   NETFLOW procedure, 198  
 KEEPGOING\_DG= option  
   INTPOINT procedure, 122  
   NETFLOW procedure, 198  
 KEEPGOING\_IB= option  
   INTPOINT procedure, 122  
   NETFLOW procedure, 198  
 KEEPGOING\_IC= option

  INTPOINT procedure, 122  
   NETFLOW procedure, 198  
 KEEPGOING\_ID= option  
   INTPOINT procedure, 122  
   NETFLOW procedure, 198

## L

lead time, 12  
 LeadTime variable  
   Indented BOM data set, 13, 15  
   Single-level BOM data set, 12  
 \_LENGTH\_ variable  
   ARCDATA data set (INTPOINT), 125  
 \_Level\_ variable  
   Indented BOM data set, 7, 15  
 linear programming problem  
   INTPOINT procedure, 66, 89, 139, 153  
 list of, BOM procedure  
   variables, 14  
 LO statement  
   INTPOINT procedure, 126  
 \_LO\_ variable  
   ARCDATA data set (INTPOINT), 126  
   CONOUT data set (INTPOINT), 142  
 loop arcs  
   INTPOINT procedure, 144  
 Low\_Code variable  
   Summarized BOM data set, 10, 17  
 \_LOWER\_ variable  
   ARCDATA data set (INTPOINT), 126  
 LOWERBD statement  
   INTPOINT procedure, 126  
 \_LOWERBD variable  
   ARCDATA data set (INTPOINT), 126  
 LP  
   INTPOINT procedure, 66  
 LP variable names  
   INTPOINT procedure, 90

## M

master production schedule, 15  
 master schedule item, 9, 14  
 mathematical description of LP  
   INTPOINT procedure, 66  
 mathematical description of NPSC  
   INTPOINT procedure, 64  
 MAX option  
   INTPOINT procedure, 109  
 MAXFLOW option  
   INTPOINT procedure, 107, 109, 113–114  
 maximal flow problem  
   INTPOINT procedure, 105, 113–114  
 MAXIMIZE option  
   INTPOINT procedure, 94, 109  
 MAXITERB= option  
   INTPOINT procedure, 120  
 MEMREP option  
   INTPOINT procedure, 109  
 MF option

INTPOINT procedure, 109  
 MINFLOW statement  
   INTPOINT procedure, 126  
   \_MINFLOW variable  
     ARCDATA data set (INTPOINT), 126  
 minimum flow problem  
   INTPOINT procedure, 109  
 missing S supply and missing D demand  
   INTPOINT procedure, 146  
 missing values  
   BOM procedure, 18  
 modularized bill of material construction, 25  
 MPS  
   See master production schedule  
 MPS item  
   See master schedule item  
 multicommodity problems  
   INTPOINT procedure, 78  
 multilevel bill of material, 3  
 multiple alternates  
   CPM procedure, 37, 39–40  
 multiple arcs  
   INTPOINT procedure, 144  
 multiprocess, multiproduct example  
   INTPOINT procedure, 79

## N

NAME statement  
   INTPOINT procedure, 109, 114, 124, 126, 133  
 \_NAME\_ variable  
   ARCDATA data set (INTPOINT), 126  
   CONOUT data set (INTPOINT), 142  
 NAMECTRL= option  
   INTPOINT procedure, 109, 126  
 NARCS= option  
   INTPOINT procedure, 112  
 NCOEFS= option  
   INTPOINT procedure, 112  
 NCONS= option  
   INTPOINT procedure, 112  
 NETDRAW procedure  
   graphics options, 191  
 NETFLOW procedure  
   affine step, 204  
   AND\_KEEPPGOING\_C= option, 199  
   AND\_KEEPPGOING\_DG= option, 199  
   AND\_KEEPPGOING\_IB= option, 199  
   AND\_KEEPPGOING\_IC= option, 199  
   AND\_KEEPPGOING\_ID= option, 199  
   AND\_STOP\_C= option, 197  
   AND\_STOP\_DG= option, 197  
   AND\_STOP\_IB= option, 197  
   AND\_STOP\_IC= option, 198  
   AND\_STOP\_ID= option, 198  
   centering step, 205  
   central path, 203  
   Cholesky factorization, 204  
   complementarity, 202  
   dual problem, 201

  excess node, 199  
   infeasibility, 202  
   Interior Point algorithm, 201  
   Karush-Kuhn-Tucker (KKT) conditions, 201, 209  
   KEEPPGOING\_C= option, 198  
   KEEPPGOING\_DG= option, 198  
   KEEPPGOING\_IB= option, 198  
   KEEPPGOING\_IC= option, 198  
   KEEPPGOING\_ID= option, 198  
   preprocessing, 201  
   Primal-Dual with Predictor-Corrector algorithm,  
     204  
   PRINTLEVEL2= option, 196  
   RTOLTOTDINF= option, 195  
   RTOLTOTPINF= option, 196  
   STOP\_C= option, 196  
   STOP\_DG= option, 196  
   STOP\_IB= option, 197  
   STOP\_IC= option, 197  
   STOP\_ID= option, 197  
   stopping criteria, 196, 206  
   symbolic factorization, 204  
   THRUNET option, 200  
   TOLTOTDINF= option, 195  
   TOLTOTPINF= option, 196  
   upper bounds, 201, 209  
 Net\_Req variable  
   Summarized BOM data set, 10, 17  
 network problems  
   INTPOINT procedure, 74  
 network programming problem with side constraints  
   INTPOINT procedure, 64  
 NNAS= option  
   INTPOINT procedure, 112  
 NNODES= option  
   INTPOINT procedure, 112  
 NODE statement  
   INTPOINT procedure, 127–128  
 \_NODE\_ variable  
   NODEDATA data set (INTPOINT), 127  
 NODEDATA= option  
   INTPOINT procedure, 82–84, 97, 104, 114, 127–  
     128  
 NON\_REPLIC= option  
   INTPOINT procedure, 112  
 nonarc variables  
   INTPOINT procedure, 80  
 nonstandard precedence relationships  
   Gantt View, 215  
 NPSC  
   INTPOINT procedure, 64  
 number pages  
   GANTT procedure, 59

## O

OBJFN statement  
   INTPOINT procedure, 124  
 OBSTYPE variable  
   Resource data set (CPM), 38



- On\_Hand variable
  - Summarized BOM data set, 13
- option overplanning, 27
- options classified by function
  - See functional summary
- output data sets
  - See also Indented BOM data set
  - See also Summarized BOM data set
  - BOM procedure, 15, 17
  - INTPOINT procedure, 104, 142
- overview
  - BOM procedure, 3
- P**
- page numbers
  - on decision tree diagram, 55
  - suppress displaying (DTREE), 55
  - suppress displaying (GANTT), 59
- Paren\_ID variable
  - Indented BOM data set, 7, 16
- \_Parent\_ variable
  - Indented BOM data set, 7, 16
- part master data file, 14
- Part variable
  - Indented BOM data set, 13, 15
  - Single-level BOM data set, 13
  - Summarized BOM data set, 13, 17
- Part\_ID variable
  - Indented BOM data set, 7, 16
- PDGAPTOL= option
  - INTPOINT procedure, 120
- PDSTEPMULT= option
  - INTPOINT procedure, 118
- PERIOD variable
  - Resource data set (CPM), 38
- phantom bill of material
  - See planning bill of material
- phantom part, 25
- planning bill of material, 25
- PM procedure
  - details, 216
  - edit project, 215
  - Gantt View, 213
  - PM Window, 213
  - Schedule data set, 216
- PM Window, 213
  - calendar information, 215
  - delete precedence constraints, 215
  - edit duration, 215
  - Gantt View, 213
  - PM procedure, 213
  - resource requirements, 215
  - user interface features, 213
- pop-up menu
  - schedule bar, 214
  - task information, 214
- preprocessing
  - INTPOINT procedure, 67, 83, 90, 118
  - NETFLOW procedure, 201

- Primal-Dual with Predictor-Corrector algorithm
  - INTPOINT procedure, 66, 70
  - NETFLOW procedure, 204
- PRINTLEVEL2= option
  - INTPOINT procedure, 119
  - NETFLOW procedure, 196
- \_Prod\_ variable
  - Indented BOM data set, 7, 16
- production-inventory-distribution problem
  - INTPOINT procedure, 74
- proportional constraints
  - INTPOINT procedure, 75, 85
- PRSLTYPE= option
  - INTPOINT procedure, 118
- pseudo bill of material
  - See planning bill of material

## Q

- QtyOnHand variable
  - Single-level BOM data set, 13
  - Summarized BOM data set, 17
- QtyPer variables
  - See Quantity variables
- Qty\_Prod variable
  - Indented BOM data set, 7, 16
- Quantity variables
  - Indented BOM data set, 13, 15
  - Single-level BOM data set, 13
- QUIT statement
  - INTPOINT procedure, 98, 127

## R

- RCHOLTINYTOL= option
  - INTPOINT procedure, 117
- \_RCOST\_ variable
  - CONOUT data set (INTPOINT), 143
- RDENSETHR= option
  - INTPOINT procedure, 117
- references
  - INTPOINT procedure, 188
  - NETFLOW procedure, 210
- Requirement variable
  - Single-level BOM data set, 13
  - Summarized BOM data set, 17
- RESID variable
  - Resource data set (CPM), 38
- resource allocation, 40
  - auxiliary resources, 41
  - multiple alternates, 39–40
  - resource dictionary, 38
  - resource-driven durations and alternate resources, 40
- resource allocation control
  - alternate resources, 37
  - options related to, 37
- Resource data set
  - CPM procedure, 38
  - multiple alternates, 40
  - variables, 38



resource requirements  
 PM Window, 215

resource-driven durations  
 CPM procedure, 40

RHS statement  
 INTPOINT procedure, 127, 133

\_RHS\_ variable  
 CONDATA data set (INTPOINT), 127

RHSOBS= option  
 INTPOINT procedure, 113, 124, 135

\_ROW, SAS variables with names that begin with  
 CONDATA data set (INTPOINT), 127

ROW statement  
 INTPOINT procedure, 124, 127, 131, 133

\_ROW\_ variable  
 CONDATA data set (INTPOINT), 127

RPDGAPTOL= option  
 INTPOINT procedure, 120

RPDSTEPMULT= option  
 INTPOINT procedure, 118

RTOLDINF= option  
 INTPOINT procedure, 116

RTOLPINF= option  
 INTPOINT procedure, 116

RTOLTOTDINF= option  
 INTPOINT procedure, 117  
 NETFLOW procedure, 195

RTOLTOTPINF= option  
 INTPOINT procedure, 117  
 NETFLOW procedure, 196

RUN statement  
 INTPOINT procedure, 98, 128

## S

SCALE= option  
 INTPOINT procedure, 113

SCDATA option  
 INTPOINT procedure, 114

Schedule data set  
 PM procedure, 216

\_SD\_ variable  
 NODEDATA data set (INTPOINT), 128

shortest path problem  
 INTPOINT procedure, 113–114

SHORTPATH option  
 INTPOINT procedure, 107, 109, 113–114

side constraints  
 INTPOINT procedure, 64, 80

single-level bill of material, 3

Single-level BOM data set, 3, 11  
 missing values, 18  
 variables, 14

single-level where-used list, 9

SINK= option  
 INTPOINT procedure, 105, 107, 109, 113

SOURCE= option  
 INTPOINT procedure, 105, 109, 113–114

SP option  
 INTPOINT procedure, 113

sparse input format  
 INTPOINT procedure, 83, 90, 95, 106, 108, 110,  
 113–115, 123–124, 127, 129, 135, 144,  
 154, 157, 172

SPARSECONDATA option  
 INTPOINT procedure, 114, 135

special rows  
 INTPOINT procedure, 124, 129, 134, 138

\_STATUS\_ variable  
 CONOUT data set (INTPOINT), 143

STOP\_C= option  
 INTPOINT procedure, 120  
 NETFLOW procedure, 196

STOP\_DG= option  
 INTPOINT procedure, 120  
 NETFLOW procedure, 196

STOP\_IB= option  
 INTPOINT procedure, 120  
 NETFLOW procedure, 197

STOP\_IC= option  
 INTPOINT procedure, 120  
 NETFLOW procedure, 197

STOP\_ID= option  
 INTPOINT procedure, 120  
 NETFLOW procedure, 197

stopping criteria  
 INTPOINT procedure, 120, 158  
 NETFLOW procedure, 196, 206

summarized bill of material, 3

Summarized BOM data set, 9, 12  
 variables, 17

summary of PROC INTPOINT Options  
 INTPOINT procedure, 99

SUPDEM statement  
 INTPOINT procedure, 128, 146

\_SUPDEM\_ variable  
 NODEDATA data set (INTPOINT), 128

super bill of material  
 See planning bill of material

SUPPLY statement  
 INTPOINT procedure, 128, 146

\_SUPPLY\_ variable  
 ARCDATA data set (INTPOINT), 129  
 CONOUT data set (INTPOINT), 143

supply-chain problem  
 INTPOINT procedure, 74

SUPPLY= option  
 INTPOINT procedure, 109, 114

symbolic factorization  
 INTPOINT procedure, 70  
 NETFLOW procedure, 204

syntax  
 INTPOINT procedure, 102

## T

table of syntax elements  
 See functional summary

TAIL statement  
 INTPOINT procedure, 129

TAILNODE statement  
     INTPOINT procedure, 127–129  
 \_TAIL\_ variable  
     ARCDATA data set (INTPOINT), 98, 129  
 task information dialog window, 213  
 THRUNET option  
     INTPOINT procedure, 114, 151  
     NETFLOW procedure, 200  
 TO statement  
     INTPOINT procedure, 125  
 \_TO\_ variable  
     ARCDATA data set (INTPOINT), 125  
     CONOUT data set (INTPOINT), 142  
 TOLDINF= option  
     INTPOINT procedure, 116  
 TOLPINF= option  
     INTPOINT procedure, 116  
 TOLTOTDINF= option  
     INTPOINT procedure, 117  
     NETFLOW procedure, 195  
 TOLTOTPINF= option  
     INTPOINT procedure, 117  
     NETFLOW procedure, 196  
 TONODE statement  
     INTPOINT procedure, 125  
 top-down hierarchical diagram, 191  
 Tot\_Lead variable  
     Indented BOM data set, 13, 16  
 TYPE statement  
     INTPOINT procedure, 129, 133  
 \_TYPE\_ variable  
     CONDATA data set (INTPOINT), 129  
 TYPEOBS= option  
     INTPOINT procedure, 115, 124, 135

## U

upper bounds  
     INTPOINT procedure, 67, 71  
     NETFLOW procedure, 201, 209  
 \_UPPER\_ variable  
     ARCDATA data set (INTPOINT), 123  
 UPPERBD statement  
     INTPOINT procedure, 123  
 \_UPPERBD variable  
     ARCDATA data set (INTPOINT), 123  
 user interface features  
     PM Window, 213

## V

VAR statement  
     INTPOINT procedure, 110, 129, 131, 133  
 variables  
     list of, BOM procedure, 14  
     treatment of missing values (BOM), 18  
 VARNAME statement  
     INTPOINT procedure, 126  
 VERBOSE= option  
     INTPOINT procedure, 115

## W

where-used list  
     single-level, 9

## Z

Z2= option  
     INTPOINT procedure, 115  
 ZERO2= option  
     INTPOINT procedure, 115  
 ZEROTOL= option  
     INTPOINT procedure, 116



# Syntax Index

## A

ACTNET statement, NETDRAW procedure  
    graphics options, 191  
AND\_KEEPPGOING\_C= option  
    PROC INTPOINT statement, 122  
    PROC NETFLOW statement, 199  
AND\_KEEPPGOING\_DG= option  
    PROC INTPOINT statement, 122  
    PROC NETFLOW statement, 199  
AND\_KEEPPGOING\_IB= option  
    PROC INTPOINT statement, 122  
    PROC NETFLOW statement, 199  
AND\_KEEPPGOING\_IC= option  
    PROC INTPOINT statement, 123  
    PROC NETFLOW statement, 199  
AND\_KEEPPGOING\_ID= option  
    PROC INTPOINT statement, 123  
    PROC NETFLOW statement, 199  
AND\_STOP\_C= option  
    PROC INTPOINT statement, 121  
    PROC NETFLOW statement, 197  
AND\_STOP\_DG= option  
    PROC INTPOINT statement, 121  
    PROC NETFLOW statement, 197  
AND\_STOP\_IB= option  
    PROC INTPOINT statement, 121  
    PROC NETFLOW statement, 197  
AND\_STOP\_IC= option  
    PROC INTPOINT statement, 121  
    PROC NETFLOW statement, 198  
AND\_STOP\_ID= option  
    PROC INTPOINT statement, 121  
    PROC NETFLOW statement, 198  
ARC\_SINGLE\_OBS option  
    PROC INTPOINT statement, 105  
ARCDATA= option  
    PROC INTPOINT statement, 104  
ARCNAME statement  
    INTPOINT procedure, 126  
ARCS\_ONLY\_ARCDATA option  
    PROC INTPOINT statement, 105

## B

BOM procedure  
    PROC BOM statement, 11  
    STRUCTURE statement, 12  
BPD= option  
    PROC INTPOINT statement, 105  
BYPASSDIV= option

    PROC INTPOINT statement, 105  
BYPASSDIVIDE= option  
    PROC INTPOINT statement, 105  
BYTES= option  
    PROC INTPOINT statement, 106

## C

CAPAC statement  
    INTPOINT procedure, 123  
CAPACITY statement  
    INTPOINT procedure, 123  
CHART statement (GANTT), 59  
COEF statement  
    INTPOINT procedure, 123  
COLUMN statement  
    INTPOINT procedure, 124  
COMP= option  
    See COMPONENT= option  
COMPONENT= option  
    STRUCTURE statement (BOM), 12  
CON\_SINGLE\_OBS option  
    PROC INTPOINT statement, 106  
CONDATA= option  
    PROC INTPOINT statement, 104  
CONOUT= option  
    PROC INTPOINT statement, 104  
CONTYPE statement  
    INTPOINT procedure, 129  
COST statement  
    INTPOINT procedure, 124  
COUT= option  
    PROC INTPOINT statement, 104

## D

DATA= option  
    PROC BOM statement, 11  
DC= option  
    PROC INTPOINT statement, 106  
DCT= option  
    PROC INTPOINT statement, 107  
DEFCAPACITY= option  
    PROC INTPOINT statement, 106  
DEFCONTYPE= option  
    PROC INTPOINT statement, 107  
DEFCOST= option  
    PROC INTPOINT statement, 107  
DEFMINFLOW= option  
    PROC INTPOINT statement, 107  
DEFTYPE= option

PROC INTPOINT statement, 107

DEMAND statement

INTPOINT procedure, 125

DEMAND= option

PROC INTPOINT statement, 107

DMF= option

PROC INTPOINT statement, 107

DTREE procedure, 55

PROC DTREE statement, 55

syntax, 55

DUR= option (BOM)

See LEADTIME= option

## F

FROM statement

INTPOINT procedure, 129

FROMNODE statement

INTPOINT procedure, 129

## G

GANTT procedure, 59

CHART statement, 59

syntax, 59

GROSSREQ= option

See REQUIREMENT= option

GROUPED= option

PROC INTPOINT statement, 107

## H

HEAD statement

INTPOINT procedure, 125

HEADNODE statement

INTPOINT procedure, 125

## I

ID statement

INTPOINT procedure, 125

ID= option

STRUCTURE statement (BOM), 12

INF= option

PROC INTPOINT statement, 109

INFINITY= option

PROC INTPOINT statement, 109

INVENTORY= option

See QTYONHAND= option

ITEM= option

See PART= option

## K

KEEPPGOING\_C= option

PROC INTPOINT statement, 121

PROC NETFLOW statement, 198

KEEPPGOING\_DG= option

PROC INTPOINT statement, 122

PROC NETFLOW statement, 198

KEEPPGOING\_IB= option

PROC INTPOINT statement, 122

PROC NETFLOW statement, 198

KEEPPGOING\_IC= option

PROC INTPOINT statement, 122

PROC NETFLOW statement, 198

KEEPPGOING\_ID= option

PROC INTPOINT statement, 122

PROC NETFLOW statement, 198

## L

LEADTIME= option

STRUCTURE statement (BOM), 12

LO statement

INTPOINT procedure, 126

LOWERBD statement

INTPOINT procedure, 126

## M

MAX option

PROC INTPOINT statement, 109

MAXFLOW option

PROC INTPOINT statement, 109

MAXIMIZE option

PROC INTPOINT statement, 109

MEMREPOption

PROC INTPOINT statement, 109

MF option

PROC INTPOINT statement, 109

MINFLOW statement

INTPOINT procedure, 126

MULTALT option

RESOURCE statement (CPM), 37

MULTIPLEALTERNATES option

RESOURCE statement (CPM), 37

## N

NAME statement

INTPOINT procedure, 126

NAMECTRL=option

PROC INTPOINT statement, 109

NARCS= option

PROC INTPOINT statement, 112

NCOEFS= option

PROC INTPOINT statement, 112

NCONS= option

PROC INTPOINT statement, 112

NNAS= option

PROC INTPOINT statement, 112

NNODES= option

PROC INTPOINT statement, 112

NODE statement

INTPOINT procedure, 127

NODEDATA= option

PROC INTPOINT statement, 104

NON\_REPLIC= option

PROC INTPOINT statement, 112

NOPAGENUM option

CHART statement (GANTT), 59

PROC DTREE statement, 55

**O**

OBJFN statement  
   INTPOINT procedure, 124  
 OBSTYPE= option  
   RESOURCE statement (CPM), 37  
 OUT= option  
   PROC BOM statement, 12

**P**

PAGENUM option  
   PROC DTREE statement, 55  
 PART= option  
   STRUCTURE statement (BOM), 13  
 PRINTLEVEL2= option  
   PROC INTPOINT statement, 119  
   PROC NETFLOW statement, 196  
 PROC BOM statement  
   DATA= option, 11  
   OUT= option, 12  
   statement options, 11  
   SUMMARYOUT= option, 12  
 PROC DTREE statement, 55  
   graphics options, 55  
 PROC INTPOINT statement  
   INTPOINT procedure, 99, 104

**Q**

QTYONHAND= option  
   STRUCTURE statement (BOM), 13  
 QTYPER= option  
   See QUANTITY= option  
 QUANTITY= option  
   STRUCTURE statement (BOM), 13  
 QUIT statement  
   INTPOINT procedure, 127

**R**

REQUIREMENT= option  
   STRUCTURE statement (BOM), 13  
 RESID= option  
   RESOURCE statement (CPM), 38  
 RESOURCE statement, CPM procedure  
   resource allocation control options, 37  
 RHS statement  
   INTPOINT procedure, 127  
 RHSOBS= option  
   PROC INTPOINT statement, 113  
 ROTATE option  
   ACTNET statement (NETDRAW), 191  
 ROW statement  
   INTPOINT procedure, 127  
 RUN statement  
   INTPOINT procedure, 128

**S**

SCALE= option  
   PROC INTPOINT statement, 113  
 SCDATA option

  PROC INTPOINT statement, 114  
 SHORTPATH option  
   PROC INTPOINT statement, 113  
 SINK= option  
   PROC INTPOINT statement, 113  
 SINKNODE= option  
   PROC INTPOINT statement, 113  
 SOURCE= option  
   PROC INTPOINT statement, 114  
 SOURCENODE= option  
   PROC INTPOINT statement, 114  
 SP option  
   PROC INTPOINT statement, 113  
 SPARSECONDATA option  
   PROC INTPOINT statement, 114  
 STOP\_C= option  
   PROC INTPOINT statement, 120  
   PROC NETFLOW statement, 196  
 STOP\_DG= option  
   PROC INTPOINT statement, 120  
   PROC NETFLOW statement, 196  
 STOP\_IB= option  
   PROC INTPOINT statement, 120  
   PROC NETFLOW statement, 197  
 STOP\_IC= option  
   PROC INTPOINT statement, 120  
   PROC NETFLOW statement, 197  
 STOP\_ID= option  
   PROC INTPOINT statement, 120  
   PROC NETFLOW statement, 197  
 STRUCTURE statement (BOM)  
   COMPONENT= option, 12  
   QTYONHAND= option, 13  
   QUANTITY= option, 13  
   REQUIREMENT= option, 13  
   BOM procedure, 12  
 SUMMARYOUT= option  
   PROC BOM statement, 12  
 SUPDEM statement  
   INTPOINT procedure, 128  
 SUPPLY statement  
   INTPOINT procedure, 128  
 SUPPLY= option  
   PROC INTPOINT statement, 114

**T**

TAIL statement  
   INTPOINT procedure, 129  
 TAILNODE statement  
   INTPOINT procedure, 129  
 THRUNET option  
   PROC INTPOINT statement, 114  
 TO statement  
   INTPOINT procedure, 125  
 TONODE statement  
   INTPOINT procedure, 125  
 TYPE statement  
   INTPOINT procedure, 129  
 TYPEOBS= option

PROC INTPOINT statement, 115

## **U**

UPPERBD statement

INTPOINT procedure, 123

## **V**

VAR statement

INTPOINT procedure, 131

VARIABLES statement (BOM)

See STRUCTURE statement

VARNAME statement

INTPOINT procedure, 126

VERBOSE= option

PROC INTPOINT, or INTPOINT statement, 115

## **Z**

Z2= option

PROC INTPOINT, or INTPOINT statement, 115

ZERO2= option

PROC INTPOINT, or INTPOINT statement, 115

ZEROTOL= option

PROC INTPOINT, or INTPOINT statement, 116

# Your Turn

---

If you have comments or suggestions about *SAS/OR® Software: Changes and Enhancements, Release 8.1*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute  
Publications Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [suggest@sas.com](mailto:suggest@sas.com)





*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS® Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

- 1.** Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
- 2.** Leave the "Subject" line blank.
- 3.** Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.

# **SAS/STAT<sup>®</sup> Software: Changes and Enhancements, Release 8.1**

---

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/STAT® Software: Changes and Enhancements, Release 8.1*, Cary, NC: SAS Institute Inc., 2000

**SAS/STAT® Software: Changes and Enhancements, Release 8.1**

Copyright © 2000 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-655-4

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, May 2000

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. indicates USA registration.

IBM® and all other International Business Machines Corporation product or service names are registered trademarks or trademarks of International Business Machines Corporation in the USA and other countries.

Oracle® and all other Oracle Corporation product or service names are registered trademarks or trademarks of Oracle Corporation in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

|                                                 |     |
|-------------------------------------------------|-----|
| Chapter 1. The CATMOD Procedure . . . . .       | 1   |
| Chapter 2. The FACTOR Procedure . . . . .       | 15  |
| Chapter 3. The FREQ Procedure . . . . .         | 31  |
| Chapter 4. The GAM Procedure . . . . .          | 35  |
| Chapter 5. The GENMOD Procedure . . . . .       | 69  |
| Chapter 6. The LOESS Procedure . . . . .        | 73  |
| Chapter 7. The LOGISTIC Procedure . . . . .     | 89  |
| Chapter 8. The MIXED Procedure . . . . .        | 109 |
| Chapter 9. The MODECLUS Procedure . . . . .     | 113 |
| Chapter 10. The MULTTEST Procedure . . . . .    | 117 |
| Chapter 11. The NLMIXED Procedure . . . . .     | 121 |
| Chapter 12. The PHREG Procedure . . . . .       | 125 |
| Chapter 13. The SURVEYMEANS Procedure . . . . . | 129 |
| Subject Index . . . . .                         | 151 |
| Syntax Index . . . . .                          | 153 |



# Chapter 1

## The CATMOD Procedure

### Chapter Table of Contents

---

|                                                                                           |    |
|-------------------------------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                                                 | 3  |
| <b>SYNTAX</b> . . . . .                                                                   | 3  |
| MODEL Statement . . . . .                                                                 | 3  |
| <b>DETAILS</b> . . . . .                                                                  | 6  |
| Missing Values . . . . .                                                                  | 6  |
| Zero Frequencies . . . . .                                                                | 6  |
| Computational Formulas . . . . .                                                          | 6  |
| <b>EXAMPLES</b> . . . . .                                                                 | 7  |
| Example 1.1 Log-Linear Independence Model with Structural and<br>Sampling Zeros . . . . . | 7  |
| Example 1.2 Identifying an Inappropriate Model . . . . .                                  | 12 |
| <b>REFERENCES</b> . . . . .                                                               | 14 |





# Chapter 1

## The CATMOD Procedure

---

### Overview

The CATMOD procedure now offers the iterative proportional fitting (IPF) algorithm for fitting hierarchical log-linear models with one population (that is, there are no independent variables and no population variables). The advantage of the IPF algorithm is that you can obtain the log likelihood,  $G^2$ , and the predicted cell counts without performing expensive parameter estimation and covariance computations. To request IPF fitting, you specify the ML=IPF option. Several options for controlling the convergence criterion are available, as well as options for controlling how the degrees of freedom for  $G^2$  are computed.

The new MISSING= option specifies whether a missing cell is treated as a sampling or structural zero. The new ZERO= option specifies whether a non-missing cell with zero weight is treated as a sampling or structural zero.

---

### Syntax

---

#### MODEL Statement

The following options have been added to the MODEL statement.

**ML** < = NR | IPF < ( options ) > >

computes maximum likelihood estimates (MLE) using either a Newton-Raphson algorithm (NR) or an iterative proportional fitting algorithm (IPF).

The option ML=NR (or simply ML) is available when you use logits or generalized logits, and is the default for generalized logits.

The ML=IPF option is available for fitting a hierarchical log-linear model with one population (that is, there are no independent variables and no population variables). The use of bar notation to describe the log-linear effects guarantees that the model is *hierarchical*, meaning that the presence of any interaction term in the model requires the presence of all its lower-order terms. The underlying table in an IPF analysis is the cross-classification of the observed levels of all dependent variables. If the table is *incomplete*, which means that a zero or missing entry occurs in at least one cell, then all missing cells and all cells with zero weight are treated as structural zeros by default. This behavior can be modified with the ZERO= and MISSING= options in the MODEL statement.

You can control the convergence of the two algorithms with the EPSILON= and MAXITER= options in the MODEL statement.

**Note:** The RESTRICT statement is not available with the ML=IPF option.

You can specify the following *options* within parentheses after the ML=IPF option.

**CONV=keyword**

**CONVCRT=keyword** specifies the method that determines when convergence of the IPF algorithm occurs. You can specify one of the following *keywords*:

- CELL** termination requires the maximum absolute difference between consecutive cell estimates to be less than 0.001 (or the value of the EPSILON= option, if specified).
- LOGL** termination requires the relative difference between consecutive estimates of the log likelihood to be less than 1E-8 (or the value of the EPSILON= option, if specified). This is the default.
- MARGIN** termination requires the maximum absolute difference between consecutive margin estimates to be less than 0.001 (or the value of the EPSILON= option, if specified).

**DF=keyword** specifies the method used to compute the degrees of freedom for the goodness of fit  $G^2$  test (labeled “Likelihood Ratio” in the “Estimates” table).

For a *complete* table (a table having nonzero entries in every cell), the degrees of freedom are calculated as the number of cells in the table ( $n_c$ ) minus the number of independent parameters specified in the model ( $n_p$ ). For incomplete tables, these degrees of freedom may be adjusted by the number of fitted zeros ( $n_z$ , which includes the number of structural zeros) and the number of non-estimable parameters due to the zeros ( $n_n$ ). If you are analyzing an incomplete table, you should verify that the degrees of freedom are correct.

You can specify one of the following *keywords*:

- UNADJ** computes the unadjusted degrees of freedom as  $n_c - n_p$ . These are the same degrees of freedom you would get if all cells in the table were positive.
- ADJ** computes the degrees of freedom as  $(n_c - n_p) - (n_z - n_n)$  (Bishop, Fienberg, and Holland 1975), which adjusts for fitted zeros and non-estimable parameters. This is the default, and for complete tables gives the same results as the UNADJ option.
- ADJEST** computes the degrees of freedom as  $(n_c - n_p) - n_z$ , which adjusts for fitted zeros only. This gives a lower bound on the true degrees of freedom.

**PARM** computes parameter estimates, generates the “ANOVA,” “Parameter Estimates,” and “Predicted Values of Response Functions” tables, and includes the predicted standard errors in the “Predicted Values of Frequencies” and “Predicted Values of Probabilities” tables.

When you specify the PARM option, the algorithm used to obtain the maximum likelihood parameter estimates is weighted least squares on the IPF-predicted frequencies. This algorithm can be much faster than the Newton-Raphson algorithm used if you just specify the ML=NR option. In the resulting ANOVA table, the likelihood ratio is computed from the initial IPF fit while the degrees of freedom are generated from the WLS analysis; you can override this with

the DF= option. The initial response function, which the WLS method usually computes from the raw data, is computed from the IPF fitted frequencies.

If there are any zero marginals in the configurations that define the model, then predicted cell frequencies of zero will result and WLS cannot be used to compute the estimates. In this case, PROC CATMOD automatically changes the algorithm from ML=IPF to ML=NR and prints a note in the log.

**MISS=***keyword* | *value*

**MISSING=***keyword* | *value*

specifies whether a missing cell is treated as a sampling or structural zero.

Structural zero cells are removed from the analysis since their expected values are zero, while sampling zero cells may have nonzero expected value. For a single population, the missing cells are treated as structural zeros by default. For multiple populations, as long as some population has a nonzero count for a given population and response profile, the missing values are treated as sampling zeros by default.

The following table displays the available *keywords* and summarizes how PROC CATMOD treats missing values for one or more populations.

| MISSING=             | One Population                                 | Multiple Populations                           |
|----------------------|------------------------------------------------|------------------------------------------------|
| STRUCTURAL (default) | structural zeros                               | sampling zeros                                 |
| SAMP   SAMPLING      | sampling zeros                                 | sampling zeros                                 |
| <i>value</i>         | sets missing weights and cells to <i>value</i> | sets missing weights and cells to <i>value</i> |

**ZERO=***keyword* | *value*

**ZEROS=***keyword* | *value*

**ZEROES=***keyword* | *value*

specifies whether a non-missing cell with zero weight in the data set is treated as a sampling or structural zero.

Structural zero cells are removed from the analysis since their expected values are zero, while sampling zero cells have nonzero expected value. For a single population, the zero cells are treated as structural zeros by default; with multiple populations, as long as some population has a nonzero count for a given population and response profile, the zeros are treated as sampling zeros by default.

The following table displays the available *keywords* and summarizes how PROC CATMOD treats zeros for one or more populations.

| ZERO=                | One Population                    | Multiple Populations              |
|----------------------|-----------------------------------|-----------------------------------|
| STRUCTURAL (default) | structural zeros                  | sampling zeros                    |
| SAMP   SAMPLING      | sampling zeros                    | sampling zeros                    |
| <i>value</i>         | sets zero weights to <i>value</i> | sets zero weights to <i>value</i> |

---

## Details

---

### Missing Values

Observations with missing values for any variable listed in the MODEL or POPULATION statement are omitted from the analysis.

If the WEIGHT variable for an observation has a missing value, the observation is by default omitted from the analysis. You can modify this behavior by specifying the MISSING= option in the MODEL statement. The option MISSING=*value* sets all missing weights to *value* and all missing cells to *value*. The option MISSING=SAMPLING causes all missing cells in a contingency table to be treated as sampling zeros.

---

### Zero Frequencies

There are two types of zero cells in a contingency table: structural and sampling. A structural zero cell has an expected value of zero, while a sampling zero cell may have nonzero expected value.

For any log-linear model analysis, PROC CATMOD creates response profiles only for the observed profiles. Thus, for an analysis with one population (the usual case), the resulting contingency table does not contain zero or missing cells, which means that these cells are treated as structural zeros. However, for a WLS or ML=NR analysis on more than one population, a zero or missing cell in the body of the contingency table is treated as a sampling zero (as long as some population has a nonzero count for that profile).

If you want to treat zero frequencies as sampling zeros, you can specify the ZERO=SAMPLING and MISSING=SAMPLING options in the MODEL statement. Alternatively, you can include a statement in the DATA step creating your SAS data set that changes each zero to a very small number (such as 1E-20).

Refer to Bishop, Fienberg, and Holland (1975) and Christensen (1997) for a discussion of the issues. See Example 1.1 on page 7 for an illustration of a log-linear model analysis of data that contain both structural and sampling zeros.

---

### Computational Formulas

The algorithm used for iterative proportional fitting is described in Haberman (1972), Bishop, Fienberg, and Holland (1975), and Agresti (1990). To illustrate the method, consider the observed three-dimensional table  $\{n_{ijk}\}$  for the variables X, Y, and Z. The statements

```
proc catmod;
  model X*Y*Z = _response_ / ml=ipf;
  loglin X|Y|Z@2;
run;
```

request that PROC CATMOD use IPF to fit the hierarchical model

$$\log(m_{ijk}) = \mu + \lambda_i^X + \lambda_j^Y + \lambda_k^Z + \lambda_{ij}^{XY} + \lambda_{ik}^{XZ} + \lambda_{jk}^{YZ}$$

where  $\{m_{ijk}\}$  are the expected frequencies of the cells in the contingency table.

PROC CATMOD begins with a table of initial cell estimates  $\{\hat{m}_{ijk}^{(0)}\}$  that are produced by setting the  $n_{sz}$  structural zero cells to 0 and all other cells to  $n/(n_c - n_{sz})$ , where  $n$  is the total weight of the table and  $n_c$  is the total number of cells in the table. It then iteratively adjusts the estimates at step  $s-1$ ,  $\{\hat{m}_{ijk}^{(s-1)}\}$ , to the observed marginal tables specified in the model by stepping through the following three-stage process to produce the estimates at step  $s$ :

$$\hat{m}_{ijk}^{(s_1)} = \hat{m}_{ijk}^{(s-1)} \frac{n_{ij\cdot}}{\hat{m}_{ij\cdot}^{(s-1)}} \quad , \quad \hat{m}_{ijk}^{(s_2)} = \hat{m}_{ijk}^{(s_1)} \frac{n_{i\cdot k}}{\hat{m}_{i\cdot k}^{(s_1)}} \quad , \quad \hat{m}_{ijk}^{(s)} = \hat{m}_{ijk}^{(s_2)} \frac{n_{\cdot jk}}{\hat{m}_{\cdot jk}^{(s_2)}}$$

where the superscripts  $(s_1)$  and  $(s_2)$  indicate the two intermediate tables in the process, and the subscript “.” indicates summation over the missing subscript. The log likelihood  $l_s$  is estimated at each step  $s$  by

$$l_s = \sum_{i,j,k} n_{ijk} \log \left( \frac{\hat{m}_{ijk}^{(s)}}{n} \right)$$

When the function  $|(l_{s-1} - l_s)/l_{s-1}|$  is less than  $1E-8$ , the iterations terminate. You can change the comparison value with the EPSILON= option, and you can change the convergence criterion with the CONV= option. The option CONV=CELL uses the maximum absolute cell difference

$$\max_{i,j,k} |d_{ijk}| < 0.001 \quad \text{where} \quad d_{ijk} = \hat{m}_{ijk}^{(s-1)} - \hat{m}_{ijk}^{(s)}$$

as the criterion while the option CONV=MARGIN uses the maximum absolute difference of the margins

$$\max \left( \max_{i,j} |d_{ij\cdot}|, \max_{i,k} |d_{i\cdot k}|, \max_{j,k} |d_{\cdot jk}| \right) < 0.001$$

---

## Examples

---

### Example 1.1. Log-Linear Independence Model with Structural and Sampling Zeros

This example illustrates a log-linear model of independence, using data that contain structural zero frequencies as well as sampling (random) zero frequencies.

In a population of six squirrel monkeys, the joint distribution of genital display with respect to active or passive role was observed. The data are from Fienberg (1980, Table 8-2). The following DATA step creates the SAS data set Display:

```

title 'Behavior of Squirrel Monkeys';
data Display;
  input Active $ Passive $ wt @@;
  datalines;
r r .   r s 1   r t 5   r u 8   r v 9   r w 0
s r 29  s s .   s t 14  s u 46  s v 4   s w 0
t r 0   t s 0   t t .   t u 0   t v 0   t w 0
u r 2   u s 3   u t 1   u u .   u v 38  u w 2
v r 0   v s 0   v t 0   v u 0   v v .   v w 1
w r 9   w s 25  w t 4   w u 6   w v 13  w w .
;

```

In this data set, since a monkey cannot have both active and passive roles in an interaction, the values on the diagonal are structural zeros. Any off-diagonal zeros are sampling zeros. Since there are two types of zeros in this data set, missing values are placed on the diagonal to represent the structural zeros.

Suppose you're interested in studying the independence of the active and passive roles. Since the diagonal cells are structural zeros, you are actually fitting a *quasi*-independence model; refer to Agresti (1990) for more information. Since monkey 't' never takes the active role, the frequencies predicted by an independence model for these cells are zero; these cells are removed from the analysis with the WHERE clause.

The following statements produce the analysis that treats the missing values on the diagonals as structural zeros (since the MISSING=STRUCTURAL option is the default for one population). The ZERO=SAMPLING option treats the remaining zeros as sampling zeros.

```

proc catmod data=Display;
  weight wt;
  where Active ^= 't';
  model Active*Passive=_response_
    / ml=ipf(parm) zero=sampling;
  loglin Active Passive;
run;

```

**Output 1.1.1.** Data Summary and Population Profile

| Behavior of Squirrel Monkeys |                |                 |     |
|------------------------------|----------------|-----------------|-----|
| The CATMOD Procedure         |                |                 |     |
| Data Summary                 |                |                 |     |
| Response                     | Active*Passive | Response Levels | 25  |
| Weight Variable              | wt             | Populations     | 1   |
| Data Set                     | DISPLAY        | Total Frequency | 220 |
| Frequency Missing            | 0              | Observations    | 25  |
| Population Profiles          |                |                 |     |
| Sample                       | Sample Size    |                 |     |
| -----                        |                |                 |     |
| 1                            | 220            |                 |     |

The response profiles, shown in Output 1.1.2, include the off-diagonal zero cells because of the ZERO=SAMPLING option.

**Output 1.1.2.** Response Profiles

| Response Profiles |        |         |
|-------------------|--------|---------|
| Response          | Active | Passive |
| -----             |        |         |
| 1                 | r      | s       |
| 2                 | r      | t       |
| 3                 | r      | u       |
| 4                 | r      | v       |
| 5                 | r      | w       |
| 6                 | s      | r       |
| 7                 | s      | t       |
| 8                 | s      | u       |
| 9                 | s      | v       |
| 10                | s      | w       |
| 11                | u      | r       |
| 12                | u      | s       |
| 13                | u      | t       |
| 14                | u      | v       |
| 15                | u      | w       |
| 16                | v      | r       |
| 17                | v      | s       |
| 18                | v      | t       |
| 19                | v      | u       |
| 20                | v      | w       |
| 21                | w      | r       |
| 22                | w      | s       |
| 23                | w      | t       |
| 24                | w      | u       |
| 25                | w      | v       |

Because the PARM option is specified, a weighted least squares analysis is performed on the IPF fitted data and the \_Response\_ Matrix is displayed (Output 1.1.3); this table can be suppressed with the NORESPONSE option.



**Output 1.1.3.** \_Response\_ Matrix

| _Response_ Matrix |    |    |    |    |    |    |    |    |    |
|-------------------|----|----|----|----|----|----|----|----|----|
|                   | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
| 1                 | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  |
| 2                 | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  |
| 3                 | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  |
| 4                 | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  |
| 5                 | 1  | 0  | 0  | 0  | -1 | -1 | -1 | -1 | -1 |
| 6                 | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0  |
| 7                 | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  |
| 8                 | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  |
| 9                 | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 1  |
| 10                | 0  | 1  | 0  | 0  | -1 | -1 | -1 | -1 | -1 |
| 11                | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 0  |
| 12                | 0  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  |
| 13                | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  |
| 14                | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  |
| 15                | 0  | 0  | 1  | 0  | -1 | -1 | -1 | -1 | -1 |
| 16                | 0  | 0  | 0  | 1  | 1  | 0  | 0  | 0  | 0  |
| 17                | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  |
| 18                | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | 0  |
| 19                | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  |
| 20                | 0  | 0  | 0  | 1  | -1 | -1 | -1 | -1 | -1 |
| 21                | -1 | -1 | -1 | -1 | 1  | 0  | 0  | 0  | 0  |
| 22                | -1 | -1 | -1 | -1 | 0  | 1  | 0  | 0  | 0  |
| 23                | -1 | -1 | -1 | -1 | 0  | 0  | 1  | 0  | 0  |
| 24                | -1 | -1 | -1 | -1 | 0  | 0  | 0  | 1  | 0  |
| 25                | -1 | -1 | -1 | -1 | 0  | 0  | 0  | 0  | 1  |

The iteration history displays the value of the log likelihood and the convergence criterion for the IPF method as discussed in the “Computational Formulas” section on page 6.

**Output 1.1.4.** Iteration History

| Maximum Likelihood Analysis  |                   |                       |
|------------------------------|-------------------|-----------------------|
| Iteration                    | -2 Log Likelihood | Convergence Criterion |
| 0                            | 1201.5105         | 1.0000                |
| 1                            | 1198.5669         | 0.002450              |
| 2                            | 1198.5604         | 5.4468E-6             |
| 3                            | 1198.5603         | 7.702E-8              |
| 4                            | 1198.5603         | 1.6932E-9             |
| The IPF algorithm converged. |                   |                       |

The “Response Functions and Design Matrix” table (Output 1.1.5) is displayed when the PARM option is specified; this table can be suppressed with the NODESIGN option. The logits are computed from the IPF fitted values rather than the original data.

**Output 1.1.5.** Response Functions, Design Matrix

| Response Functions and Design Matrix |                 |                   |   |   |   |   |    |    |    |    |    |
|--------------------------------------|-----------------|-------------------|---|---|---|---|----|----|----|----|----|
| Sample                               | Function Number | Response Function | 1 | 2 | 3 | 4 | 5  | 6  | 7  | 8  | 9  |
| 1                                    | 1               | -0.97354          | 2 | 1 | 1 | 1 | 0  | 1  | 0  | 0  | -1 |
|                                      | 2               | -1.72504          | 2 | 1 | 1 | 1 | 0  | 0  | 1  | 0  | -1 |
|                                      | 3               | -0.52752          | 2 | 1 | 1 | 1 | 0  | 0  | 0  | 1  | -1 |
|                                      | 4               | -0.73927          | 2 | 1 | 1 | 1 | 0  | 0  | 0  | 0  | 0  |
|                                      | 5               | -3.56052          | 2 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -2 |
|                                      | 6               | 0.32061           | 1 | 2 | 1 | 1 | 1  | 0  | 0  | 0  | -1 |
|                                      | 7               | -0.29932          | 1 | 2 | 1 | 1 | 0  | 0  | 1  | 0  | -1 |
|                                      | 8               | 0.89820           | 1 | 2 | 1 | 1 | 0  | 0  | 0  | 1  | -1 |
|                                      | 9               | 0.68645           | 1 | 2 | 1 | 1 | 0  | 0  | 0  | 0  | 0  |
|                                      | 10              | -2.13480          | 1 | 2 | 1 | 1 | -1 | -1 | -1 | -1 | -2 |
|                                      | 11              | -0.24152          | 1 | 1 | 2 | 1 | 1  | 0  | 0  | 0  | -1 |
|                                      | 12              | -0.10995          | 1 | 1 | 2 | 1 | 0  | 1  | 0  | 0  | -1 |
|                                      | 13              | -0.86145          | 1 | 1 | 2 | 1 | 0  | 0  | 1  | 0  | -1 |
|                                      | 14              | 0.12432           | 1 | 1 | 2 | 1 | 0  | 0  | 0  | 0  | 0  |
|                                      | 15              | -2.69693          | 1 | 1 | 2 | 1 | -1 | -1 | -1 | -1 | -2 |
|                                      | 16              | -4.14787          | 1 | 1 | 1 | 2 | 1  | 0  | 0  | 0  | -1 |
|                                      | 17              | -4.01631          | 1 | 1 | 1 | 2 | 0  | 1  | 0  | 0  | -1 |
|                                      | 18              | -4.76780          | 1 | 1 | 1 | 2 | 0  | 0  | 1  | 0  | -1 |
|                                      | 19              | -3.57029          | 1 | 1 | 1 | 2 | 0  | 0  | 0  | 1  | -1 |
|                                      | 20              | -6.60328          | 1 | 1 | 1 | 2 | -1 | -1 | -1 | -1 | -2 |
|                                      | 21              | -0.36584          | 0 | 0 | 0 | 0 | 1  | 0  | 0  | 0  | -1 |
|                                      | 22              | -0.23427          | 0 | 0 | 0 | 0 | 0  | 1  | 0  | 0  | -1 |
|                                      | 23              | -0.98577          | 0 | 0 | 0 | 0 | 0  | 0  | 1  | 0  | -1 |
|                                      | 24              | 0.21175           | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 1  | -1 |

The ANOVA table and the parameter estimates are a by-product of running WLS on the IPF-fitted values. Note that the likelihood ratio chi-square (goodness-of-fit  $G^2$ ) in the ANOVA table is computed from the IPF routine; however, the degrees of freedom for  $G^2$  are calculated through WLS. If the PARM option was not specified, then only the likelihood ratio test would be displayed.

**Output 1.1.6.** ANOVA

| Maximum Likelihood Analysis of Variance |    |            |            |
|-----------------------------------------|----|------------|------------|
| Source                                  | DF | Chi-Square | Pr > ChiSq |
| Active                                  | 4  | 56.57      | <.0001     |
| Passive                                 | 5  | 47.94      | <.0001     |
| Likelihood Ratio                        | 15 | 135.17     | <.0001     |

**Output 1.1.7.** Parameter Estimates

| Analysis of Maximum Likelihood Estimates |           |          |                |            |            |
|------------------------------------------|-----------|----------|----------------|------------|------------|
| Effect                                   | Parameter | Estimate | Standard Error | Chi-Square | Pr > ChiSq |
| Active                                   | 1         | 0.00284  | 0.2660         | 0.00       | 0.9915     |
|                                          | 2         | 1.4286   | 0.2277         | 39.35      | <.0001     |
|                                          | 3         | 0.8664   | 0.2428         | 12.73      | 0.0004     |
|                                          | 4         | -3.0399  | 0.8031         | 14.33      | 0.0002     |
| Passive                                  | 5         | 0.3334   | 0.1739         | 3.67       | 0.0552     |
|                                          | 6         | 0.4650   | 0.1990         | 5.46       | 0.0195     |
|                                          | 7         | -0.2865  | 0.2019         | 2.01       | 0.1558     |
|                                          | 8         | 0.9110   | 0.1615         | 31.81      | <.0001     |
|                                          | 9         | 0.6992   | 0.1530         | 20.88      | <.0001     |

Since the PARM option is specified, the predicted response functions are computed from the WLS fit (this table is not shown here). For the IPF method, the “Maximum Likelihood Predicted Values for Frequencies” table is displayed by default; however, the predicted standard errors are not computed unless the PARM option is specified. The predicted standard errors are computed through WLS.

**Output 1.1.8.** Predicted Frequencies

| Maximum Likelihood Predicted Values for Frequencies |         |                    |                |                     |                |          |
|-----------------------------------------------------|---------|--------------------|----------------|---------------------|----------------|----------|
| Active                                              | Passive | -----Observed----- |                | -----Predicted----- |                | Residual |
|                                                     |         | Frequency          | Standard Error | Frequency           | Standard Error |          |
| r                                                   | s       | 1                  | 0.997725       | 5.259562            | 1.361573       | -4.25956 |
| r                                                   | t       | 5                  | 2.210512       | 2.48072             | 0.691065       | 2.51928  |
| r                                                   | u       | 8                  | 2.776525       | 8.21586             | 1.855129       | -0.21586 |
| r                                                   | v       | 9                  | 2.937996       | 6.648033            | 1.509317       | 2.351967 |
| r                                                   | w       | 0                  | 0              | 0.395767            | 0.240267       | -0.39577 |
| s                                                   | r       | 29                 | 5.017696       | 19.18631            | 3.147955       | 9.813693 |
| s                                                   | t       | 14                 | 3.620648       | 10.32189            | 2.16963        | 3.678112 |
| s                                                   | u       | 46                 | 6.031734       | 34.18491            | 4.428728       | 11.81509 |
| s                                                   | v       | 4                  | 1.981735       | 27.66143            | 3.722828       | -23.6614 |
| s                                                   | w       | 0                  | 0              | 1.646726            | 0.952727       | -1.64673 |
| u                                                   | r       | 2                  | 1.407771       | 10.93611            | 2.12318        | -8.93611 |
| u                                                   | s       | 3                  | 1.720201       | 12.47391            | 2.554314       | -9.47391 |
| u                                                   | t       | 1                  | 0.997725       | 5.88343             | 1.380627       | -4.88343 |
| u                                                   | v       | 38                 | 5.606814       | 15.76689            | 2.684647       | 22.23311 |
| u                                                   | w       | 2                  | 1.407771       | 0.938627            | 0.551631       | 1.061373 |
| v                                                   | r       | 0                  | 0              | 0.219965            | 0.22182        | -0.21997 |
| v                                                   | s       | 0                  | 0              | 0.250896            | 0.253756       | -0.2509  |
| v                                                   | t       | 0                  | 0              | 0.118337            | 0.120336       | -0.11834 |
| v                                                   | u       | 0                  | 0              | 0.39192             | 0.393325       | -0.39192 |
| v                                                   | w       | 1                  | 0.997725       | 0.018879            | 0.021731       | 0.981121 |
| w                                                   | r       | 9                  | 2.937996       | 9.657617            | 1.808652       | -0.65762 |
| w                                                   | s       | 25                 | 4.707344       | 11.01564            | 2.275041       | 13.98436 |
| w                                                   | t       | 4                  | 1.981735       | 5.195624            | 1.18445        | -1.19562 |
| w                                                   | u       | 6                  | 2.415857       | 17.20731            | 2.772074       | -11.2073 |
| w                                                   | v       | 13                 | 3.497402       | 13.92365            | 2.241575       | -0.92365 |

The model of independence does not fit since the likelihood ratio test for the interaction is significant. In other words, active and passive behaviors of the squirrel monkeys are dependent behavior roles.

Results from using the ML=NR option instead of the ML=IPF option are very similar, since these are just two different algorithms for maximum likelihood estimation. Due to the sampling zeros in the table, use of the WLS method is not recommended.

## Example 1.2. Identifying an Inappropriate Model

Suppose you have the following data, and you want to use IPF to fit the “no three-factor effect” model:

```
data pathological;
  input X Y Z count @@;
  datalines;
  1 1 1 0 1 1 2 15
  1 2 1 15 1 2 2 24
```

```

2 1 1 17 2 1 2 14
2 2 1 16 2 2 2 0
;

```

For this model, it turns out that  $n_{111} = n_{222} = 0$  implies the cell frequency estimates  $\hat{m}_{111} = \hat{m}_{222} = 0$ . This means that the table has only 6 degrees of freedom (non-structural zero cells) available, while the model requires 7 degrees of freedom (one degree for each of the mean, X, Y, Z, XY, XZ, and YZ). Therefore, in order to analyze the data appropriately, these two cells should be dropped from the table and treated as structural zeros, and the model should be reduced. You may be able to identify cases like this with PROC CATMOD by observing convergence problems or by noting that the predicted frequency of a cell seems to be converging to zero:

```

proc catmod data=pathological;
  weight count;
  model X*Y*Z=_response_ / ml=ipf zero=sampling;
  loglin X|Y|Z@2;
run;

```

**Output 1.2.1.** ML=IPF with ZERO=SAMPLING

**WARNING: The IPF algorithm failed to converge.**

When the sampling zeros are replaced by structural zeros, the adjusted degrees of freedom for the likelihood ratio are negative; this is another signal that the model is inappropriate for the data:

```

proc catmod data=pathological;
  weight count;
  model X*Y*Z=_response_ / ml=ipf;
  loglin X|Y|Z@2;
run;

```

**Output 1.2.2.** ML=IPF with Structural Zeros

**The IPF algorithm converged.**

**Maximum Likelihood Analysis of Variance**

| Source           | DF | Chi-Square | Pr > ChiSq |
|------------------|----|------------|------------|
| Likelihood Ratio | 0  | .          | .          |

**WARNING: Negative adjusted degrees of freedom were calculated for the Likelihood Ratio test. The model may be inappropriate.**

When using ML=NR, you receive a note about having redundant parameters in the model, and you may get messages about having infinite parameters:

```

proc catmod data=pathological;
  weight count;
  model X*Y*Z=_response_ / ml=nr;
  loglin X|Y|Z@2;
run;

```

**Output 1.2.3.** ML=NR with Structural Zeros

| Maximum likelihood computations converged.                                               |    |            |            |
|------------------------------------------------------------------------------------------|----|------------|------------|
| Maximum Likelihood Analysis of Variance                                                  |    |            |            |
| Source                                                                                   | DF | Chi-Square | Pr > ChiSq |
| X                                                                                        | 1  | 0.00       | 0.9931     |
| Y                                                                                        | 1  | 0.73       | 0.3930     |
| X*Y                                                                                      | 1  | 1.23       | 0.2682     |
| Z                                                                                        | 1  | 0.32       | 0.5723     |
| X*Z                                                                                      | 1  | 1.85       | 0.1739     |
| Y*Z                                                                                      | 0* | .          | .          |
| Likelihood Ratio                                                                         | 0  | .          | .          |
| NOTE: Effects marked with '*' contain one or more<br>redundant or restricted parameters. |    |            |            |

This example is discussed further in Bishop, Fienberg, and Holland (1975, p. 115), Agresti (1990, p. 245), and Christensen (1997, p. 292).

---

## References

- Agresti, A. (1990), *Categorical Data Analysis*, New York: John Wiley & Sons, Inc.
- Bishop, Y.M.M., Fienberg, S.E., and Holland, P.W. (1975), *Discrete Multivariate Analysis: Theory and Practice*, Cambridge, MA: The MIT Press.
- Christensen, R. (1997), *Log-Linear Models and Logistic Regression*, New York: Springer-Verlag.
- Haberman, S.J. (1972), "Log-Linear Fit for Contingency Tables," *Applied Statistics*, 21, 218–225.
- Haslett, S. (1990), "Degrees of Freedom and Parameter Estimability in Hierarchical Models for Sparse Complete Contingency Tables," *Computational Statistics and Data Analysis*, 9, 179–195.

# Chapter 2

## The FACTOR Procedure

### Chapter Table of Contents

---

|                                                                                    |    |
|------------------------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                                          | 17 |
| <b>SYNTAX</b> . . . . .                                                            | 17 |
| PROC FACTOR Statement . . . . .                                                    | 17 |
| <b>DETAILS</b> . . . . .                                                           | 21 |
| Output Data Sets . . . . .                                                         | 21 |
| Confidence Intervals and the Saliency of Factor Loadings . . . . .                 | 21 |
| Simplicity Functions for Rotations . . . . .                                       | 23 |
| <b>EXAMPLE</b> . . . . .                                                           | 24 |
| Example 2.1 Using Confidence Intervals to Locate Salient Factor Loadings . . . . . | 24 |
| <b>REFERENCES</b> . . . . .                                                        | 29 |



## Chapter 2

# The FACTOR Procedure

---

### Overview

Enhancements to the FACTOR procedure include:

- the generalized Crawford-Ferguson family of rotations (Jennrich 1973) using the ROTATE= option, including the direct oblimin, the quartimin, and many other specialized orthogonal and oblique rotations
- control of the number of rotation cycles and the criterion for rotational convergence using the RITER= and the RCONVERGE= options
- standard error estimates for the unrotated and various rotated solutions using the SE option under the maximum likelihood estimation (Archer and Jennrich 1973, Hayashi and Yung 1999, Jennrich 1973)
- confidence intervals and coverage displays for the factor loadings, the factor correlations, and the factor structure loadings using the CI= option under the maximum likelihood estimation
- control of the percentage coverage of the confidence interval using the ALPHA= option

---

### Syntax

New options for the PROC FACTOR statement are added.

---

### PROC FACTOR Statement

**PROC FACTOR** < *options* > ;

The following new or updated options are available:

**ALPHA=***p*

specifies the level of confidence  $1-p$  for interval construction. By default,  $p = 0.05$ , corresponding to  $1-p = 95\%$  confidence intervals. If  $p$  is greater than one, it is interpreted as a percentage and divided by 100. Because the coverage probability is not controlled simultaneously, you may consider supplying a nonconventional  $p$  using methods such as Bonferroni adjustment.

**COVER** <=*p*>

**CI** <=*p*>

computes the confidence intervals and optionally specifies the value of factor loading



for coverage detection. By default,  $p = 0$ . The specified value is represented by an asterisk '\*' in the coverage display. This is useful for determining the salience of loadings. For example, if COVER=.4, a display '0\*[ ]' indicates that the entire confidence interval is above 0.4, implying strong evidence for the salience of the loading. See the section "Confidence Intervals and the Salience of Factor Loadings" on page 21 for more details.

**HKPOWER= $p$**

**HKP= $p$**

specifies the power of the square roots of the eigenvalues used to rescale the eigenvectors for Harris-Kaiser (ROTATE=HK) rotation, assuming that the factors are extracted by the principal factor method. If the principal factor method is not used for factor extraction, the eigenvectors are replaced by the normalized columns of the unrotated factor matrix, and the eigenvalues replaced by the column normalizing constants. HKPOWER= values between 0.0 and 1.0 are reasonable. The default value is 0.0, yielding the independent cluster solution, in which each variable tends to have a large loading on only one factor. An HKPOWER= value of 1.0 is equivalent to an orthogonal rotation, with the varimax rotation as the default. You can also specify the HKPOWER= option with ROTATE=QUARTIMAX, ROTATE=BIQUARTIMAX, ROTATE=EQUAMAX, or ROTATE=ORTHOMAX, and so on. The only restriction is that the Harris-Kaiser rotation must be associated with an orthogonal rotation.

**PREROTATE=*name***

**PRE=*name***

specifies the prerotation method for the option ROTATE=PROMAX. Any rotation method other than PROMAX or PROCRUSTES can be used. See the ROTATE= option for the available prerotation methods. The default is PREROTATE=VARIMAX. If a previously rotated pattern is read using the option METHOD=PATTERN, you should specify the PREROTATE=NONE option.

**RCONVERGE= $p$**

**RCONV= $p$**

specifies the convergence criterion for rotation cycles. Rotation stops when the scaled change of the simplicity function value is less than the RCONVERGE= value. The default convergence criterion is

$$|f_{new} - f_{old}|/K < \epsilon$$

where  $f_{new}$  and  $f_{old}$  are simplicity function values of the current cycle and the previous cycle, respectively,  $K = \max(1, |f_{old}|)$  is a scaling factor, and  $\epsilon$  is 1E-9 by default and is modified by the RCONVERGE= value.

**RTITER= $n$**

specifies the maximum number of cycles for factor rotation. Except for promax and Procrustes, you can use the RTITER= option with all rotation methods. The default is the maximum between 100 and ten times of the number of variables.

**ROTATE=*name***

**R=*name***

specifies the rotation method. The default is ROTATE=NONE.

Valid *names* for orthogonal rotations are as follows:

BIQUARTIMAX | BIQMAX specifies orthogonal biquartimax rotation. This corresponds to the specification ROTATE=ORTHOMAX(.5).

EQUAMAX | E specifies orthogonal equamax rotation. This corresponds to the specification ROTATE=ORTHOMAX with GAMMA=*number of factors*/2.

FACTORPARSIMAX | FPA specifies orthogonal factor parsimax rotation. This corresponds to the specification ROTATE=ORTHOMAX with GAMMA=*number of variables*.

NONE | N specifies that no rotation be performed, leaving the original orthogonal solution.

ORTHCF(*p1,p2*) | ORCF(*p1,p2*) specifies the orthogonal Crawford-Ferguson rotation with the weights *p1* and *p2* for variable and factor parsimony, respectively. See the definitions of weights in the section “Simplicity Functions for Rotations” on page 23.

ORTHGENCF(*p1,p2,p3,p4*) | ORGENCF(*p1,p2,p3,p4*) specifies the orthogonal generalized Crawford-Ferguson rotation with the four weights *p1*, *p2*, *p3*, and *p4*. See the definitions of weights in the section “Simplicity Functions for Rotations” on page 23.

ORTHOMAX<(p)> | ORMAX<(p)> specifies the orthomax rotation. If ROTATE=ORTHOMAX is used, the orthomax weight is specified by the GAMMA= option. You can also specify the GAMMA= value in the parentheses of ROTATE=ORTHOMAX(*p*). See the definition of the orthomax weight in the section “Simplicity Functions for Rotations” on page 23.

PARSIMAX | PA specifies orthogonal parsimax rotation. This corresponds to the specification ROTATE=ORTHOMAX with

$$\text{GAMMA} = \frac{nvar \times (nfact - 1)}{nvar + nfact - 2}$$

where *nvar* is the number of variables, and *nfact* is the number of factors.

QUARTIMAX | QMAX | Q specifies orthogonal quartimax rotation. This corresponds to the specification ROTATE=ORTHOMAX(0).

VARIMAX | V specifies orthogonal varimax rotation. This corresponds to the specification ROTATE=ORTHOMAX with GAMMA=1.

Valid *names* for oblique rotations are as follows:

BIQUARTIMIN | BIQMIN specifies biquartimin rotation. It corresponds to the specification ROTATE=OBLIMIN(.5) or ROTATE=OBLIMIN with TAU=.5.

COVARIMIN | CVMIN specifies covarimin rotation. It corresponds to the specification ROTATE=OBLIMIN(1) or ROTATE=OBLIMIN with TAU=1.

HK<(p)> | H<(p)> specifies Harris-Kaiser case II orthoblique rotation. When specifying this option, you can use the HKPOWER= option to set the power of the square roots of the eigenvalues by which the eigenvectors are scaled, assuming that the factors are extracted by the principal factor method. For other extraction methods, the unrotated factor pattern is column normalized. The power is then applied to the column normalizing constants, instead of the eigenvalues. You can also use ROTATE=HK(p), with  $p$  representing the HKPOWER= value. The default associated orthogonal rotation with ROTATE=HK is the varimax rotation without Kaiser normalization. You may associate the Harris-Kaiser with other orthogonal rotations using the ROTATE= option together with the HKPOWER= option.

OBBIQUARTIMAX | OBIQMAX specifies oblique biquartimax rotation.

OBEQUAMAX | OE specifies oblique equamax rotation.

OBFACTORPARSIMAX | OFPA specifies oblique factor parsimax rotation.

OBLICF( $p1, p2$ ) | OBCF( $p1, p2$ ) specifies the oblique Crawford-Ferguson rotation with the weights  $p1$  and  $p2$  for variable and factor parsimony, respectively. See the definitions of weights in the section “Simplicity Functions for Rotations” on page 23.

OBLIGENCF( $p1, p2, p3, p4$ ) | OBGENCF( $p1, p2, p3, p4$ ) specifies the oblique generalized Crawford-Ferguson rotation with the four weights  $p1$ ,  $p2$ ,  $p3$ , and  $p4$ . See the definitions of weights in the section “Simplicity Functions for Rotations” on page 23.

OBLIMIN<(p)> | OBMIN<(p)> specifies the oblimin rotation. If ROTATE=OBLIMIN is used, the oblimin weight is specified by the TAU= option. Alternatively, ROTATE=OBLIMIN( $p$ ) specifies  $p$  as the TAU= value. See the definition of the oblimin weight in the section “Simplicity Functions for Rotations” on page 23.

OBPARSIMAX | OPA specifies oblique parsimax rotation.

OBQUARTIMAX | OQMAX specifies oblique quartimax rotation. This is the same as the QUARTIMIN method.

OBVARIMAX | OV specifies oblique varimax rotation.

PROCRUSTES specifies oblique Procrustes rotation with the target pattern provided by the TARGET= data set. The unrestricted least squares method is used with factors scaled to unit variance after rotation.

PROMAX<(p)> | P<(p)> specifies oblique promax rotation. You can use the PREROTATE= option to set the desirable prerotation method, orthogonal or oblique. When using with ROTATE=PROMAX, the POWER= option lets you specify the power for forming the target. You can also use ROTATE=PROMAX( $p$ ), where  $p$  represents the POWER= value.

QUARTIMIN | QMIN specifies quartimin rotation. It is the same as the oblique quartimax method. It also corresponds to the specification ROTATE=OBLIMIN(0) or ROTATE=OBLIMIN with TAU=0.

**SE  
STDERR**

computes standard errors for various classes of unrotated and rotated solutions under the maximum likelihood estimation.

---

## Details

---

### Output Data Sets

#### *The OUTSTAT= Data Set*

If standard error estimates are available, the output data set contains the following new observations:

| <u>_TYPE_</u> | <u>Contents</u>                                                                                                         |
|---------------|-------------------------------------------------------------------------------------------------------------------------|
| SE_PREFC      | standard error estimates for pre-rotated interfactor correlations. The _NAME_ variable contains the name of the factor. |
| SE_PREPA      | standard error estimates for the pre-rotated loadings. The _NAME_ variable contains the name of the factor.             |
| SE_PREST      | standard error estimates for pre-rotated structure loadings. The _NAME_ variable contains the name of the factor.       |
| SE_FCORR      | standard error estimates for interfactor correlations. The _NAME_ variable contains the name of the factor.             |
| SE_PAT        | standard error estimates for the rotated loadings. The _NAME_ variable contains the name of the factor.                 |
| SE_STRUC      | standard error estimates for structure loadings. The _NAME_ variable contains the name of the factor.                   |

---

## Confidence Intervals and the Salience of Factor Loadings

The traditional approach to determining salient loadings (loadings that are considered large in absolute values) employs rules-of-thumb such as 0.3 or 0.4. However, this does not utilize the statistical evidence efficiently. The asymptotic normality of the distribution of factor loadings enables you to construct confidence intervals to gauge the salience of factor loadings. To guarantee the range-respecting properties of confidence intervals, a transformation procedure such as in CEFA (Browne, Cudeck, Tateneni, and Mels 1998) is used. For example, because the orthogonal rotated factor loading  $\theta$  must be bounded between  $-1$  and  $+1$ , the Fisher transformation

$$\varphi = \frac{1}{2} \log\left(\frac{1 + \theta}{1 - \theta}\right)$$

is employed so that  $\varphi$  is an unbounded parameter. Assuming the asymptotic normality of  $\hat{\varphi}$ , a symmetric confidence interval for  $\varphi$  is constructed. Then, a back-transformation on the confidence limits yields an asymmetric confidence interval for  $\theta$ . Applying the results of Browne (1982), a  $(1-\alpha)100\%$  confidence interval for the orthogonal factor loading  $\theta$  is

$$(\hat{\theta}_l = \frac{a/b - 1}{a/b + 1}, \hat{\theta}_u = \frac{a \times b - 1}{a \times b + 1})$$

where

$$a = \frac{1 + \hat{\theta}}{1 - \hat{\theta}}, \quad b = \exp(z_{\alpha/2} \times \frac{2\hat{\sigma}}{1 - \hat{\theta}^2})$$

and  $\hat{\theta}$  is the estimated factor loading,  $\hat{\sigma}$  is the standard error estimate of the factor loading, and  $z_{\alpha/2}$  is the  $(1 - \alpha/2)100$  percentile point of a standard normal distribution.

Once the confidence limits are constructed, you can use the corresponding coverage displays for determining the salience of the variable-factor relationship. In a coverage display, the COVER= value is represented by an asterisk '\*'. The following table summarizes the various displays and their interpretations.

**Table 2.1.** Interpretations of the Coverage Displays

| Positive Estimate | Negative Estimate | COVER=0 specified | Interpretation                                                                                                                                                                                                                                               |
|-------------------|-------------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [0]*              | *[0]              |                   | The estimate is not significantly different from zero and the CI covers a region of values that are smaller in magnitude than the COVER= value. This is strong statistical evidence for the non-salience of the variable-factor relationship.                |
| 0[ ]*             | *[ ]0             |                   | The estimate is significantly different from zero but the CI covers a region of values that are smaller in magnitude than the COVER= value. This is strong statistical evidence for the non-salience of the variable-factor relationship.                    |
| [0*]              | [*0]              | [0]               | The estimate is not significantly different from zero or the COVER= value. The population value might have been larger or smaller in magnitude than the COVER= value. There is no statistical evidence for the salience of the variable-factor relationship. |
| 0[*]              | [*]0              |                   | The estimate is significantly different from zero but not from the COVER= value. This is marginal statistical evidence for the salience of the variable-factor relationship.                                                                                 |
| 0*[ ]             | [ ]*0             | 0[ ] or [ ]0      | The estimate is significantly different from zero and the CI covers a region of values that are larger in magnitude than the COVER= value. This is strong statistical evidence for the salience of the variable-factor relationship.                         |

See Example 2.1 on page 24 for an illustration of the use of confidence intervals for interpreting factors.

---

## Simplicity Functions for Rotations

To rotate a factor pattern is to apply a non-singular linear transformation to the unrotated factor pattern matrix. To arrive at an optimal transformation you must define a so-called simplicity function for assessing the optimal point. For the promax or the Procrustean transformation, the simplicity function is defined as the sum of squared differences between the rotated factor pattern and the target matrix. Thus, the solution of the optimal transformation is easily obtained by the familiar least-squares method.

For the class of the generalized Crawford-Ferguson family (Jennrich 1973), the simplicity function being optimized is

$$f = k_1 Z + k_2 H + k_3 V + k_4 Q$$

where

$$Z = \left( \sum_j \sum_i b_{ij}^2 \right)^2, \quad H = \sum_i \left( \sum_j b_{ij}^2 \right)^2$$

$$V = \sum_j \left( \sum_i b_{ij}^2 \right)^2, \quad Q = \sum_j \sum_i b_{ij}^4$$

$k_1, k_2, k_3$ , and  $k_4$  are constants, and  $b_{ij}$  represents an element of the rotated pattern matrix. Except for specialized research purposes, it is rare in practice to use this simplicity function for rotation. However, it reduces to many well-known classes and special cases of rotations. One of these is the Crawford-Ferguson family (Crawford and Ferguson 1970), which minimizes

$$f_{cf} = c_1(H - Q) + c_2(V - Q)$$

where  $c_1$  and  $c_2$  are constants and  $(H - Q)$  represents variable (row) parsimony and  $(V - Q)$  represents factor (column) parsimony. Therefore, the relative importance of the variable and the factor parsimony is adjusted via the constants  $c_1$  and  $c_2$ . The orthomax class (Carroll, see Harman 1976) maximizes the function

$$f_{or} = pQ - \gamma V$$

where  $\gamma$  is the orthomax weight and is usually between 0 and the number of variables  $p$ . The oblimin class minimizes the function

$$f_{ob} = p(H - Q) - \tau(Z - V)$$

where  $\tau$  is the oblimin weight and is usually between 0 and the number of variables  $p$ .

All the above definitions are for rotations without row normalization. For rotations with Kaiser normalization the definition of  $b_{ij}$  is replaced by  $b_{ij}/h_i$ , where  $h_i$  is the communality of variable  $i$ .

---

## Example

---

### Example 2.1. Using Confidence Intervals to Locate Salient Factor Loadings

This example illustrates how you can utilize the standard errors and confidence intervals to understand the pattern of factor loadings under the maximum likelihood estimation. There are nine tests and you want a three-factor solution for a correlation matrix based on 200 observations. You apply quartimin rotation with (default) Kaiser normalization. You define loadings with magnitudes greater than 0.45 to be salient and use 90% confidence intervals to judge the salience:

```

data test(type=corr);
  title 'Quartimin-Rotated Factor Solution with Standard Errors';
  input _name_ $ test1-test9;
  _type_ = 'corr';
datalines;
Test1      1 .561 .602 .290 .404 .328 .367 .179 -.268
Test2    .561  1 .743 .414 .526 .442 .523 .289 -.399
Test3    .602 .743  1 .286 .343 .361 .679 .456 -.532
Test4    .290 .414 .286  1 .677 .446 .412 .400 -.491
Test5    .404 .526 .343 .677  1 .584 .408 .299 -.466
Test6    .328 .442 .361 .446 .584  1 .333 .178 -.306
Test7    .367 .523 .679 .412 .408 .333  1 .711 -.760
Test8    .179 .289 .456 .400 .299 .178 .711  1 -.725
Test9   -.268 -.399 -.532 -.491 -.466 -.306 -.760 -.725  1
;
proc factor data=test method=ml reorder rotate=quartimin
  nobs=200 n=3 se cover=.45 alpha=.1;
  title2 'A nine-variable-three-factor example';
run;

```

Output 2.1.1. Quartimin-Rotated Factor Solution with Standard Errors

| Quartimin-Rotated Factor Solution with Standard Errors |         |         |         |
|--------------------------------------------------------|---------|---------|---------|
| A nine-variable-three-factor example                   |         |         |         |
| The FACTOR Procedure                                   |         |         |         |
| Rotation Method: Quartimin                             |         |         |         |
| Inter-Factor Correlations                              |         |         |         |
| With 90% confidence limits                             |         |         |         |
| Estimate/StdErr/LowerCL/UpperCL                        |         |         |         |
|                                                        | Factor1 | Factor2 | Factor3 |
| Factor1                                                | 1.00000 | 0.41283 | 0.38304 |
|                                                        | 0.00000 | 0.06267 | 0.06060 |
|                                                        | .       | 0.30475 | 0.27919 |
|                                                        | .       | 0.51041 | 0.47804 |
| Factor2                                                | 0.41283 | 1.00000 | 0.47006 |
|                                                        | 0.06267 | 0.00000 | 0.05116 |
|                                                        | 0.30475 | .       | 0.38177 |
|                                                        | 0.51041 | .       | 0.54986 |
| Factor3                                                | 0.38304 | 0.47006 | 1.00000 |
|                                                        | 0.06060 | 0.05116 | 0.00000 |
|                                                        | 0.27919 | 0.38177 | .       |
|                                                        | 0.47804 | 0.54986 | .       |

After the quartimin rotation, the correlation matrix for factors is shown in Output 2.1.1. The factors are medium to highly correlated. The confidence intervals seem to be very wide, suggesting that the estimation of factor correlations may not be very accurate for this sample size. For example, the 90% confidence interval for the correlation between Factor1 and Factor2 is (0.30, 0.51), a range of 0.21. You may need a larger sample to get a narrower interval, or a better estimation.



**Output 2.1.2.** Interpretations of Factors Using Rotated Factor Pattern

| A nine-variable-three-factor example                          |          |          |          |
|---------------------------------------------------------------|----------|----------|----------|
| The FACTOR Procedure                                          |          |          |          |
| Rotation Method: Quartimin                                    |          |          |          |
| Rotated Factor Pattern (Standardized Regression Coefficients) |          |          |          |
| With 90% confidence limits; Cover  *  = 0.45?                 |          |          |          |
| Estimate/StdErr/LowerCL/UpperCL/Coverage Display              |          |          |          |
|                                                               | Factor1  | Factor2  | Factor3  |
| test8                                                         | 0.86810  | -0.05045 | 0.00114  |
|                                                               | 0.03282  | 0.03185  | 0.03087  |
|                                                               | 0.80271  | -0.10265 | -0.04959 |
|                                                               | 0.91286  | 0.00204  | 0.05187  |
|                                                               | 0*[]     | *[0]     | [0]*     |
| test7                                                         | 0.73204  | 0.27296  | 0.01098  |
|                                                               | 0.04434  | 0.05292  | 0.03838  |
|                                                               | 0.65040  | 0.18390  | -0.05211 |
|                                                               | 0.79697  | 0.35758  | 0.07399  |
|                                                               | 0*[]     | 0[]*     | [0]*     |
| test9                                                         | -0.79654 | -0.01230 | -0.17307 |
|                                                               | 0.03948  | 0.04225  | 0.04420  |
|                                                               | -0.85291 | -0.08163 | -0.24472 |
|                                                               | -0.72180 | 0.05715  | -0.09955 |
|                                                               | []*0     | *[0]     | *[]0     |
| test3                                                         | 0.27715  | 0.91156  | -0.19727 |
|                                                               | 0.05489  | 0.04877  | 0.02981  |
|                                                               | 0.18464  | 0.78650  | -0.24577 |
|                                                               | 0.36478  | 0.96481  | -0.14778 |
|                                                               | 0[]*     | 0*[]     | *[]0     |
| test2                                                         | 0.01063  | 0.71540  | 0.20500  |
|                                                               | 0.05060  | 0.05148  | 0.05496  |
|                                                               | -0.07248 | 0.61982  | 0.11310  |
|                                                               | 0.09359  | 0.79007  | 0.29342  |
|                                                               | [0]*     | 0*[]     | 0[]*     |
| test1                                                         | -0.07356 | 0.63815  | 0.13983  |
|                                                               | 0.04245  | 0.05380  | 0.05597  |
|                                                               | -0.14292 | 0.54114  | 0.04682  |
|                                                               | -0.00348 | 0.71839  | 0.23044  |
|                                                               | *[]0     | 0*[]     | 0[]*     |
| test5                                                         | 0.00863  | 0.03234  | 0.91282  |
|                                                               | 0.04394  | 0.04387  | 0.04509  |
|                                                               | -0.06356 | -0.03986 | 0.80030  |
|                                                               | 0.08073  | 0.10421  | 0.96323  |
|                                                               | [0]*     | [0]*     | 0*[]     |
| test4                                                         | 0.22357  | -0.07576 | 0.67925  |
|                                                               | 0.05956  | 0.03640  | 0.05434  |
|                                                               | 0.12366  | -0.13528 | 0.57955  |
|                                                               | 0.31900  | -0.01569 | 0.75891  |
|                                                               | 0[]*     | *[]0     | 0*[]     |
| test6                                                         | -0.04295 | 0.21911  | 0.53183  |
|                                                               | 0.05114  | 0.07481  | 0.06905  |
|                                                               | -0.12656 | 0.09319  | 0.40893  |
|                                                               | 0.04127  | 0.33813  | 0.63578  |
|                                                               | *[0]     | 0[]*     | 0[*]     |

The coverage displays in Output 2.1.2 show that **Test8**, **Test7**, and **Test9** have salient relationships with **Factor1**. The coverage displays are either '0\*[ ]' or '[ ]\*0', indicating that the entire 90% confidence intervals for the corresponding loadings are beyond the salience value at 0.45. On the other hand, the coverage display for **Test3** on **Factor1** is '0[ ]\*'. This indicates that even though the loading estimate is significantly larger than zero, it is not large enough to be salient. Similarly, **Test3**, **Test2**, and **Test1** have salient relationships with **Factor2**, while **Test5** and **Test4** have salient relationships with **Factor3**. For **Test6**, its relationship with **Factor3** is a little bit ambiguous; the 90% confidence interval covers approximately values between 0.40 and 0.64. This means that the population value might have been smaller or larger than 0.45. It is marginal evidence for a salient relationship.

**Output 2.1.3.** Interpretations of Factors Using Factor Structure

| A nine-variable-three-factor example             |          |          |          |
|--------------------------------------------------|----------|----------|----------|
| The FACTOR Procedure                             |          |          |          |
| Rotation Method: Quartimin                       |          |          |          |
| Factor Structure (Correlations)                  |          |          |          |
| With 90% confidence limits; Cover  *  = 0.45?    |          |          |          |
| Estimate/StdErr/LowerCL/UpperCL/Coverage Display |          |          |          |
|                                                  | Factor1  | Factor2  | Factor3  |
| test8                                            | 0.84771  | 0.30847  | 0.30994  |
|                                                  | 0.02871  | 0.06593  | 0.06263  |
|                                                  | 0.79324  | 0.19641  | 0.20363  |
|                                                  | 0.88872  | 0.41257  | 0.40904  |
|                                                  | 0*[]     | 0[]*     | 0[]*     |
| test7                                            | 0.84894  | 0.58033  | 0.41970  |
|                                                  | 0.02688  | 0.05265  | 0.06060  |
|                                                  | 0.79834  | 0.48721  | 0.31523  |
|                                                  | 0.88764  | 0.66041  | 0.51412  |
|                                                  | 0*[]     | 0*[]     | 0[*]     |
| test9                                            | -0.86791 | -0.42248 | -0.48396 |
|                                                  | 0.02522  | 0.06187  | 0.05504  |
|                                                  | -0.90381 | -0.51873 | -0.56921 |
|                                                  | -0.81987 | -0.31567 | -0.38841 |
|                                                  | []*0     | []*0     | []*0     |
| test3                                            | 0.57790  | 0.93325  | 0.33738  |
|                                                  | 0.05069  | 0.02953  | 0.06779  |
|                                                  | 0.48853  | 0.86340  | 0.22157  |
|                                                  | 0.65528  | 0.96799  | 0.44380  |
|                                                  | 0*[]     | 0*[]     | 0[]*     |
| test2                                            | 0.38449  | 0.81615  | 0.54535  |
|                                                  | 0.06143  | 0.03106  | 0.05456  |
|                                                  | 0.27914  | 0.75829  | 0.44946  |
|                                                  | 0.48070  | 0.86126  | 0.62883  |
|                                                  | 0[*]     | 0*[]     | 0[*]     |
| test1                                            | 0.24345  | 0.67351  | 0.41162  |
|                                                  | 0.06864  | 0.04284  | 0.05995  |
|                                                  | 0.12771  | 0.59680  | 0.30846  |
|                                                  | 0.35264  | 0.73802  | 0.50522  |
|                                                  | 0[]*     | 0*[]     | 0[*]     |
| test5                                            | 0.37163  | 0.46498  | 0.93132  |
|                                                  | 0.06092  | 0.04979  | 0.03277  |
|                                                  | 0.26739  | 0.37923  | 0.85159  |
|                                                  | 0.46727  | 0.54282  | 0.96894  |
|                                                  | 0[*]     | 0[*]     | 0*[]     |
| test4                                            | 0.45248  | 0.33583  | 0.72927  |
|                                                  | 0.05876  | 0.06289  | 0.04061  |
|                                                  | 0.35072  | 0.22867  | 0.65527  |
|                                                  | 0.54367  | 0.43494  | 0.78941  |
|                                                  | 0[*]     | 0[]*     | 0*[]     |
| test6                                            | 0.25122  | 0.45137  | 0.61837  |
|                                                  | 0.07140  | 0.05858  | 0.05051  |
|                                                  | 0.13061  | 0.34997  | 0.52833  |
|                                                  | 0.36450  | 0.54232  | 0.69465  |
|                                                  | 0[]*     | 0[*]     | 0*[]     |

For oblique factor solutions, some researchers prefer to examine the factor structure loadings, which represent correlations, for determining salient relationships. In Output 2.1.3, the factor structure loadings and the associated standard error estimates and coverage displays are shown. The interpretations based on the factor structure matrix do not change much except for **Test3** and **Test9**. **Test9** now has a salient correlation with **Factor3**. For **Test3**, it has salient correlations with both **Factor1** and **Factor2**. Fortunately, there are still tests that only have salient correlations with either **Factor1** or **Factor2** (but not both). This would make interpretations of factors less problematic.

---

## References

- Archer, C.O. and Jennrich, R.I. (1973), "Standard Errors for Orthogonally Rotated Factor Loadings," *Psychometrika*, 38, 581–592.
- Browne, M.W. (1982), "Covariance Structures," in *Topics in Applied Multivariate Analysis*, ed. D.M. Hawkins, Cambridge: Cambridge University Press, 72–141.
- Browne, M.W., Cudeck, R., Tateneni, K., and Mels, G. (1998), *CEFA: Comprehensive Exploratory Factor Analysis*. [<http://quantrm2.psy.ohio-state.edu/browne/>].
- Crawford, C.B. and Ferguson, G.A. (1970), "A General Rotation Criterion and Its Use in Orthogonal Rotation," *Psychometrika*, 35, 321–332.
- Harman, H.H. (1976), *Modern Factor Analysis*, Third Edition, Chicago: University of Chicago Press.
- Hayashi, K. and Yung, Y.F. (1999), "Standard Errors for the Class of Orthomax-Rotated Factor Loadings: Some Matrix Results," *Psychometrika*, 64, 451–460.
- Jennrich, R.I. (1973), "Standard Errors for Obliquely Rotated Factor Loadings," *Psychometrika*, 38, 593–604.



# Chapter 3

## The FREQ Procedure

### Chapter Table of Contents

---

|                            |    |
|----------------------------|----|
| <b>OVERVIEW</b> . . . . .  | 33 |
| <b>SYNTAX</b> . . . . .    | 33 |
| TABLES Statement . . . . . | 33 |



## Chapter 3

# The FREQ Procedure

---

### Overview

---

PROC FREQ has three new TABLES statement options to customize procedure output. The CONTENTS= option specifies the HTML contents link for crosstabulation tables. The FORMAT= option formats the frequencies displayed in crosstabulation tables. The OUTCUM option includes cumulative frequencies and cumulative percentages in the output data set for one-way tables.

### Syntax

---

#### TABLES Statement

---

The following options have been added to the TABLES statement:

**CONTENTS=***link-text*

specifies the text for the HTML contents file links to crosstabulation tables. For information on HTML output, refer to *The Complete Guide to the SAS Output Delivery System*. The CONTENTS= option affects only the HTML contents file, and not the HTML body file.

If you omit the CONTENTS= option, by default the HTML link text for crosstabulation tables is “Cross-Tabular Freq Table.”

Note that links to all crosstabulation tables produced by a single TABLES statement use the same text. To specify different text for different crosstabulation table links, request the tables in separate TABLES statements and use the CONTENTS= option in each TABLES statement.

The CONTENTS= option affects only links to crosstabulation tables. It does not affect links to other PROC FREQ tables. To specify link text for any other PROC FREQ table, you can use PROC TEMPLATE to create a customized table definition. The CONTENTS\_LABEL attribute in the DEFINE TABLE statement of PROC TEMPLATE specifies the contents file link for the table. For detailed information, refer to the chapter titled “The TEMPLATE Procedure” in *The Complete Guide to the SAS Output Delivery System*.

**FORMAT=***format-name*

specifies a format for the following crosstabulation table cell values: frequency, expected frequency, and deviation. PROC FREQ also uses this format to display the total row and column frequencies for crosstabulation tables.

You can specify any standard SAS numeric format or a numeric format defined with the FORMAT procedure. The format length must not exceed 24. If you omit



FORMAT=, by default PROC FREQ uses the BEST6. format to display frequencies less than 1E6, and the BEST7. format otherwise.

**OUTCUM**

includes the cumulative frequency and the cumulative percentage for one-way tables in the output data set when you specify the OUT= option in the TABLES statement. The variable CUM\_FREQ contains the cumulative frequency for each level of the analysis variable, and the variable CUM\_PCT contains the cumulative percentage for each level. The OUTCUM option has no effect for two-way or multiway tables.

# Chapter 4

## The GAM Procedure

### Chapter Table of Contents

---

|                                                         |    |
|---------------------------------------------------------|----|
| <b>OVERVIEW</b>                                         | 37 |
| <b>GETTING STARTED</b>                                  | 37 |
| <b>SYNTAX</b>                                           | 42 |
| PROC GAM Statement                                      | 42 |
| BY Statement                                            | 42 |
| CLASS Statement                                         | 43 |
| FREQ Statement                                          | 43 |
| ID Statement                                            | 44 |
| MODEL Statement                                         | 44 |
| OUTPUT Statement                                        | 45 |
| SCORE Statement                                         | 46 |
| <b>DETAILS</b>                                          | 46 |
| Nonparametric Regression                                | 46 |
| Additive Models and Generalized Additive Models         | 47 |
| Back-Fitting and Local Scoring Algorithms               | 48 |
| Smoothers                                               | 51 |
| Selection of Smoothing Parameters                       | 52 |
| Distribution Family and Canonical Link                  | 53 |
| Forms of Additive Models                                | 53 |
| ODS Tables Produced by PROC GAM                         | 54 |
| <b>EXAMPLES</b>                                         | 54 |
| Example 4.1 Generalized Additive Model with Binary Data | 54 |
| Example 4.2 Comparing PROC GAM with PROC TPSPLINE       | 61 |
| <b>REFERENCES</b>                                       | 66 |



## Chapter 4

# The GAM Procedure

---

### Overview

The GAM procedure is a new, experimental procedure that fits generalized additive models as those models are defined by Hastie and Tibshirani (1990). This procedure provides an array of powerful tools for data analysis, based on nonparametric regression and smoothing techniques.

Nonparametric regression relaxes the usual assumption of linearity and enables you to uncover structure in the relationship between the independent variables and the dependent variable that might otherwise be missed. The SAS System provides many procedures for nonparametric regression, such as the LOESS procedure for local regression and the TPSPLINE procedure for thin-plate smoothing splines. The generalized additive models fit by the GAM procedure combine

- an additive assumption (Stone 1985) that allows relatively many nonparametric relationships to be explored simultaneously with
- the distributional flexibility of generalized linear models (Nelder 1972)

Thus, you can use the GAM procedure when you have multiple independent variables whose effect you want to model nonparametrically, or when the dependent variable is not normally distributed. Refer to the “Nonparametric Regression” section on page 46 for more details on the form of generalized additive models.

The GAM procedure

- provides nonparametric estimates for additive models
- supports the use of multidimensional data
- supports multiple SCORE statements
- fits both generalized semiparametric additive models and generalized additive models
- enables you to choose a particular model by specifying the model degrees of freedom or smoothing parameter

---

### Getting Started

The following example illustrates the use of the GAM procedure to explore in a nonparametrical way how two factors affect a response. The data come from a study (Sackett et al. 1987) of the factors affecting patterns of insulin-dependent diabetes

mellitus in children. The objective is to investigate the dependence of the level of serum C-peptide on various other factors in order to understand the patterns of residual insulin secretion. The response measurement is the logarithm of C-peptide concentration (pmol/ml) at diagnosis, and the predictor measurements are age and base deficit (a measure of acidity):

```

title 'Patterns of Diabetes';
data diabetes;
  input Age BaseDeficit CPeptide @@;
  logCP = log(CPeptide);
  datalines;
    5.2   -8.1   4.8   8.8  -16.1   4.1  10.5   -0.9   5.2
  10.6   -7.8   5.5  10.4  -29.0   5.0   1.8  -19.2   3.4
  12.7  -18.9   3.4  15.6  -10.6   4.9   5.8   -2.8   5.6
    1.9 -25.0   3.7   2.2   -3.1   3.9   4.8   -7.8   4.5
    7.9 -13.9   4.8   5.2   -4.5   4.9   0.9  -11.6   3.0
   11.8   -2.1   4.6   7.9   -2.0   4.8  11.5   -9.0   5.5
   10.6 -11.2   4.5   8.5   -0.2   5.3  11.1   -6.1   4.7
   12.8   -1.0   6.6  11.3   -3.6   5.1   1.0   -8.2   3.9
   14.5   -0.5   5.7  11.9   -2.0   5.1   8.1   -1.6   5.2
   13.8 -11.9   3.7  15.5   -0.7   4.9   9.8   -1.2   4.8
   11.0 -14.3   4.4  12.4   -0.8   5.2  11.1  -16.8   5.1
    5.1   -5.1   4.6   4.8   -9.5   3.9   4.2  -17.0   5.1
    6.9   -3.3   5.1  13.2   -0.7   6.0   9.9   -3.3   4.9
   12.5 -13.6   4.1  13.2   -1.9   4.6   8.9  -10.0   4.9
   10.8 -13.5   5.1
  ;
run;

```

The following statements perform the desired analysis. The PROC GAM statement invokes the procedure and specifies the `diabetes` data set as input. The MODEL statement specifies `logCP` as the response variable and requests that univariate BSPLINES with 4 degrees of freedom be used to model the effect of `Age` and `BaseDeficit`. The OUTPUT statement specifies that partial prediction curves are to be saved in the data set `estimates`:

```

proc gam data=diabetes;
  model logCP = spline(age) spline(BaseDeficit);
  output out=estimates p;
run;

```

The results are shown in Figure 4.1 and Figure 4.2.

| Patterns of Diabetes                                       |              |
|------------------------------------------------------------|--------------|
| The GAM Procedure                                          |              |
| Dependent Variable: logCP                                  |              |
| Smoothing Model Component: spline(Age) spline(BaseDeficit) |              |
| Iteration Summary and Fit Statistics                       |              |
| Final number of backfitting iterations                     | 5            |
| Final backfitting criterion                                | 5.542743E-10 |
| Final residual sum of squares                              | 0.4180802183 |
| Summary of Input Data Set                                  |              |
| Number of Observations                                     | 43           |
| Number of Missing Observations                             | 0            |
| Distribution                                               | Gaussian     |
| Link Function                                              | Identity     |

Figure 4.1. Summary Statistics

Figure 4.1 shows two tables. The first table summarizes the convergence criterion for back-fitting, and the second one summarizes the input data set and the distribution family used for the model.

| Patterns of Diabetes                                       |                     |                |          |                    |
|------------------------------------------------------------|---------------------|----------------|----------|--------------------|
| The GAM Procedure                                          |                     |                |          |                    |
| Dependent Variable: logCP                                  |                     |                |          |                    |
| Smoothing Model Component: spline(Age) spline(BaseDeficit) |                     |                |          |                    |
| Regression Model Analysis                                  |                     |                |          |                    |
| Parameter Estimates                                        |                     |                |          |                    |
| Parameter                                                  | Parameter Estimate  | Standard Error | t Value  | Pr >  t            |
| Intercept                                                  | 1.48141             | 0.02588        | 57.24    | <.0001             |
| L_Age                                                      | 0.01437             | 0.00437        | 3.28     | 0.0024             |
| L_BaseDeficit                                              | 0.00807             | 0.00240        | 3.35     | 0.0020             |
| Smoothing Model Analysis                                   |                     |                |          |                    |
| Fit Statistics of Smoothing Components                     |                     |                |          |                    |
| Component                                                  | Smoothing Parameter | DF             | GCV      | No. of Unique Obs. |
| spline(Age)                                                | 0.995582            | 4.000000       | 0.011675 | 37                 |
| spline(BaseDeficit)                                        | 0.995299            | 4.000000       | 0.012437 | 39                 |
| Smoothing Model Analysis                                   |                     |                |          |                    |
| Analysis of Deviance                                       |                     |                |          |                    |
| Source                                                     | DF                  | Sum of Squares | F Value  | Pr > F             |
| spline(Age)                                                | 4.000000            | 0.150760       | 12.26    | 0.0155             |
| spline(BaseDeficit)                                        | 4.000000            | 0.081272       | 6.61     | 0.1580             |

Figure 4.2. Analysis of Model

Figure 4.2 displays summary statistics for the model. It consists of three tables. The first is the Parameter Estimates table for the parametric part of the model. It indicates that the linear trends for both Age and BaseDeficit are highly significant with  $p$ -values of 0.0024 and 0.0020. The second table is the summary of smoothing components of the nonparametric part of the model. Since the GAM fit used the default  $DF = 4$ , the main point of this table is to present the smoothing parameter values that yield this  $DF$  for each component. Finally, the third table shows the Analysis of Deviance table for the nonparametric component of the model.

The `P` option in the `OUTPUT` statement puts the partial predictions for Age and BaseDeficit in the output data set. You can compute the entire partial prediction effect for each factor by adding the estimated linear terms to the respective partial predictions, as in the following statement:

```
data estimates; set estimates;
  P2_age      = P_age      + 0.01437*age;
  P2_BaseDeficit = P_BaseDeficit + 0.00807*BaseDeficit;
run;
```

Plotting the partial predictions is one way to explore the overall shape of the relationship between each factor and the response. First of all, the following statements set up the graphics options:

```
axis1 label=(angle=90 rotate=0) minor=none;
axis2 minor=none order=(0 to 16 by 4);
symbol1 color=red interpol=join value=none line=1;
symbol2 color=blue interpol=join value=none line=2;
```

Then the following statements plot the partial prediction curves for Age and BaseDeficit:

```
proc sort data=estimates;
  by age;

proc gplot data=estimates;
  plot P_age *age = 1
       P2_Age*Age = 2 /overlay legend frame cframe=ligr
       name='gam1' vaxis=axis1 haxis=axis2;
run;

proc sort data=estimates;
  by BaseDeficit;

proc gplot data=estimates;
  plot P_BaseDeficit *BaseDeficit = 1
       P2_BaseDeficit*BaseDeficit = 2 /
       overlay legend frame cframe=ligr name='gam2'
       vaxis=axis1 haxis=axis2;
run;
```

Finally, the following statements redisplay the curves side by side for easy comparison:

```

goptions display;
proc greplay tc=tempcat nofs;
  igout gseg;
  tdef newtwo des='two plots of equal size'
  1/llx=0    lly=0
    ulx=0    uly=100
    urx=50   ury=100
    lrx=50   lry=0
  2/llx=50   lly=0
    ulx=50   uly=100
    urx=100  ury=100
    lrx=100  lry=0
;
template newtwo;
  treplay 1:gam1
          2:gam2;
run; quit;

```

The resulting plots for each predictor with and without the linear term are shown in Figure 4.3.

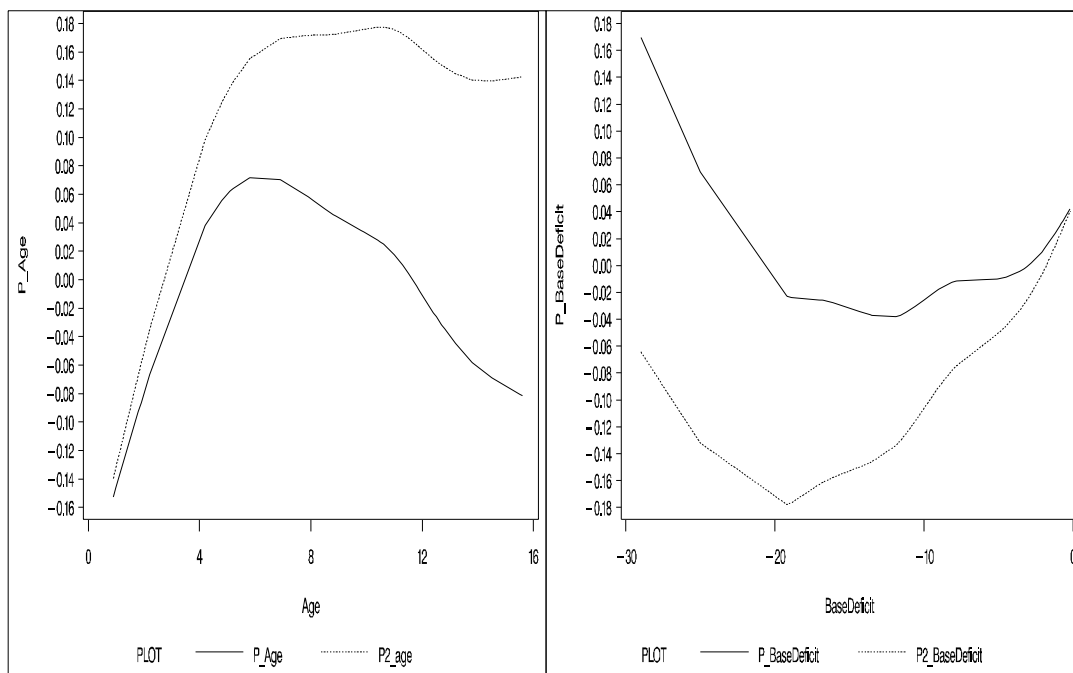


**Figure 4.3.** Partial Prediction for Each Predictor

Both plots show a strong quadratic pattern, with a possible indication of higher-order behavior. Further investigation is required to determine whether these patterns are real or not.



---

## Syntax

```

PROC GAM < option > ;
  CLASS variables ;
  MODEL dependent = < PARAM(effects) >
                        smoothing effects < /options > ;
  SCORE data=SAS-data-set out=SAS-data-set ;
  OUTPUT < out=SAS-data-set > keyword < ...keyword > < /option > ;
  BY variables ;
  ID variables ;
  FREQ variable ;

```

The syntax of the GAM procedure is similar to that of other regression procedures in the SAS System. The PROC GAM and MODEL statements are required. The SCORE statement can appear multiple times; all other statements appear only once.

The syntax for PROC GAM is described in the following sections in alphabetical order after the description of the PROC GAM statement.

---

### PROC GAM Statement

```
PROC GAM< option > ;
```

The PROC GAM statement invokes the procedure. You can specify the following option.

**DATA=SAS-data-set**

specifies the SAS data set to be read by PROC GAM. The default value is the most recently created data set.

---

### BY Statement

```
BY variables ;
```

You can specify a BY statement with PROC GAM to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the GAM procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (ac-

cording to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.

- Create an index for the BY variables using the DATASETS procedure.

For more information on the BY statement, refer to the discussion in *SAS Language Reference: Concepts*. For more information on the DATASETS procedure, refer to the discussion in the *SAS Procedures Guide*.

---

## CLASS Statement

**CLASS** *variables* ;

The CLASS statement names the classification variables to be used in the analysis. Typical class variables are TREATMENT, SEX, RACE, GROUP, and REPLICATION. If the CLASS statement is used, it must appear before the MODEL statement.

Classification variables can be either character or numeric. Class levels are determined from the formatted values of the CLASS variables. Thus, you can use formats to group values into levels. Refer to the discussion of the FORMAT procedure in the *SAS Procedures Guide*, and the discussions for the FORMAT statement and SAS formats in *SAS Language Reference: Dictionary*.

---

## FREQ Statement

**FREQ** *variable* ;

The FREQ statement names a variable that provides frequencies for each observation in the DATA= data set. Specifically, if  $n$  is the value of the FREQ variable for a given observation, then that observation is used  $n$  times.

The analysis produced using a FREQ statement reflects the expanded number of observations. You can produce the same analysis (without the FREQ statement) by first creating a new data set that contains the expanded number of observations. For example, if the value of the FREQ variable is 5 for the first observation, the first five observations in the new data set are identical. Each observation in the old data set is replicated  $n_i$  times in the new data set, where  $n_i$  is the value of the FREQ variable for that observation.

If the value of the FREQ variable is missing or is less than 1, the observation is not used in the analysis. If the value is not an integer, only the integer portion is used.

## ID Statement

**ID** *variables* ;

The variables in the ID statement are copied from the input data set to the OUT= data set. If you omit the ID statement, only the variables used in the MODEL statement and requested statistics are included in the output data set.

## MODEL Statement

**MODEL** *dependent* = <PARAM(*effects*)> <*smoothing effects*> </options> ;

The MODEL statement specifies the dependent variable and the independent effects you want to use to model its values. Specify the independent parametric variables inside the parentheses of PARAM( ). The parametric variables can be either CLASS variables or continuous variables. Any number of smoothing effects can be specified, as follows:

| Smoothing Effect                         | Meaning                                                   |
|------------------------------------------|-----------------------------------------------------------|
| spline(variable, <df=number>)            | fit smoothing spline with the variable and with DF=number |
| spline2(variable, variable, <df=number>) | fit bivariate thin-plate spline with DF=number            |

Both parametric effects and smoothing effects are optional, but at least one of them must be present.

If only parametric variables are present, PROC GAM fits a parametric linear model using the terms inside the parentheses of PARAM( ). If only smoothing effects are present, PROC GAM fits a nonparametric additive model. If both types of effect are present, PROC GAM fits a semiparametric model using the parametric effects as the linear part of the model.

The following table shows how to specify various models for a dependent variable *y* and independent variables *x*, *x1*, and *x2*.

**Table 4.1.** Syntax for Common GAM Models

| Type of Model     | Syntax                                                    | Mathematical Form                       |
|-------------------|-----------------------------------------------------------|-----------------------------------------|
| Parametric        | model <i>y</i> = param( <i>x</i> );                       | $E(y) = \beta_0 + \beta_1 x$            |
| Nonparametric     | model <i>y</i> = spline( <i>x</i> );                      | $E(y) = \beta_0 + s(x_2)$               |
| Semiparametric    | model <i>y</i> = param( <i>x1</i> ) spline( <i>x2</i> );  | $E(y) = \beta_0 + \beta_1 x_1 + s(x_2)$ |
| Additive          | model <i>y</i> = spline( <i>x1</i> ) spline( <i>x2</i> ); | $E(y) = \beta_0 + s_1(x_1) + s_2(x_2)$  |
| Thin-plate spline | model <i>y</i> = spline( <i>x1</i> , <i>x2</i> );         | $E(y) = \beta_0 + s(x_1, x_2)$          |

You can specify the following options in the MODEL statement.

**ALPHA=number**

specifies the significance level  $\alpha$  of the confidence limits on the final nonparametric component estimates when you request confidence limits to be included in the output data set. Specify *number* as a value between 0 and 1. The default value is 0.05. Refer to the “OUTPUT Statement” section on page 45 for more information on the OUTPUT statement.

**DIST=distribution-id**

specifies the distribution family used in the model. The *distribution-id* can be GAUSSIAN or LOGISTIC. The canonical link is used with those distributions. Although theoretically, alternative links are possible, with nonparametric models the final fit is relatively insensitive to the precise choice of link function. Therefore, only the canonical link for each distribution family is implemented in PROC GAM.

**EPSILON=number**

specifies the convergence criterion for the back-fitting algorithm.

**ITPRINT**

produces an iteration table for the smoothing effects.

**MAXITER=number**

specifies the maximum number of iterations for the back-fitting algorithm.

**METHOD=GCV**

specifies that the value of the smoothing parameter should be selected by generalized cross validation. If you specify both METHOD=GCV and the DF= option for the smoothing effects, the user-specified DF= is used, and the METHOD=GCV option is ignored. Refer to the “Selection of Smoothing Parameters” section on page 52 for more details on the GCV method.

---

## OUTPUT Statement

**OUTPUT** *OUT=SAS-data-set* < *keyword* ... *keyword* > ;

The OUTPUT statement creates a new SAS data set containing diagnostic measures calculated after fitting the model.

You can request a variety of diagnostic measures that are calculated for each observation in the data set. The new data set contains the variables specified in the MODEL statement in addition to the requested variables. If no *keyword* is present, the data set contains only the predicted values.

Details on the specifications in the OUTPUT statement are as follows.

**OUT=SAS-data-set**

specifies the name of the new data set to contain the diagnostic measures. This specification is required.

*keyword*

specifies the statistics to include in the output data set. The keywords and the statistics they represent are as follows:

|       |                                                                    |
|-------|--------------------------------------------------------------------|
| PRED  | predicted values                                                   |
| ADIAG | diagonal element of the hat matrix associated with the observation |

The names of the new variables that contain the statistics are formed by using a prefix of one or more characters that identify the statistic, followed by an underscore (\_), followed by the dependent variable name.

For example, suppose that you have a dependent variable *y*, and you specify the keywords PRED and ADIAG. In this case, the output SAS data set will contain the variables *P\_y* and *ADIAG\_y*.

---

## SCORE Statement

**SCORE** *DATA=SAS-data-set OUT=SAS-data-set ;*

The SCORE statement calculates predicted values for a new data set. If you have multiple data sets to predict, you can specify multiple SCORE statements. You must use a SCORE statement for each data set.

The following keywords must be specified in the SCORE statement.

**DATA=SAS-data-set**

specifies an input SAS data set containing all the variables included in independent effects in the MODEL statement. The predicted response is computed for each observation in the SCORE DATA= data set.

**OUT=SAS-data-set**

specifies the name of the SAS data set to contain the predictions.

---

## Details

---

### Nonparametric Regression

Nonparametric regression relaxes the usual assumption of linearity and enables you to explore the data visually, uncovering structure in the data that might otherwise be missed.

However, many forms of nonparametric regression do not perform well when the number of independent variables in the model is large. The sparseness of data in this setting causes the variances of the estimates to be unacceptably large unless the sample size is extremely large. The problem of rapidly increasing variance for increasing dimensionality is sometimes referred to as the “curse of dimensionality.” Interpretability is another problem with nonparametric regression based on kernel and smoothing spline estimates. The information these estimates contain about the relationship between the dependent and independent variables is often difficult to comprehend.

To overcome these difficulties, Stone (1985) proposed additive models. These models estimate an additive approximation to the multivariate regression function. The

benefits of an additive approximation are at least twofold. First, since each of the individual additive terms is estimated using a univariate smoother, the curse of dimensionality is avoided, at the cost of not being able to approximate universally. Second, estimates of the individual terms explain how the dependent variable changes with the corresponding independent variables.

To extend the additive model to a wide range of distribution families, Hastie and Tibshirani (1990) proposed generalized additive models. These models enable the mean of the dependent variable to depend on an additive predictor through a non-linear link function. The models permit the response probability distribution to be any member of the exponential family of distributions. Many widely used statistical models belong to this general class; they include additive models for Gaussian data, nonparametric logistic models for binary data, and nonparametric log-linear models for Poisson data.

---

## Additive Models and Generalized Additive Models

This section describes the methodology and the fitting procedure behind generalized additive models.

Let  $Y$  be a response random variable and  $X_1, X_2, \dots, X_p$  be a set of predictor variables. A regression procedure can be viewed as a method for estimating the expected value of  $Y$  given the values of  $X_1, X_2, \dots, X_p$ . The standard linear regression model assumes a linear form for the conditional expectation

$$E(Y|X_1, X_2, \dots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

Given a sample, estimates of  $\beta_0, \beta_1, \dots, \beta_p$  are usually obtained by the least squares method.

The additive model generalizes the linear model by modeling the conditional expectation as

$$E(Y|X_1, X_2, \dots, X_p) = s_0 + s_1(X_1) + s_2(X_2) + \dots + s_p(X_p)$$

where  $s_i(X), i = 1, 2, \dots, p$  are smooth functions.

In order to be estimable, the smooth functions  $s_i$  have to satisfy standardized conditions such as  $E s_j(X_j) = 0$ . These functions are not given a parametric form but instead are estimated in a nonparametric fashion.

While traditional linear models and additive models can be used in most statistical data analysis, there are types of problems for which they are not appropriate. For example, the normal distribution may not be adequate for modeling discrete responses such as counts or bounded responses such as proportions.

Generalized additive models address these difficulties, extending additive models to many other distributions besides just the normal. Thus, generalized additive models can be applied to a much wider range of data analysis problems.

Similar to generalized linear models, generalized additive models consist of a random component, an additive component, and a link function relating the two components. The response  $Y$ , the random component, is assumed to have exponential family density

$$f_Y(y; \theta; \phi) = \exp \left\{ \frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi) \right\}$$

where  $\theta$  is called the natural parameter and  $\phi$  is the scale parameter. The mean of the response variable  $\mu$  is related to the set of covariates  $X_1, X_2, \dots, X_p$  by  $g(\mu) = \eta$ . Here,  $\eta$  is defined as

$$\eta = s_0 + \sum_{i=1}^p s_i(X_i)$$

where  $s_1(\cdot), \dots, s_p(\cdot)$  are smooth functions, the quantity  $\eta$  is the linear component, and  $g(\cdot)$  is the link function. The most commonly used link for a given  $f$  is called the canonical link, for which  $\eta = \theta$ .

Generalized additive models and generalized linear models can be applied in similar situations, but they serve different analytic purposes. Generalized linear models emphasize estimation and inference for the parameters of the model, while generalized additive models focus on exploring data nonparametrically. Generalized additive models are more suitable for exploring the data set and visualizing the relationship between the dependent variable and the independent variables.

---

## Back-Fitting and Local Scoring Algorithms

Consider the estimation of the smoothing terms  $s_0, s_1(\cdot), \dots, s_p(\cdot)$  in the additive model

$$\mu(X) = s_0 + \sum_{i=1}^p s_i(X_i)$$

where  $E[s_j(X_j)] = 0$  for every  $j$ . Since the algorithm for additive models is the basis for fitting generalized additive models, the algorithm for additive models is discussed first.

Many ways are available to approach the formulation and estimation of additive models. The back-fitting algorithm is a general algorithm that can fit an additive model using any regression-type fitting mechanisms.

Define the partial residual as

$$R_j = Y - s_0 - \sum_{k \neq j} s_k(X_k)$$

Then  $E(R_j|X_j) = s_j(X_j)$ . This observation provides a way for estimating each smoothing function  $s_j(\cdot)$  given estimates  $\{\hat{s}_i(\cdot), i \neq j\}$  for all the others. The resulting iterative procedure is known as the back-fitting algorithm (Friedman and Stuetzle 1981).

### The Back-Fitting Algorithm

**1. Initialization:**

$$s_0 = E(Y), s_1^1 = s_2^1 = \cdots = s_p^1 = 0, m = 0.$$

**2. Iterate:**

$$m = m + 1$$

for  $j = 1$  to  $p$  do:

$$R_j = Y - s_0 - \sum_{k=1}^{j-1} s_k^m(X_k) - \sum_{k=j+1}^p s_k^{m-1}(X_k)$$

$$s_j^m = E(R_j|X_j).$$

**3. Until:**

$$RSS = \text{Avg}(Y - s_0 - \sum_{j=1}^p s_j^m(X_j))^2 \text{ fails to decrease.}$$

In the above notation,  $s_j^m(\cdot)$  denotes the estimate of  $s_j(\cdot)$  at the  $m$ th iteration. It can be shown that RSS never increases at any step, which implies that the algorithm always converges. However, the individual functions need not be unique, since dependence among the covariates can lead to more than one representation for the same fitted surface.

A weighted back-fitting algorithm has the same form as for the unweighted case, except that the smoothers are weighted. The weights might represent the relative precision of each observation or might arise as part of another iterative procedure. For example, weights are used in the local scoring procedure described later in this section.

The algorithm so far described fits just additive models. The algorithm for generalized additive models is a little more complicated. Generalized additive models extend generalized linear models in the same manner that additive models extend linear regression models, that is, by replacing form  $\alpha + \sum_j X_j \beta_j$  with the additive form  $\alpha + \sum_j f_j(X_j)$ . Thus, it is helpful to review the iteratively reweighted least-square procedure for computing the maximum likelihood estimates in a generalized linear model.

For generalized linear models, the maximum likelihood estimate of  $\beta$  is defined by the score equations

$$\sum_{i=1}^n x_{ij} \left( \frac{\partial \mu_i}{\partial \eta_i} \right) V_i^{-1}(y_i - \mu_i), j = 0, 1, \dots, p$$

where  $V_i$  is the variance matrix for  $Y_i$ . The Fisher scoring procedure is the standard method for solving these equations. It involves a Newton-Raphson algorithm using the expected (as opposed to the observed) information matrix. An equivalent procedure that is convenient for this problem is called dependent variable regression and is a form of iteratively reweighted least squares. Given a current coefficient vector



$\beta^0$ , with corresponding linear predictor  $\eta^0$  and fitted values  $\mu^0$ , construct the adjusted dependent variable

$$z_i = \eta_i^0 + (y_i - \mu_i^0) \left( \frac{\partial \eta_i}{\partial \mu_i} \right)_0$$

Define weights  $w_i$  by

$$w_i^{-1} = \left( \frac{\partial \eta_i}{\partial \mu_i} \right)_0^2 V_i^0$$

The algorithm proceeds by regressing  $z_i$  on  $x$  with weight  $w_i$  to obtain a revised estimate  $\beta$ . Then a new  $\mu^0$  and  $\eta^0$  are computed, new  $z_i$ s are computed, and the process is repeated until the change in the deviance

$$D(y; \hat{\mu}) = 2(l(\mu_{max}; y) - l(\hat{\mu}'y))$$

is sufficiently small.

Some adjusted dependent variables and weights for commonly used models are listed in the following table.

| Distribution         | Link     | Adjusted Dependent(Z)            | Weights(w)      |
|----------------------|----------|----------------------------------|-----------------|
| Normal               | identity | $y$                              | 1               |
| $\text{Bin}(n, \mu)$ | logit    | $\eta + (y - \mu)/n\mu(1 - \mu)$ | $n\mu(1 - \mu)$ |
| Gamma                | log      | $\eta + (y - \mu)/\mu$           | 1               |
| Poisson              | log      | $\eta + (y - \mu)/\mu$           | $\mu$           |

Generalized additive models differ from generalized linear models in that an additive predictor replaces the linear predictor. Estimation of the additive terms is accomplished by replacing the weighted linear regression in the adjusted dependent variable regression by the weighted back-fitting algorithm for fitting a weighted additive mode. This results in the algorithm described below as the *local scoring algorithm*. The name “local scoring” derives from the fact that local averaging is used to generalize the Fisher scoring procedure.

### The General Local Scoring Algorithm

#### 1. Initialization:

$$s_i = g(E(y)), s_1^0 = s_2^0 = \cdots = s_p^0 = 0, m = 0.$$

#### 2. Iterate:

$$m = m + 1$$

Form the adjusted dependent variable, predictor, and mean based on the previous iteration

$$Z = \eta^{m-1} + (Y - \mu^{m-1})(\partial \eta / \partial \mu^{m-1})$$

$$\eta^{m-1} = s_0 + \sum_{j=1}^p s_j^{m-1}(X_j)$$

$$\mu^{m-1} = g^{-1}(\eta^{m-1}).$$

Form the weights

$$w_i = (\partial \mu^{m-1} / \partial \eta^{m-1})^2 V_i^{-1}.$$

Fit an additive model to  $Z$  using the back-fitting algorithm with weights  $W$  to obtain estimated functions  $s_j^m(\cdot)$ .

**3. Until:**

$\text{Avg}(D(Y, \mu^m))$  fails to decrease, where  $\text{Avg}(D(Y, \mu^m))$  is an average of the deviance of estimate  $\mu^m$ .

The estimating procedure for generalized additive models consists of two loops. Inside each step of the local scoring algorithm (outer loop), a weighted back-fitting algorithm (inner loop) is used until convergence, that is, until the RSS fails to decrease. Then, based on the estimates from this weighted back-fitting algorithm, a new set of weights is calculated and the next iteration of the scoring algorithm starts. The scoring algorithm stops when the deviance of the estimates ceases to decrease.

---

## Smoothers

A smoother is a tool for summarizing the trend of a response measurement  $Y$  as a function of one or more predictor measurements  $X_1, \dots, X_p$ . It produces an estimate of the trend that is less variable than  $Y$  itself. An important property of a smoother is its nonparametric nature. It doesn't assume a rigid form for the dependence of  $Y$  on  $X_1, \dots, X_p$ . This section gives a brief overview of the smoothers that can be used with the GAM procedure.

### **Cubic Smoothing Spline**

A smoothing spline is the solution to the following optimization problem: among all functions  $s(x)$  with two continuous derivatives, find one that minimizes the penalized least square

$$\sum_{i=1}^n (y_i - s(x_i))^2 + \lambda \int_a^b (f''(t))^2 dt$$

where  $\lambda$  is a fixed constant, and  $a \leq x_1 \leq \dots \leq x_n \leq b$ . The first term measures closeness to the data while the second term penalizes curvature in the function. It can be shown that there exists an explicit, unique minimizer, and that minimizer is a natural cubic spline with knots at the unique values of  $x_i$ .

The parameter  $\lambda$  is the smoothing parameter. Large values of  $\lambda$  produce smoother curves while smaller values produce wiggly curves.

### **Thin-Plate Smoothing Spline**

The theoretical foundations for the thin-plate smoothing spline are described in Duchon (1976, 1977) and Meinguet (1979). Further results and applications are given in Wahba and Wendelberger (1980). Refer to "The TPSPLINE Procedure" in *SAS/STAT User's Guide, Version 8* for more details.

## Selection of Smoothing Parameters

### CV and GCV

The smoothers discussed here have a single smoothing parameter. In choosing the smoothing parameter, cross validation can be used. Cross validation works by leaving points  $(x_i, y_i)$  out one at a time, estimating the squared residual for smooth function at  $x_i$  based on the remaining  $n - 1$  data points, and choosing the smoother to minimize the sum of those squared residuals. This mimics the use of training and test samples for prediction. The cross validation function is defined as

$$CV(\lambda) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{s}_{\lambda}^{-i}(x_i))^2$$

where  $\hat{s}_{\lambda}^{-i}(x_i)$  indicates the fit at  $x_i$ , computed by leaving out the  $i$ th data point. The quantity  $nCV(\lambda)$  is sometimes called the prediction sum of squares or *PRESS* (Allen 1974).

All of the smoothers fit by the GAM procedure can be formulated as a linear combination of the sample responses

$$\hat{s}(x) = A(\lambda)Y$$

for some matrix  $A(\lambda)$ , which depends on  $\lambda$ . (The matrix  $A(\lambda)$  depends on  $x$  and the sample data, as well, but this dependence is suppressed in the preceding equation.) Let  $a_{ii}$  be the diagonal elements of the  $A(\lambda)$ . Then the *CV* function can be expressed as

$$CV(\lambda) = \frac{1}{n} \sum_{i=1}^n \left( \frac{(y_i - \hat{s}_{\lambda}(x_i))}{1 - a_{ii}} \right)^2$$

In most cases, it is very time consuming to compute the quantity  $a_{ii}$ . To solve this computational problem, Wahba (1990) has proposed the generalized cross validation function (*GCV*) that can be used to solve a wide variety of problems involving selection of a parameter to minimize the prediction risk.

The *GCV* function is defined as

$$GCV(\lambda) = \frac{\sum_{i=1}^n (y_i - \hat{s}_{\lambda}(x_i))^2}{(n - \text{tr}(A(\lambda)))^2}$$

The *GCV* formula simply replaces the  $a_{ii}$  with  $\text{tr}(A(\lambda))/n$ . Therefore, it can be viewed as a weighted version of *CV*. In most of the cases of interest, *GCV* is closely related to *CV* but much easier to compute. The GAM procedure uses the *GCV* function as the criterion for choosing the smoothing parameters.

The  $A$  matrix has the same role as the projection matrix in linear regression; therefore, nonparametric degrees of freedom (DF) for the model can be defined as  $\text{tr}(A)$ .

---

## Distribution Family and Canonical Link

For each distribution, more than one link can exist. Different link functions may result in a slight difference in estimates for parametric models. However, the difference will be less pronounced for nonparametric models because of the flexibility of nonparametric model forms. To simplify the calculation, the GAM procedure uses the canonical link.

The GAM procedure can fit the data from the Gaussian and binomial distributions:

### ***The Gaussian Model***

With this model, the link function is the identical function, and the generalized additive model is the additive model.

### ***The Logistic Model***

A binomial response model assumes that the proportion of successes  $Y$  is such that  $Y$  has a  $Bin(n(x), p(x))$  distribution. The  $Bin(n(x), p(x))$  refers to the binomial distribution with parameters  $n(x)$  and  $p(x)$ . Often the data are binary, in which case  $n(x) = 1$ . The canonical link is

$$g(p(x)) = \log \frac{p(x)}{1 - p(x)} = \eta(x)$$

---

## Forms of Additive Models

Suppose that  $y$  is a continuous variable and  $x1$  and  $x2$  are two explanatory variables of interest. To fit an additive model, you can use a MODEL statement similar to that used in many regression procedures in the SAS system:

```
model y = spline(x1) spline(x2);
```

This model statement requires the procedure to fit the following model:

$$f(x1, x2) = \text{Intercept} + s_1(x1) + s_2(x2)$$

where the  $s_i()$  terms denote nonparametric spline functions of the respective explanatory variables.

The GAM procedure can fit semiparametric models. The following MODEL statement assumes a linear relation with  $x1$  and an unknown functional relation with  $x2$ :

```
model y = param(x1) spline(x2);
```

If you want to fit a model containing a functional two-way interaction between  $x1$  and  $x2$ , you can use the following MODEL statement:

```
model y = spline2(x1, x2);
```

In this case, the GAM procedure fits a model equivalent to that of PROC TPSPLINE.

## ODS Tables Produced by PROC GAM

PROC GAM assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, refer to “Using the Output Delivery System” in *SAS/STAT User’s Guide, Version 8*.

**Table 4.2.** ODS Tables Produced by PROC GAM

| ODS Table Name     | Description                                        | Statement | Option  |
|--------------------|----------------------------------------------------|-----------|---------|
| ANOVA              | ANOVA table for parametric fit                     | PROC      | default |
| ANODEV             | Analysis of Deviance table for smoothing variables | PROC      | default |
| ClassSummary       | Summary of class variables                         | PROC      | default |
| DataSummary        | Data summary                                       | PROC      | default |
| IterSummary        | Iteration summary                                  | PROC      | default |
| FitSummary         | Fit parameters and fit summary                     | PROC      | default |
| ParameterEstimates | Parameter estimation for regression variables      | PROC      | default |
| Iteration          | Iteration history table                            | MODEL     | ITPRINT |

By referring to the names of such tables, you can use the ODS OUTPUT statement to place one or more of these tables in output data sets.

## Examples

### Example 4.1. Generalized Additive Model with Binary Data

The following example illustrates the capabilities of the GAM procedure and compares it to the GENMOD procedure.

The data used in this example are based on a study by Bell et al. (1989). Bell and his associates studied the result of multiple-level thoracic and lumbar laminectomy, a corrective spinal surgery commonly performed on children. The data in the study consist of retrospective measurements on 83 patients. The specific outcome of interest is the presence (1) or absence (0) of kyphosis, defined as a forward flexion of the spine of at least 40 degrees from vertical. The available predictor variables are Age in months at time of the operation, the starting of vertebrae levels involved in the operation (StartVert), and the number of levels involved (NumVert). The goal of this analysis is to identify risk factors for kyphosis. PROC GENMOD can be used to investigate the relationship among kyphosis and the predictors. The following DATA step creates the data kyphosis:

```

title 'Comparing PROC GAM with PROC GENMOD';
data kyphosis;
  input Age StartVert NumVert Kyphosis @@;
  datalines;
71 5 3 0      158 14 3 0      128 5 4 1
2 1 5 0      1 15 4 0      1 16 2 0

```

```

61 17 2 0      37 16 3 0      113 16 2 0
59 12 6 1      82 14 5 1      148 16 3 0
18 2 5 0       1 12 4 0      243 8 8 0
168 18 3 0     1 16 3 0      78 15 6 0
175 13 5 0     80 16 5 0     27 9 4 0
22 16 2 0     105 5 6 1     96 12 3 1
131 3 2 0     15 2 7 1     9 13 5 0
12 2 14 1     8 6 3 0      100 14 3 0
4 16 3 0      151 16 2 0     31 16 3 0
125 11 2 0    130 13 5 0     112 16 3 0
140 11 5 0    93 16 3 0      1 9 3 0
52 6 5 1      20 9 6 0      91 12 5 1
73 1 5 1      35 13 3 0     143 3 9 0
61 1 4 0      97 16 3 0     139 10 3 1
136 15 4 0    131 13 5 0     121 3 3 1
177 14 2 0    68 10 5 0      9 17 2 0
139 6 10 1    2 17 2 0      140 15 4 0
72 15 5 0     2 13 3 0      120 8 5 1
51 9 7 0      102 13 3 0     130 1 4 1
114 8 7 1     81 1 4 0      118 16 3 0
118 16 4 0    17 10 4 0     195 17 2 0
159 13 4 0    18 11 4 0     15 16 5 0
158 15 4 0    127 12 4 0     87 16 4 0
206 10 4 0    11 15 3 0     178 15 4 0
157 13 3 1    26 13 7 0     120 13 2 0
42 6 7 1      36 13 4 0
;

proc genmod;
  model Kyphosis = Age StartVert NumVert
              / link=logit dist=binomial;
run;

```

Output 4.1.1. GENMOD Analysis: Partial Output

| Comparing PROC GAM with PROC GENMOD |    |          |                |                            |        |            |            |
|-------------------------------------|----|----------|----------------|----------------------------|--------|------------|------------|
| The GENMOD Procedure                |    |          |                |                            |        |            |            |
| Analysis Of Parameter Estimates     |    |          |                |                            |        |            |            |
| Parameter                           | DF | Estimate | Standard Error | Wald 95% Confidence Limits |        | Chi-Square | Pr > ChiSq |
| Intercept                           | 1  | 1.2497   | 1.2424         | -1.1853                    | 3.6848 | 1.01       | 0.3145     |
| Age                                 | 1  | -0.0061  | 0.0055         | -0.0170                    | 0.0048 | 1.21       | 0.2713     |
| StartVert                           | 1  | 0.1972   | 0.0657         | 0.0684                     | 0.3260 | 9.01       | 0.0027     |
| NumVert                             | 1  | -0.3031  | 0.1790         | -0.6540                    | 0.0477 | 2.87       | 0.0904     |
| Scale                               | 0  | 1.0000   | 0.0000         | 1.0000                     | 1.0000 |            |            |

NOTE: The scale parameter was held fixed.

The GENMOD analysis of the independent variable effects is shown in Output 4.1.1. Based on these results, the only significant factor is **StartVert** with odds ratio of  $-0.1972$ . The variable **NumVert** has a  $p$ -value of 0.0904 with odds ratio of 0.3031.

The GENMOD procedure assumes a strict linear relationship between the response and the predictors. The following SAS statements use PROC GAM to investigate a less restrictive model, with moderately flexible spline terms for each of the predictors:

```

title 'Comparing PROC GAM with PROC GENMOD';
proc gam data=kyphosis;
    model Kyphosis=spline(Age,df=3) spline(StartVert,df=3)
              spline(NumVert,df=3) /dist = logist;
    output out=estimate p;
run;

```

The MODEL statement requests an additive model using a univariate BSPLINE for each term. The option `dist=logist` specifies a logistic model. Each term is fitted using a smoothing spline with three degrees of freedom. Although this might seem to be an unduly modest amount of flexibility, it is better to be conservative with a data set this small. An output data set `estimate` containing predicted values is requested by the OUTPUT statement.

Output 4.1.2 and Output 4.1.3 list the output from PROC GAM.

#### Output 4.1.2. Summary Statistics

| Comparing PROC GAM with PROC GENMOD                                      |              |
|--------------------------------------------------------------------------|--------------|
| The GAM Procedure                                                        |              |
| Dependent Variable: Kyphosis                                             |              |
| Smoothing Model Component: spline(Age) spline(StartVert) spline(NumVert) |              |
| Iteration Summary and Fit Statistics                                     |              |
| Number of local score iterations                                         | 9            |
| Local score convergence criterion                                        | 3.9786363E-9 |
| Final number of backfitting iterations                                   | 1            |
| Final backfitting criterion                                              | 5.2326788E-9 |
| Final residual sum of squares                                            | 46.610928346 |
| Summary of Input Data Set                                                |              |
| Number of Observations                                                   | 83           |
| Number of Missing Observations                                           | 0            |
| Distribution                                                             | Binomial     |
| Link Function                                                            | Logit        |

**Output 4.1.3.** Analysis of Model

| Comparing PROC GAM with PROC GENMOD                                      |                     |                |            |                    |
|--------------------------------------------------------------------------|---------------------|----------------|------------|--------------------|
| The GAM Procedure                                                        |                     |                |            |                    |
| Dependent Variable: Kyphosis                                             |                     |                |            |                    |
| Smoothing Model Component: spline(Age) spline(StartVert) spline(NumVert) |                     |                |            |                    |
| Regression Model Analysis                                                |                     |                |            |                    |
| Parameter Estimates                                                      |                     |                |            |                    |
| Parameter                                                                | Parameter Estimate  | Standard Error | t Value    | Pr >  t            |
| Intercept                                                                | -2.01545            | 0.74274        | -2.71      | 0.0083             |
| L_Age                                                                    | 0.01213             | 0.00622        | 1.95       | 0.0552             |
| L_StartVert                                                              | -0.18615            | 0.06061        | -3.07      | 0.0030             |
| L_NumVert                                                                | 0.38347             | 0.15264        | 2.51       | 0.0142             |
| Smoothing Model Analysis                                                 |                     |                |            |                    |
| Fit Statistics of Smoothing Components                                   |                     |                |            |                    |
| Component                                                                | Smoothing Parameter | DF             | GCV        | No. of Unique Obs. |
| spline(Age)                                                              | 0.999996            | 3.000000       | 328.513619 | 66                 |
| spline(StartVert)                                                        | 0.999551            | 3.000000       | 317.647039 | 16                 |
| spline(NumVert)                                                          | 0.921758            | 3.000000       | 20.144078  | 10                 |
| Smoothing Model Analysis                                                 |                     |                |            |                    |
| Analysis of Deviance                                                     |                     |                |            |                    |
| Source                                                                   | DF                  | Sum of Squares | F Value    | Pr > F             |
| spline(Age)                                                              | 3.000000            | 10.494366      | 16.44      | 0.0009             |
| spline(StartVert)                                                        | 3.000000            | 5.494965       | 8.61       | 0.0135             |
| spline(NumVert)                                                          | 3.000000            | 2.184514       | 3.42       | 0.3311             |

The critical part of the GAM results is the Analysis of Deviance table, shown in Output 4.1.3. For each smoothing effect in the model, this table gives an  $F$ -test comparing the deviance between the full model and the model without this variable. In this case, the analysis of deviance results indicates that the effect of Age and StartVert are highly significant, while the effect of NumVert is insignificant. Plots of predictions against predictor can be used to investigate why PROC GAM and PROC GENMOD produce different results.

The GAM statement requests an output data set of predicted values to be created. Since the estimate of the generalized additive model is the sum of functional estimates of individual predictors, plus a constant, the output data set will contain a column of partial prediction for each predictor. If requested, a Bayesian confidence interval or a point-wise standard-error band, as defined in Hastie and Tibshirani (1990), can be produced in the output data set.

Using the following statements, the data set estimate is plotted in Output 4.1.4:



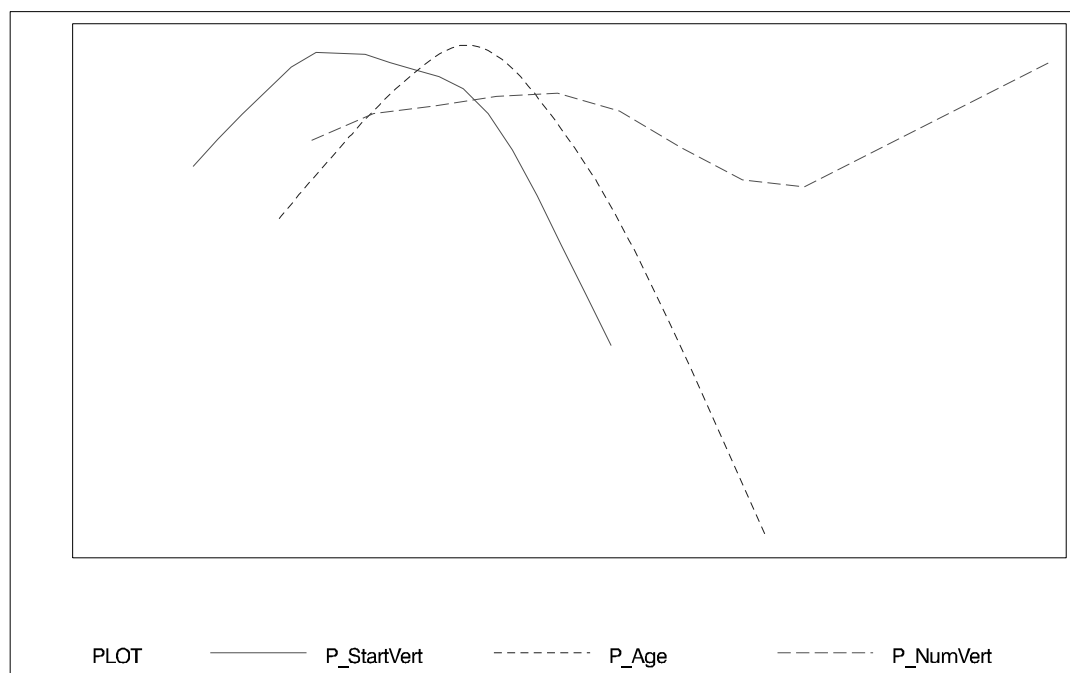
```

proc sort data=estimate(keep=StartVert P_StartVert)
    out =StartVert;
    by StartVert;
proc sort data=estimate(keep=Age      P_Age      )
    out =Age;
    by Age;
proc sort data=estimate(keep=NumVert  P_NumVert  )
    out =NumVert;
    by NumVert;
data Plot; merge StartVert Age NumVert;
proc standard m=0 s=1 data=Plot out=Plot;
    var StartVert Age NumVert;
run;

legend1 frame cframe=ligr cborder=black label=none
    position=center;
axis1 label=(angle=90 rotate=0 " ") minor=none
    value=NONE major=NONE;
axis2 minor=none label=(" ") major=NONE value=NONE;
symbol1 color=red   interpol=join value=none line=1;
symbol2 color=blue  interpol=join value=none line=2;
symbol3 color=green interpol=join value=none line=3;
proc gplot data=Plot;
    title;
    plot P_StartVert*StartVert=1
        P_Age      *Age      =2
        P_NumVert  *NumVert  =3 / overlay legend frame
        cframe=ligr vaxis=axis1 haxis=axis2;
run;

```

Output 4.1.4. Partial Prediction for Each Predictor



The plot shows that the partial predictions corresponding to both Age and StartVert have a strong quadratic pattern, while NumVert has a more complicated but weaker pattern. However, in the plot for NumVert, notice that about half the vertical range of the function is determined by the point at the upper extreme. It would be a good idea, therefore, to re-run the analysis without this point, to see how much it affects the conclusions. You can do this simply by including a WHERE clause when specifying the data set for the GAM procedure, as in the following code:

```

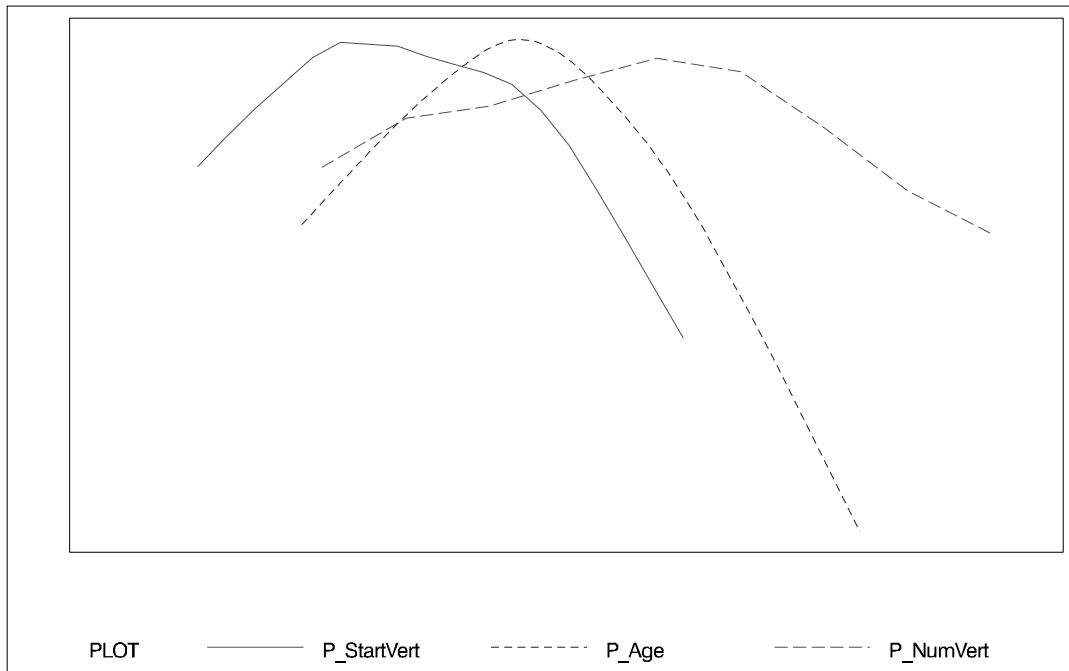
title 'Comparing PROC GAM with PROC GENMOD';
proc gam data=kyphosis(where=(NumVert^=14));
  model Kyphosis=spline(Age,df=3) spline(StartVert,df=3)
    spline(NumVert, df=3) /dist = logist;
  output out=estimate p;
run;

```

The analysis of deviance table from this re-analysis is shown in Output 4.1.5, and Output 4.1.6 shows the re-computed partial predictor plots.

**Output 4.1.5.** Analysis After Removing NumVert=14

| Comparing PROC GAM with PROC GENMOD                                      |          |                |         |        |
|--------------------------------------------------------------------------|----------|----------------|---------|--------|
| The GAM Procedure                                                        |          |                |         |        |
| Dependent Variable: Kyphosis                                             |          |                |         |        |
| Smoothing Model Component: spline(Age) spline(StartVert) spline(NumVert) |          |                |         |        |
| Smoothing Model Analysis                                                 |          |                |         |        |
| Analysis of Deviance                                                     |          |                |         |        |
| Source                                                                   | DF       | Sum of Squares | F Value | Pr > F |
| spline(Age)                                                              | 3.000000 | 10.587568      | 16.90   | 0.0007 |
| spline(StartVert)                                                        | 3.000000 | 5.477104       | 8.74    | 0.0126 |
| spline(NumVert)                                                          | 3.000000 | 3.209100       | 5.12    | 0.1630 |

**Output 4.1.6.** Partial Prediction After Removing NumVert=14

After removing data point NumVert=14, the predictors `age` and `StartVert` are still significant and the variable `NumVert` still insignificant. Based on the plot in Output 4.1.6, the removed point has almost no effect on estimates of curve shape for variables `Age` and `StartVert`. But the removal has a dramatic effect on the variable `NumVert`: this curve for this variable `NumVert` now also seems quadratic, though it is much less pronounced than for the other two variables.

Having used the GAM procedure to discover an appropriate form of the dependence of Kyphosis on each of the three independent variables, you can use the GENMOD procedure to fit and assess the corresponding parametric model. The following code fits a GENMOD model with quadratic terms for all three variables, including tests for the joint linear and quadratic effects of each variable. The resulting contrast tests are shown in Output 4.1.7.

```

title 'Comparing PROC GAM with PROC GENMOD';
proc genmod data=kyphosis(where=(NumVert^=14));
  model kyphosis = Age      Age      *Age
                  StartVert StartVert*StartVert
                  NumVert  NumVert  *NumVert
                  /link=logit dist=binomial;
  contrast 'Age'      Age      1, Age*Age      1;
  contrast 'StartVert' StartVert 1, StartVert*StartVert 1;
  contrast 'NumVert'  NumVert  1, NumVert*NumVert 1;
run;

```

**Output 4.1.7.** Joint Linear and Quadratic Tests

| Comparing PROC GAM with PROC GENMOD |    |            |            |      |
|-------------------------------------|----|------------|------------|------|
| The GENMOD Procedure                |    |            |            |      |
| Contrast Results                    |    |            |            |      |
| Contrast                            | DF | Chi-Square | Pr > ChiSq | Type |
| Age                                 | 2  | 13.63      | 0.0011     | LR   |
| StartVert                           | 2  | 15.41      | 0.0005     | LR   |
| NumVert                             | 2  | 3.56       | 0.1684     | LR   |

The results for the quadratic GENMOD model are now quite consistent with the GAM results.

From this example, you can see that PROC GAM is very useful in visualizing the data and detecting the nonlinearity among the variables.

---

**Example 4.2. Comparing PROC GAM with PROC TPSPLINE**

This example compares the GAM procedure with the TPSPLINE procedure, another nonparametric procedure that fits a smooth surface to multivariate data. It does not assume additivity of the model and uses very general basis functions for model fitting, making the TPSPLINE procedure much slower than the GAM procedure. For more details about the TPSPLINE procedure, refer to “The TPSPLINE Procedure” in *SAS/STAT User’s Guide, Version 8*.

The data used here is also analyzed in “The TPSPLINE Procedure” in *SAS/STAT User’s Guide, Version 8*. It presents age-adjusted melanoma incidences for 37 years from the Connecticut Tumor Registry (Houghton, Flannery, and Viola 1980):

```

title 'Comparing PROC GAM with PROC TPSPLINE';
data melanoma;
  input year incidences @@;
  datalines;
1936 0.9 1937 0.8 1938 0.8 1939 1.3
1940 1.4 1941 1.2 1942 1.7 1943 1.8
1944 1.6 1945 1.5 1946 1.5 1947 2.0
1948 2.5 1949 2.7 1950 2.9 1951 2.5
1952 3.1 1953 2.4 1954 2.2 1955 2.9
1956 2.5 1957 2.6 1958 3.2 1959 3.8
1960 4.2 1961 3.9 1962 3.7 1963 3.3
1964 3.7 1965 3.9 1966 4.1 1967 3.8
1968 4.7 1969 4.4 1970 4.8 1971 4.8
1972 4.8
;
run;

```

The variable `incidences` records the number of melanoma cases per 100,000 people for the years 1936 to 1972.

Four to five degrees of freedom for each nonparametric term in a generalized additive model fits most data well. However, to select DF more objectively you can use the GCV option to minimize the generalized cross validation function, as shown in the following PROC GAM code:

```
proc gam data=melanoma;
  model incidences = spline(year) /method = GCV;
  output out=gam p;
run;
```

The results are listed in Output 4.2.1 and Output 4.2.2.

**Output 4.2.1.** Summary Statistics

| Comparing PROC GAM with PROC TPSPLINE   |              |
|-----------------------------------------|--------------|
| The GAM Procedure                       |              |
| Dependent Variable: incidences          |              |
| Smoothing Model Component: spline(year) |              |
| Iteration Summary and Fit Statistics    |              |
| Final number of backfitting iterations  | 2            |
| Final backfitting criterion             | 0            |
| Final residual sum of squares           | 1.2242517494 |
| Summary of Input Data Set               |              |
| Number of Observations                  | 37           |
| Number of Missing Observations          | 0            |
| Distribution                            | Gaussian     |
| Link Function                           | Identity     |

Output 4.2.2. Analysis of Model

| Comparing PROC GAM with PROC TPSPLINE   |                     |                |          |                    |
|-----------------------------------------|---------------------|----------------|----------|--------------------|
| The GAM Procedure                       |                     |                |          |                    |
| Dependent Variable: incidences          |                     |                |          |                    |
| Smoothing Model Component: spline(year) |                     |                |          |                    |
| Regression Model Analysis               |                     |                |          |                    |
| Parameter Estimates                     |                     |                |          |                    |
| Parameter                               | Parameter Estimate  | Standard Error | t Value  | Pr >  t            |
| Intercept                               | -212.69706          | 7.00491        | -30.36   | <.0001             |
| L_year                                  | 0.11029             | 0.00358        | 30.77    | <.0001             |
| Smoothing Model Analysis                |                     |                |          |                    |
| Fit Statistics of Smoothing Components  |                     |                |          |                    |
| Component                               | Smoothing Parameter | DF             | GCV      | No. of Unique Obs. |
| spline(year)                            | 0.634903            | 13.414936      | 0.088803 | 37                 |
| Smoothing Model Analysis                |                     |                |          |                    |
| Analysis of Deviance                    |                     |                |          |                    |
| Source                                  | DF                  | Sum of Squares | F Value  | Pr > F             |
| spline(year)                            | 13.414936           | 2.736763       | 50.49    | <.0001             |

Based on the summary of the model, the final model has a DF = 13.414936 and the nonparametric trend is highly significant. Note that this DF is much greater than the default value of 4, indicating that there is a great deal of structure in the yearly incidence rates of melanoma. A prediction plot should reveal the nature of this structure:

```

legend1 frame cframe=ligr cborder=black label=none
      position=center;
axis1  label=(angle=90 rotate=0);
axis2  minor=none;
symbol1 color=red interpol=join value=none line=1;

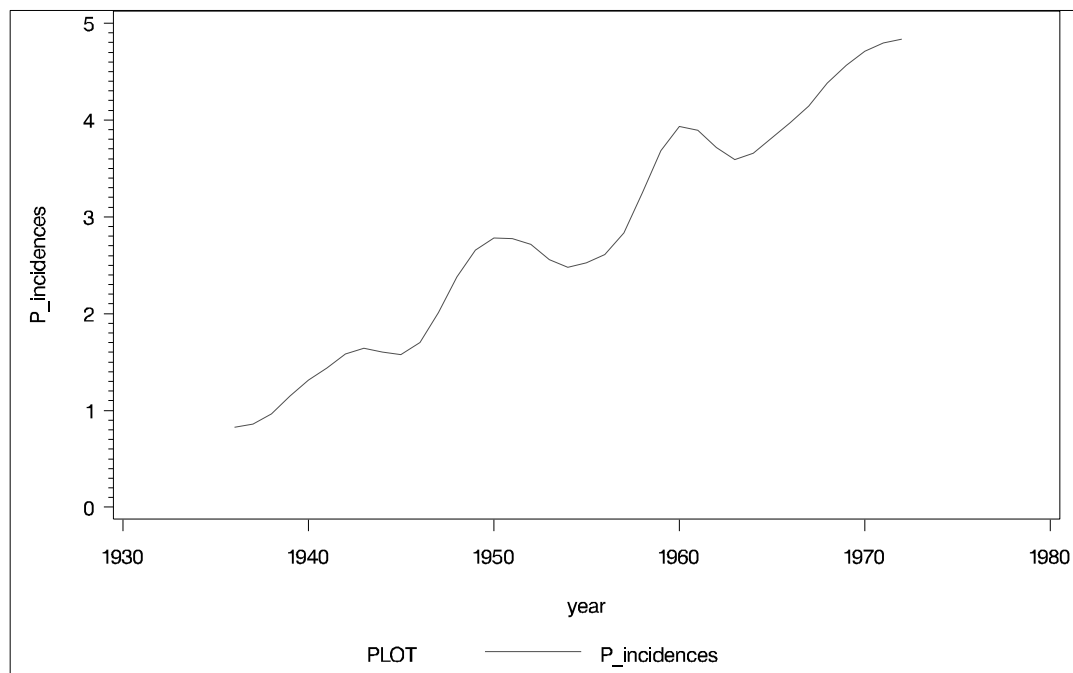
proc sort data=gam; by year;
proc gplot data=gam;
  title;
  plot p_incidences*year = 1 /overlay legend
      frame cframe=ligr vaxis=axis1 haxis=axis2;
run;

```

Output 4.2.3 shows the predicted melanoma rate over time. Two features stand out on this plot:

- Melanoma incidence on the whole increases over the period of the study.

- A strong periodic effect is evident, with a period of a little more than a decade. This has been attributed to the 11-year sunspot cycle: the more sunspots there are, the more melanoma cases there are likely to be.

**Output 4.2.3.** Predicted Melanoma Incidence Rates By Year

Since PROC TPSPLINE also fits a nonparametric model, PROC TPSPLINE and PROC GAM fits should be very similar for this univariate case. The following code produces the TPSPLINE analysis shown in Output 4.2.4:

```
title 'Comparing PROC GAM with PROC GENMOD';
proc tpspline data=melanoma;
  model incidences = (year);
  output out=tpspline p;
run;
```

**Output 4.2.4.** Analysis from PROC TPSPLINE

| Comparing PROC GAM with PROC TPSPLINE     |         |
|-------------------------------------------|---------|
| The TPSPLINE Procedure                    |         |
| Dependent Variable: incidences            |         |
| Summary of Input Data Set                 |         |
| Number of Non-Missing Observations        | 37      |
| Number of Missing Observations            | 0       |
| Unique Smoothing Design Points            | 37      |
| Summary of Final Model                    |         |
| Number of Regression Variables            | 0       |
| Number of Smoothing Variables             | 1       |
| Order of Derivative in the Penalty        | 2       |
| Dimension of Polynomial Space             | 2       |
| Summary Statistics<br>of Final Estimation |         |
| log10(n*Lambda)                           | -0.0607 |
| Smoothing Penalty                         | 0.5171  |
| Residual SS                               | 1.2243  |
| Tr(I-A)                                   | 22.5852 |
| Model DF                                  | 14.4148 |
| Standard Deviation                        | 0.2328  |

The TPSPLINE model analysis shows that the DF for the model is 14.4148. This is consistent with the GAM model because the DF value in GAM excludes the degree of freedom of the linear `L_year` term. The OUTPUT statements in the PROC GAM and PROC TPSPLINE code create `gam` and `tpspline` data sets containing the predicted values for the respective procedures. You can use the following code to look at the values for the two procedures side by side:

```
data both; merge gam      (rename=(p_incidence=gam      ))
                tpspline(rename=(p_incidence=tpspline));
proc print data=both;
  var year gam tpspline;
run;
```

The results, the first ten of which are displayed in Output 4.2.5, show that PROC GAM and PROC TPSPLINE give essentially the same predictions for this problem.



**Output 4.2.5.** Melanoma Predictions for First Ten Years

| Comparing PROC GAM with PROC TPSPLINE |     |      |         |          |
|---------------------------------------|-----|------|---------|----------|
|                                       | Obs | year | gam     | tpspline |
|                                       | 1   | 1936 | 0.82425 | 0.82424  |
|                                       | 2   | 1937 | 0.85580 | 0.85580  |
|                                       | 3   | 1938 | 0.96379 | 0.96379  |
|                                       | 4   | 1939 | 1.15046 | 1.15046  |
|                                       | 5   | 1940 | 1.31044 | 1.31044  |
|                                       | 6   | 1941 | 1.43881 | 1.43881  |
|                                       | 7   | 1942 | 1.58218 | 1.58218  |
|                                       | 8   | 1943 | 1.64382 | 1.64382  |
|                                       | 9   | 1944 | 1.60148 | 1.60148  |
|                                       | 10  | 1945 | 1.57498 | 1.57499  |

## References

- Allen, D.M. (1974), "The relationship between variable selection and data augmentation and a method of prediction," *Technometrics*, 16, 125–127.
- Bell, D., Walker, J., O'Connor, G., Orrel, J. and Tibshirani, R. (1989), "Spinal Deformation Following Multi-Level Thoracic and Lumbar Laminectomy in Children." Submitted for publication.
- Duchon, J. (1976), "Fonctions-Spline et Esperances Conditionnelles de Champs Gaussiens," *Ann. Sci. Univ. Clermont Ferrand II Math.*, 14, 19–27.
- Duchon, J. (1977), "Splines Minimizing Rotation-Invariant Semi-Norms in Sobolev Spaces," in *Constructive Theory of Functions of Several Variables*, eds. W. Schempp and K. Zeller, New York: Springer-Verlag, 85–100.
- Hastie, T.J. and Tibshirani, R.J. (1990), *Generalized Additive Models*, New York: Chapman and Hall.
- Houghton, A. N., Flannery, J. and Viola, M. V. (1980), "Malignant Melanoma in Connecticut and Denmark," *International Journal of Cancer*, 25, 95–104.
- Meinguet, J. (1979), "Multivariate Interpolation at Arbitrary Points Made Simple," *Journal of Applied Mathematics and Physics (ZAMP)*, 30, 292–304.
- Nelder, J.A. and Wedderburn, R.W.M. (1972) "Generalized Linear models" *Journal of the Royal Statistical Society, Series A*, 135, 370–384.
- SAS Institute Inc. (1999), *SAS/STAT User's Guide, Version 8*, Cary, NC: SAS Institute Inc.
- Socketk, E.B., Daneman, D., Clarson, C., and Ehrich, R.M. (1987), "Factors Affecting and Patterns of Residual Insulin Secretion During the First Year of Type I (Insulin Dependent) Diabetes Mellitus in Children," *Diabet*, 30, 453–459.
- Stone, C.J. (1985), "Additive Regression and Other Nonparametric Models," *Annals of Statistics*, 13, 689–705.
- Wahba, G. (1990), *Spline Models for Observational Data*, Philadelphia: Society for Industrial and Applied Mathematics.

- Wahba, G. and Wendelberger, J. (1980), "Some New Mathematical Methods for Variational Objective Analysis Using Splines and Cross Validation," *Monthly Weather Review*, 108, 1122–1145.



# Chapter 5

## The GENMOD Procedure

### Chapter Table of Contents

---

|                           |    |
|---------------------------|----|
| <b>OVERVIEW</b> . . . . . | 71 |
|---------------------------|----|



## Chapter 5

# The GENMOD Procedure

---

### Overview

The probability of the lower level of binary (dichotomous) response variables specified with the single response variable syntax and the binomial distribution is now modeled by default in the GENMOD procedure. Previously, the higher level of the response variable was modeled.

The DESCENDING option in the PROC GENMOD statement now applies to binary response variables.



# Chapter 6

## The LOESS Procedure

### Chapter Table of Contents

---

|                                                     |    |
|-----------------------------------------------------|----|
| <b>OVERVIEW</b>                                     | 75 |
| <b>SYNTAX</b>                                       | 75 |
| MODEL Statement                                     | 75 |
| SCORE Statement                                     | 77 |
| <b>DETAILS</b>                                      | 77 |
| kd Trees and Blending                               | 77 |
| Automatic Smoothing Parameter Selection             | 78 |
| ODS Table Names                                     | 81 |
| <b>EXAMPLE</b>                                      | 81 |
| Example 6.1 Automatic Smoothing Parameter Selection | 81 |
| <b>REFERENCES</b>                                   | 86 |





# Chapter 6

## The LOESS Procedure

---

### Overview

---

Enhancements to the LOESS procedure include:

- automatic smoothing parameter selection
- higher order blending for models with one or two regressors
- a table summarizing the fits obtained for each value of the smoothing parameter

### Syntax

---

#### MODEL Statement

The following new or updated options are available in the MODEL statement after a slash (/).

**DETAILS** < ( *tables* ) >

selects which tables to display, where *tables* is one or more of KDTREE, MODEL-SUMMARY, OUTPUTSTATISTICS, and PREDATVERTICES:

- KDTREE displays the kd tree structure.
- MODELSUMMARY displays the fit criteria for all smoothing parameter values that are specified in the SMOOTH= option in the MODEL statement, or which are fit with automatic smoothing parameter selection.
- OUTPUTSTATISTICS displays the predicted values and other requested statistics at the points in the input data set.
- PREDATVERTICES displays fitted values and coordinates of the kd tree vertices where the local least squares fitting is done.

The KDTREE and PREDATVERTICES specifications are ignored if the DIRECT option is specified in the MODEL statement. Specifying the option DETAILS with no qualifying list outputs all tables.

**INTERP= LINEAR | CUBIC**

specifies the degree of the interpolating polynomials used for blending local polynomial fits at the kd tree vertices. This option is ignored if the DIRECT option is specified in the model statement. INTERP=CUBIC is not supported for models with more than two regressors. The default is INTERP=LINEAR.

**TRACEL**

specifies that the trace of the prediction matrix as well as the GCV and AICC statistics are to be included in the “FIT Summary” table. The use of any of the MODEL statement options ALL, CLM, DFMETHOD=EXACT, DIRECT, SELECT=, or T implicitly selects the TRACEL option.

**SELECT=criterion < ( <GLOBAL> <STEPS> <RANGE(lower,upper)> )>**

specifies that automatic smoothing parameter selection be done using the named *criterion*, where the *criterion* is one of AICC, AICC1, or GCV. The smoothing parameter value is selected to yield a minimum of the *criterion*, as follows:

- If you specify the SMOOTH=*value-list* option, then PROC LOESS selects the largest value in this list that yields the global minimum of the specified criterion.
- If you do not specify the SMOOTH= option, then PROC LOESS finds a local minimum of the specified criterion using a golden section search of values less than or equal to one.

You can specify the following modifiers in parentheses after the specified criterion to alter the behavior of the SELECT= option:

- GLOBAL specifies that a global minimum be found within the range of smoothing parameter values examined. This modifier has no effect if you also specify the SMOOTH= option in the MODEL statement.
- STEPS specifies that all models evaluated in the selection process be displayed.
- RANGE(*lower,upper*) specifies that only smoothing parameter values greater than or equal to *lower* and less than or equal to *upper* be examined.

For models with one dependent variable, if you specify neither the SELECT= nor the SMOOTH= options in the MODEL statement, then PROC LOESS uses SELECT=AICC.

The following table summarizes how the smoothing parameter values are chosen for various combinations of the SMOOTH= option, the SELECT= option, and the SELECT= option modifiers.

**Table 6.1.** Smoothing Parameter Value(s) Used for Combinations of SMOOTH= and SELECT= Options for Models with One Dependent Variable

| SYNTAX                                                                        | SEARCH METHOD             | SEARCH DOMAIN                                        |
|-------------------------------------------------------------------------------|---------------------------|------------------------------------------------------|
| <i>default</i>                                                                | golden section using AICC | (0, 1]                                               |
| SMOOTH= <i>list</i>                                                           | no selection              | values in <i>list</i>                                |
| SMOOTH= <i>list</i> SELECT= <i>criterion</i>                                  | global                    | values in <i>list</i>                                |
| SMOOTH= <i>list</i> SELECT= <i>criterion</i> ( RANGE( <i>l</i> , <i>u</i> ) ) | global                    | values in <i>list</i> within [ <i>l</i> , <i>u</i> ] |
| SELECT= <i>criterion</i>                                                      | golden section            | (0, 1]                                               |
| SELECT= <i>criterion</i> ( RANGE( <i>l</i> , <i>u</i> ) )                     | golden section            | [ <i>l</i> , <i>u</i> ]                              |
| SELECT= <i>criterion</i> ( GLOBAL )                                           | global                    | (0, 1]                                               |
| SELECT= <i>criterion</i> ( GLOBAL RANGE( <i>l</i> , <i>u</i> ) )              | global                    | [ <i>l</i> , <i>u</i> ]                              |

**Note:** The SELECT= option cannot be used for models with more than one dependent variable.

#### **SMOOTH=***value-list*

specifies a list of positive smoothing parameter values. If you do not SELECT= option in the MODEL statement, then a separate fit is obtained for each SMOOTH= value specified. If you do specify the SELECT= option, then models with all values specified in the SMOOTH= list are examined, and PROC LOESS selects the value that minimizes the criterion specified in the SELECT= option.

For models with two or more dependent variables, if the SMOOTH= option is not specified in the MODEL statement, then SMOOTH=0.5 is used as a default.

---

## SCORE Statement

The following new option is available in the SCORE statement after a slash (/).

#### **STEPS**

requests that all models evaluated during smoothing parameter value selection be scored, provided that the SELECT= option together with the STEPS modifier is specified in the MODEL statement. By default only the selected model is scored.

---

## Details

---

### kd Trees and Blending

PROC LOESS uses a kd tree to divide the box (also called the *initial cell* or *bucket*) enclosing all the predictor data points into rectangular cells. The vertices of these cells are the points at which local least squares fitting is done.

Starting from the initial cell, the direction of the longest cell edge is selected as the split direction. The median of this coordinate of the data in the cell is the split value. The data in the starting cell are partitioned into two child cells. The left child consists of all data from the parent cell whose coordinate in the split direction is less than the

split value. The above procedure is repeated for each child cell that has more than a prespecified number of points, called the *bucket size* of the kd tree.

You can specify the bucket size with the BUCKET= option in the MODEL statement. If you do not specify the BUCKET= option, the default value used is the largest integer less than or equal to  $ns/5$ , where  $n$  is the number of observations and  $s$  is the value of the smoothing parameter. Note that if fitting is being done for a range of smoothing parameter values, the bucket size may change for each value.

The set of vertices of all the cells of the kd tree are the points at which PROC LOESS performs its local fitting. The fitted value at an original data point (or at any other point within the original data cell) is obtained by blending the fitted values at the vertices of the kd tree cell that contains that data point.

The univariate blending methods available in PROC LOESS are linear and cubic polynomial interpolation, with linear interpolation being the default. You can request cubic interpolation by specifying the INTERP=CUBIC option in the MODEL statement. In this case, PROC LOESS uses the unique cubic polynomial whose values and first derivatives match those of the fitted local polynomials evaluated at the two endpoints of the kd tree cell edge.

In the multivariate case, such univariate interpolating polynomials are computed on each edge of the kd-tree cells, and are combined using blending functions (Gordon 1971). In the case of two regressors, if you specify INTERP=CUBIC in the MODEL statement, PROC LOESS uses Hermite cubic polynomials as blending functions. If you do not specify INTERP=CUBIC, or if you specify a model with more than two regressors, then PROC LOESS uses linear polynomials as blending functions. In these cases, the blending method reduces to tensor product interpolation from the  $2^p$  vertices of each kd tree cell, where  $p$  is the number of regressors.

While the details of the kd tree and the fitted values at the vertices of the kd tree are implementation details that seldom need to be examined, PROC LOESS does provide options for their display. Each kd tree subdivision of the data used by PROC LOESS is placed in the “kdTree” table. The predicted values at the vertices of each kd tree are placed in the “PredAtVertices” table. You can request these tables using the DETAILS option in the MODEL statement.

---

## Automatic Smoothing Parameter Selection

There are several methodologies for automatic smoothing parameter selection. One class of methods chooses the smoothing parameter value to minimize a criterion that incorporates both the tightness of the fit and model complexity. Such a criterion can usually be written as a function of the error mean square,  $\hat{\sigma}^2$ , and a penalty function designed to decrease with increasing smoothness of the fit. This penalty function is usually defined in terms of the matrix  $L$  such that

$$\hat{y} = Ly$$

where  $y$  is the vector of observed values and  $\hat{y}$  is the corresponding vector of predicted values of the dependent variable. Examples of specific criteria are general-

ized cross-validation (Craven and Wahba 1979) and the Akaike information criterion (Akaike 1973). These classical selectors have two undesirable properties when used with local polynomial and kernel estimators: they tend to undersmooth and tend to be non-robust in the sense that small variations of the input data can change the choice of smoothing parameter value significantly. Hurvich, Simonoff, and Tsai (1998) obtained several bias-corrected AIC criteria that limit these unfavorable properties and perform comparably with the *plug-in selectors* (Ruppert, Sheather, and Wand 1995). PROC LOESS provides automatic smoothing parameter selection using two of these bias-corrected AIC criteria, named  $AIC_{C_1}$  and  $AIC_C$  in Hurvich, Simonoff, and Tsai (1998), and generalized cross-validation, denoted by the acronym GCV.

The relevant formulae are

$$\begin{aligned} AIC_{C_1} &= n \log(\hat{\sigma}^2) + n \frac{\delta_1 / \delta_2 (n + \nu_1)}{\delta_1^2 / \delta_2 - 2} \\ AIC_C &= \log(\hat{\sigma}^2) + 1 + \frac{2 (\text{Trace}(L) + 1)}{n - \text{Trace}(L) - 2} \\ GCV &= \frac{n \hat{\sigma}^2}{(n - \text{Trace}(L))^2} \end{aligned}$$

where  $n$  is the number of observations and

$$\begin{aligned} \delta_1 &\equiv \text{Trace}(I - L)^T (I - L) \\ \delta_2 &\equiv \text{Trace}((I - L)^T (I - L))^2 \\ \nu_1 &\equiv \text{Trace}(L^T L) \end{aligned}$$

You invoke automatic smoothing parameter selection by specifying the `SELECT=criterion` option in the MODEL statement, where *criterion* is one of AICC1, AICC, or GCV. PROC LOESS evaluates the specified criterion for a sequence of smoothing parameter values and selects the value in this sequence that minimizes the specified criterion. If multiple values yield the optimum, then the largest of these values is selected. The results are summarized in the “Smoothing Criterion” table. This table is displayed whenever automatic smoothing parameter selection is performed. You can obtain details of the sequence of models examined by specifying the `DETAILS(MODELSUMMARY)` option in the model statement to display the “Model Summary” table.

There are several ways in which you can control the sequence of models examined by PROC LOESS. If you specify the `SMOOTH=value-list` option in the MODEL statement, then only the values in this list are examined in performing the selection. For example, the following statements select the model that minimizes the AICC1 criterion among the three models with smoothing parameter values 0.1, 0.3, and 0.4:

```
proc loess data=notReal;
  model y= x1/ smooth=0.1 0.3 0.4 select=AICC1;
run;
```

If you do not specify the SMOOTH= option in the model statement, then by default PROC LOESS uses a golden section search method to find a local minimum of the specified criterion in the range (0, 1]. You can use the RANGE(*lower,upper*) modifier in the SELECT= option to change the interval in which the golden section search is performed. For example, the following statements request a golden section search to find a local minimizer of the GCV criterion for smoothing parameter values in the interval [0.1,0.5]:

```
proc loess data=notReal;
  model y= x1/select=GCV( range(0.1,0.5) );
run;
```

If you want to be sure of obtaining a global minimum in the range of smoothing parameter values examined, you can specify the GLOBAL modifier in the SELECT= option. For example, the following statements request that a global minimizer of the AICC criterion be obtained for smoothing parameter values in the interval [0.2, 0.8]:

```
proc loess data=notReal;
  model y= x1/select=AICC( global range(0.2,0.8) );
run;
```

Note that even though the smoothing parameter is a continuous variable, a given range of smoothing parameter values corresponds to a finite set of local models. For example, for a data set with 100 observations, the range [0.2, 0.4] corresponds to models with 20, 21, 22, . . . , 40 points in the local neighborhoods. If the GLOBAL modifier is specified, all possible models in the range are evaluated sequentially.

Note that by default PROC LOESS displays a “Fit Summary” and other optionally requested tables only for the selected model. You can request that these tables be displayed for all models in the selection process by adding the STEPS modifier in the SELECT= option. Also note that by default scoring requested with SCORE statements is done only for the selected model. However, if you specify the STEPS in both the MODEL and SCORE statements, then all models evaluated in the selection process are scored.

In terms of computation,  $AIC_C$  and  $GCV$  depend on the smoothing matrix  $L$  only through its trace. In the direct method, this trace can be computed efficiently. In the interpolated method using kd trees, there is some additional computational cost but the overall work is not significant compared to the rest of the computation. In contrast, the quantities  $\delta_1$ ,  $\delta_2$ , and  $\nu_1$ , which appear in the  $AIC_{C1}$  criterion, depend on the entire  $L$  matrix and for this reason, the time needed to compute these quantities dominates the time required for the model fitting. Hence SELECT=AICC1 is much more computationally expensive than SELECT=AICC and SELECT=GCV, especially when combined with the GLOBAL modifier. Hurvich, Simonoff, and Tsai

(1998) note that  $AIC_C$  can be regarded as an approximation of  $AIC_{C_1}$  and that “the  $AIC_C$  selector generally performs well in all circumstances.”

For models with one dependent variable, PROC LOESS uses `SELECT=AICC` as its default, if you specify neither the `SMOOTH=` nor `SELECT=` options in the `MODEL` statement. With two or more dependent variables, automatic smoothing parameter selection needs to be done separately for each dependent variable. For this reason automatic smoothing parameter selection is not available for models with multiple dependent variables. In such cases you should use a separate PROC LOESS step for each dependent variable, if you want to use automatic smoothing parameter selection.

---

## ODS Table Names

PROC LOESS assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. The following new tables have been added.

**Table 6.2.** ODS Tables Produced by PROC LOESS

| ODS Table Name     | Description                                            | Statement | Option                |
|--------------------|--------------------------------------------------------|-----------|-----------------------|
| ModelSummary       | Summary of all models evaluated                        | MODEL     | DETAILS(ModelSummary) |
| SmoothingCriterion | Criterion value and selected smoothing parameter value | MODEL     | SELECT                |

---

## Example

---

### Example 6.1. Automatic Smoothing Parameter Selection

The following data set contains measurements of monthly averaged atmospheric pressure differences between Easter Island and Darwin, Australia, for a period of 168 months (NIST 1998):

```
data ENSO;
  input Pressure @@;
  Month=_N_;
  format Pressure 4.1;
  format Month 3.0;
datalines;
12.9 11.3 10.6 11.2 10.9 7.5 7.7 11.7
12.9 14.3 10.9 13.7 17.1 14.0 15.3 8.5
5.7 5.5 7.6 8.6 7.3 7.6 12.7 11.0
12.7 12.9 13.0 10.9 10.4 10.2 8.0 10.9
13.6 10.5 9.2 12.4 12.7 13.3 10.1 7.8
4.8 3.0 2.5 6.3 9.7 11.6 8.6 12.4
10.5 13.3 10.4 8.1 3.7 10.7 5.1 10.4
```



```

10.9  11.7  11.4  13.7  14.1  14.0  12.5  6.3
 9.6  11.7   5.0  10.8  12.7  10.8  11.8  12.6
15.7  12.6  14.8   7.8   7.1  11.2   8.1   6.4
 5.2  12.0  10.2  12.7  10.2  14.7  12.2   7.1
 5.7   6.7   3.9   8.5   8.3  10.8  16.7  12.6
12.5  12.5   9.8   7.2   4.1  10.6  10.1  10.1
11.9  13.6  16.3  17.6  15.5  16.0  15.2  11.2
14.3  14.5   8.5  12.0  12.7  11.3  14.5  15.1
10.4  11.5  13.4   7.5   0.6   0.3   5.5   5.0
 4.6   8.2   9.9   9.2  12.5  10.9   9.9   8.9
 7.6   9.5   8.4  10.7  13.6  13.7  13.7  16.5
16.8  17.1  15.4   9.5   6.1  10.1   9.3   5.3
11.2  16.6  15.6  12.0  11.5   8.6  13.8   8.7
 8.6   8.6   8.7  12.8  13.2  14.0  13.4  14.8
;

```

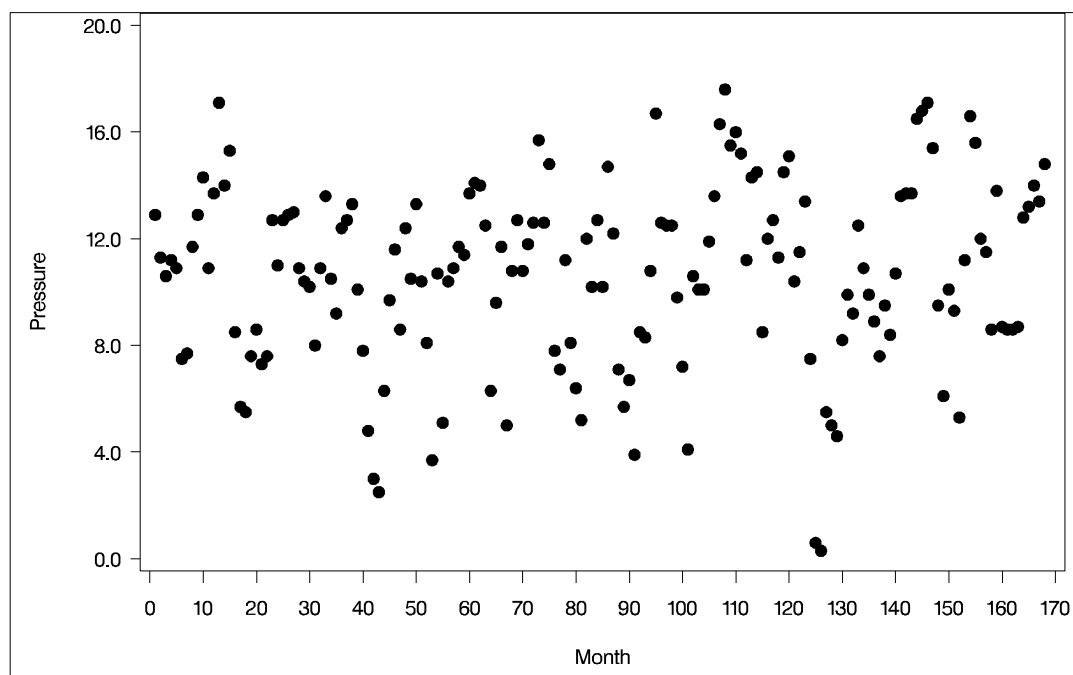
The following PROC Gplot statements produce the simple scatter plot of these data, displayed in Output 6.1.1:

```

symbol1 color=black value=dot ;
proc gplot data=ENSO;
  plot Pressure*Month /
    hminor = 0
    vminor = 0
    vaxis  = axis1
    frame cframe=ligr;
    axis1 label = ( r=0 a=90 ) order=(0 to 20 by 4);
run;

```

Output 6.1.1. Scatter Plot of ENSO Data



You can compute a loess fit and plot the results for these data using the following statements:

```
ods output OutputStatistics=ENS0stats;

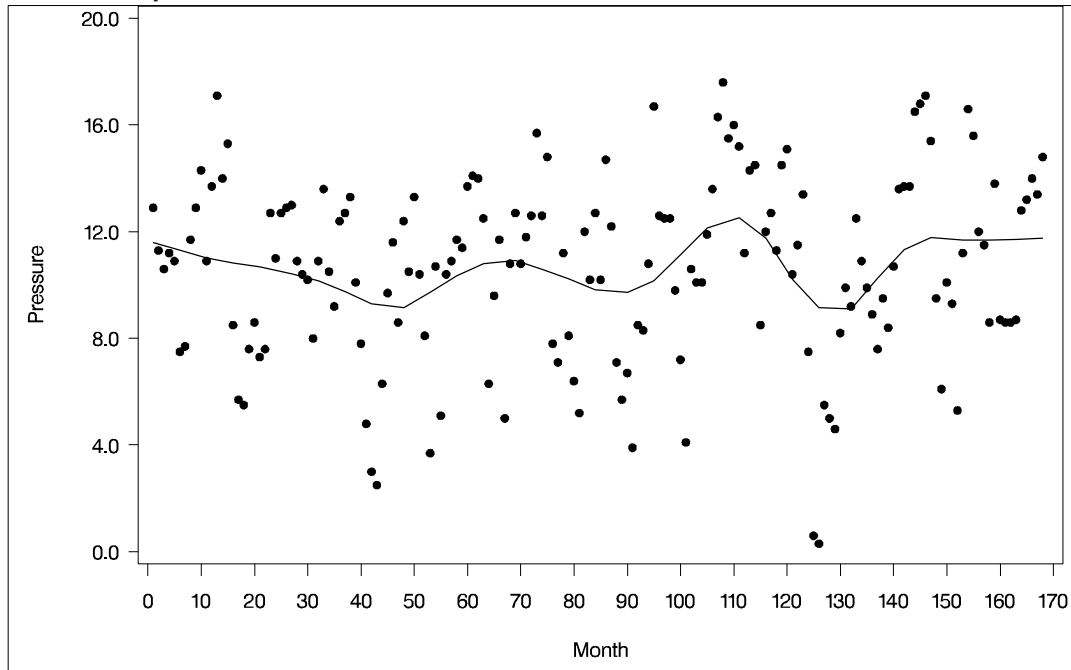
proc loess data=ENS0;
  model Pressure=Month ;
run;

symbol1 color=black value=dot h=2.5 pct;
symbol2 color=black interpol=join value=none width=2;
proc gplot data=ENS0stats;
  plot (depvar pred)*Month / overlay
    hminor = 0
    vminor = 0
    vaxis  = axis1
    frame cframe=ligr;
    axis1 label = ( r=0 a=90 ) order=(0 to 20 by 4);
run; quit;
```

The “Smoothing Criterion” and “Fit Summary” tables are shown in Output 6.1.2 and the fit is plotted in Figure 6.1.3.

**Output 6.1.2.** Output from PROC LOESS

|                                     |                     |
|-------------------------------------|---------------------|
| The LOESS Procedure                 |                     |
| Dependent Variable: Pressure        |                     |
| Optimal Smoothing Criterion         |                     |
| AICC                                | Smoothing Parameter |
| 3.41105                             | 0.22321             |
| The LOESS Procedure                 |                     |
| Selected Smoothing Parameter: 0.223 |                     |
| Dependent Variable: Pressure        |                     |
| Fit Summary                         |                     |
| Fit Method                          | kd Tree             |
| Blending                            | Linear              |
| Number of Observations              | 168                 |
| Number of Fitting Points            | 33                  |
| kd Tree Bucket Size                 | 7                   |
| Degree of Local Polynomials         | 1                   |
| Smoothing Parameter                 | 0.22321             |
| Points in Local Neighborhood        | 37                  |
| Residual Sum of Squares             | 1654.27725          |
| Trace[L]                            | 8.74180             |
| GCV                                 | 0.06522             |
| AICC                                | 3.41105             |

**Output 6.1.3.** Oversmoothed Loess Fit for the ENSO Data

The smoothing parameter value used for the loess fit shown in Figure 6.1.3 was chosen using the default method of PROC LOESS, namely a golden section minimization of the AICC criterion over the interval  $(0, 1]$ . The fit seems to be oversmoothed. What accounts for this poor fit?

One possibility is that the golden section search has found a local rather than a global minimum of the AICC criterion. You can test this by redoing the fit requesting a global minimum. It is also helpful to plot the AICC criterion as a function of the smoothing parameter value used. You do this with the following statements:

```
ods output ModelSummary=ENSOSummary;

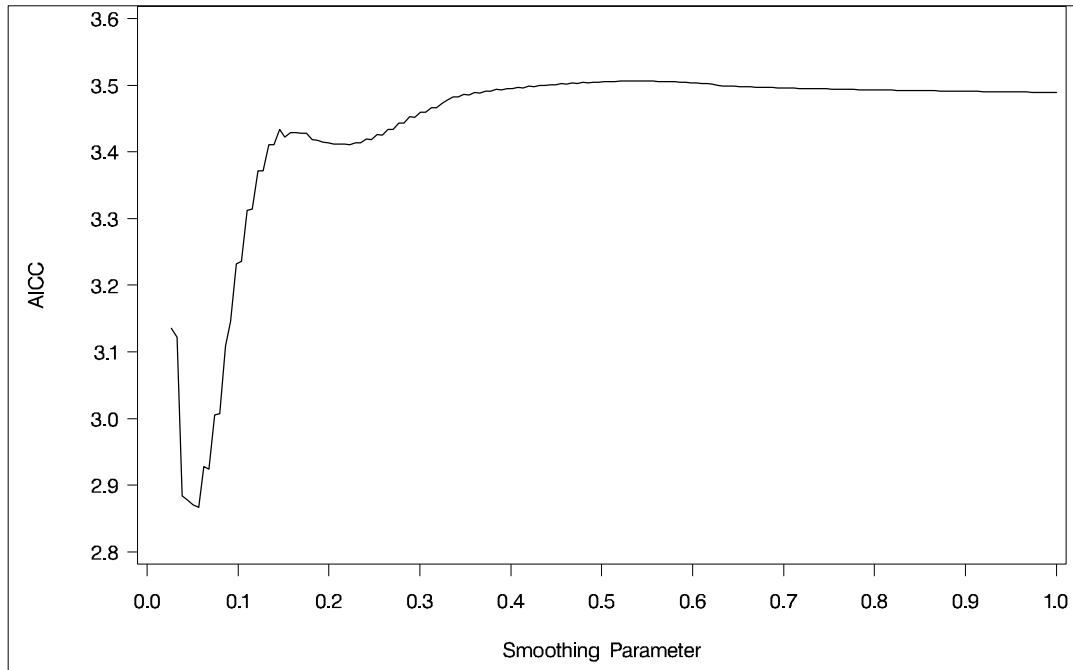
proc loess data=ENSO;
  model Pressure=Month/select=AICC(global);
run;

proc sort data=ENSOSummary;
  by smooth;
run;

symbol1 color=black interpol=join value=none width=2;
proc gplot data=ENSOSummary;
  format AICC f4.1;
  format smooth f4.1;
  plot AICC*Smooth /
    hminor = 0 vminor = 0
    vaxis = axis1 frame cframe=ligr;
    axis1 label = ( r=0 a=90 );
run; quit;
```

The results are shown in Figure 6.1.4.

**Output 6.1.4.** AICC versus Smoothing Parameter Showing Local Minima



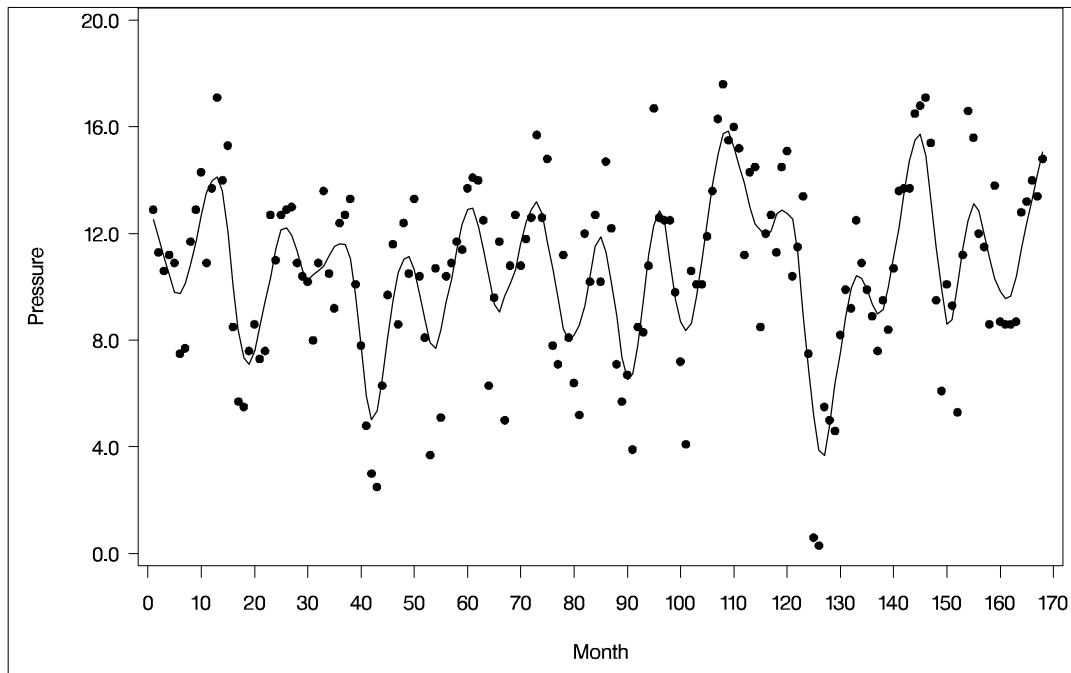
The explanation for the oversmoothed fit in Figure 6.1.3 is now apparent. The golden section search algorithm found the local minimum that occurs near the value 0.22 of the smoothing parameter rather than the global minimum that occurs near 0.06. Note that if you restrict the range of smoothing parameter values examined to lie below 0.2, then the golden section search finds the global minimum as the following statements demonstrate:

```
ods output OutputStatistics=ENS0stats;

proc loess data=ENS0;
  model Pressure=Month/select=AICC( range(0.03,0.2) );
run;

symbol1 color=black value=dot h=2.5 pct;
symbol2 color=black interpol=join value=none width=2;
proc gplot data=ENS0stats;
  plot (depvar pred)*Month / overlay
      hminor = 0
      vminor = 0
      vaxis  = axis1
      frame cframe=ligr;
  axis1 label = ( r=0 a=90 ) order=(0 to 20 by 4);
run; quit;
```

The fit obtained is shown in Figure 6.1.5.

**Output 6.1.5.** Loess Fit for the ENSO Data

The loess fit shown in Figure 6.1.5 clearly shows an annual cycle in the data. An interesting question is whether there is some phenomenon captured in the data that would explain the presence of the local minimum near 0.22 in the AICC curve. Note that there is some evidence of a cycle of about 42 months in the oversmoothed fit in Figure 6.1.3. You can see this cycle because the strong annual cycle in Figure 6.1.5 has been smoothed out. The physical phenomenon that accounts for the existence of this cycle has been identified as the periodic warming of the Pacific Ocean known as “El Niño.”

---

## References

- Akaike, H. (1973), “Information Theory and an Extension of the Maximum Likelihood Principle,” in *Proceedings of the Second International Symposium on Information Theory*, eds. Petrov and Csaki, 267–281.
- Craven, P. and Wahba, G. (1979), “Smoothing Noisy Data with Spline Functions,” *Numerical Mathematics*, 31, 377–403.
- Gordon, W.J. (1971) “Blending-function Methods of Bivariate and Multivariate Interpolation and Approximation,” *SIAM Journal of Numerical Analysis*, 8, No. 1, 158–177.
- Hurvich, C.M., Simonoff, J.S., and Tsai, C.L. (1998), “Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion” *Journal of the Royal Statistical Society B*, 60, 271–293.
- NIST (1998), “Statistical Reference Data Sets,” [<http://www.nist.gov/itl/div898/strd/>], accessed 20 January 1999.

- Ruppert, D., Sheather, S.J., and Wand, M.P. (1995), “An Effective Bandwidth Selector for Local Least Squares Regression,” *Journal of the American Statistical Association*, 90, 1257–1270.



# Chapter 7

## The LOGISTIC Procedure

### Chapter Table of Contents

---

|                                                |     |
|------------------------------------------------|-----|
| <b>OVERVIEW</b> . . . . .                      | 91  |
| <b>SYNTAX</b> . . . . .                        | 91  |
| PROC LOGISTIC Statement . . . . .              | 91  |
| EXACT Statement . . . . .                      | 92  |
| <b>DETAILS</b> . . . . .                       | 94  |
| Exact Conditional Analysis . . . . .           | 94  |
| OUTDIST= Output Data Set . . . . .             | 98  |
| Displayed Output . . . . .                     | 99  |
| ODS Table Names . . . . .                      | 100 |
| <b>EXAMPLES</b> . . . . .                      | 101 |
| Example 7.1 Dose-Response Study . . . . .      | 101 |
| Example 7.2 Crossover Clinical Trial . . . . . | 105 |
| <b>REFERENCES</b> . . . . .                    | 106 |





# Chapter 7

## The LOGISTIC Procedure

---

### Overview

Exact conditional inference capability for binary (dichotomous) response variables is now available in the LOGISTIC procedure. Exact methods can be useful for analyzing data sets for which the usual asymptotic assumptions are inadequate; for example, when sample sizes are small or the data are sparse or skewed.

### Syntax

The additional syntax to the LOGISTIC procedure consists of one or more EXACT statements for specifying the effects to be analyzed and the EXACTONLY and (global) EXACTOPTIONS options in the PROC LOGISTIC statement. You still need the MODEL statement to specify the effects in the model. Loosely speaking, the exact inference on the effects in the EXACT statement are conditional on all other effects in the MODEL statement.

### PROC LOGISTIC Statement

The following options are added to the PROC LOGISTIC statement:

**EXACTONLY**

requests only the exact analyses. The unconditional likelihood analysis that PROC LOGISTIC usually performs is suppressed.

**EXACTOPTIONS(*options*)**

specifies options that apply to every EXACT statement in the program. The following options are available:

**MAXTIME=seconds** specifies the maximum clock time (in seconds) that PROC LOGISTIC can use to calculate the permutation distributions. If the limit is exceeded, the procedure halts all computations. The default maximum clock time is seven days.

**STATUSTIME=seconds** specifies the time interval (in seconds) for printing a status line in the SAS Log. You can use this status line to track the progress of the computation of the exact conditional distributions. The time interval you specify is a lower limit; the actual time interval may vary for larger problems. By default, no status reports are produced.

---

## EXACT Statement

**EXACT** <'label'> < Intercept > < effects > < / options > ;

The EXACT statement performs the exact tests of the parameters for the specified effects and optionally estimates the parameters and outputs the exact conditional distributions. You can specify the keyword INTERCEPT and any effects in the MODEL statement. Inference on parameters for the specified effects is performed by conditioning on the sufficient statistics of all the other model parameters (possibly including the intercept).

You can specify several EXACT statements, but they must follow the MODEL statement. Each statement can optionally include an identifying label. If several EXACT statements are specified, any statement without a label is assigned a label of the form "Exact $n$ ", where " $n$ " indicates the  $n$ th EXACT statement. The label is included in the headers of the exact analysis tables.

The CONTRAST, TEST, and UNITS statements do not apply to the exact analyses. Exact analyses are not performed when you specify a WEIGHT statement, a link other than LINK=LOGIT, an offset variable, the NOFIT option, or a model-selection method.

The effects specified in the MODEL statement can have at most 32 parameters. If you have too many observations with the same covariate values, an integer overflow may occur and computations will cease.

The following options can be specified in each EXACT statement after a slash (/):

### **ALPHA= $p$**

specifies the level of significance  $p$  for  $100(1 - p)\%$  confidence limits for the parameters or odds ratios. The value  $p$  must be between 0 and 1. If the ALPHA= option is not specified,  $p$  is equal to 0.05 or the value of the ALPHA= option in the PROC LOGISTIC statement.

### **ESTIMATE** < =keyword >

estimates the individual parameters (conditional on all other parameters) for the effects specified in the EXACT statement. For each parameter, a point estimate, a confidence interval, and a  $p$ -value for a two-sided test that the parameter is zero are displayed. Note that the two-sided  $p$ -value is twice the one-sided  $p$ -value. You can optionally specify one of the following keywords:

**PARM** specifies that the parameters be estimated. This is the default.

**ODDS** specifies that the odds ratios be estimated. For classification variables, use of the reference parameterization (the PARAM=REF option in the CLASS statement) is recommended.

**BOTH** specifies that the parameters and odds ratios be estimated.

### **JOINT**

performs the joint test that all of the parameters for the EXACT statement are simul-

taneously equal to zero along with tests that the parameters of the individual variables are zero. The joint test of all the parameters is indicated in the “Conditional Exact Tests” table by the label “Joint.”

#### **JOINTONLY**

performs only the joint test that all of the parameters for the EXACT statement are simultaneously equal to zero. The test is indicated in the “Conditional Exact Tests” table by the label “Joint.” When this option is specified, tests for the parameters of the individual variables are not performed.

#### **ONESIDED**

requests one-sided confidence intervals and  $p$ -values for the individual parameter estimates and odds ratios. The one-sided  $p$ -value is the smaller of the left and right tail probabilities for the observed sufficient statistic for the parameter under the null hypothesis that the parameter is zero. The two-sided  $p$ -values (default) are twice the one-sided  $p$ -values.

#### **OUTDIST=SAS-data-set**

names the SAS data set containing the exact conditional distributions. This data set contains all of the exact conditional distributions required to process the corresponding EXACT statement. The data set contains the possible sufficient statistics for the parameters of the effects specified in the EXACT statement, the counts, and, when hypothesis tests are performed on the parameters, the probability of occurrence and the score value for each sufficient statistic. When you request an OUTDIST= data set, the observed sufficient statistics are displayed in the “Sufficient Statistics” table. See the “OUTDIST= Output Data Set” section on page 98 for more information.

### **EXACT Statement Examples**

- In the following example, two exact tests are computed: one is for  $x_1$  and the other is for  $x_2$ . The test for  $x_1$  is based on the exact conditional distribution of the sufficient statistic for the  $x_1$  parameter given the observed values of the sufficient statistics for the intercept,  $x_2$ , and  $x_3$  parameters; likewise, the test for  $x_2$  is conditional on the observed sufficient statistics for the intercept,  $x_1$ , and  $x_3$ :

```
proc logistic;
  model y = x1 x2 x3;
  exact x1 x2;
run;
```

- If  $x_3$  is a CLASS variable with three levels and the reference parameterization is specified, then there are two associated parameters. If you submit the C13 statement, an individual test that the parameter of  $x_1$  is zero and a joint test for testing that the two parameters of  $x_3$  are both zero is performed. If you specify the CJ13 statement, both of these tests and a joint test that all the parameters for the EXACT statement are zero are performed. If you request the CE3 statement, then a joint test for testing that the two parameters of  $x_3$  are both zero is performed, and estimates for each of the two parameters are produced:

```
class x3 / param=ref;
exact 'C13' x1 x3;
exact 'CJ13' x1 x3 / joint;
exact 'CE3' x3 / estimate;
```

- You can specify multiple EXACT statements in the same PROC LOGISTIC invocation. PROC LOGISTIC determines, from all the EXACT statements, the distinct conditional distributions that need to be evaluated. For example, only one exact conditional distribution is required for the following two EXACT statements:

```
exact x1 / estimate=parm;
exact x1 / estimate=parm onesided;
```

- For each EXACT statement, individual tests for the parameters of the specified effects are computed unless the JOINTONLY option is specified. Consider the following EXACT statements:

```
exact 'E12' x1 x2 / estimate;
exact 'E1'  x1    / estimate;
exact 'E2'  x2    / estimate;
exact 'J12' x1 x2 / joint;
```

In the E12 statement, the parameters for x1 and x2 are estimated and tested separately. Specifying the E12 statement is equivalent to specifying both the E1 and E2 statements. In the J12 statement, the joint test for the parameters of x1 and x2 is computed as well as the individual tests for x1 and x2.

- All exact conditional distributions for the tests and estimates computed in a single EXACT statement are output to the corresponding OUTDIST= data set. For example, consider the following EXACT statements:

```
exact 'O1'  x1    / outdist=o1;
exact 'OJ12' x1 x2 / jointonly outdist=oj12;
exact 'OA12' x1 x2 / joint    outdist=oa12;
exact 'OE12' x1 x2 / estimate outdist=oe12;
```

The O1 statement outputs a single exact conditional distribution. The OJ12 statement outputs only the joint distribution for x1 and x2. The OA12 statement outputs three conditional distributions: one for x1, one for x2, and one jointly for x1 and x2. The OE12 statement outputs two conditional distributions: one for x1 and the other for x2. Data set oe12 contains both the x1 and x2 variables; the distribution for x1 has missing values in the x2 column while the distribution for x2 has missing values in the x1 column.

See the “OUTDIST= Output Data Set” section on page 98 for more information.

---

## Details

---

### Exact Conditional Analysis

Asymptotic methods may be inadequate when sample sizes are small or the data are sparse or skewed. In such cases, the asymptotic  $p$ -values are not close approximations to the true  $p$ -values. Exact conditional inference remains valid in such situations.

The following sections summarize the unconditional and conditional analyses, discuss the network algorithm used in the computations and the necessary computational resources, and provide details on the statistics displayed in the output tables.

### Background

The theory of exact conditional logistic regression analysis was originally laid out by Cox (1970). Other useful references include Cox and Snell (1989), Agresti (1990, 1992), and Mehta and Patel (1995).

Consider  $N$  independent Bernoulli random variables  $Y_1, \dots, Y_N$  having observed values  $\mathbf{y}_0 = (y_{01}, \dots, y_{0N})'$ . For each observation  $i = 1, \dots, N$ , let  $\mathbf{x}_i = (x_{i1}, \dots, x_{in}, x_{i,n+1}, \dots, x_{i,n+m})'$  be an  $n + m = s$  vector of explanatory variables, and denote  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)'$ . Let  $p_i = p(\mathbf{x}_i) = \Pr(Y_i = 1 | \mathbf{x}_i)$  be the event probability for each  $i = 1, \dots, N$ , and denote  $\mathbf{p} = (p_1, \dots, p_N)'$ . Then the logistic regression model is  $\text{logit}(\mathbf{p}) = \mathbf{X}\boldsymbol{\beta}$ , or

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1 - p_i}\right) = \mathbf{x}_i' \boldsymbol{\beta}$$

where  $\boldsymbol{\beta} = (\beta_1, \dots, \beta_s)'$  is the unknown parameter vector.

Unconditional likelihood inference is based on maximizing the likelihood function:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^N p_i^{y_{0i}} (1 - p_i)^{1 - y_{0i}} = \frac{\exp(\mathbf{y}_0' \mathbf{X} \boldsymbol{\beta})}{\prod_{i=1}^N [1 + \exp(\mathbf{x}_i' \boldsymbol{\beta})]}$$

To perform conditional inference, first observe that the sufficient statistics for the  $\beta_j$  in the unconditional likelihood function are the corresponding  $T_j = \sum_{i=1}^N y_i x_{ij}$ , where  $y_i$  is a realization of  $Y_i$ . To create the probability density function (pdf) for  $\mathbf{T} = (T_1, \dots, T_s)'$ , sum over all binary sequences  $\mathbf{y}$  that generate an observable  $\mathbf{t}$ :

$$\Pr(\mathbf{T} = \mathbf{t}) = \frac{C(\mathbf{t}) \exp(\mathbf{t}' \boldsymbol{\beta})}{\prod_{i=1}^N [1 + \exp(\mathbf{x}_i' \boldsymbol{\beta})]}$$

where  $C(\mathbf{t}) = ||\{\mathbf{y} : \mathbf{y}' \mathbf{X} = \mathbf{t}\}||$  is the number of sequences  $\mathbf{y}$  that generate  $\mathbf{t}$ . Suppose the  $m$  parameters  $\boldsymbol{\beta}_2 = (\beta_{n+1}, \dots, \beta_{n+m})'$  are *nuisance* parameters; that is, the current analysis is geared toward the first  $n$  parameters  $\boldsymbol{\beta}_1 = (\beta_1, \dots, \beta_n)'$ . Denote the sufficient statistics for the parameters of interest as  $\mathbf{T}_1 = (T_1, \dots, T_n)'$ , the corresponding observed values as  $\mathbf{t}_1$ , and the corresponding columns of  $\mathbf{X}$  as  $\mathbf{X}_1$ . Similarly, define  $\mathbf{T}_2$ ,  $\mathbf{t}_2$ , and  $\mathbf{X}_2$  for the nuisance parameters. The nuisance parameters can be removed from the analysis by conditioning on their sufficient statistics to create the conditional likelihood

$$\Pr(\mathbf{T}_1 = \mathbf{t}_1 | \mathbf{T}_2 = \mathbf{t}_2) = \frac{\Pr(\mathbf{T} = \mathbf{t})}{\Pr(\mathbf{T}_2 = \mathbf{t}_2)} = \frac{C(\mathbf{t}) \exp(\mathbf{t}_1' \boldsymbol{\beta}_1)}{\sum_{\mathbf{u}} C(\mathbf{u}, \mathbf{t}_2) \exp(\mathbf{u}' \boldsymbol{\beta}_1)}$$

where  $C(\mathbf{u}, \mathbf{t}_2)$  is the number of vectors  $\mathbf{y}$  such that  $\mathbf{y}' \mathbf{X}_1 = \mathbf{u}$  and  $\mathbf{y}' \mathbf{X}_2 = \mathbf{t}_2$ .

Conditional asymptotic inference is performed by maximizing the conditional likelihood and producing conditional statistics similar to those for the unconditional likelihood case.

Conditional exact inference is based on generating the conditional distribution for the parameters of interest. This distribution is called the *permutation* or *exact conditional* distribution. The conditional pdf  $\Pr(\mathbf{T}_1 = \mathbf{t}_1 | \mathbf{T}_2 = \mathbf{t}_2)$  is denoted as  $f_{\beta_1}(\mathbf{t}_1 | \mathbf{t}_2)$ .

### Computational Algorithm

The goal of the exact conditional analysis is to determine how likely the observed response  $\mathbf{y}_0$  is with respect to all  $2^N$  possible responses  $\mathbf{y} = (y_1, \dots, y_N)'$ . One way to proceed is to generate every  $\mathbf{y}$  vector for which  $\mathbf{y}'\mathbf{X}_2 = \mathbf{t}_2$  and count the number of vectors  $\mathbf{y}$  for which  $\mathbf{y}'\mathbf{X}_1$  is equal to each unique  $\mathbf{t}_1$ . Generating the conditional distribution from complete enumeration of the joint distribution is conceptually simple; however, this method becomes computationally infeasible very quickly. For example, if you had only 30 observations, you'd have to scan through  $2^{30}$  different  $\mathbf{y}$  vectors.

PROC LOGISTIC employs a network algorithm developed by Hirji, Mehta, and Patel (1987) to generate and count the  $\mathbf{y}$  vectors. The algorithm is based on the following observation: given any  $\mathbf{y} = (y_1, \dots, y_N)'$  and a design  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)'$ , let  $\mathbf{y}_{(i)} =$

$$(y_1, \dots, y_i)' \text{ and } \mathbf{X}_{(i)} = (\mathbf{x}_1, \dots, \mathbf{x}_i)' = \begin{pmatrix} x_{1,1} & \dots & x_{1,s} \\ \vdots & & \vdots \\ x_{i,1} & \dots & x_{i,s} \end{pmatrix} \text{ be the first } i \text{ rows of}$$

each matrix. Write the sufficient statistic based on these  $i$  rows as  $\mathbf{t}_{(i)}' = \mathbf{y}_{(i)}'\mathbf{X}_{(i)}$ . A recursion relation results:  $\mathbf{t}_{(i+1)} = \mathbf{t}_{(i)} + y_{i+1}\mathbf{x}_{i+1}$ . Combining this relation with a method of determining which  $\mathbf{y}$  vectors produce the  $\mathbf{t}_2$  margins makes the generation of the permutation distribution feasible.

The bulk of the computation time and memory is consumed by the creation of the exact joint distribution. After the joint distribution for a set of effects is created, the computational effort required to produce hypothesis tests and parameter estimates for any subset of the effects is (relatively) trivial.

### Computational Resources

PROC LOGISTIC uses a relatively fast and efficient algorithm for the exact analyses. This recently developed algorithm, together with improvements in computer power, now make it feasible to perform exact computations for data sets where previously only asymptotic methods could be applied. Nevertheless, there are still large problems that may require a prohibitive amount of time and memory for exact computations, depending on the speed and memory available on your computer. For large problems, consider whether exact methods are really needed or whether asymptotic methods might give results quite close to the exact results, while requiring much less computer time and memory.

A formula does not exist that can predict the amount of time and memory necessary to generate the permutation distributions for a particular problem. The time and memory required depend on several factors, including the total sample size, the number of parameters of interest, the number of nuisance parameters, and the order in which the parameters are processed. If you run out of memory, refer to the SAS Companion for your system to see how to allocate more.

You can use the MAXTIME= option in the EXACTOPTIONS option to limit the total amount of time PROC LOGISTIC uses to derive all of the exact distributions. If PROC LOGISTIC does not finish within that time, the procedure terminates. If you

need to derive several distributions, it may be more feasible to request one distribution at a time.

At any time while PROC LOGISTIC is deriving the distributions, you can terminate the computations by pressing the system interrupt key sequence (refer to the SAS Companion for your system) and choosing to stop computations.

### Hypothesis Tests

Consider testing the null hypothesis  $H_0: \beta_1 = \mathbf{0}$  against the alternative  $H_A: \beta_1 \neq \mathbf{0}$ , conditional on  $\mathbf{T}_2 = \mathbf{t}_2$ . Under the null hypothesis, the test statistic for the *exact probability test* is just  $f_{\beta_1=0}(\mathbf{t}_1|\mathbf{t}_2)$ , while the corresponding  $p$ -value is the probability of getting a less likely (more extreme) statistic,

$$p(\mathbf{t}_1|\mathbf{t}_2) = \sum_{\mathbf{u} \in \Omega_p} f_0(\mathbf{u}|\mathbf{t}_2)$$

where  $\Omega_p = \{\mathbf{u}: \text{for some } \mathbf{y}, \mathbf{y}'\mathbf{X}_1 = \mathbf{u}, \mathbf{y}'\mathbf{X}_2 = \mathbf{t}_2, \text{ and } f_0(\mathbf{u}|\mathbf{t}_2) \leq f_0(\mathbf{t}_1|\mathbf{t}_2)\}$ .

For the *exact conditional scores test*, the conditional mean  $\boldsymbol{\mu}_1$  and variance matrix  $\boldsymbol{\Sigma}_1$  of the  $\mathbf{T}_1$  (conditional on  $\mathbf{T}_2 = \mathbf{t}_2$ ) are calculated, and the score statistic for the observed value,

$$s = (\mathbf{t}_1 - \boldsymbol{\mu}_1)' \boldsymbol{\Sigma}_1^{-1} (\mathbf{t}_1 - \boldsymbol{\mu}_1)$$

is compared to the score for each member of the distribution

$$S(\mathbf{T}_1) = (\mathbf{T}_1 - \boldsymbol{\mu}_1)' \boldsymbol{\Sigma}_1^{-1} (\mathbf{T}_1 - \boldsymbol{\mu}_1)$$

The resulting  $p$ -value is

$$p(\mathbf{t}_1|\mathbf{t}_2) = Pr(S \geq s) = \sum_{\mathbf{u} \in \Omega_s} f_0(\mathbf{u}|\mathbf{t}_2)$$

where  $\Omega_s = \{\mathbf{u}: \text{for some } \mathbf{y}, \mathbf{y}'\mathbf{X}_1 = \mathbf{u}, \mathbf{y}'\mathbf{X}_2 = \mathbf{t}_2, \text{ and } S(\mathbf{u}) \geq s\}$ .

The mid- $p$  statistic was originally proposed by Lancaster (1961) to compensate for the discreteness of the distribution. Hirji, Tsiatis, and Mehta (1989) recommend its use with small or sparse data sets; Vollset, Hirji, and Afifi (1991) suggest the statistic for matched case-control studies; and Hirji and Tang (1998) recommend the statistic for tests of trend. The mid- $p$  is defined as

$$\frac{1}{2} f_{\beta_1}(\mathbf{t}_1|\mathbf{t}_2) + \sum_{\mathbf{u} \in \Omega} f_{\beta_1}(\mathbf{u}|\mathbf{t}_2)$$

where  $\Omega$  is either  $\Omega_s$  or  $\Omega_p$  except that strict inequalities are used.

### Inference for a Single Parameter

Exact parameter estimates are derived for a single parameter  $\beta_i$  by regarding all the other parameters  $\boldsymbol{\beta}_2 = (\beta_1, \dots, \beta_{i-1}, \beta_{i+1}, \dots, \beta_s)'$  as nuisance parameters. The appropriate sufficient statistics are  $\mathbf{T}_1 = T_i$  and  $\mathbf{T}_2 = (T_1, \dots, T_{i-1}, T_{i+1}, \dots, T_s)'$ , with



their observed values denoted by the lowercase  $t$ . Hence, the conditional pdf used to create the parameter estimate for  $\beta_i$  is

$$f_{\beta_i}(t_i|\mathbf{t}_2) = \frac{C(t_i, \mathbf{t}_2) \exp(t_i \beta_i)}{\sum_{u \in \Omega} C(u, \mathbf{t}_2) \exp(u \beta_i)}$$

for  $\Omega = \{u: \text{for some } \mathbf{y}, T_i = u \text{ and } \mathbf{T}_2 = \mathbf{t}_2\}$ .

The maximum exact conditional likelihood estimate is the quantity  $\hat{\beta}_i$  which maximizes the conditional pdf. A Newton-Raphson algorithm is used to perform this search. However, if the observed  $t_i$  attains either its minimum or maximum value in the permutation distribution (that is, either  $t_i = \min\{u : u \in \Omega\}$  or  $t_i = \max\{u : u \in \Omega\}$ ), then the conditional pdf is monotonically increasing in  $\beta_i$  and cannot be maximized. In this case, a median unbiased estimate (Hirji, Tsiatis, and Mehta 1989; Hirji and Tang 1998)  $\hat{\beta}_i$  is produced that satisfies  $\sum_u f_{\hat{\beta}_i}(u|\mathbf{t}_2) = \frac{1}{2}$ , where the sum is over all possible values of  $T_i$ , and a Newton-Raphson-type algorithm is used to perform the search.

Likelihood ratio tests based on the conditional pdf are used to test the null  $H_0: \beta_i = 0$  against various alternatives. For testing against the alternative  $H_A: \beta_i > 0$ , the critical region for the UMP test consists of the upper tail of values for  $T_i$  in the permutation distribution. Thus, the one-sided significance level  $p_G(t_i; 0)$  is the probability of a more extreme (greater) value:

$$p_G(t_i; 0) = \sum_{u \geq t_i} f_0(u|\mathbf{t}_2)$$

The one-sided significance level  $p_L(t_i; 0)$  against  $H_A: \beta_i < 0$  is

$$p_L(t_i; 0) = \sum_{u \leq t_i} f_0(u|\mathbf{t}_2)$$

The minimum of these one-sided levels is reported when the ONESIDED option is specified. The two-sided significance level  $p(t_i; 0)$  against  $H_A: \beta_i \neq 0$  is calculated as

$$p(t_i; 0) = 2 \min[p_L(t_i; 0), p_G(t_i; 0)]$$

An upper  $100(1 - 2\epsilon)\%$  confidence limit for  $\hat{\beta}_i$  corresponding to the observed  $t_i$  is the solution  $\beta_U(t_i)$  of  $\epsilon = p_L(t_i, \beta_U(t_i))$ , while the lower confidence limit is the solution  $\beta_L(t_i)$  of  $\epsilon = p_G(t_i, \beta_L(t_i))$ . A Newton-Raphson procedure is used to search for the solutions.

---

## OUTDIST= Output Data Set

The OUTDIST= data set contains every exact conditional distribution necessary to process the EXACT statement. For example, the following statements create one distribution for the x1 parameter and another for the x2 parameters, and produces the data set dist shown in the following table:

```

proc logistic;
  class x2 / param=ref;
  model y=x1 x2;
  exact x1 x2 / outdist=dist;
proc print data=dist;
run;

```

| Obs | x1 | x20 | x21 | Count | Score   | Prob    |
|-----|----|-----|-----|-------|---------|---------|
| 1   | .  | 0   | 0   | 3     | 5.81151 | 0.03333 |
| 2   | .  | 0   | 1   | 15    | 1.66031 | 0.16667 |
| 3   | .  | 0   | 2   | 9     | 3.12728 | 0.10000 |
| 4   | .  | 1   | 0   | 15    | 1.46523 | 0.16667 |
| 5   | .  | 1   | 1   | 18    | 0.21675 | 0.20000 |
| 6   | .  | 1   | 2   | 6     | 4.58644 | 0.06667 |
| 7   | .  | 2   | 0   | 19    | 1.61869 | 0.21111 |
| 8   | .  | 2   | 1   | 2     | 3.27293 | 0.02222 |
| 9   | .  | 3   | 0   | 3     | 6.27189 | 0.03333 |
| 10  | 2  | .   | .   | 6     | 3.03030 | 0.12000 |
| 11  | 3  | .   | .   | 12    | 0.75758 | 0.24000 |
| 12  | 4  | .   | .   | 11    | 0.00000 | 0.22000 |
| 13  | 5  | .   | .   | 18    | 0.75758 | 0.36000 |
| 14  | 6  | .   | .   | 3     | 3.03030 | 0.06000 |

The first nine observations in the `dist` data set contain a permutation distribution for the parameters of the `x2` effect (hence the values for the `x1` parameter are missing), and the remaining five observations are for the `x1` parameter. If a joint distribution was created, there would be observations with values for both the `x1` and `x2` parameters. For CLASS variables, the corresponding parameters in the `dist` data set are identified by concatenating the variable name with the appropriate classification level.

The data set contains the possible sufficient statistics of the parameters for the effects specified in the EXACT statement, and the `Count` variable contains the number of different responses that yield these statistics. For example, there were six possible response vectors  $\mathbf{y}$  for which the product  $\mathbf{y}'\mathbf{x}_1$  was equal to 2, and for which  $\mathbf{y}'\mathbf{x}_{20}$ ,  $\mathbf{y}'\mathbf{x}_{21}$ , and  $\mathbf{y}'\mathbf{1}$  were equal to their actual observed values (displayed in the “Sufficient Statistics” table). When hypothesis tests are performed on the parameters, the `Prob` variable contains the probability of obtaining that statistic (which is just the count divided by the total count), and the `Score` variable contains the score for that statistic. For more information, see the “EXACT Statement Examples” section on page 93.

---

## Displayed Output

The displayed exact conditional analysis output of the LOGISTIC procedure includes the following:

- the “Conditional Exact Tests” table provides two tests for the null hypothesis that the parameters for the specified effects are zero: the exact probability test and the exact conditional scores test. For each test, the test statistic, an exact  $p$ -value (the probability of obtaining a more extreme statistic than the observed,

assuming the null hypothesis), and a mid  $p$ -value (adjusts for the discreteness of the distribution) are displayed.

Individual hypothesis tests for the parameter of each continuous effect, and joint tests for the parameters of classification variables, are generated by default. A joint test for all effects may be requested with the JOINT or JOINTONLY options.

- if you specify the ESTIMATE, ESTIMATE=PARM, or ESTIMATE=BOTH options, the “Exact Parameter Estimates” table displays individual parameter estimates for each parameter conditional on the values of all the other parameters in the model. These are either the exact conditional maximum likelihood estimates (MLE) or, in cases where the conditional MLE does not exist, the median unbiased estimates.

Also displayed are one-sided or two-sided confidence limits for the estimate, and a one-sided or two-sided  $p$ -value for testing that the parameter estimate is zero. The one-sided  $p$ -value is the smaller of the left and right tail probabilities for the observed sufficient statistic for the parameter under the null hypothesis that the parameter is zero, while the two-sided  $p$ -value is twice the one-sided  $p$ -value.

- if you specify the ESTIMATE=ODDS or ESTIMATE=BOTH options, the “Exact Odds Ratios” table displays odds ratios for individual parameters, confidence limits, and a  $p$ -value for testing that the odds ratio is 1.
- if you request an OUTDIST= data set, the “Sufficient Statistics” table is displayed before printing any of the exact analysis results. The table lists the parameters and their observed sufficient statistics.

---

## ODS Table Names

PROC LOGISTIC assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. The names for the exact conditional analyses are listed in the following table.

**Table 7.1.** ODS Tables Produced in PROC LOGISTIC

| ODS Table Name | Description             | Statement | Option                                       |
|----------------|-------------------------|-----------|----------------------------------------------|
| ExactTests     | Conditional Exact Tests | EXACT     | default                                      |
| ExactParmEst   | Parameter Estimates     | EXACT     | ESTIMATE,<br>ESTIMATE=PARM,<br>ESTIMATE=BOTH |
| ExactOddsRatio | Exact Odds Ratios       | EXACT     | ESTIMATE=ODDS,<br>ESTIMATE=BOTH              |
| SuffStats      | Sufficient Statistics   | EXACT     | OUTDIST=                                     |

---

## Examples

The following examples illustrate different types of exact analysis. Example 7.1 illustrates how to use exact conditional analysis and how to adjust for within-strata correlation. It also discusses how the MLE for the unconditional likelihood analysis may not exist, rendering the asymptotic inference impossible, while the exact conditional inference is still plausible. Example 7.2 is a phase II analysis for the pharmaceutical industry.

---

### Example 7.1. Dose-Response Study

Researchers are interested in analyzing how mortality rates change with respect to dosage of a drug. The data contains life/death outcomes for six levels of drug dosage (0 to 5). One subject in each of three different research centers (A to C) is given a specific dose of the drug, and the **Response** is recorded as 0 for survival and 1 for death; the **Z** variable will be used in a later analysis:

```
data dose;
  input Center $ Dose Response @@;
  Z = 2 - Response;
  datalines;
A 0 0      B 0 0      C 0 0
A 1 0      B 1 0      C 1 0
A 2 0      B 2 0      C 2 0
A 3 0      B 3 0      C 3 0
A 4 1      B 4 0      C 4 0
A 5 1      B 5 0      C 5 1
;
```

Since all of the cells have counts that are less than 5, the applicability of large sample theory is questionable. For each subject  $i$  receiving dosage  $x_i$ ,  $i = 1, \dots, 18$ , let  $Y_i = 1$  if the subject died,  $Y_i = 0$  otherwise, and  $\pi_i = \Pr(Y_i = 1|x_i)$ . Then the linear logistic model for this problem is  $\text{logit}(\pi_i) = \log\left(\frac{\pi_i}{1-\pi_i}\right) = \alpha + x_i\beta$ , which fits a common intercept and slope for the  $i$  subjects. In the PROC LOGISTIC invocation below, the EXACT statement requests an exact analysis and the ESTIMATE option produces exact parameter estimates:

```
proc logistic data=dose descending;
  model Response = Dose;
  exact Dose / estimate=both;
run;
```

Output 7.1.1 displays some of the unconditional asymptotic results that are produced by default. The likelihood ratio and score tests reject the null hypothesis that  $\beta$  is zero. However, the Wald test does not reject this null hypothesis. The conflicting conclusions of these tests are a telltale sign that the large sample approximation is unreliable. The estimates for the intercept  $\alpha$  and the slope  $\beta$  both have  $p$ -values greater than 0.05, indicating marginal influence. The confidence limits for the odds

ratio of the dose parameter contains the value 1, from which you could conclude, if you accept the model, that there is no change in mortality with a change in dosage.

**Output 7.1.1.** Output from Asymptotic Analysis

| The LOGISTIC Procedure                   |    |                |                            |            |            |
|------------------------------------------|----|----------------|----------------------------|------------|------------|
| Testing Global Null Hypothesis: BETA=0   |    |                |                            |            |            |
| Test                                     |    | Chi-Square     | DF                         | Pr > ChiSq |            |
| Likelihood Ratio                         |    | 8.1478         | 1                          | 0.0043     |            |
| Score                                    |    | 5.7943         | 1                          | 0.0161     |            |
| Wald                                     |    | 2.7249         | 1                          | 0.0988     |            |
| Analysis of Maximum Likelihood Estimates |    |                |                            |            |            |
| Parameter                                | DF | Estimate       | Standard Error             | Chi-Square | Pr > ChiSq |
| Intercept                                | 1  | -9.4745        | 5.5677                     | 2.8958     | 0.0888     |
| Dose                                     | 1  | 2.0804         | 1.2603                     | 2.7249     | 0.0988     |
| Odds Ratio Estimates                     |    |                |                            |            |            |
| Effect                                   |    | Point Estimate | 95% Wald Confidence Limits |            |            |
| Dose                                     |    | 8.007          | 0.677                      | 94.679     |            |

Output 7.1.2 shows the results from the exact conditional analysis. The  $p$ -values in the “Conditional Exact Tests” table lead to rejecting the null hypothesis that  $\beta$  is zero (no conclusions can be made about  $\alpha$  since it is “conditioned” away). The “Exact Parameter Estimates” table shows that the slope  $\beta$  is estimated to be  $\hat{\beta} = 1.8$ , and since the 95% confidence interval for the exponential of  $\hat{\beta}$  does not contain 1, the odds of death increase significantly with dosage. Note that the exact tests do not produce standard errors for the estimates.

**Output 7.1.2.** Output from EXACT Analysis

| The LOGISTIC Procedure     |             |                       |                 |         |
|----------------------------|-------------|-----------------------|-----------------|---------|
| Exact Conditional Analysis |             |                       |                 |         |
| Conditional Exact Tests    |             |                       |                 |         |
| Effect                     | Test        | Statistic             | --- p-Value --- |         |
|                            |             |                       | Exact           | Mid     |
| Dose                       | Score       | 5.4724                | 0.0245          | 0.0190  |
|                            | Probability | 0.0110                | 0.0245          | 0.0190  |
| Exact Parameter Estimates  |             |                       |                 |         |
| Parameter                  | Estimate    | 95% Confidence Limits |                 | p-Value |
| Dose                       | 1.7999      | 0.1157                | 5.8665          | 0.0245  |
| Exact Odds Ratios          |             |                       |                 |         |
| Parameter                  | Estimate    | 95% Confidence Limits |                 | p-Value |
| Dose                       | 6.049       | 1.123                 | 353.000         | 0.0245  |

The unconditional asymptotic and conditional exact results produce conflicting conclusions for this example. Stokes, Davis, and Koch (1995) recommend looking at the exact results when sample sizes are small and the approximate  $p$ -values are less than 0.10. For this example, the small sample size and the conflicting results for the asymptotic hypothesis tests indicate that an exact analysis is appropriate.

You can also perform a stratified analysis to control for the research centers. The strata are treated as nuisance parameters and a conditional likelihood removes them from the analysis. Your model contains a different intercept term for each stratum:

$$\text{logit}(\pi_{hi}) = \alpha_h + \mathbf{x}_{hi}\boldsymbol{\beta}$$

where  $h$  indexes the strata,  $\alpha_h$  are the strata intercepts, and  $i$  indexes the subjects within the strata.

With PROC LOGISTIC, you can specify a stratification variable by including it in the CLASS statement. For example, a stratification variable that has three levels can be parameterized as

| Stratum | Level 1 | Level 2 |
|---------|---------|---------|
| 1       | 1       | 0       |
| 2       | 0       | 1       |
| 3       | 0       | 0       |

where the usual intercept term represents the last strata level, and the other strata levels are a combination of the intercept and the appropriate level term. This is defined in the CLASS statement with the PARAM=REF option.

You can perform a stratified analysis by using the following statements, where `Center` is defined to be a classification variable and is conditioned out of the analysis by specifying only `Dose` in the `EXACT` statement.

```
proc logistic data=dose descending;
  class Center / param=ref;
  model Response = Center Dose;
  exact Dose / estimate=both;
run;
```

The usual asymptotic analysis indicates that there is complete separation of the data. (This means that unique maximum likelihood estimates do not exist.) You can see that the parameter estimates do not seem to converge if you specify both the `ITPRINT` and `NOCHECK` options in the `MODEL` statement. However, the exact analysis is still valid in this case, and exact tests and estimates for the conditional analysis are computed and displayed in Output 7.1.3.

**Output 7.1.3.** Stratified Output from EXACT Analysis

| The LOGISTIC Procedure                        |             |                       |                 |         |
|-----------------------------------------------|-------------|-----------------------|-----------------|---------|
| Exact Conditional Analysis                    |             |                       |                 |         |
| Conditional Exact Tests                       |             |                       |                 |         |
| Effect                                        | Test        | Statistic             | --- p-Value --- |         |
|                                               |             |                       | Exact           | Mid     |
| Dose                                          | Score       | 5.5714                | 0.0222          | 0.0167  |
|                                               | Probability | 0.0111                | 0.0222          | 0.0167  |
| Exact Parameter Estimates                     |             |                       |                 |         |
| Parameter                                     | Estimate    | 95% Confidence Limits |                 | p-Value |
| Dose                                          | 1.3204*     | 0.1354                | Infinity        | 0.0222  |
| NOTE: * indicates a median unbiased estimate. |             |                       |                 |         |
| Exact Odds Ratios                             |             |                       |                 |         |
| Parameter                                     | Estimate    | 95% Confidence Limits |                 | p-Value |
| Dose                                          | 3.745*      | 1.145                 | Infinity        | 0.0222  |
| NOTE: * indicates a median unbiased estimate. |             |                       |                 |         |

The median unbiased estimate is created instead of the conditional MLE because the value of the observed sufficient statistic lies at an extreme of the derived distribution, implying that the conditional MLE does not exist. The confidence interval for the exact odds ratio does not include 1, so you can conclude that the odds of death increases significantly with dosage. Even though the asymptotic results are unreliable, the exact analysis allows you to conclude that there is a significant effect due to Dose.

This exact analysis should be compared to an asymptotic conditional likelihood analysis, which is available with the PHREG procedure. First, define a variable *Z* to be 1 if the response is an event and 2 if the response is a nonevent. This variable is used as the time variable as well as the censoring indicator (with 2 as the censored value) in the MODEL statement of PROC PHREG. Also specify the TIES=DISCRETE option to request the discrete logistic model, and the STRATA statement to specify the strata on which to condition:

```
proc phreg data=dose;
  strata Center;
  model Z*Z(2)=Dose / ties=discrete;
run;
```

For these data, the PHREG procedure does not converge and the maximum likelihood estimates are not valid. Generally, the conditional score statistics for testing the overall null hypothesis should be the same for both the asymptotic conditional analysis in PROC PHREG and the exact analysis in PROC LOGISTIC. However, PROC PHREG computes the *p*-value by comparing the value of the conditional score statistic to a chi-squared distribution, while PROC LOGISTIC derives its *p*-value from the exact conditional distribution. Also, inference on individual parameters is often not the same.

---

## Example 7.2. Crossover Clinical Trial

One common use of conditional logistic regression is in a crossover clinical trial. In this example, the subjects are given a sequence of drugs and their response to each drug is recorded. Each subject is considered to be a separate stratum. The goal is to determine if the drugs have the same effect, adjusting for period and carryover effects. In this example, researchers give 15 different subjects three different drugs (A, B, P=placebo) in three consecutive periods (P1, P2, P3), and their response in each period is 1 for improvement and 0 for no improvement. The carryover effect is a classification variable indicating which drug was given in the preceding period:

```
data Crossover (drop=P1 P2 P3);
  input Subject P1$ P2$ P3$ Improve @@;
  Period=1; Drug=P1; Carry='0'; output;
  input Improve @@;
  Period=2; Drug=P2; Carry=P1; output;
  input Improve @@;
  Period=3; Drug=P3; Carry=P2; output;
  datalines;
1  A B P 0 0 0      8  B P A 0 0 1
2  A B P 1 1 0      9  B P A 1 0 1
3  A B P 0 1 1     10  B P A 0 1 0
4  A P B 1 0 1     11  P A B 0 1 0
5  A P B 1 0 0     12  P B A 1 0 1
6  B A P 0 0 0     13  P B A 0 0 1
7  B A P 1 1 0     14  P B A 0 1 0
                   15  P B A 0 1 1
;
```



The model to be fit is

$$\begin{aligned}\text{logit}(\pi_{hi}) = & \alpha_h + I(\text{Drug}_{hi} = \text{A})\beta_1 + I(\text{Drug}_{hi} = \text{B})\beta_2 \\ & + I(i = 1)\beta_3 + I(i = 2)\beta_4\end{aligned}$$

where  $h$  indexes the subject,  $\alpha_h$  are the subject intercepts,  $i$  indexes the period, and the  $I(\cdot)$  are indicator variables taking the value 1 when the condition is true. Note that this model ignores carryover effects. The following statements perform the analysis:

```
proc logistic data=Crossover descending exactonly;
  class Subject Drug Period/ param=ref;
  model Improve=Subject Drug Period;
  exact Drug Period/ joint;
run;
```

The exact conditional score  $p$ -value for the test of significance of all the parameters is 0.1835; hence, you cannot reject the null hypothesis. However, the exact conditional score  $p$ -value for the test of no drug effects,  $\beta_1 = \beta_2 = 0$ , is 0.0583, while the  $p$ -value for the test of no period effects,  $\beta_3 = \beta_4 = 0$ , is 0.8605, which suggests that the period term should be dropped from this model.

---

## References

- Agresti, Alan (1990), *Categorical Data Analysis*, New York: John Wiley & Sons, Inc.
- Agresti, Alan (1992), "A Survey of Exact Inference for Contingency Tables," *Statistical Science*, 7, 131–177.
- Cox, D.R. (1970), *Analysis of Binary Data*, New York: Chapman and Hall.
- Cox, D.R. and Snell, E.J. (1989), *Analysis of Binary Data*, Second Edition, New York: Chapman and Hall.
- Derr, Robert E. (2000), "Performing Exact Logistic Regression with the SAS System," *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc.
- Hirji, Karim F., Mehta, Cyrus R., and Patel, Nitin R. (1987), "Computing Distributions for Exact Logistic Regression," *Journal of the American Statistical Association*, 82, 1110–1117.
- Hirji, Karim F. and Tang, Man-Lai (1998), "A Comparison of Tests for Trend," *Communications in Statistics—Theory and Methods*, 27, 943–963.
- Hirji, Karim F., Tsiatis, Anastasios A., and Mehta, Cyrus R. (1989), "Median Unbiased Estimation for Binary Data," *American Statistician*, 43, 7–11.
- Lancaster, H. O., (1961), "Significance Tests in Discrete Distributions," *Journal of the American Statistical Association*, 56, 223–234.

Mehta, Cyrus R. and Patel, Nitin R. (1995), “Exact Logistic Regression: Theory and Examples,” *Statistics in Medicine*, 14, 2143–2160.

Stokes, Maura E., Davis, Charles S., and Koch, Gary G. (1995), *Categorical Data Analysis Using the SAS System*, Cary, NC: SAS Institute Inc.

Vollset, Stein E., Hirji, Karim F., and Afifi, Abdelmonem A. (1991), “Evaluation of Exact and Asymptotic Interval Estimators in Logistic Analysis of Matched Case-Control Studies,” *Biometrics*, 47, 1311–1325.



# Chapter 8

## The MIXED Procedure

### Chapter Table of Contents

---

|                             |     |
|-----------------------------|-----|
| <b>DETAILS</b> . . . . .    | 111 |
| Default Output . . . . .    | 111 |
| <b>REFERENCES</b> . . . . . | 111 |



# Chapter 8

## The MIXED Procedure

---

### Details

---

#### Default Output

##### *Fit Statistics*

AIC and BIC are now printed in smaller-is-better forms in the “Fit Statistics” table. A finite-sample corrected version of AIC (AICC) is also included. When you specify METHOD=ML, these criteria now incorporate the effective number of fixed-effects parameters (the rank of the  $X$  matrix) in addition to the number of estimated covariance parameters. Refer to Burnham and Anderson (1998) for additional details.

### References

---

Burnham, K.P. and Anderson, D.R. (1998), *Model Selection and Inference: A Practical Information-Theoretic Approach*, New York: Springer-Verlag.



# Chapter 9

## The MODECLUS Procedure

### Chapter Table of Contents

---

|                              |     |
|------------------------------|-----|
| <b>DETAILS</b> . . . . .     | 115 |
| Density Estimation . . . . . | 115 |





## Chapter 9

# The MODECLUS Procedure

---

### Details

---

#### Density Estimation

The MODECLUS procedure now provides a default smoothing parameter if none of the options (DR=, CR=, R=, DK=, CK=, and K= ) is specified. The formula for computing this default value is given by

$$\left[ \frac{2^{v+2}(v+2)\Gamma(.5v+1)}{nv^2} \right]^{1/(v+4)} \sqrt{\sum_{l=1}^v s_l^2}$$

If the data are distances, the factor  $\sqrt{\sum s_l^2}$  can be replaced by an average root-mean-square Euclidean distance divided by  $\sqrt{2}$ .



# Chapter 10

## The MULTTEST Procedure

### Chapter Table of Contents

---

|                                   |     |
|-----------------------------------|-----|
| <b>SYNTAX</b> . . . . .           | 119 |
| PROC MULTTEST Statement . . . . . | 119 |
| STRATA Statement . . . . .        | 119 |
| <b>DETAILS</b> . . . . .          | 119 |
| $p$ -Value Adjustments . . . . .  | 119 |
| <b>REFERENCES</b> . . . . .       | 120 |



# Chapter 10

## The MULTTEST Procedure

---

### Syntax

---

#### PROC MULTTEST Statement

Two new  $p$ -value adjustment methods are available: Fisher combination and Hommel. You can obtain these adjustments by specifying the FISHER\_C and HOMMEL options, respectively.

---

#### STRATA Statement

The WEIGHT= option for the STRATA statement specifies the type of strata weighting to use when computing the Freeman-Tukey and t-tests for the mean. Valid values for the WEIGHT= option in the STRATA statement are SAMPLESIZE, HARMONIC, and EQUAL. SAMPLESIZE requests weights proportional to the within-stratum sample sizes, and is the default method. HARMONIC sets up weights equal to the harmonic mean of the non-missing within-stratum CLASS sizes, and is similar to a Type 2 analysis in PROC GLM. EQUAL specifies equal weights, and is similar to a Type 3 analysis in PROC GLM.

---

### Details

---

#### $p$ -Value Adjustments

The FISHER\_C option requests adjusted  $p$ -values using closed tests, based on the idea of Fisher's combination test. The Fisher combination test for a joint test of any set of  $S$  hypotheses with  $p$ -values uses the Chi-Square statistic  $\chi^2 = -2 \sum \log(p_i)$ , with  $2S$  degrees of freedom. The FISHER\_C adjusted  $p$ -value for test  $j$  is the maximum of all  $p$ -values for the combination tests, taken over all joint tests that include  $j$  as one of their components. Independence of  $p$ -values is absolutely required for this method.

Hommel's (1988) method is a closed testing procedure based on Simes' (1986) test. The Simes  $p$ -value for a joint test of any set of  $S$  hypotheses with  $p$ -values  $p_1 \leq p_2 \leq \dots \leq p_S$  is  $\min((S/1)p_1, (S/2)p_2, \dots, (S/S)p_S)$ . The Hommel adjusted  $p$ -value for test  $j$  is the maximum of all such Simes  $p$ -values, taken over all joint tests that include  $j$  as one of their components. Hochberg adjusted  $p$ -values are always as large or larger than Hommel adjusted  $p$ -values. Sarkar and Chang (1997) showed that Simes' method is valid under independent or positively dependent  $p$ -values, so Hommel's and Hochberg's methods also are valid in such cases by the closure principle.

Westfall et al. (1999) and Westfall and Wolfinger (2000) are new references dealing with multiplicity issues and PROC MULTTEST.

---

## References

- Hommel, G. (1988), "A Comparison of Two Modified Bonferroni Procedures," *Biometrika*, 75, 383–386.
- Sarkar, S. and Chang, C.K. (1997), "Simes Method for Multiple Hypothesis Testing with Positively Dependent Test Statistics," *Journal of the American Statistical Association*, 92, 1601–1608.
- Simes, R.J. (1986), "An Improved Bonferroni Procedure for Multiple Tests of Significance," *Biometrika*, 73, 751–754.
- Westfall, P.H., Tobias, R.D., Rom, D., Wolfinger, R.D., and Hochberg, Y. (1999), *Multiple Comparisons and Multiple Tests using the SAS System*, SAS Institute Inc., Cary, NC.
- Westfall, P.H. and Wolfinger, R.D. (2000), "Closed Multiple Testing Procedures and PROC MULTTEST," *Observations*, SAS Institute Inc., to appear.

# Chapter 11

## The NLMIXED Procedure

### Chapter Table of Contents

---

|                             |     |
|-----------------------------|-----|
| <b>SYNTAX</b> . . . . .     | 123 |
| MODEL Statement . . . . .   | 123 |
| <b>DETAILS</b> . . . . .    | 123 |
| Displayed Output . . . . .  | 123 |
| <b>REFERENCES</b> . . . . . | 123 |





# Chapter 11

## The NLMIXED Procedure

---

### Syntax

---

#### MODEL Statement

The gamma and negative binomial distributions are now available in the MODEL statement. They are specified as *gamma(a,b)* and *negbin(n,p)*.

---

### Details

---

#### Displayed Output

##### Fitting Information

Only the smaller-is-better forms of AIC and BIC are now printed in the “Fit Statistics” table. A finite-sample corrected version of AIC (AICC) is also included. The criteria are computed as follows:

$$\begin{aligned}AIC &= 2f(\hat{\theta}) + 2p \\AICC &= 2f(\hat{\theta}) + 2pn/(n - p - 1) \\BIC &= 2f(\hat{\theta}) + p \log(s)\end{aligned}$$

where  $f()$  is the marginal likelihood function,  $\hat{\theta}$  is the vector of parameter estimates,  $p$  is the number of parameters,  $n$  is the number of observations, and  $s$  is the number of subjects. Refer to Burnham and Anderson (1998) for additional details.

---

### References

Burnham, K.P. and Anderson, D.R. (1998), *Model Selection and Inference: A Practical Information-Theoretic Approach*, New York: Springer-Verlag.



# Chapter 12

## The PHREG Procedure

### Chapter Table of Contents

---

|                                                    |     |
|----------------------------------------------------|-----|
| <b>OVERVIEW</b> . . . . .                          | 127 |
| <b>SYNTAX</b> . . . . .                            | 127 |
| PROC PHREG Statement . . . . .                     | 127 |
| <b>DETAILS</b> . . . . .                           | 127 |
| Robust Estimate of the Covariance Matrix . . . . . | 127 |
| Modified Score Statistic . . . . .                 | 128 |
| Fitting a Null Model . . . . .                     | 128 |
| <b>REFERENCES</b> . . . . .                        | 128 |



# Chapter 12

## The PHREG Procedure

---

### Overview

The PHREG procedure now offers the Wei and Lin (1989) robust estimate of the covariance matrix. Also in this release, you can fit a null model by not specifying any explanatory variables in the MODEL statement.

---

### Syntax

---

#### PROC PHREG Statement

**COVSANDWICH < (AGGREGATE) >**

**COVS < (AGGREGATE) >**

requests the robust sandwich estimate of Lin and Wei (1989) for the covariance matrix. When this option is specified, this robust sandwich estimate is used in the Wald tests for testing the global null hypothesis, null hypotheses of individual parameters, and the hypotheses in the TEST statements. In addition, a modified score test is computed in the testing of the global null hypothesis, and the parameter estimates table has an additional StdErrRatio column, which contains the ratios of the robust estimate of the standard error relative to the corresponding model-based estimate. Optionally, you can specify the keyword AGGREGATE enclosed in parentheses after the COVSANDWICH (or COVS) option, which requests a summing up of the score residuals for each distinct ID pattern in the computation of the robust sandwich covariance estimate. This AGGREGATE option has no effects if the ID statement is not specified.

---

### Details

---

#### Robust Estimate of the Covariance Matrix

The robust variance of Lin and Wei (1989) is a sandwich estimate given by

$$\hat{V}^s = I^{-1}(U'U)I^{-1}$$

where  $I$  is the information matrix evaluated at the maximum likelihood estimate  $\hat{\beta}$  and  $U$  is the  $n \times p$  matrix of score residuals. Since the matrix of DFBETA residuals  $D$  can be written as  $D = UI^{-1}$ , the robust sandwich variance estimate  $\hat{V}^s$  can be computed as

$$\hat{V}^s = D'D$$

---

## Modified Score Statistic

Let  $\mathbf{U}_0$  be the  $n \times p$  matrix of efficient scores and  $\mathbf{I}_0$  be the  $p \times p$  information matrix, both evaluated at  $\beta = \mathbf{0}$ ; let  $\mathbf{1}$  be a column  $n$ -vector of 1's. Let  $\hat{\beta}_1$  be the one-step estimate of  $\beta$ ; that is,

$$\hat{\beta}_1 = \mathbf{I}_0^{-1}(\mathbf{U}_0' \mathbf{1})$$

the covariance matrix is estimated by  $\hat{\mathbf{V}}(\beta_1) = \mathbf{I}_0^{-1}$ .

The score statistic for testing  $H_0: \beta = \mathbf{0}$  can be expressed as a Wald test statistic (Therneau and Grambsch [2000]):

$$\begin{aligned} (\mathbf{U}_0' \mathbf{1})' \mathbf{I}_0^{-1} (\mathbf{U}_0' \mathbf{1}) &= [\mathbf{I}_0^{-1} (\mathbf{U}_0' \mathbf{1})]' \mathbf{I}_0 [\mathbf{I}_0^{-1} (\mathbf{U}_0' \mathbf{1})] \\ &= \hat{\beta}_1' [\hat{\mathbf{V}}(\beta_1)]^{-1} \hat{\beta}_1 \end{aligned}$$

The modified score test statistic for testing  $H_0: \beta = \mathbf{0}$  is obtained by replacing  $\hat{\mathbf{V}}(\beta_1)$  in the score statistic by the robust sandwich estimate  $\hat{\mathbf{V}}_0^s = \mathbf{D}_0' \mathbf{D}_0$  where  $\mathbf{D}_0 = \mathbf{U}_0 \mathbf{I}_0^{-1}$ :

$$\begin{aligned} \hat{\beta}_1' (\hat{\mathbf{V}}_0^s)^{-1} \hat{\beta}_1 &= [\mathbf{I}_0^{-1} (\mathbf{U}_0' \mathbf{1})]' (\mathbf{D}_0' \mathbf{D}_0)^{-1} [\mathbf{I}_0^{-1} (\mathbf{U}_0' \mathbf{1})] \\ &= (\mathbf{U}_0' \mathbf{1})' (\mathbf{U}_0' \mathbf{U}_0)^{-1} (\mathbf{U}_0' \mathbf{1}) \end{aligned}$$

---

## Fitting a Null Model

In some situations you may want to fit a null model. For instance, by fitting a null model, you can trick PROC PHREG into providing the Kaplan-Meier estimate of the survivor function for a set of survival times with right censoring and left truncation. In the following SAS program, `t2` is the survival time, `t1` is the left truncation time, and `c` is the censoring indicator with value 1 indicating censored observations. By not specifying any explanatory variables in the `MDOEL` statement, you are fitting a null model. The `sdf` variable in the output data set `out1` contains the product-limit estimates.

```
proc phreg;
  model t2*c(1)= /entrytime=t1 ;
  output out=out1 survival= sdf;
run;
```

---

## References

- Lin, D.Y. and Wei, L.J. (1989), "The Robust Inference for the Proportional Hazards Model," *Journal of the American Statistical Association*, 84, 1074–1078.
- Therneau, T. M. and Grambsch, P.M. (2000), *Modeling Survival Data: Extending the Cox Model*. New York: Springer-Verlag.

# Chapter 13

## The SURVEYMEANS Procedure

### Chapter Table of Contents

---

|                                                      |     |
|------------------------------------------------------|-----|
| <b>OVERVIEW</b>                                      | 131 |
| <b>GETTING STARTED</b>                               | 131 |
| Stratified Sampling                                  | 131 |
| <b>SYNTAX</b>                                        | 133 |
| DOMAIN Statement                                     | 133 |
| <b>DETAILS</b>                                       | 134 |
| Domain Analysis                                      | 134 |
| Missing Values                                       | 134 |
| Statistical Computations                             | 136 |
| Displayed Output                                     | 139 |
| ODS Table Names                                      | 139 |
| <b>EXAMPLES</b>                                      | 139 |
| Example 13.1 Stratified Cluster Sample Design        | 140 |
| Example 13.2 Domain Analysis                         | 143 |
| Example 13.3 Analyze Survey Data with Missing Values | 146 |
| <b>REFERENCES</b>                                    | 149 |





# Chapter 13

## The SURVEYMEANS Procedure

---

### Overview

The new DOMAIN statement in the SURVEYMEANS procedure enables you to perform domain analysis for survey data.

Examples and computational details are revised and improved.

---

### Getting Started

#### Stratified Sampling

This example illustrates how you can use PROC SURVEYMEANS to estimate population means and proportions from sample survey data. The study population is a junior high school with a total of 4,000 students in grades 7, 8, and 9. Researchers want to know how much these students spend weekly for ice cream, on average, and what percentage of students spend at least \$10 weekly for ice cream.

Suppose that the sample of students was selected using stratified random sampling. In stratified sampling, the study population is divided into nonoverlapping strata, and samples are selected from each stratum independently.

The list of students in this junior high school was stratified by grade, yielding three strata: grades 7, 8, and 9. A simple random sample of students was selected from each grade. Table 13.1 shows the total number of students in each grade.

**Table 13.1.** Number of Students by Grade

| Grade | Number of Students |
|-------|--------------------|
| 7     | 1,824              |
| 8     | 1,025              |
| 9     | 1,151              |
| Total | 4,000              |

A sample of 40 students was selected from the entire student population. Each student selected for the sample was asked how much he or she spends for ice cream per week, on average. The SAS data set named `IceCream` saved the responses of the 40 students:

```
data IceCream;
  input Grade Spending @@;
  if (Spending < 10) then Group='less';
  else Group='more';
  datalines;
```

```

7 7 7 7 8 12 9 10 7 1 7 10 7 3 8 20 8 19 7 2
7 2 9 15 8 16 7 6 7 6 7 6 9 15 8 17 8 14 9 8
9 8 9 7 7 3 7 12 7 4 9 14 8 18 9 9 7 2 7 1
7 4 7 11 9 8 8 10 8 13 7 2 9 6 9 11 7 2 7 9
;

```

The variable `Grade` contains a student's grade. The variable `Spending` contains a student's response on how much was spent per week for ice cream, in dollars. The variable `Group` is created to indicate whether a student spends at least \$10 weekly for ice cream: `Group='more'` if a student spends at least \$10, or `Group='less'` if a student spends less than \$10.

To analyze this stratified sample, you need to provide the population totals for each stratum to PROC SURVEYMEANS. The SAS data set named `StudentTotals` contains the information from Table 13.1:

```

data StudentTotals;
    input Grade _total_; datalines;
7 1824
8 1025
9 1151
;

```

The variable `Grade` is the stratum identification variable, and the variable `_TOTAL_` contains the total number of students for each stratum. PROC SURVEYMEANS requires you to use the variable name `_TOTAL_` for the stratum population totals.

The procedure uses the stratum population totals to adjust variance estimates for the effects of sampling from a finite population. If you do not provide population totals or sampling rates, then the procedure assumes that the proportion of the population in the sample is very small, and the computation does not involve a finite population correction.

In a stratified sample design, when the sampling rates in the strata are unequal, you need to use sampling weights to reflect this information in order to produce an unbiased mean estimator. In this example, the appropriate sampling weights are reciprocals of the probabilities of selection. You can use the following data step to create the sampling weights:

```

data IceCream;
    set IceCream;
    if Grade=7 then Prob=20/1824;
    if Grade=8 then Prob=9/1025;
    if Grade=9 then Prob=11/1151;
    Weight=1/Prob;

```

If you use PROC SURVEYSELECT to select your sample, it creates these sampling weights for you.

The following SAS statements perform the stratified analysis of the survey data:

```

title1 'Analysis of Ice Cream Spending';
title2 'Stratified Simple Random Sample Design';
proc surveymeans data=IceCream total=StudentTotals;
    stratum Grade / list;
    var Spending Group;
    weight Weight;
run;

```

The PROC SURVEYMEANS statement invokes the procedure. The DATA= option names the SAS data set `IceCream` as the input data set to be analyzed. The TOTAL= option names the data set `StudentTotals` as the input data set containing the stratum population totals. Notice that the TOTAL=StudentTotals option is used here instead of the TOTAL=4000 option. In this stratified sample design, the population totals are different for different strata, and so you need to provide them to PROC SURVEYMEANS in a SAS data set.

The STRATA statement identifies the stratification variable `Grade`. The LIST option in the STRATA statement requests that the procedure display stratum information. The WEIGHT statement tells the procedure that the variable `Weight` contains the sampling weights.

---

## Syntax

The following statement is available in PROC SURVEYMEANS.

```

DOMAIN variables < variable*variable
                        variable*variable*variable ... > ;

```

---

## DOMAIN Statement

```

DOMAIN | SUBGROUP variables < variable*variable
                        variable*variable*variable ... > ;

```

The DOMAIN statement requests analysis for subpopulations, or domains, in addition to analysis for the entire study population. The DOMAIN statement names the variables that identify domains, which are called domain variables.

It is common practice to compute statistics for domains. The formation of these domains may be unrelated to the sample design. Therefore, the sample sizes for the domains are random variables. In order to incorporate this variability into the variance estimation, you should use a DOMAIN statement.

Note that a DOMAIN statement is different from a BY statement. In a BY statement, you treat the sample sizes as fixed in each subpopulation, and you perform analysis within each BY group independently. See the section “Domain Analysis” on page 136 for more details.

A domain variable can be either character or numeric. However, the procedure treats domain variables as categorical variables. If a variable appears by itself in a

DOMAIN statement, each level of this variable determines a domain in the study population. If two or more variables are joined by asterisks (\*), then every possible combination of levels of the variables determines a domain. The procedure performs a descriptive analysis within each domain defined by the domain variables.

The formatted values of the domain variables determine the categorical variable levels. Thus, you can use formats to group values into levels. Refer to the discussion of the FORMAT procedure in the *SAS Procedures Guide* and to the discussions of the FORMAT statement and SAS formats in *SAS Language Reference: Dictionary*.

---

## Details

---

### Domain Analysis

It is common practice to compute statistics for subpopulations, or domains, in addition to computing statistics for the entire study population. Analysis for domains using the entire sample is called *domain analysis* (or subgroup analysis, subpopulation analysis, subdomain analysis). The formation of these subpopulations of interest may be unrelated to the sample design. Therefore, the sample sizes for the subpopulations may actually be random variables.

In order to incorporate this variability into the variance estimation, you should use a DOMAIN statement. Note that using a BY statement provides completely separate analyses of the BY groups. It does not provide a statistically valid subpopulation or domain analysis, where the total number of units in the subpopulation is not known with certainty. For more detailed information about domain analysis, refer to Kish (1965) and Statistical Laboratory (1989).

---

### Missing Values

When computing statistics for an analysis variable, PROC SURVEYMEANS omits observations with missing values for that variable. The procedure bases statistics for each variable only on observations that have nonmissing values for that variable. If you specify the MISSING option in the PROC SURVEYMEANS statement, the procedure treats missing values of a categorical variable as a valid category.

An observation is also excluded if it has a missing value for any STRATA or CLUSTER variable, unless the MISSING option is used.

If an observation has a missing value or a nonpositive value for the WEIGHT variable, then PROC SURVEYMEANS excludes that observation from the analysis.

The procedure performs univariate analysis and analyzes each VAR variable separately. Thus, the number of missing observations may be different for different variables. You can specify the keyword NMISS in the PROC SURVEYMEANS statement to display the number of missing values for each analysis variable in the “Statistics” table.

If you have missing values in your survey data for any reason (such as nonresponse), this can compromise the quality of your survey results. An observation without

missing values is called a complete respondent, and an observation with missing values is called an incomplete respondent. If the complete respondents are different from the incomplete respondents with regard to a survey effect or outcome, then survey estimates will be biased and will not accurately represent the survey population. There are a variety of techniques in sample design and survey operations that can reduce nonresponse. Once data collection is complete, you can use imputation to replace missing values with acceptable values, and you can use sampling weight adjustments to compensate for nonresponse. You should complete this data preparation and adjustment before you analyze your data with PROC SURVEYMEANS. Refer to Cochran (1977), Kalton and Kaspyzyk (1986), and Brick and Kalton (1996) for more details.

If there is evidence indicating that complete respondents are different from incomplete respondents for your study, you can use the DOMAIN statement to compute the descriptive statistics “among complete respondents” from your survey data without imputation on incomplete respondents. See Example 13.3 on page 146.

If missing values result in empty strata in the sample, then they will have an impact on the statistical computation, which uses the total number of strata. If all the observations in a stratum have missing weights or missing values for the current analysis variable, this stratum is an *empty stratum*. For example,

```
data new;
  input stratum y z w;
  datalines;
1 . 13 40
1 2 9 .
1 . 5 25
2 5 10 20
2 8 60 15
;
proc surveymeans df mean nobis nmiss;
  strata stratum;
  var y z;
  weight w;
run;
```

You analyze variable Y and Z, with weight variable W and stratum variable STRATUM. For variable Y, all observations have missing values or missing weights in STRATUM=1, therefore, the analysis for variable Y uses only observations in STRATUM=2. Thus, for variable Y, STRATUM=1 is an empty stratum and STRATUM=2 is a non-empty stratum. Note, however, that STRATUM=1 is a non-empty stratum for variable Z.

If your sample design contains stratification, PROC SURVEYMEANS analyzes only the data in non-empty strata. Therefore, the total number of strata for an analysis variable means the total number of *non-empty* strata. In this example, the total number of strata for Y and Z is one and two, respectively.

## Statistical Computations

### *t* Test for the Mean

If you specify the keyword `T`, PROC SURVEYMEANS computes the *t* value for testing that the population mean equals zero,  $H_0 : \bar{Y} = 0$ . The test statistic equals

$$t(\hat{Y}) = \hat{Y} / \text{StdErr}(\hat{Y})$$

The two-sided *p*-value for this test is

$$\text{Prob}(|T| > |t(\hat{Y})|)$$

where *T* is a random variable with the *t* distribution with *df* degrees of freedom.

PROC SURVEYMEANS calculates the degrees of freedom for the *t* test as the number of clusters minus the number of strata. If there are no clusters, then *df* equals the number of observations minus the number of strata. If the design is not stratified, then *df* equals the number of clusters minus one. The procedure displays *df* for the *t* test if you specify the keyword `DF` in the PROC SURVEYMEANS statement.

If missing values or missing weights are present in your data, the number of strata, the number of observations, and the number of clusters are counted based on the observations in non-empty strata. See the section “Missing Values” on page 134 for details. For degrees of freedom in domain analysis, see the section “Domain Analysis” on page 136.

### Domain Analysis

When you use a `DOMAIN` statement to request a domain analysis, the procedure computes the requested statistics for each domain.

For a domain *D*, let  $I_D$  be the corresponding indicator variable:

$$I_D(h, i, j) = \begin{cases} 1 & \text{if observation } (h, i, j) \text{ belongs to } D \\ 0 & \text{otherwise} \end{cases}$$

Let

$$z_{hij} = y_{hij} I_D(h, i, j) = \begin{cases} y_{hij} & \text{if observation } (h, i, j) \text{ belongs to } D \\ 0 & \text{otherwise} \end{cases}$$

The requested statistics for variable *y* in domain *D* are computed based on the values of *z*.

**Domain Mean** The estimated mean of *y* in the domain *D* is

$$\hat{Y}_D = \left( \sum_{h=1}^H \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} v_{hij} z_{hij} \right) / v_{...}$$

where

$$v_{hij} = w_{hij} I_D(h, i, j) = \begin{cases} w_{hij} & \text{if observation } (h, i, j) \text{ belongs to } D \\ 0 & \text{otherwise} \end{cases}$$

$$v_{...} = \sum_{h=1}^H \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} v_{hij}$$

The variance of  $\widehat{Y}_D$  is estimated by

$$\widehat{V}(\widehat{Y}_D) = \sum_{h=1}^H \frac{n_h(1-f_h)}{n_h-1} \sum_{i=1}^{n_h} (r_{hi\cdot} - \bar{r}_{h\cdot\cdot})^2$$

where

$$r_{hi\cdot} = \left( \sum_{j=1}^{m_{hi}} v_{hij} (z_{hij} - \widehat{Y}_D) \right) / v_{...}$$

$$\bar{r}_{h\cdot\cdot} = \left( \sum_{i=1}^{n_h} r_{hi\cdot} \right) / n_h$$

**Domain Total** The estimated total in domain  $D$  is

$$\widehat{Y}_D = \sum_{h=1}^H \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} v_{hij} z_{hij}$$

and its estimated variance is

$$\widehat{V}(\widehat{Y}_D) = \sum_{h=1}^H \frac{n_h(1-f_h)}{n_h-1} \sum_{i=1}^{n_h} (z_{hi\cdot} - \bar{z}_{h\cdot\cdot})^2$$

where

$$z_{hi\cdot} = \sum_{j=1}^{m_{hi}} v_{hij} z_{hij}$$

$$\bar{z}_{h\cdot\cdot} = \left( \sum_{i=1}^{n_h} z_{hi\cdot} \right) / n_h$$

**Degrees of Freedom** For domain analysis, PROC SURVEYMEANS computes the degrees of freedom for  $t$  tests as the number of clusters in the non-empty strata minus the number of non-empty strata. When the sample design has no clusters, the degrees of freedom equals the number of observations in non-empty strata minus the number of non-empty strata. As discussed in the section “Missing Values” on page 134, missing values and missing weights can result in empty strata. In domain analysis, an empty stratum can also occur when the stratum contains no observations in the specified domain. If no observations in a whole stratum belong to a domain, then this stratum is called an empty stratum for that domain.



For example,

```
data new;
  input str clu y w d;
  datalines;
1 1 . 40 9
1 2 2 . 9
1 3 . 25 9
2 4 5 20 9
2 5 8 15 9
3 6 5 30 7
3 7 9 89 7
3 8 6 23 7
;
proc surveymeans df nobn nclu nmiss;
  strata str;
  cluster clu;
  var y;
  weight w;
  domain d;
run;
```

**Table 13.2.** Calculations of  $df$  for Y

|                                      | Domain D=7              | Domain D=9      |
|--------------------------------------|-------------------------|-----------------|
| <b>Non Empty Strata</b>              | STR=3                   | STR=2           |
| <b>Clusters Used in the Analysis</b> | CLU=6, CLU=7, and CLU=8 | CLU=4 and CLU=5 |
| $df$                                 | $3 - 1 = 2$             | $2 - 1 = 1$     |

Although there are three strata in the data set, STR=1 is an empty stratum for variable Y because of missing values and missing weights. In addition, no observations in stratum STR=3 belong to domain D=9. Therefore, STR=3 becomes an empty stratum as well for variable Y in domain D=9. As a result, the total number of non-empty strata for domain D=9 is one. The non-empty stratum for domain D=9 and variable Y is stratum STR=2. The total number of clusters for domain D=9 is two, which belong to stratum STR=2. Thus, for variable Y in domain D=9, the degrees of freedom for the  $t$  tests of the domain mean is  $df = 2 - 1 = 1$ . Similarly, for domain D=7, strata STR=1 and STR=2 are both empty strata, so the total number of strata is one (STR=3), and the total number of clusters is three (CLU=6, CLU=7, and CLU=8). Table 13.2 illustrates how domains affect the total number of clusters and total number of strata in the  $df$  calculation. Figure 13.1 shows the  $df$  computed by the procedure.

| The SURVEYMEANS Procedure |          |   |        |          |    |
|---------------------------|----------|---|--------|----------|----|
| Domain Analysis: d        |          |   |        |          |    |
| d                         | Variable | N | N Miss | Clusters | DF |
| 7                         | y        | 3 | 0      | 3        | 2  |
| 9                         | y        | 2 | 2      | 2        | 1  |

**Figure 13.1.** Degrees of Freedoms in Domain Analysis

---

## Displayed Output

### Domain Analysis

If you use a DOMAIN statement, the procedure displays statistics in each domain in a “Domain Analysis” table. A “Domain Analysis” table contains all the columns in the “Statistics” table, plus columns of domain variable values.

Note that depending on how you define the domains with domain variables, the procedure may produce more than one “Domain Analysis” table. For example, in the following DOMAIN statement

```
domain A B*C*D A*C C;
```

you use four definitions to define domains:

- A: all the levels of A
- C: all the levels of C
- A\*C: all the interactive levels of A and C
- B\*C\*D: all the interactive levels of B, C, and D

The procedure displays four “Domain Analysis” tables, one for each domain definition.

---

## ODS Table Names

PROC SURVEYMEANS assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

**Table 13.3.** ODS Tables Produced in PROC SURVEYMEANS

| ODS Table Name | Description             | Statement | Option  |
|----------------|-------------------------|-----------|---------|
| ClassVarInfo   | Class level information | CLASS     | default |
| Domain         | Statistics in domains   | DOMAIN    | default |
| Statistics     | Statistics              | PROC      | default |
| StrataInfo     | Stratum information     | STRATA    | LIST    |
| Summary        | Data summary            | PROC      | default |

---

## Examples

The “Getting Started” section on page 131 contains an example of analyzing data from stratified simple random sample designs. This section provides more examples that illustrate how to use PROC SURVEYMEANS.

### Example 13.1. Stratified Cluster Sample Design

Consider the example in the section “Stratified Sampling” on page 131. The study population is a junior high school with a total of 4,000 students in grades 7, 8, and 9. Researchers want to know how much these students spend weekly for ice cream, on the average, and what percentage of students spend at least \$10 weekly for ice cream.

The example in the section “Stratified Sampling” on page 131 assumes that the sample of students was selected using a stratified simple random sample design. This example shows analysis based on a more complex sample design.

Suppose that every student belongs to a study group and that study groups are formed within each grade level. Each study group contains between two and four students. Table 13.4 shows the total number of study groups for each grade.

**Table 13.4.** Study Groups and Students by Grade

| Grade | Number of Study Groups | Number of Students |
|-------|------------------------|--------------------|
| 7     | 608                    | 1,824              |
| 8     | 252                    | 1,025              |
| 9     | 403                    | 1,151              |
| Total | 617                    | 4,000              |

It is quicker and more convenient to collect data from students in the same study group than to collect data from students individually. Therefore, this study uses a stratified clustered sample design. The primary sampling units, or clusters, are study groups. The list of all study groups in the school is stratified by grade level. From each grade level, a sample of study groups is randomly selected, and all students in each selected study group are interviewed. The sample consists of eight study groups from the 7th grade, three groups from the 8th grade, and five groups from the 9th grade.

The SAS data set named IceCreamStudy saves the responses of the selected students:

```
data IceCreamStudy;
  input Grade StudyGroup Spending @@;
  if (Spending < 10) then Group='less';
  else Group='more';
  datalines;
7 34 7      7 34 7      7 412 4      9 27 14
7 34 2      9 230 15     9 27 15     7 501 2
9 230 8      9 230 7      7 501 3      8 59 20
7 403 4      7 403 11     8 59 13     8 59 17
8 143 12     8 143 16     8 59 18     9 235 9
8 143 10     9 312 8      9 235 6      9 235 11
9 312 10     7 321 6      8 156 19     8 156 14
7 321 3      7 321 12     7 489 2      7 489 9
7 78 1       7 78 10      7 489 2      7 156 1
7 78 6       7 412 6      7 156 2      9 301 8
;
```

In the data set `IceCreamStudy`, the variable `Grade` contain a student's grade. The variable `StudyGroup` identifies a student's study group. It is possible for students from different grades to have the same study group number because study groups are sequentially numbered within each grade. The variable `Spending` contains a student's response to how much he or she spends per week for ice cream, in dollars. The variable `GROUP` indicates whether a student spends at least \$10 weekly for ice cream. It is not necessary to store the data in order of grade and study group.

The SAS data set `StudyGroup` is created to provide PROC SURVEYMEANS with the sample design information shown in Table 13.4:

```
data StudyGroups;
  input Grade _total_; datalines;
7 608
8 252
9 403
;
```

The variable `Grade` identifies the strata, and the variable `_TOTAL_` contains the total number of study groups in each stratum. The population totals stored in the variable `_TOTAL_` should be expressed in terms of the primary sampling units (PSUs), which are study groups in this example. Therefore, the variable `_TOTAL_` contains the total number of study groups for each grade, rather than the total number of students.

In order to obtain unbiased estimates, you create sampling weights using the following SAS statements:

```
data IceCreamStudy;
  set IceCreamStudy;
  if Grade=7 then Prob=8/608;
  if Grade=8 then Prob=3/252;
  if Grade=9 then Prob=5/403;
  Weight=1/Prob;
```

The sampling weights are the reciprocals of the probabilities of selections. The variable `Weight` contains the sampling weights. Because the sampling design is clustered, and all students from each selected cluster are interviewed, the sampling weights equal the inverse of the cluster (or study group) selection probabilities.

The following SAS statements perform the analysis for this sample design:

```
title1 'Analysis of Ice Cream Spending';
title2 'Stratified Clustered Sample Design';
proc surveymeans data=IceCreamStudy total=StudyGroups;
  stratum Grade / list;
  cluster StudyGroup;
  var Spending Group;
  weight Weight;
run;
```

**Output 13.1.1.** Data Summary and Class Information

| Analysis of Ice Cream Spending<br>Stratified Clustered Sample Design |        |           |
|----------------------------------------------------------------------|--------|-----------|
| The SURVEYMEANS Procedure                                            |        |           |
| Data Summary                                                         |        |           |
| Number of Strata                                                     |        | 3         |
| Number of Clusters                                                   |        | 16        |
| Number of Observations                                               |        | 40        |
| Sum of Weights                                                       |        | 3162.6    |
| Class Level Information                                              |        |           |
| Class<br>Variable                                                    | Levels | Values    |
| Group                                                                | 2      | less more |

Output 13.1.1 provides information on the sample design and the input data set. There are 3 strata in the sample design, and the sample contains 16 clusters and 40 observations. The variable **Group** has two levels, 'less' and 'more'.

**Output 13.1.2.** Stratum Information

| Analysis of Ice Cream Spending<br>Stratified Clustered Sample Design |       |                     |                  |       |            |    |          |
|----------------------------------------------------------------------|-------|---------------------|------------------|-------|------------|----|----------|
| The SURVEYMEANS Procedure                                            |       |                     |                  |       |            |    |          |
| Stratum Information                                                  |       |                     |                  |       |            |    |          |
| Stratum<br>Index                                                     | Grade | Population<br>Total | Sampling<br>Rate | N Obs | Variable   | N  | Clusters |
| 1                                                                    | 7     | 608                 | 1.32%            | 20    | Spending   | 20 | 8        |
|                                                                      |       |                     |                  |       | Group=less | 17 | 8        |
|                                                                      |       |                     |                  |       | Group=more | 3  | 3        |
| 2                                                                    | 8     | 252                 | 1.19%            | 9     | Spending   | 9  | 3        |
|                                                                      |       |                     |                  |       | Group=less | 0  | 0        |
|                                                                      |       |                     |                  |       | Group=more | 9  | 3        |
| 3                                                                    | 9     | 403                 | 1.24%            | 11    | Spending   | 11 | 5        |
|                                                                      |       |                     |                  |       | Group=less | 6  | 4        |
|                                                                      |       |                     |                  |       | Group=more | 5  | 4        |

Output 13.1.2 displays information for each stratum. Since the primary sampling units in this design are study groups, the population totals shown in Output 13.1.2 are the total numbers of study groups for each stratum or grade. Output 13.1.2 also displays the number of clusters for each stratum and analysis variable.

**Output 13.1.3. Statistics**

| Analysis of Ice Cream Spending<br>Stratified Clustered Sample Design |    |          |                      |                          |                          |
|----------------------------------------------------------------------|----|----------|----------------------|--------------------------|--------------------------|
| The SURVEYMEANS Procedure                                            |    |          |                      |                          |                          |
| Statistics                                                           |    |          |                      |                          |                          |
| Variable                                                             | N  | Mean     | Std Error<br>of Mean | Lower 95%<br>CL for Mean | Upper 95%<br>CL for Mean |
| Spending                                                             | 40 | 8.923860 | 0.650859             | 7.517764                 | 10.329957                |
| Group=less                                                           | 23 | 0.561437 | 0.056368             | 0.439661                 | 0.683213                 |
| Group=more                                                           | 17 | 0.438563 | 0.056368             | 0.316787                 | 0.560339                 |

Output 13.1.3 displays the estimates of the average weekly ice cream expense and the percentage of students spending at least \$10 weekly for ice cream.

## Example 13.2. Domain Analysis

Suppose that you are studying profiles of the 800 top-performing companies to provide information on their impact on the economy. You are also interested in the company profiles within each market type. A sample of 66 companies is selected with unequal probability across market types. However, market type is not included in the sample design. Thus, the number of companies within each market type is a random variable in your sample. To obtain statistics within each market type, you should use domain analysis. The data of the 66 companies are saved in the following data set:

```
data Company;
  length Type $14;
  input Type$ Asset Sale Value Profit Employee Weight;
  datalines;
Other      2764.0  1828.0  1850.3  144.0  18.7  9.6
Energy     13246.2  4633.5  4387.7  462.9  24.3  42.6
Finance    3597.7   377.8   93.0   14.0   1.1  12.2
Transportation 6646.1  6414.2  2377.5  348.2  47.1  21.8
HiTech     1068.4  1689.8  1430.2   72.9   4.6   4.3
Manufacturing 1125.0  1719.4  1057.5   98.1  20.4   4.5
Other      1459.0  1241.4   452.7   24.5  20.1   5.5
Finance    2672.3   262.5   296.2   23.1   2.2   9.3
Finance     311.0   566.2   932.0   52.8   2.7   1.9
Energy     1148.6  1014.6   485.1   60.6   4.0   4.5
Finance    5327.0   572.4   372.9   25.2   4.2  17.7
Energy     1602.7   678.4   653.0   75.6   2.8   6.0
Energy     5808.8  1288.4  2007.0  318.8   5.9  19.2
Medical     268.8   204.4   820.9   45.6   3.7   1.8
Transportation 5222.6  2627.8  1910.0  245.6  22.8  17.4
Other       872.7  1419.4   939.3   69.7  12.2   3.7
Retail     4461.7  8946.8  4662.7  289.0  132.1  15.0
HiTech     6719.2  6942.0  8240.2  381.3   85.8  22.1
Retail      833.4  1538.8  1090.3   64.9  15.4   3.5
Finance     415.9   167.3  1126.8   56.8   0.7   2.2
```

|                |         |         |        |       |       |      |
|----------------|---------|---------|--------|-------|-------|------|
| HiTech         | 442.4   | 1139.9  | 1039.9 | 57.6  | 22.7  | 2.3  |
| Other          | 801.5   | 1157.0  | 664.2  | 56.9  | 15.5  | 3.4  |
| Finance        | 4954.8  | 468.8   | 366.4  | 41.7  | 3.0   | 16.5 |
| Finance        | 2661.9  | 257.9   | 181.1  | 21.2  | 2.1   | 9.3  |
| Finance        | 5345.8  | 530.1   | 337.4  | 36.4  | 4.3   | 17.8 |
| Energy         | 3334.3  | 1644.7  | 1407.8 | 157.6 | 6.4   | 11.4 |
| Manufacturing  | 1826.6  | 2671.7  | 483.2  | 71.3  | 25.3  | 6.7  |
| Retail         | 618.8   | 2354.7  | 767.7  | 58.6  | 19.0  | 2.9  |
| Retail         | 1529.1  | 6534.0  | 826.3  | 58.3  | 65.8  | 5.7  |
| Manufacturing  | 4458.4  | 4824.5  | 3132.1 | 28.9  | 67.0  | 15.0 |
| HiTech         | 5831.7  | 6611.1  | 9464.7 | 459.6 | 86.7  | 19.3 |
| Medical        | 6468.3  | 4199.2  | 3170.4 | 270.1 | 59.5  | 21.3 |
| Energy         | 1720.7  | 473.1   | 811.1  | 86.6  | 1.6   | 6.3  |
| Energy         | 1679.7  | 1379.9  | 721.1  | 91.8  | 4.5   | 6.2  |
| Retail         | 4018.2  | 16823.4 | 2038.3 | 178.1 | 162.0 | 13.6 |
| Other          | 227.1   | 575.8   | 1083.8 | 62.6  | 1.9   | 1.6  |
| Finance        | 3872.8  | 362.0   | 209.3  | 27.6  | 2.4   | 13.1 |
| Retail         | 3359.3  | 4844.7  | 2651.4 | 224.1 | 75.6  | 11.5 |
| Energy         | 1295.6  | 356.9   | 180.8  | 162.3 | 0.6   | 5.0  |
| Energy         | 1658.0  | 626.6   | 688.0  | 126.0 | 3.5   | 6.1  |
| Finance        | 12156.7 | 1345.5  | 680.7  | 106.6 | 9.4   | 39.2 |
| HiTech         | 3982.6  | 4196.0  | 3946.8 | 313.9 | 64.3  | 13.5 |
| Finance        | 8760.7  | 886.4   | 1006.9 | 90.0  | 7.5   | 28.5 |
| Manufacturing  | 2362.2  | 3153.3  | 1080.0 | 137.0 | 25.2  | 8.4  |
| Transportation | 2499.9  | 3419.0  | 992.6  | 47.2  | 25.3  | 8.8  |
| Energy         | 1430.4  | 1610.0  | 664.3  | 77.7  | 3.5   | 5.4  |
| Energy         | 13666.5 | 15465.4 | 2736.7 | 411.4 | 26.6  | 43.9 |
| Manufacturing  | 4069.3  | 4174.7  | 2907.6 | 289.2 | 38.2  | 13.7 |
| Energy         | 2924.7  | 711.9   | 1067.8 | 146.7 | 3.4   | 10.1 |
| Transportation | 1262.1  | 1716.0  | 364.3  | 71.2  | 14.5  | 4.9  |
| Medical        | 684.4   | 672.9   | 287.4  | 61.8  | 6.0   | 3.1  |
| Energy         | 3069.3  | 1719.0  | 1439.0 | 196.4 | 4.9   | 10.6 |
| Medical        | 246.5   | 318.8   | 924.1  | 43.8  | 3.1   | 1.7  |
| Finance        | 11562.2 | 1128.5  | 580.4  | 64.2  | 6.7   | 37.3 |
| Finance        | 9316.0  | 1059.4  | 816.5  | 95.9  | 8.0   | 30.2 |
| Retail         | 1094.3  | 3848.0  | 563.3  | 29.4  | 44.7  | 4.4  |
| Retail         | 1102.1  | 4878.3  | 932.4  | 65.2  | 47.3  | 4.4  |
| HiTech         | 466.4   | 675.8   | 845.7  | 64.5  | 5.2   | 2.4  |
| Manufacturing  | 10839.4 | 5468.7  | 1895.4 | 232.8 | 47.8  | 35.0 |
| Manufacturing  | 733.5   | 2135.3  | 96.6   | 10.9  | 2.7   | 3.2  |
| Manufacturing  | 10354.2 | 14477.4 | 5607.2 | 321.9 | 188.5 | 33.5 |
| Energy         | 1902.1  | 2697.9  | 329.3  | 34.2  | 2.2   | 6.9  |
| Other          | 2245.2  | 2132.2  | 2230.4 | 198.9 | 8.0   | 8.0  |
| Transportation | 949.4   | 1248.3  | 298.9  | 35.4  | 10.4  | 3.9  |
| Retail         | 2834.4  | 2884.6  | 458.2  | 41.2  | 49.8  | 9.8  |
| Retail         | 2621.1  | 6173.8  | 1992.7 | 183.7 | 115.1 | 9.2  |

;

For each company in your sample,

- The variable `Type` identifies the type of market for the company.
- The variable `Asset` contains the company's assets in millions of dollars.

- The variable **Sale** contains sales in millions of dollars.
- The variable **Value** contains the market value of the company in millions of dollars.
- The variable **Profit** contains the profit in millions of dollars.
- The variable **Employee** stores the number of employees in thousands.
- The variable **Weight** contains the sampling weight.

The following SAS statements use PROC SURVEYMEANS to perform the domain analysis, estimating means and other statistics for the overall population and also for the subpopulations (or domain) defined by market type. The DOMAIN statement specifies **Type** as the domain variable:

```

title1 'Top Companies Profile Study';
proc surveymeans data=Company total=800 mean sum;
  var Asset Sale Value Profit Employee;
  weight Weight;
  domain Type;
run;

```

**Output 13.2.1.** Company Profile Study

| Top Companies Profile Study |             |                      |         |             |
|-----------------------------|-------------|----------------------|---------|-------------|
| The SURVEYMEANS Procedure   |             |                      |         |             |
| Data Summary                |             |                      |         |             |
| Number of Observations      |             | 66                   |         |             |
| Sum of Weights              |             | 799.8                |         |             |
| Statistics                  |             |                      |         |             |
| Variable                    | Mean        | Std Error<br>of Mean | Sum     | Std Dev     |
| Asset                       | 6523.488510 | 720.557075           | 5217486 | 1073829     |
| Sale                        | 4215.995799 | 839.132506           | 3371953 | 847885      |
| Value                       | 2145.935121 | 342.531720           | 1716319 | 359609      |
| Profit                      | 188.788210  | 25.057876            | 150993  | 30144       |
| Employee                    | 36.874869   | 7.787857             | 29493   | 7148.003298 |

Output 13.2.1 shows that there are 66 observations in the sample. The sum of the sampling weights equals 799.8, which is close to the total number of companies in the study population.

The “Statistics” table in Output 13.2.1 displays the estimates of the mean and total for all analysis variables for the entire 800 companies, while Output 13.2.2 shows the mean and total estimates for each company type.



**Output 13.2.2.** Domain Analysis for Company Profile Study

| Top Companies Profile Study |          |             |                      |             |             |
|-----------------------------|----------|-------------|----------------------|-------------|-------------|
| The SURVEYMEANS Procedure   |          |             |                      |             |             |
| Domain Analysis: Type       |          |             |                      |             |             |
| Type                        | Variable | Mean        | Std Error<br>of Mean | Sum         | Std Dev     |
| Energy                      | Asset    | 7868.302932 | 1941.699163          | 1449341     | 785962      |
|                             | Sale     | 5419.679099 | 2416.214417          | 998305      | 673373      |
|                             | Value    | 2249.297177 | 520.295162           | 414321      | 213580      |
|                             | Profit   | 289.564658  | 52.512141            | 53338       | 25927       |
|                             | Employee | 14.151194   | 3.974697             | 2606.650000 | 1481.777769 |
| Finance                     | Asset    | 7890.190264 | 1057.185336          | 1855773     | 704506      |
|                             | Sale     | 829.210502  | 115.762531           | 195030      | 74436       |
|                             | Value    | 565.068197  | 76.964547            | 132904      | 48156       |
|                             | Profit   | 63.716837   | 10.099341            | 14986       | 5801.108513 |
|                             | Employee | 5.806293    | 0.811555             | 1365.640000 | 519.658410  |
| HiTech                      | Asset    | 5031.959781 | 732.436967           | 321542      | 183302      |
|                             | Sale     | 5464.292019 | 731.296997           | 349168      | 196013      |
|                             | Value    | 6707.828482 | 1194.160584          | 428630      | 249154      |
|                             | Profit   | 346.407042  | 42.299004            | 22135       | 12223       |
|                             | Employee | 70.766980   | 8.683595             | 4522.010000 | 2524.778281 |
| Manufacturing               | Asset    | 7403.004250 | 1454.921083          | 888361      | 492577      |
|                             | Sale     | 7207.638833 | 2112.444703          | 864917      | 501679      |
|                             | Value    | 2986.442750 | 799.121544           | 358373      | 196979      |
|                             | Profit   | 211.933583  | 39.993255            | 25432       | 13322       |
|                             | Employee | 83.314333   | 31.089019            | 9997.720000 | 6294.309490 |
| Medical                     | Asset    | 5046.570609 | 1218.444638          | 140799      | 131942      |
|                             | Sale     | 3313.219713 | 758.216303           | 92439       | 85655       |
|                             | Value    | 2561.614695 | 530.802245           | 71469       | 64663       |
|                             | Profit   | 218.682796  | 44.051447            | 6101.250000 | 5509.560969 |
|                             | Employee | 46.518996   | 11.135955            | 1297.880000 | 1213.651734 |
| Other                       | Asset    | 1850.250000 | 338.128984           | 58838       | 31375       |
|                             | Sale     | 1620.784906 | 168.686773           | 51541       | 24593       |
|                             | Value    | 1432.820755 | 297.869828           | 45564       | 24204       |
|                             | Profit   | 115.089937  | 27.970560            | 3659.860000 | 2018.201371 |
|                             | Employee | 14.306604   | 2.313733             | 454.950000  | 216.327710  |
| Retail                      | Asset    | 2939.845750 | 393.692369           | 235188      | 94605       |
|                             | Sale     | 7395.453500 | 1746.187580          | 591636      | 263263      |
|                             | Value    | 2103.863125 | 529.756409           | 168309      | 78304       |
|                             | Profit   | 157.171875  | 31.734253            | 12574       | 5478.281027 |
|                             | Employee | 93.624000   | 15.726743            | 7489.920000 | 3093.832061 |
| Transportation              | Asset    | 4712.047359 | 888.954411           | 267644      | 163516      |
|                             | Sale     | 4030.233275 | 1015.555708          | 228917      | 142669      |
|                             | Value    | 1703.330282 | 313.841326           | 96749       | 58947       |
|                             | Profit   | 224.762324  | 56.168925            | 12767       | 8287.585418 |
|                             | Employee | 30.946303   | 6.786270             | 1757.750000 | 1066.586615 |

### Example 13.3. Analyze Survey Data with Missing Values

As described in the section “Missing Values” on page 134, the SURVEYMEANS procedure excludes an observation from the analysis if it has a missing value for the analysis variable or a nonpositive value for the WEIGHT variable.

However, if there is evidence indicating that the nonrespondents are different from the respondents for your study, you can use the DOMAIN statement to compute descriptive statistics “among respondents” from your survey data without imputation for nonrespondents.

Consider the ice cream example in the section “Stratified Sampling” on page 131. Suppose that some of the students failed to provide the amounts spent on ice cream, as shown in the following data set IceCream:

```
data IceCream;
  input Grade Spending @@; datalines;
7 7 7 7 8 . 9 10 7 . 7 10 7 3 8 20 8 19 7 2
7 . 9 15 8 16 7 6 7 6 7 6 9 15 8 17 8 14 9 .
9 8 9 7 7 3 7 12 7 4 9 14 8 18 9 9 7 2 7 1
7 4 7 11 9 8 8 . 8 13 7 . 9 . 9 11 7 2 7 9
;
data StudentTotals;
  input Grade _total_; datalines;
7 1824
8 1025
9 1151
;
```

Considering the possibility that those students who didn’t respond spend differently than those students who did respond, you can create an indicator variable to identify the respondents and non-respondents using the following SAS DATA step:

```
data IceCream;
  set IceCream;
  if Spending=. then Indicator='Nonrespondent';
  else do;
    Indicator='Respondent';
    if (Spending < 10) then Group='less';
    else Group='more';
  end;
  if Grade=7 then Prob=20/1824;
  if Grade=8 then Prob=9/1025;
  if Grade=9 then Prob=11/1151;
  Weight=1/Prob;
```

Variable Indicator identifies a student in the data set as either a respondent or a non-respondent. Variable Group specifies whether a student spent more than \$10 among the respondents.

The following SAS statements analyze the incomplete ice cream data:

```
title1 'Analysis of Ice Cream Spending';
proc surveymeans data=IceCream total=StudentTotals mean sum;
  stratum Grade / list;
  var Spending Group;
  weight Weight;
  domain Indicator;
run;
```

**Output 13.3.1.** Analyze Incomplete Ice Cream Data Excluding Observations with Missing Values

| Analysis of Ice Cream Spending |          |                      |             |             |
|--------------------------------|----------|----------------------|-------------|-------------|
| The SURVEYMEANS Procedure      |          |                      |             |             |
| Data Summary                   |          |                      |             |             |
| Number of Strata               |          | 3                    |             |             |
| Number of Observations         |          | 40                   |             |             |
| Sum of Weights                 |          | 4000                 |             |             |
| Statistics                     |          |                      |             |             |
| Variable                       | Mean     | Std Error<br>of Mean | Sum         | Std Dev     |
| Spending                       | 9.770542 | 0.541381             | 32139       | 1780.792065 |
| Group=less                     | 0.515404 | 0.067092             | 1695.345455 | 220.690305  |
| Group=more                     | 0.484596 | 0.067092             | 1594.004040 | 220.690305  |

Output 13.3.1 shows the mean and total estimates excluding those students who failed to provide the spending amount on ice cream.

**Output 13.3.2.** Analyze Incomplete Ice Cream Data Treating Respondents as a Domain

| Analysis of Ice Cream Spending |            |          |                      |             |             |
|--------------------------------|------------|----------|----------------------|-------------|-------------|
| The SURVEYMEANS Procedure      |            |          |                      |             |             |
| Domain Analysis: Indicator     |            |          |                      |             |             |
| Indicator                      | Variable   | Mean     | Std Error<br>of Mean | Sum         | Std Dev     |
| Nonrespondent                  | Spending   | .        | .                    | .           | .           |
|                                | Group=less | .        | .                    | .           | .           |
|                                | Group=more | .        | .                    | .           | .           |
| Respondent                     | Spending   | 9.770542 | 0.652347             | 32139       | 3515.126876 |
|                                | Group=less | 0.515404 | 0.067257             | 1695.345455 | 221.232029  |
|                                | Group=more | 0.484596 | 0.067257             | 1594.004040 | 221.232029  |

Output 13.3.2 shows the mean and total estimates treating respondents as a domain in the student population. Compared to the estimates in Output 13.3.1, the point estimates are the same, but the variance estimates are slightly higher.

---

## References

- Brick, J.M. and Kalton, G. (1996), "Handling Missing Data in Survey Research," *Statistical Methods in Medical Research*, 5, 215–238.
- Cochran, W.G. (1977), *Sampling Techniques*, Third Edition, New York: John Wiley & Sons, Inc.
- Fuller, W.A., Kennedy, W., Schnell, D., Sullivan, G., and Park, H.J. (1989), *PC CARP*, Ames, IA: Statistical Laboratory, Iowa State University.
- Hidiroglou, M.A., Fuller, W.A., and Hickman, R.D. (1980), *SUPER CARP*, Ames, IA: Statistical Laboratory, Iowa State University.
- Kalton, G. and Kaspyzyk, D. (1986), "The Treatment of Missing Survey Data," *Survey Methodology*, 12, 1–16.
- Kish, L. (1965), *Survey Sampling*, New York: John Wiley & Sons, Inc.



# Subject Index

## B

biquartimax method, 19–20  
biquartimin method, 19

## C

CATMOD procedure  
  iterative proportional fitting, 3  
  maximum likelihood estimation, 3  
  missing values, 6  
  syntax, 3  
  zeros, structural and sampling, 6  
confidence intervals, FACTOR procedure, 21  
confidence limits, FACTOR procedure, 21  
covarimin method, 20  
coverage displays  
  FACTOR procedure, 23  
Crawford-Ferguson method, 19–20

## D

degrees of freedom  
  SURVEYMEANS procedure, 136–137  
domain analysis  
  SURVEYMEANS procedure, 134, 136, 139  
domain mean  
  SURVEYMEANS procedure, 136  
domain total  
  SURVEYMEANS procedure, 137

## E

empty stratum  
  SURVEYMEANS procedure, 135, 137  
equamax method, 19–20  
exact logistic regression  
  LOGISTIC procedure, 92, 94

## F

factor parsimax method, 19–20  
FACTOR procedure  
  coverage displays, 23  
  simplicity functions, 18, 23

## G

GAM procedure  
  comparing PROC GAM with PROC TPSPLINE,  
    61  
  generalized additive model with binary data, 54  
  ODS table names, 54  
generalized Crawford-Ferguson method, 19–20

## H

Harris-Kaiser method, 20

## I

iterative proportional fitting  
  estimation (CATMOD), 3  
  formulas (CATMOD), 6

## L

LOESS procedure  
  automatic smoothing parameter selection, 78  
  kd trees and blending, 77  
  output table names, 81  
LOGISTIC procedure  
  displayed output, 99  
  exact logistic regression, 92, 94  
  ODS table names, 100  
  output data sets, 98

## M

maximum likelihood estimation  
  CATMOD procedure, 3  
missing values  
  SURVEYMEANS procedure, 134, 146  
ML factor analysis  
  and confidence intervals, 21  
MODECLUS procedure  
  smoothing parameter, default, 115

## O

oblimin method, 20  
orthomax method, 19  
output data sets  
  LOGISTIC procedure, 98  
output table names  
  SURVEYMEANS procedure, 139

## P

parsimax method, 19–20  
Procrustes rotation, 20  
promax method, 20

## Q

quartimax method, 19–20  
quartimin method, 21

**S**

salience of loadings, FACTOR procedure, 21

sampling zeros

and structural zeros (CATMOD), 6

simplicity functions

FACTOR procedure, 18, 23

smoothing parameter, default

MODECLUS procedure, 115

statistical computation

SURVEYMEANS procedure, 136

stratified cluster sample

SURVEYMEANS procedure, 140

stratified sampling

SURVEYMEANS procedure, 131

subdomain analysis

SURVEYMEANS procedure, 134

subgroup analysis

SURVEYMEANS procedure, 134

subpopulation analysis

SURVEYMEANS procedure, 134

SURVEYMEANS procedure

degrees of freedom, 136–137

domain analysis, 134, 136, 139

domain mean, 136

domain total, 137

domain variables, 133

empty stratum, 135, 137

missing values, 134, 146

ODS table names, 139

output table names, 139

statistical computation, 136

stratified cluster sample, 140

stratified sampling, 131

subdomain analysis, 134

subgroup analysis, 134

subpopulation analysis, 134

*t* test, 136

**T**

*t* test

SURVEYMEANS procedure, 136

**V**

varimax method, 19–20

**Z**

zeros, structural and sampling

CATMOD procedure, 6

# Syntax Index

## A

- ALPHA= option
  - EXACT statement (LOGISTIC), 92
  - MODEL statement (GAM), 45
  - PROC FACTOR statement, 17

## B

- BY statement
  - GAM procedure, 42

## C

- CATMOD procedure, MODEL statement, 3
  - MISS= option, 5
  - MISSING= option, 5
  - ML option, 3
  - ZERO= option, 5
  - ZEROES= option, 5
  - ZEROS= option, 5
- CLASS statement
  - GAM procedure, 43
- CONTENTS= option
  - TABLES statement (FREQ), 33
- COVER= option
  - PROC FACTOR statement, 17
- COVS option
  - PROC PHREG statement, 127
- COVSANDWICH option
  - PROC PHREG statement, 127

## D

- DATA= option
  - PROC GAM statement, 42
  - SCORE statement (GAM), 46
- DETAILS option
  - MODEL statement (LOESS), 75
- DIST = option
  - MODEL statement (GAM), 45
- DOMAIN statement
  - SURVEYMEANS procedure, 133

## E

- EPSILON = option
  - MODEL statement (GAM), 45
- ESTIMATE option
  - EXACT statement (LOGISTIC), 92
- EXACT statement
  - LOGISTIC procedure, 92
- EXACTONLY option
  - PROC LOGISTIC statement, 91

- EXACTOPTIONS option
  - PROC LOGISTIC statement, 91

## F

- FACTOR procedure, PROC FACTOR statement, 17
  - ALPHA= option, 17
  - COVER= option, 17
  - HKPOWER= option, 18
  - PREROTATE= option, 18
  - RCONVERGE= option, 18
  - RITER= option, 18
  - ROTATE= option, 18
  - SE option, 21
- FORMAT= option
  - TABLES statement (FREQ), 33
- FREQ procedure
  - syntax, 33
- FREQ procedure, TABLES statement, 33
  - CONTENTS= option, 33
  - FORMAT= option, 33
  - OUTCUM option, 34
- FREQ statement
  - GAM procedure, 43

## G

- GAM procedure, 42
  - syntax, 42
- GAM procedure, BY statement, 42
- GAM procedure, CLASS statement, 43
- GAM procedure, FREQ statement, 43
- GAM procedure, ID statement, 44
- GAM procedure, MODEL statement
  - ALPHA= option, 45
  - DIST= option, 45
  - EPSILON= option, 45
  - MAXITER= option, 45
  - METHOD= option, 45
- GAM procedure, OUTPUT statement, 45
  - OUT= option, 45
- GAM procedure, PROC GAM statement, 42
  - DATA= option, 42
- GAM procedure, SCORE statement, 46
  - DATA= option, 46
  - OUT= option, 46

## H

- HKPOWER= option
  - PROC FACTOR statement, 18



**I**

ID statement  
     GAM procedure, 44  
 INTERP= option  
     MODEL statement (LOESS), 76

**J**

JOINT option  
     EXACT statement (LOGISTIC), 92  
 JOINTONLY option  
     EXACT statement (LOGISTIC), 93

**L**

LOESS procedure, MODEL statement  
     DETAILS option, 75  
     INTERP= option, 76  
     SELECT= option, 76  
     SMOOTH= option, 77  
     TRACEL option, 76  
 LOESS procedure, SCORE statement  
     STEPS option, 77  
 LOGISTIC procedure, 91  
     syntax, 91  
 LOGISTIC procedure, EXACT statement, 92  
     ALPHA= option, 92  
     ESTIMATE option, 92  
     JOINT option, 92  
     JOINTONLY option, 93  
     ONESIDED option, 93  
     OUTDIST= option, 93  
 LOGISTIC procedure, PROC LOGISTIC statement,  
     91  
     EXACTONLY option, 91  
     EXACTOPTIONS option, 91

**M**

MAXITER = option  
     MODEL statement (GAM), 45  
 METHOD= option  
     MODEL statement (GAM), 45  
 MISS= option  
     MODEL statement (CATMOD), 5  
 MISSING= option  
     MODEL statement (CATMOD), 5  
 ML option  
     MODEL statement (CATMOD), 3  
 MODEL statement  
     CATMOD procedure, 3  
     TPSPLINE procedure, 44

**O**

ONESIDED option  
     EXACT statement (LOGISTIC), 93  
 OUT= option  
     OUTPUT statement (GAM), 45  
     SCORE statement (GAM), 46  
 OUTCUM option  
     TABLES statement (FREQ), 34  
 OUTDIST= option

EXACT statement (LOGISTIC), 93  
 OUTPUT statement  
     GAM procedure, 45

**P**

PHREG procedure, PROC PHREG statement  
     COVS option, 127  
     COVSANDWICH option, 127  
 PREROTATE= option  
     PROC FACTOR statement, 18  
 PROC FACTOR statement  
     See FACTOR procedure  
 PROC GAM statement  
     See GAM procedure  
 PROC LOGISTIC statement  
     See LOGISTIC procedure

**R**

RCONVERGE= option  
     PROC FACTOR statement, 18  
 RITER= option  
     PROC FACTOR statement, 18  
 ROTATE= option  
     PROC FACTOR statement, 18

**S**

SCORE statement, GAM procedure, 46  
 SE option  
     PROC FACTOR statement, 21  
 SELECT= option  
     MODEL statement (LOESS), 76  
 SMOOTH= option  
     MODEL statement (LOESS), 77  
 STEPS option  
     SCORE statement (LOESS), 77  
 SUBGROUP statement  
     SURVEYMEANS procedure, 133  
 SURVEYMEANS procedure  
     syntax, 133  
 SURVEYMEANS procedure, DOMAIN statement,  
     133

**T**

TABLES statement  
     FREQ procedure, 33  
 TPSPLINE procedure, MODEL statement, 44  
 TRACEL option  
     MODEL statement (LOESS), 76

**Z**

ZERO= option  
     MODEL statement (CATMOD), 5  
 ZEROES= option  
     MODEL statement (CATMOD), 5  
 ZEROS= option  
     MODEL statement (CATMOD), 5



# Your Turn

---

If you have comments or suggestions about *SAS/STAT® Software: Changes and Enhancements, Release 8.1*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute  
Publications Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [suggest@sas.com](mailto:suggest@sas.com)



*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS<sup>®</sup> Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

- 1.** Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
- 2.** Leave the "Subject" line blank.
- 3.** Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.





















SAS Publishing

# What's New in SAS® Software for Release 8.2



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *What's New in SAS® Software for Release 8.2*, Cary, NC: SAS Institute Inc., 2001.

**What's New in SAS® Software for Release 8.2**

Copyright © 2001 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-817-4

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, February 2001

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

IBM® and all other International Business Machines Corporation product or service names are registered trademarks or trademarks of International Business Machines Corporation in the USA and other countries.

Oracle® and all other Oracle Corporation product or service names are registered trademarks or trademarks of Oracle Corporation in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.



---

# Contents

## Important Information About This Book ix

## **PART 1** Base SAS Software 1

### **Chapter 1** △ Base SAS Software: General Features 3

Cross-Environment Data Access (CEDA) 3

SAS Output: Output Delivery System 3

SAS I/O Enhancements 4

European Currency Conversion 5

Windowing Environment 5

Sending E-mail from SAS Using the SMTP Interface 5

### **Chapter 2** △ Base SAS Software: SAS Language Elements 7

Commands 7

Data Set Options 7

Formats 8

Functions and CALL Routines 9

Informats 10

Macro Facility 11

ARM Macros 11

Statements 11

System Options 11

### **Chapter 3** △ Base SAS Software: Procedures 13

The FREQ Procedure 13

The OPTLOAD Procedure 13

The OPTSAVE Procedure 13

The PRINT Procedure 14

The PRINTTO Procedure 14

The PRTDEF Procedure 14

The REGISTRY Procedure 14

The REPORT Procedure 14

The SQL Procedure 15

The UNIVARIATE Procedure 15

### **Chapter 4** △ SAS Software in the CMS Environment 17

Byte File System Support 17

Sending E-mail from SAS Using the SMTP Interface 17

Universal Printing 17

Writing to the SAS Log From a Pipeline 17

SAS Language Elements 18

### **Chapter 5** △ SAS Software in the Microsoft Windows Environment 21

Enhanced Editor 21

|                                                            |           |
|------------------------------------------------------------|-----------|
| Basic Operation of the SAS System                          | 21        |
| SAS Language Elements                                      | 22        |
| <b>Chapter 6 △ SAS Software in the OpenVMS Environment</b> | <b>25</b> |
| SAS Language Elements                                      | 25        |
| <b>Chapter 7 △ SAS Software in the OS/2 Environment</b>    | <b>27</b> |
| SAS Language Elements                                      | 27        |
| <b>Chapter 8 △ SAS Software in the OS/390 Environment</b>  | <b>29</b> |
| Universal Printing                                         | 29        |
| Sending E-mail from SAS Using the SMTP Interface           | 29        |
| SAS Language Elements                                      | 29        |
| <b>Chapter 9 △ SAS Software in UNIX Environments</b>       | <b>31</b> |
| SAS Language Elements                                      | 31        |
| SAS Resources                                              | 32        |

## **PART 2    Other SAS Software Products    33**

|                                                                |           |
|----------------------------------------------------------------|-----------|
| <b>Chapter 10 △ SAS/ACCESS Software</b>                        | <b>35</b> |
| SAS/ACCESS Software for Relational Databases                   | 35        |
| SAS/ACCESS Interface to CA-DATACOM/DB (Available under OS/390) | 37        |
| SAS/ACCESS Interface to IMS-DL/I (Available under OS/390)      | 37        |
| SAS/ACCESS Software for PC File Formats                        | 37        |
| SAS/ACCESS Interface to PeopleSoft                             | 37        |
| SAS/ACCESS Interface to R/3                                    | 37        |
| <b>Chapter 11 △ SAS ADX Interface</b>                          | <b>39</b> |
| Process Variables                                              | 39        |
| Import Design Wizard                                           | 39        |
| Design Details                                                 | 39        |
| <b>Chapter 12 △ SAS/ASSIST Software</b>                        | <b>41</b> |
| Improved Performance Through the Use of Metadata               | 41        |
| New Data Sources Window                                        | 41        |
| New Facility for Defining Your Own Metadata                    | 41        |
| New Sample Programs                                            | 42        |
| New Table/Column Search                                        | 42        |
| For Additional Information                                     | 42        |
| <b>Chapter 13 △ SAS Component Language</b>                     | <b>43</b> |
| SCL List Initialization Syntax Supports Brackets               | 43        |
| <b>Chapter 14 △ SAS/CONNECT Software</b>                       | <b>45</b> |
| Enhancements to RSUBMIT                                        | 45        |
| Monitoring SAS/CONNECT Tasks                                   | 45        |
| RSUBMIT and Macro Processing                                   | 45        |

|                                                                |           |
|----------------------------------------------------------------|-----------|
| Complete Set of Agent Services                                 | 46        |
| Common Messaging Interface                                     | 46        |
| SAS/CONNECT Spawners                                           | 46        |
| File Compression Improves Performance for Large Data Transfers | 46        |
| CEDA Technology Is Now Part of Base SAS                        | 46        |
| <b>Chapter 15 △ SAS/EIS Software</b>                           | <b>47</b> |
| Distinct Values (NUNIQUE) Window                               | 47        |
| Computed Values Window                                         | 47        |
| Sorting by Unformatted Category Values                         | 47        |
| Specifying Labels for Statistics                               | 48        |
| Activating Access Control                                      | 48        |
| <b>Chapter 16 △ SAS/ETS Software</b>                           | <b>49</b> |
| ARIMA Procedure                                                | 49        |
| COUNTREG Procedure                                             | 49        |
| DATETIME Formats                                               | 49        |
| Investment Analysis System                                     | 50        |
| MDC Procedure                                                  | 50        |
| MODEL Procedure                                                | 50        |
| QLIM Procedure                                                 | 50        |
| Time Series Forecasting System                                 | 50        |
| VARMAX Procedure                                               | 51        |
| X12 Procedure                                                  | 51        |
| For Additional Information                                     | 51        |
| <b>Chapter 17 △ SAS/GIS Software</b>                           | <b>53</b> |
| PROC GIS MAP ACTION Statement                                  | 53        |
| Updated Tutorial                                               | 54        |
| <b>Chapter 18 △ SAS/GRAPH Software</b>                         | <b>55</b> |
| GCHART Procedure                                               | 55        |
| GCONTOUR Procedure                                             | 55        |
| GPLOT Procedure                                                | 55        |
| GMAP Procedure                                                 | 56        |
| GRADAR Procedure                                               | 56        |
| GREPLAY Procedure                                              | 56        |
| Generating Graphs for the Web                                  | 56        |
| <b>Chapter 19 △ SAS/IML Software</b>                           | <b>57</b> |
| Wavelet Analysis                                               | 57        |
| For Additional Information                                     | 57        |
| <b>Chapter 20 △ SAS Integration Technologies Software</b>      | <b>59</b> |
| SAS Business Intelligence Portal                               | 59        |
| SAS Publishing Framework                                       | 59        |
| Directory Services                                             | 60        |

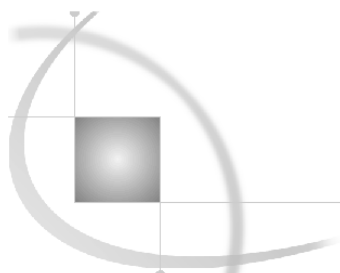
|                                                                   |           |
|-------------------------------------------------------------------|-----------|
| Distributed Objects                                               | 60        |
| Message Queuing                                                   | 60        |
| Enhancements to SAS Package Retriever                             | 60        |
| Enhancements to SAS Publisher                                     | 61        |
| For Additional Information                                        | 62        |
| <b>Chapter 21 <math>\triangle</math> SAS/IntrNet Software</b>     | <b>63</b> |
| Application Dispatcher                                            | 63        |
| Java Tool                                                         | 64        |
| MDDB Report Viewer                                                | 64        |
| New Tool: SAS Design-Time Controls                                | 64        |
| Xplore Sample Application                                         | 64        |
| For Additional Information                                        | 64        |
| <b>Chapter 22 <math>\triangle</math> SAS/MDDB Server Software</b> | <b>65</b> |
| New Methods for the MDDB_M Class                                  | 65        |
| Enhanced Method for the MDDB_M Class                              | 65        |
| <b>Chapter 23 <math>\triangle</math> SAS OLAP Server Software</b> | <b>67</b> |
| Support for Multiple Environments                                 | 67        |
| Experimental User Interface for Building MDDBs                    | 67        |
| Activating Access Control                                         | 67        |
| <b>Chapter 24 <math>\triangle</math> SAS/OR Software</b>          | <b>69</b> |
| BOM Procedure                                                     | 69        |
| CPM Procedure                                                     | 69        |
| GANTT Procedure                                                   | 69        |
| INTPOINT Procedure                                                | 70        |
| PM Procedure                                                      | 70        |
| For Additional Information                                        | 70        |
| <b>Chapter 25 <math>\triangle</math> SAS/QC Software</b>          | <b>71</b> |
| CAPABILITY Procedure                                              | 71        |
| RELIABILITY Procedure                                             | 71        |
| SHEWHART Procedure                                                | 72        |
| For Additional Information                                        | 73        |
| <b>Chapter 26 <math>\triangle</math> SAS/SHARE Software</b>       | <b>75</b> |
| Services Support                                                  | 75        |
| <b>Chapter 27 <math>\triangle</math> SAS/STAT Software</b>        | <b>77</b> |
| BOXPLOT Procedure                                                 | 77        |
| CATMOD Procedure                                                  | 78        |
| FACTOR Procedure                                                  | 78        |
| FREQ Procedure                                                    | 78        |
| GAM Procedure                                                     | 78        |
| LIFEREG Procedure                                                 | 79        |

|                                                          |           |
|----------------------------------------------------------|-----------|
| LOGISTIC Procedure                                       | 79        |
| LOESS Procedure                                          | 79        |
| MI Procedure                                             | 80        |
| MIANALYZE Procedure                                      | 80        |
| NPAR1WAY Procedure                                       | 80        |
| PROBIT Procedure                                         | 80        |
| REG Procedure                                            | 81        |
| SURVEYMEANS Procedure                                    | 81        |
| TRANSREG Procedure                                       | 81        |
| References                                               | 81        |
| For Additional Information                               | 81        |
| <b>Chapter 28 △ SAS/Warehouse Administrator Software</b> | <b>83</b> |
| Metadata API Enhancements                                | 83        |
| LDAP Security Interface Enhancements                     | 83        |
| Closer Integration with Enterprise Reporter              | 84        |
| OLAP Data Store Enhancements                             | 84        |
| Process Library Enhancements                             | 84        |
| Getting Started Tutorial                                 | 85        |

## **PART 3    New Products    87**

|                                                         |           |
|---------------------------------------------------------|-----------|
| <b>Chapter 29 △ SAS/ACCESS Interface to PeopleSoft</b>  | <b>89</b> |
| Overview                                                | 89        |
| For Additional Information                              | 89        |
| <b>Chapter 30 △ SAS Data Quality - Cleanse Software</b> | <b>91</b> |
| Overview                                                | 91        |
| SAS Data Quality – Cleanse Features                     | 91        |
| <b>Index</b>                                            | <b>93</b> |





## Important Information About This Book

---

*Where to Find Complete Information About New and Enhanced Features* ix  
*Get the Latest News* ix

---

### Where to Find Complete Information About New and Enhanced Features

This book gives you a quick, convenient overview of features that have been added or enhanced in base SAS software and in approximately 20 other SAS software products for Release 8.2. Detailed information for most products is located in the SAS System Help.

A new Web site provides information about what's new in Release 8.2 of SAS Software, as well as the changes and enhancements reports for SAS/ASSIST, SAS/ETS, SAS/IML, SAS/OR, SAS/QC, and SAS/STAT software. All of these reports can be accessed and printed in HTML or PDF formats at [www.sas.com/newversion](http://www.sas.com/newversion).

Visit this site often for the most up-to-date information about Release 8.2. The Release 8.2 changes and enhancements reports are also available to you in PDF format on CD-ROM by contacting the SAS Distribution Center at 919-531-7850. If you prefer hardcopy books, you can purchase the book version of selected\* Release 8.2 documents from SAS Fulfillment Services at 800-727-3228.

---

### Get the Latest News

To stay up-to-date, you can subscribe to NEWDOCNEWS, a monthly electronic newsletter. This newsletter provides SAS software users with information about new books that were published during the previous month, notice of new Observations articles on our Web page, announcements of online documentation, notice of documentation updates, and general news items about SAS documentation.

Customers in the USA can subscribe to this electronic newsletter either by filling out a Web form at [www.sas.com/service/doc/newdoc/newdocintro.html](http://www.sas.com/service/doc/newdoc/newdocintro.html) or as follows:

- 1 Send an e-mail message to [listserv@vm.sas.com](mailto:listserv@vm.sas.com).
- 2 Leave the Subject line blank.

---

\* Changes and enhancements reports for SAS/ASSIST, SAS/IML, and SAS/QC software are not available in hardcopy.

**3** The text of the message should be **Subscribe NEWDOCNEWS-L John Doe** (replace John Doe with your name).

**4** Send the message.

You will receive a message confirming that you have been added to the NEWDOCNEWS mailing list. The message will also contain instructions on how to unsubscribe from the mailing list, in case you choose to do so in the future.

International customers can subscribe to the newsletter as follows:

**1** Send an e-mail message to **pubs@unx.sas.com**.

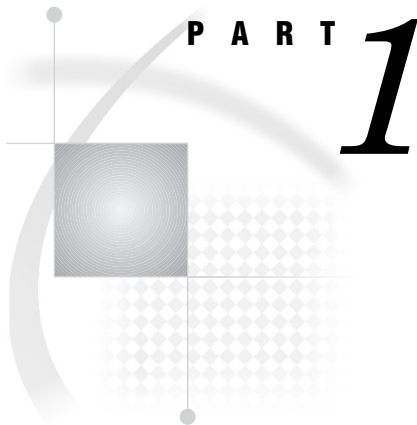
**2** The subject line should be **Add to NEWDOCNEWS**.

**3** The text of the message should be **Subscribe NEWDOCNEWS John Doe** (replace John Doe with your name).

**4** Send the message.

We will forward your name and e-mail address to your local SAS office, and they will add you to the mailing list for their electronic newsletter.

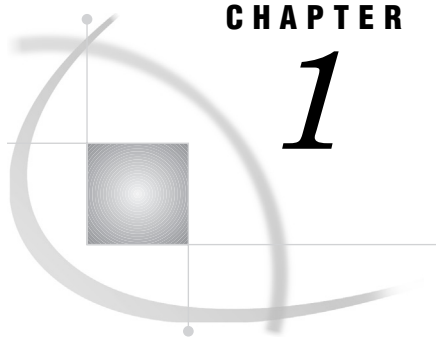




## **Base SAS Software**

|                  |                                                          |           |
|------------------|----------------------------------------------------------|-----------|
| <i>Chapter 1</i> | <b>Base SAS Software: General Features</b>               | <b>3</b>  |
| <i>Chapter 2</i> | <b>Base SAS Software: SAS Language Elements</b>          | <b>7</b>  |
| <i>Chapter 3</i> | <b>Base SAS Software: Procedures</b>                     | <b>13</b> |
| <i>Chapter 4</i> | <b>SAS Software in the CMS Environment</b>               | <b>17</b> |
| <i>Chapter 5</i> | <b>SAS Software in the Microsoft Windows Environment</b> | <b>21</b> |
| <i>Chapter 6</i> | <b>SAS Software in the OpenVMS Environment</b>           | <b>25</b> |
| <i>Chapter 7</i> | <b>SAS Software in the OS/2 Environment</b>              | <b>27</b> |
| <i>Chapter 8</i> | <b>SAS Software in the OS/390 Environment</b>            | <b>29</b> |
| <i>Chapter 9</i> | <b>SAS Software in UNIX Environments</b>                 | <b>31</b> |





## CHAPTER

## 1

# Base SAS Software: General Features

---

|                                                         |   |
|---------------------------------------------------------|---|
| <i>Cross-Environment Data Access (CEDA)</i>             | 3 |
| <i>SAS Output: Output Delivery System</i>               | 3 |
| <i>SAS I/O Enhancements</i>                             | 4 |
| <i>European Currency Conversion</i>                     | 5 |
| <i>Windowing Environment</i>                            | 5 |
| <i>Sending E-mail from SAS Using the SMTP Interface</i> | 5 |

---

## Cross-Environment Data Access (CEDA)

CEDA, which enables your SAS programs to directly access SAS data files that were created in another operating environment, is now available as part of base SAS software. You no longer need to license SAS/CONNECT software in order to use CEDA functionality.

---

## SAS Output: Output Delivery System

- The following ODS statements are now available:
  - The ODS \_ALL\_ CLOSE statement closes all open ODS output destinations, including the Listing destination.
  - The ODS ESCAPECHAR statement enables you to apply simple formatting to a cell or paragraph in ODS output.
  - The ODS PROCLABEL statement enables you to change a procedure label in the table of contents.
  - The ODS RESULTS statement displays ODS output in the Results window.
- For all destinations except the Output and Listing destinations, the following style attributes are now supported:
  - BACKGROUNDIMAGE specifies an image to use as the background for your output.
  - PREIMAGE specifies an image to place before a table or cell.
  - POSTIMAGE specifies an image to place after a table or cell.
- The ODS HTML statement now supports the ID= option, which enables you to run multiple instances of the same destination at the same time.
- The ODS PATH statement now supports the following actions and path arguments. The actions enable you to place one or more locations in the ODS path or to remove one or more locations from the ODS path.

- APPEND adds one or more locations to the end of a path.
- PREPEND adds one or more locations to the beginning of a path.
- REMOVE removes one or more locations from the path.

The path arguments set or display the ODS path.

- RESET sets the ODS path to the default setting SASUSER.TEMPLAT (UPDATE) SASHELP.TMPLMST (READ).
- SHOW displays the current ODS path.
- VERIFY sets the ODS path to include only templates that SAS supplies. VERIFY is the same as specifying ODS PATH SASHELP.TMPLMST (READ).
- The ODS PRINTER statement now supports the following options:
  - The DISPLAY= option opens, manages, or closes the DISPLAY destination.
  - The ID= option enables you to run multiple instances of the same destination at the same time.
  - The PCL= option opens, manages, or closes the PCL destination.
  - The PDF= option opens, manages, or closes the PDF destination.
  - The PS= option opens, manages, or closes the PostScript destination.
  - The STARTPAGE= option controls page breaks.
- The ODS RTF statement now supports the following options:
  - The ID= option enables you to run multiple instances of the same destination at the same time.
  - The STARTPAGE= option controls page breaks.
- PROC TEMPLATE has been enhanced as follows:
  - New access modes have been added for the STORE= option:
    - READ provides read-only access.
    - WRITE provides write access (always creating a new template store). If the store name already exists, it will be overwritten.
    - UPDATE provides update access (creating a new template store only if the one that is specified does not exist). UPDATE is the default access mode.
  - New PATH statement actions enable you to place one or more locations in the ODS path or to remove one or more locations from the path:
    - APPEND adds one or more locations to the end of a path.
    - PREPEND adds one or more locations to the beginning of a path.
    - REMOVE removes one or more locations from a path.
  - New PATH statement path arguments set or display the ODS path:
    - RESET sets the ODS path to the default setting SASUSER.TEMPLAT (UPDATE) SASHELP.TMPLMST (READ).
    - SHOW displays the current ODS path.
    - VERIFY sets the ODS path to include only templates that SAS supplies. VERIFY is the same as specifying ODS PATH SASHELP.TMPLMST (READ).

---

## SAS I/O Enhancements

When a request is made to compress a SAS data set with either the COMPRESS= data set option or the COMPRESS= system option, SAS now determines whether the

resulting compressed file will be larger than an uncompressed file. If so, SAS creates an uncompressed data set and issues a warning message stating that the file has not been compressed.

---

## European Currency Conversion

SAS enables you to convert European currencies by providing a group of formats, informats, and a function that convert

- an amount in one country's currency to an equivalent amount in another country's currency
- a country's currency to euros
- euros to a specific country's currency.

---

## Windowing Environment

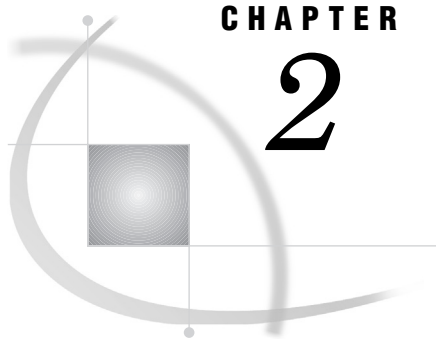
- For the External File Interface (EFI), the option **Number of guessing rows** has been added to the Options for Import window. The option specifies how many rows (observations) the EFI examines when you request that variables be automatically created.
- Color-coded syntax is now supported in code-editing windows, such as the Program Editor or Notepad. You can use the new Edit Scheme window to modify color schemes. You can also use the new COLOR SYNTAX command to invoke color-coded syntax at any time.
- There is a new Printer Setup window for Universal Printing.

---

## Sending E-mail from SAS Using the SMTP Interface

You can now use the FILENAME, EMAIL (SMTP) Access Method to send electronic mail programmatically from SAS in all operating environments in which SAS runs. In addition, you can change encoding for processing the message as well as attachments. To send e-mail to an SMTP (Simple Mail Transfer Protocol) server, you first use the EMAILSYS system option to specify the SMTP e-mail interface. Next, use the FILENAME statement to specify the EMAIL device type, and then submit SAS statements in a DATA step or in SCL code. For example, you can use the logic of a DATA step to subset e-mail distribution based on a large data set of e-mail addresses. You can also direct output through e-mail based on the results of processing.





## CHAPTER

## 2

# Base SAS Software: SAS Language Elements

---

|                                    |    |
|------------------------------------|----|
| <i>Commands</i>                    | 7  |
| <i>Data Set Options</i>            | 7  |
| <i>Formats</i>                     | 8  |
| <i>Functions and CALL Routines</i> | 9  |
| <i>Informats</i>                   | 10 |
| <i>Macro Facility</i>              | 11 |
| <i>ARM Macros</i>                  | 11 |
| <i>Statements</i>                  | 11 |
| <i>System Options</i>              | 11 |

---

## Commands

The following commands are new:

- ❑ The COLOR SYNTAX command applies color-coded syntax to code-editing windows, such as the Notepad or the Program Editor.
- ❑ The DMOPTLOAD command reads SAS system option settings that are stored in the SAS registry or in a SAS data set, and puts them into effect.
- ❑ The DMOPTSAVE command saves the current system option settings either to a registry key or to a data set that contains two variables named OPTNAME and OPTVALUE.

The following commands have been changed or enhanced:

- ❑ The DMPRTCREATE command opens a Wizard that lets you define a new printer definition.
  - ❑ The DMPRTSETUP command opens the Print Setup dialog box.
  - ❑ The SYNCOLOR command toggles the color-coded syntax in the Program Editor, Notepad, or SCL Editor on and off.
  - ❑ The SYNCONFIG command opens the Edit Scheme window, which enables you to change the color syntax scheme for the color editor.
- 

## Data Set Options

When a request is made to compress a SAS data set with the COMPRESS= data set option, SAS now determines whether the resulting compressed file will be larger than an uncompressed file. If so, SAS creates an uncompressed data set and issues a warning message stating that the file has not been compressed.

## Formats

- There are four new Unicode formats:
  - The \$UTF8X*w*. format writes character data in 8-bit UCS2 (Unicode).
  - The \$UCS2B*w*. format writes a SAS value in big endian 16-bit UCS2 Unicode without a byte-order mark.
  - The \$UCS2L*w*. format writes a SAS value in little endian 16-bit UCS2 Unicode without a byte-order mark.
  - The \$UCS2X*w*. format writes 16-bit UCS2 Unicode without a byte-order mark.
- The following Date and Time formats are new:
  - The DTMONY*Yw*. format writes the date part of a datetime value as the month and year in the form *mmmyy* or *mmmyyyy*.
  - The DTWKDATX*w*. format writes the date part of a datetime value as the day of the week and the date in the form *day-of-week, dd mmm yy* (or *yyyy*).
  - The DTYEAR*w*. format writes the date part of a datetime value as the year in the form *yy* or *yyyy*.
  - The DTYYQC*w*. format writes the date part of a datetime value as the year and the quarter and separates them with a colon (:).
- The following binary formats are available:
  - The S370FHEX*w.d* format writes real binary (floating-point) values in hexadecimal representation.
  - The VAXRB*w.d* format writes real binary (floating-point) data in VMS format.
- The following European currency conversion formats are available:
  - The EUROW*w.d* and EUROX*w.d* formats write values that contain the euro (€) symbol:
    - The EUROW*w.d* format writes numeric values with a leading euro (€) symbol, a comma that separates every three digits, and a period that separates the decimal fraction.
    - The EUROX*w.d* format writes numeric values with a leading euro (€) symbol, a period that separates every three digits, and a comma that separates the decimal fraction. This format is commonly used in European countries.
  - The formats that begin with EUROFR convert an amount in a specific European currency to an equivalent amount in euros:
    - EURFRAT*Sw.d* converts an amount from Austrian schillings to euros.
    - EURFRBEF*w.d* converts an amount from Belgian francs to euros.
    - EURFRCHF*w.d* converts an amount from Swiss francs to euros.
    - EURFRCKZ*w.d* converts an amount from Czech koruny to euros.
    - EURFRDEM*w.d* converts an amount from German marks to euros.
    - EURFRDKK*w.d* converts an amount from Danish kroner to euros.
    - EURFRESP*w.d* converts an amount from Spanish pesetas to euros.
    - EURFRFIM*w.d* converts an amount from Finnish markkaa to euros.
    - EURFRFRF *w.d* converts an amount from French francs to euros.
    - EURFRGBP*w.d* converts an amount from British pounds to euros.
    - EURFRGRD*w.d* converts an amount from Greek drachmas to euros.



- EURFRHUF*w.d* converts an amount from Hungarian forints to euros.
- EURFRIEP*w.d* converts an amount from Irish pounds to euros.
- EURFRITL*w.d* converts an amount from Italian lire to euros.
- EURFRLUF*w.d* converts an amount from Luxembourg francs to euros.
- EURFRNLG*w.d* converts an amount from Dutch guilders to euros.
- EURFRNOK*w.d* converts an amount from Norwegian krone to euros.
- EURFRPLZ*w.d* converts an amount from Polish zlotys to euros.
- EURFRPTE *w.d* converts an amount from Portuguese escudos to euros.
- EURFRROL*w.d* converts an amount from Romanian lei to euros.
- EURFRRUR*w.d* converts an amount from Russian rubles to euros.
- EURFRSEK*w.d* converts an amount from Swedish kronor to euros.
- EURFRSIT*w.d* converts an amount from Slovenian tolar to euros.
- EURFRTRL*w.d* converts an amount from Turkish liras to euros.
- EURFRYUD*w.d* converts an amount from Yugoslavian dinars to euros.
- The formats that begin with EUROTO convert an amount in euros to an equivalent amount in a specific European currency:
  - EURTOATSw*d* converts an amount in euros to Austrian schillings.
  - EURTOBEF*w.d* converts an amount in euros to Belgian francs.
  - EURTOCHF*w.d* converts an amount in euros to Swiss francs.
  - EURTOCZK*w.d* converts an amount in euros to Czech koruny.
  - EURTOFIM*w.d* converts an amount in euros to Finnish markkaa.
  - EURTODEM*w.d* converts an amount in euros to German marks.
  - EURTODKK*w.d* converts an amount in euros to Danish kroner.
  - EURTOESP*w.d* converts an amount in euros to Spanish pesetas.
  - EURTOFRF*w.d* converts an amount in euros to French francs.
  - EURTOGBP*w.d* converts an amount in euros to British pounds.
  - EURTOGRD*w.d* converts an amount in euros to Greek drachmas.
  - EURTOHUF*w.d* converts an amount in euros to Hungarian forints.
  - EURTOIEP*w.d* converts an amount in euros to Irish pounds.
  - EURTOITL*w.d* converts an amount in euros to Italian lire.
  - EURTOLUF*w.d* converts an amount in euros to Luxembourg francs.
  - EURTONLG*w.d* converts an amount in euros to Dutch guilders.
  - EURTONOK*w.d* converts an amount in euros to Norwegian krone.
  - EURTOPLZ*w.d* converts an amount in euros to Polish zlotys.
  - EURTOPTE*w.d* converts an amount in euros to Portuguese escudos.
  - EURTOROL*w.d* converts an amount in euros to Romanian lei.
  - EURTORUR*w.d* converts an amount in euros to Russian rubles.
  - EURTOSEK*w.d* converts an amount in euros to Swedish kronor.
  - EURTOSIT*w.d* converts an amount in euros to Slovenian tolar.
  - EURTOTRL*w.d* converts an amount in euros to Turkish liras.
  - EURTOYUD*w.d* converts an amount in euros to Yugoslavian dinars.

---

## Functions and CALL Routines

- The following function and CALL routine are experimental in Release 8.2:

- The RXPATTERN function parses a pattern and returns a value.
  - The CALL RXSTYLE routine specifies using the registry key for pattern matching.
- The EUROCURRE function converts one European currency to another and returns a value.
- The PATHNAME function now has an optional argument called *search-ref*. If the name of a fileref is identical to the name of a libref, you can use the *search-ref* argument to specify whether you want to search for a fileref or for a libref. If you specify a value of F, SAS searches for a fileref. If you specify a value of L, SAS searches for a libref.
- Six new functions are available to use with the ARM wrapper macros. Invoking the SAS interface to ARM through macros is preferable to using the ARM functions, because the ARM macros are easier to use and have additional capabilities. See ARM Macros.
- The new CALL SYMDEL routine deletes the specified variable from the macro global symbol table.
- SAS now supports processing on 64-bit platforms as well as on 32-bit platforms. This functionality provides a new set of functions and a CALL routine that enable you to perform pointer arithmetic on character values. The following new functions and CALL routine can be used on both 32-bit and 64-bit platforms:
  - The ADDRLONG function returns the memory address of a variable.
  - The PEEKLONG function stores the contents of a memory address in a numeric variable.
  - The PEEKCLONG function stores the contents of a memory address in a character variable.
  - The POKELONG function writes a value directly into memory.
  - The CALL POKELONG routine writes a value directly into memory.
  - The PTRLONGADD function returns the pointer address as a character variable.
- Because SAS now supports processing on 64-bit platforms as well as on 32-bit platforms, the following functions and CALL routine are now restricted to use on 32-bit platforms only:
  - ADDR
  - PEEK
  - PEEKC
  - POKE
  - CALL POKE

---

## Informats

- There are four new Unicode informats:
  - The \$UTF8Xw. informat reads character data in 8-byte Unicode without a byte-order mark.
  - The \$UCS2Bw. informat reads character data in big endian 16-bit UCS2 Unicode without a byte-order mark.
  - The \$UCS2Lw. informat reads character data in little endian 16-bit UCS2 Unicode without a byte-order mark.
  - The \$UCS2Xw. informat reads character data in 16-bit UCS2 Unicode without a byte-order mark.

- The following binary informat is available:
  - The S370FHEX*w.d* informat reads real binary (floating-point) values in hexadecimal representation.
- The following European currency conversion informats are available:
  - The EURO*w.d* informat reads numeric values and removes embedded characters.
  - The EUROX*w.d* informat reads numeric values and removes embedded characters in values where a comma is the separator between a whole number and the decimal portion.

---

## Macro Facility

- The new %SYMDEL macro deletes the listed variable(s) from the macro global symbol table.
- The following automatic macro variables are new:
  - SYSPROCNAME is an automatic macro variable whose value indicates what the SAS language processor is currently executing. The value can be either DATASTEP or the name of a SAS procedure.
  - SYSMACRONAME is an automatic macro variable that returns the name of the currently executing macro.

---

## ARM Macros

- The new Application Response Measurement (ARM) macros provide a way to measure the performance of an application as the application is executing. The ARM macros are not part of the SAS Macro Facility. They are used only with the ARM function calls. Invoking the SAS interface to ARM through macros is preferable to using the functions, because the ARM macros are easier to use and have additional capabilities.

---

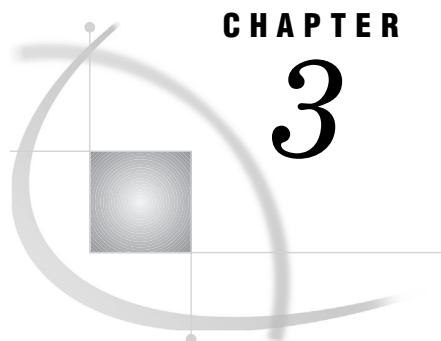
## Statements

- The FILE, FILENAME, and INFILE statements support the new ENCODING= statement option. This option enables you to dynamically change the character-set encoding for processing external data.
- You can now use the FILENAME, EMAIL (SMTP) access method to send electronic mail programmatically from SAS in all operating environments in which SAS runs. In addition, you can change encoding for processing the message as well as attachments. To send e-mail to an SMTP (Simple Mail Transfer Protocol) server, you first use the EMAILSYS system option to specify the SMTP e-mail interface. Next, use the FILENAME statement to specify the EMAIL device type, and then submit SAS statements in a DATA step or in SCL code. For example, you can use the logic of a DATA step to subset e-mail distribution based on a large data set of e-mail addresses. You can also direct output through e-mail based on the results of processing.

---

## System Options

- The following system options are new:
  - When set to YES, AUTOSIGNON= causes the RSUBMIT command or statement to automatically execute a SIGNON command, if no current connection exists.
  - SYSPRINTFONT specifies which font to use when printing with the current default printer.
  - UUIDCOUNT= specifies the number of UUIDs (Universally Unique Identifiers) to acquire each time the UUID generator daemon is used.
  - UUIDGENDHOST= identifies the host and port of the UUID generator daemon.
- The following system options have been changed or enhanced:
  - When a request is made to compress a SAS data set with the COMPRESS= system option, SAS now determines whether the resulting compressed file will be larger than an uncompressed file. If so, SAS creates an uncompressed data set and issues a warning message stating that the file has not been compressed.
  - The PROGSIG= system option no longer has any effect on the printed output of statistical procedures. It has been superseded by features in the TEMPLATE procedure, which is part of ODS, the Output Delivery System.



## CHAPTER

## 3

## Base SAS Software: Procedures

---

|                                 |    |
|---------------------------------|----|
| <i>The FREQ Procedure</i>       | 13 |
| <i>The OPTLOAD Procedure</i>    | 13 |
| <i>The OPTSAVE Procedure</i>    | 13 |
| <i>The PRINT Procedure</i>      | 14 |
| <i>The PRINTTO Procedure</i>    | 14 |
| <i>The PRTDEF Procedure</i>     | 14 |
| <i>The REGISTRY Procedure</i>   | 14 |
| <i>The REPORT Procedure</i>     | 14 |
| <i>The SQL Procedure</i>        | 15 |
| <i>The UNIVARIATE Procedure</i> | 15 |

---

### The FREQ Procedure

- The new POINT option in the EXACT statement requests exact point probabilities for the test statistics.
- The BINOMIAL option now includes a new LEVEL= suboption to specify which variable level PROC FREQ uses to compute the proportion of observations. By default, PROC FREQ computes the proportion of observations for the first variable level that appears in the output.
- The new BINOMIALC option in the TABLES statement computes the BINOMIAL option statistics for one-way tables and includes a continuity correction in the asymptotic confidence limits and asymptotic test. The BINOMIAL keyword in the OUTPUT statement writes these statistics to a SAS data set.
- The new RISKDIFFC option in the TABLES statement computes the RISKDIFF option statistics for 2×2 tables, and includes a continuity correction in the asymptotic confidence limits. The RISKDIFF keyword in the OUTPUT statement writes these statistics to a SAS data set.

---

### The OPTLOAD Procedure

The new OPTLOAD procedure reads SAS system option settings that are stored in the SAS Registry or in a SAS data set and puts them into effect.

---

### The OPTSAVE Procedure

The new OPTSAVE procedure saves the current SAS system option settings in the SAS Registry or in a SAS data set.

---

## The PRINT Procedure

The new STYLE= option enables you to control presentation attributes like font face, font weight, and color when using the Output Delivery System. This option is supported for all destinations except Listing and Output. STYLE= is available on the PROC PRINT statement, ID statement, SUM statement, and VAR statement.

---

## The PRINTTO Procedure

The LOG= and PRINT= options now support printing to a SAS catalog entry that is specified with a fileref.

---

## The PRTDEF Procedure

The new PRTDEF procedure enables you to create printer definitions in batch mode either for yourself or for all SAS users at a site.

---

## The REGISTRY Procedure

The new UPCASE option enables you to convert all incoming key names to upper case.

---

## The REPORT Procedure

- The REPORT procedure now supports the use of a SAS format or a user-defined format as the value of a style attribute.
- The REPORT procedure now supports Quantile statistics. These include the median, P1, P5, P10, P25, P75, P90, P95, and P99. The new options in the PROC statement control how the procedure computes the quantile statistics.
  - The QMARKERS= option specifies the sample size to use for the P2 quantile estimation method.
  - The QMETHOD= option specifies the quantile estimation method.
  - The QNTLDEF= option specifies the mathematical definition to calculate quantiles.
- The new COMPLETECOLS | NOCOMPLETECOLS option in the PROC statement creates all possible combinations of across variable values.
- The new COMPLETEROWS | NOCOMPLETEROWS option in the PROC statement creates all possible combinations of group variable values.
- The new EXCLUSIVE option in the DEFINE statement excludes all combinations of an item that are not found in a preloaded range of user-defined formats.
- The new MISSING option in the DEFINE statement treats missing values as valid values for an item.

- The new PRELOADFMT option in the DEFINE statement specifies that all formats are preloaded for an item.
- The new WEIGHT= option in the DEFINE statement specifies a numeric variable whose values weight the value of the analysis variable.

---

## The SQL Procedure

- The new BTRIM function trims a single character from the beginning, end or both beginning and end of a string.
- The SQL procedure contains three new joins:
  - The new CROSS JOIN produces the crossproduct of two tables.
  - The new NATURAL JOIN selects rows from two tables that have equal values in columns that share the same name.
  - The new UNION JOIN returns a union of the columns of both tables.
- The new fold functions change the case of their arguments.
- The new truncated string comparison operators compare two strings after making the strings the same length by truncating the longer string to the same length as the shorter string.
- The new SUBSTRING function returns part of a character expression.
- The LIKE condition has a new ESCAPE clause that overrides the default behavior of the \_ (underscore) and the % (percent sign).

---

## The UNIVARIATE Procedure

- The HISTOGRAM statement supports the following new options:
  - CTEXTSIDE= specifies the color for the row labels that are displayed along the left side of the comparative histogram.
  - CTEXTTOP= specifies the color for the row labels that are displayed along the top of the comparative histogram.
  - ENDPOINTS= uses the endpoints as the tick mark values for the horizontal axis instead of the midpoints.
  - The HISTOGRAM statement provides some additional goodness-of-fit tests based on the empirical distribution function (EDF). The following table summarizes the different combinations of the estimated parameters for which EDF tests are available.

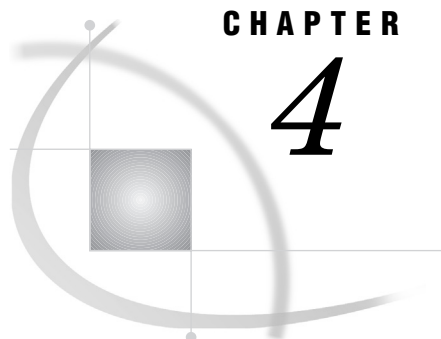
**Table 3.1**   Availability of EDF Tests

| Distribution | Parameters       |                  |                             | Tests Available |
|--------------|------------------|------------------|-----------------------------|-----------------|
|              | Threshold        | Scale            | Shape                       |                 |
| Beta         | $\theta$ known   | $\sigma$ known   | $\alpha, \beta$ known       | all             |
|              | $\theta$ known   | $\sigma$ known   | $\alpha, \beta < 5$ unknown | all             |
| Exponential  | $\theta$ known   | $\sigma$ known   |                             | all             |
|              | $\theta$ known   | $\sigma$ unknown |                             | all             |
|              | $\theta$ unknown | $\sigma$ known   |                             | all             |
|              | $\theta$ unknown | $\sigma$ unknown |                             | all             |

| Distribution | Parameters       |                  |                      | Tests Available |
|--------------|------------------|------------------|----------------------|-----------------|
|              | Threshold        | Scale            | Shape                |                 |
| Gamma        | $\theta$ known   | $\sigma$ known   | $\alpha$ known       | all             |
|              | $\theta$ known   | $\sigma$ unknown | $\alpha$ known       | all             |
|              | $\theta$ known   | $\sigma$ known   | $\alpha$ unknown     | all             |
|              | $\theta$ known   | $\sigma$ unknown | $\alpha$ unknown     | all             |
|              | $\theta$ unknown | $\sigma$ known   | $\alpha > 1$ known   | all             |
|              | $\theta$ unknown | $\sigma$ unknown | $\alpha > 1$ known   | all             |
|              | $\theta$ unknown | $\sigma$ known   | $\alpha > 1$ unknown | all             |
|              | $\theta$ unknown | $\sigma$ unknown | $\alpha > 1$ unknown | all             |
| Lognormal    | $\theta$ known   | $\zeta$ known    | $\sigma$ known       | all             |
|              | $\theta$ known   | $\zeta$ known    | $\sigma$ unknown     | $A^2$ and $W^2$ |
|              | $\theta$ known   | $\zeta$ unknown  | $\sigma$ known       | $A^2$ and $W^2$ |
|              | $\theta$ known   | $\zeta$ unknown  | $\sigma$ unknown     | all             |
|              | $\theta$ unknown | $\zeta$ known    | $\sigma < 3$ known   | all             |
|              | $\theta$ unknown | $\zeta$ known    | $\sigma < 3$ unknown | all             |
|              | $\theta$ unknown | $\zeta$ unknown  | $\sigma < 3$ known   | all             |
|              | $\theta$ unknown | $\zeta$ unknown  | $\sigma < 3$ unknown | all             |
| Normal       | $\theta$ known   | $\sigma$ known   |                      | all             |
|              | $\theta$ known   | $\sigma$ unknown |                      | $A^2$ and $W^2$ |
|              | $\theta$ unknown | $\sigma$ known   |                      | $A^2$ and $W^2$ |
|              | $\theta$ unknown | $\sigma$ unknown |                      | all             |
| Weibull      | $\theta$ known   | $\sigma$ known   | $c$ known            | all             |
|              | $\theta$ known   | $\sigma$ unknown | $c$ known            | $A^2$ and $W^2$ |
|              | $\theta$ known   | $\sigma$ known   | $c$ unknown          | $A^2$ and $W^2$ |
|              | $\theta$ known   | $\sigma$ unknown | $c$ unknown          | $A^2$ and $W^2$ |
|              | $\theta$ unknown | $\sigma$ known   | $c > 2$ known        | all             |
|              | $\theta$ unknown | $\sigma$ unknown | $c > 2$ known        | all             |
|              | $\theta$ unknown | $\sigma$ known   | $c > 2$ unknown      | all             |
|              | $\theta$ unknown | $\sigma$ unknown | $c > 2$ unknown      | all             |

- The PROBPLOT statement supports the following new options:
  - CTEXTSIDE= specifies the color for the row labels that are displayed along the left side of the comparative probability plot.
  - CTEXTTOP= specifies the color for the row labels that are displayed along the top of the comparative probability plot.
- The QQPLOT statement supports the following new options:
  - CTEXTSIDE= specifies the color for the row labels that are displayed along the left side of the comparative quantile-quantile plot.
  - CTEXTTOP= specifies the color for the row labels that are displayed along the left side of the comparative quantile-quantile plot.





## CHAPTER

## 4

# SAS Software in the CMS Environment

---

|                                                         |    |
|---------------------------------------------------------|----|
| <i>Byte File System Support</i>                         | 17 |
| <i>Sending E-mail from SAS Using the SMTP Interface</i> | 17 |
| <i>Universal Printing</i>                               | 17 |
| <i>Writing to the SAS Log From a Pipeline</i>           | 17 |
| <i>SAS Language Elements</i>                            | 18 |
| <i>Commands</i>                                         | 18 |
| <i>Macros</i>                                           | 18 |
| <i>Procedures</i>                                       | 18 |
| <i>Statements</i>                                       | 18 |
| <i>System Options</i>                                   | 18 |

---

## Byte File System Support

IBM Open Edition for VM/ESA implements a directory-based file system, known as Byte File System (BFS), that is very similar to the file systems used in UNIX. The SAS System under CMS enables you to read and write BFS files and to issue OPENVM CMS commands.

---

## Sending E-mail from SAS Using the SMTP Interface

When you use the FILENAME, EMAIL (SMTP) access method, you can now specify a character-set encoding for processing messages and attachments.

---

## Universal Printing

Universal Printing has been enabled under CMS. Universal Printing is a printing mechanism that provides printing support for all operating environments. It is especially helpful for those operating environments in which printing can be a challenge.

---

## Writing to the SAS Log From a Pipeline

A new SAS pipeline command, SASLOG, enables you to write messages to the SAS log from pipelines that are defined by FILENAME, INFILE, or FILE statements

(including pipelines that are invoked by the CALLPIPE or ADDPIPE CMS pipeline subcommands under such pipelines).

---

## SAS Language Elements

---

### Commands

The following commands have been enhanced:

- The FILE and INCLUDE commands support a new ENCODING= option. This option enables you to dynamically change the character-set encoding when reading from or writing to external files.

---

### Macros

The %KEYDEF macro, for defining function keys, is obsolete.

---

### Procedures

The PRTDEF procedure, which enables you to define Universal Printers in batch mode, has a host-specific requirement for printing directly to a printer. You must specify PIPE as the DEVICE value and SASPRT as the DEST value.

---

### Statements

The following statements have been enhanced:

- The FILE, FILENAME, and INFILE statements support the BFS file type and BFS file and directory names in the file-specification argument.
- The FILENAME statement supports the UPRINTER device type for Universal Printing.
- The FILE, FILENAME, and INFILE statements support a new ENCODING= option. This option enables you to dynamically change the character-set encoding when reading from or writing to external files.

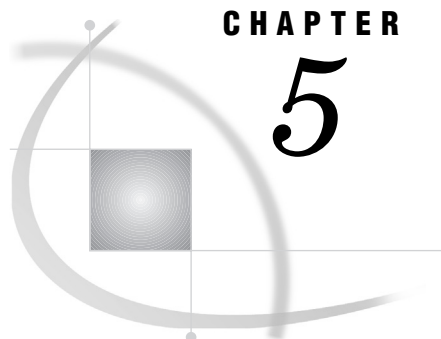
---

### System Options

- The following system options are new:
  - ENCODING= specifies the character-set encoding for reading from or writing to external files.
  - LOCALE= specifies the locale for the SAS session. The locale reflects the local conventions, language, and culture for a geographical region.
  - NLSCOMPATMODE specifies whether to use the default character-set encoding to process data.
  - REALMEMSIZE= specifies the amount of virtual memory that SAS can allocate to procedures and applications.

- VIOBUF= specifies the size of the virtual storage area to be used for members of the WORK library.
- The following system options have been modified:
  - SYSIN= now accepts a concatenated list of filenames in the *file-specification* argument. For each filename, a filetype of "SAS" and a filemode of "\*" is assumed. Previously, only complete file specifications could be concatenated.
  - The default value for UNIVERSALPRINT is now UNIVERSALPRINT.





## CHAPTER

## 5

# SAS Software in the Microsoft Windows Environment

---

|                                          |    |
|------------------------------------------|----|
| <i>Enhanced Editor</i>                   | 21 |
| <i>Basic Operation of the SAS System</i> | 21 |
| <i>SAS Language Elements</i>             | 22 |
| <i>Commands</i>                          | 22 |
| <i>Functions and CALL Routines</i>       | 22 |
| <i>Macros</i>                            | 22 |
| <i>Statements</i>                        | 22 |
| <i>System Options</i>                    | 22 |

---

## Enhanced Editor

The following Enhanced Editor features are new:

- ☐ You can open multiple views of the same file.
  - ☐ Using the SUBTOP command, you can submit a specified number of lines from the beginning of a program for processing. You can also submit your program from the Open dialog box.
  - ☐ You can search for matching DO-END pairs.
  - ☐ Scrolling and line number commands can be entered from the command bar.
  - ☐ The previous search string is the initial Find text value in the Find and Replace dialog boxes.
  - ☐ When an external file on disk has been modified, the Enhanced Editor automatically reloads unchanged files.
  - ☐ When the Enhanced Editor option Strip Ctrl + Z characters is selected, the Ctrl + Z characters are removed from files that were created in a DOS editor.
  - ☐ You can change the background color of the Enhanced Editor window by using the background color selection for normal text.
- 

## Basic Operation of the SAS System

The basic operation of the SAS System has been enhanced as follows:

- ☐ The F11 key default command is now set to COMMAND FOCUS, which places the window focus in the command bar regardless of whether the command bar is docked to the main SAS window or undocked and contained in a separate window.
- ☐ When your preferences are set to turn off the message area of the status line, messages appear in the active window.

---

## SAS Language Elements

---

### Commands

The following SAS commands have been enhanced:

- In the COMMAND command, the new FOCUS argument places the window focus in the command bar regardless of whether the command bar is docked to the main SAS window or undocked and contained in a separate window.
- The SUBTOP command is now supported in the Enhanced Editor as well as in the Program Editor.
- In the FILE and INCLUDE commands, the new ENCODING argument specifies a character-set encoding for processing external data that is in a different language.
- In the FILE and INCLUDE commands, the *file-specification* argument now accepts filerefs.
- In the FILE and INCLUDE commands, the host option IGNOREDOSEOF specifies that ^Z is interpreted as character data and not as an end-of-file marker.

---

### Functions and CALL Routines

The documentation for the following CALL routine has been updated:

- The CALL SOUND duration is measured in milliseconds.

---

### Macros

The SYSSCPL automatic macro variable returns a value for the Windows Millennium Edition operating environment.

---

### Statements

The following SAS statements have been enhanced:

- In the FILENAME, FILE, and INFILE statements, the new ENCODING option specifies a character-set encoding for processing external data that is in a different language.
- In the %INCLUDE, FILENAME, FILE, and INFILE statements, the host option IGNOREDOSEOF specifies that ^Z is interpreted as character data and not as an end-of-file marker.

---

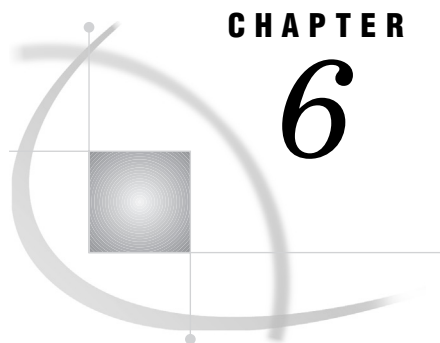
### System Options

- The following system options are new:
  - ENCODING specifies the character-set encoding for your SAS session.
  - LOCALE specifies the locale of your SAS session. The locale reflects the local conventions, language, and culture for a geographic region.
  - NLSCOMPATMODE controls the default character-set encoding for processing data.

- PRTABORTDLGS specifies whether you want the Print Abort dialog box to be displayed.
- PRTSETFORMS displays or removes the **Use Forms** check box from the Print Setup dialog box.
- REALMEMSIZE specifies how much virtual memory SAS can allocate.
- The following system options have been enhanced:
  - The default value for SGIO is now NOSGIO.
  - DBCSTYPE now accepts WINDOWS as an alias for the PCMS character-set encoding method.
  - The default value for SASUSER has been modified to include (as part of the path) a Windows variable whose value is a user's Personal folder or My Documents folder.







## CHAPTER

## 6

## SAS Software in the OpenVMS Environment

---

*SAS Language Elements* 25

*Commands* 25

*Statements* 25

*System Options* 25

---

### SAS Language Elements

---

#### Commands

The FONTLIST command, which lists available software fonts, is now specific to the OpenVMS environment.

---

#### Statements

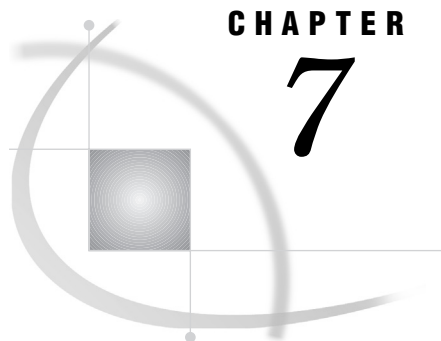
RECFM=N is now supported for the FILE, FILENAME, and INFILE statements and the FILENAME function. RECFM=N specifies that the record format of the file is binary.

---

#### System Options

- ☐ The default for the SEQENGINE system option has been changed to TAPE.
- ☐ The REALMEMSIZE system option is not supported on OpenVMS.
- ☐ The ENCODING system option, which specifies the default character-set encoding for external data, has been added.
- ☐ The LOCALE system option, which reflects the local conventions, language and culture for a geographic region, has been added.
- ☐ The NLSCOMPATMODE system option, which uses the default character-set encoding to process data, has been added.





## CHAPTER

## 7

# SAS Software in the OS/2 Environment

---

*SAS Language Elements* 27

*Commands* 27

*Functions and CALL Routines* 27

*Statements* 27

*System Options* 27

---

## SAS Language Elements

---

### Commands

The following commands have been enhanced:

- In the COMMAND command, the new FOCUS argument places the window focus in the command bar regardless of whether the command bar is docked to the main SAS window or undocked and contained in a separate window.
- In the FILE and INCLUDE commands, the new ENCODING argument specifies a character-set encoding for processing external data that is in a different language.
- In the FILE and INCLUDE commands, the *file-specification* argument now accepts filerefs.

---

### Functions and CALL Routines

The documentation for the following CALL routine has been updated:

- The CALL SOUND duration is measured in milliseconds.

---

### Statements

The following SAS statements have been enhanced:

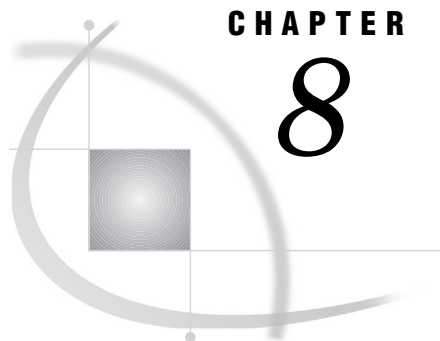
- In the FILENAME, FILE, and INFILE statements, the new ENCODING option specifies a character-set encoding for processing external data that is in a different language.

---

### System Options

- The following system options are new:

- ☐ `ENCODING` specifies the character-set encoding for your SAS session.
  - ☐ `LOCALE` specifies the locale of your SAS session. The locale reflects the local conventions, language, and culture for a geographic region.
  - ☐ `NLSCOMPATMODE` controls the default character-set encoding for processing data.
  - ☐ `PRTABORTDLGS` specifies whether you want the Print Abort dialog box to be displayed.
  - ☐ `PRTSETFORMS` displays or removes the **Use Forms** check box from the Print Setup dialog box.
  - ☐ `REALMEMSIZE` specifies how much virtual memory SAS can allocate.
- 
- ☐ The following system option has been enhanced:
    - ☐ `DBCSTYPE` now accepts `WINDOWS` as an alias for the `PCMS` character-set encoding method.



## CHAPTER

## 8

# SAS Software in the OS/390 Environment

---

|                                                         |    |
|---------------------------------------------------------|----|
| <i>Universal Printing</i>                               | 29 |
| <i>Sending E-mail from SAS Using the SMTP Interface</i> | 29 |
| <i>SAS Language Elements</i>                            | 29 |
| <i>Commands</i>                                         | 29 |
| <i>Macros</i>                                           | 29 |
| <i>Statements</i>                                       | 30 |
| <i>System Options</i>                                   | 30 |

---

## Universal Printing

Universal Printing has been enabled under OS/390. Universal Printing is a printing mechanism that provides printing support to all operating environments. It is especially helpful for those operating environments in which printing can be a challenge.

---

## Sending E-mail from SAS Using the SMTP Interface

When you use the FILENAME, EMAIL (SMTP) access method, you can now specify a character-set encoding for processing messages and attachments.

---

## SAS Language Elements

---

### Commands

The following commands have been enhanced:

- INCLUDE supports the \* wildcard character for specifying a concatenation of UNIX System Services files.
  - The FILE and INCLUDE commands support a new ENCODING= option. This option enables you to dynamically change the character-set encoding when reading from or writing to external files.
- 

### Macros

The %KEYDEF macro, for defining function keys, is obsolete.

---

## Statements

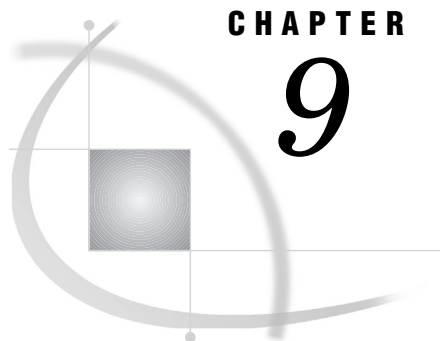
The following statements have been enhanced:

- The FILENAME statement supports REP as a valid value for the DISP= option. Specifying DISP=REP enables you to create or replace an external file without knowing whether the file already exists.
- The FILENAME statement supports the UPRINTER device type for Universal Printing.
- The FILENAME, INFILE, and %INCLUDE statements support the \* wildcard character for specifying a concatenation of UNIX System Services files.
- The FILE, FILENAME, and INFILE statements support a new ENCODING= option. This option enables you to dynamically change the character-set encoding when reading from or writing to external files.

---

## System Options

- The following system options are new:
  - ENCODING= specifies the character-set encoding for reading from or writing to external files.
  - LOCALE= specifies the locale for the SAS session. The locale reflects the local conventions, language, and culture of a geographical region.
  - NLSCOMPATMODE controls the encoding that is used for processing character data.
  - REALMEMSIZE= specifies the amount of virtual memory that SAS can allocate to procedures and applications.
  - SET= defines environment variables.
- The following system option has been modified:
  - The default value for UNIVERSALPRINT is now UNIVERSALPRINT.



## CHAPTER

## 9

# SAS Software in UNIX Environments

---

|                              |           |
|------------------------------|-----------|
| <i>SAS Language Elements</i> | <b>31</b> |
| <i>Commands</i>              | <b>31</b> |
| <i>Statements</i>            | <b>31</b> |
| <i>System Options</i>        | <b>31</b> |
| <i>SAS Resources</i>         | <b>32</b> |

---

## SAS Language Elements

---

### Commands

- The new SETDMSFONT command specifies a windowing environment font for the current SAS session.
- The FILE and INCLUDE commands support the new ENCODING option.

---

### Statements

The FILE, FILENAME, and INFILE statements support the new ENCODING option. This option enables you to dynamically change the character-set encoding when reading from or writing to external files.

---

### System Options

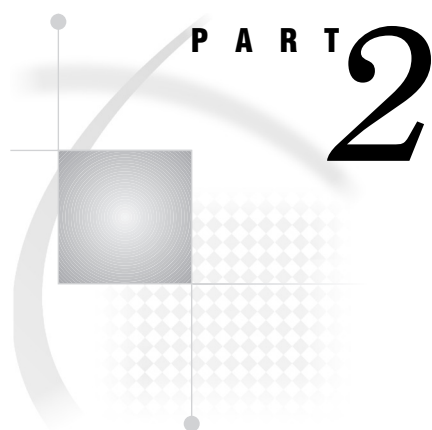
- The following SAS system options are new:
  - The ENCODING system option specifies a default character-set encoding for processing external data.
  - The LOCALE system option specifies the locale for the SAS session. The locale reflects the local conventions, language, and culture for a geographical region.
  - The NLSCOMPATMODE system option specifies whether to use the default character-set encoding to process data.
- The following system options have been changed or enhanced:
  - The default for the SORTPGM system option has been changed to BEST.
  - The EMAILSYS system option now supports SMTP (Simple Mail Transport Protocol).

---

## SAS Resources

The new resource **SAS.suppressMenuIcons** suppresses the use of icons in menus.





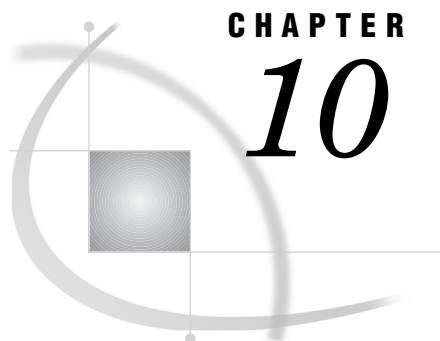
## Other SAS Software Products

|                                |                                              |           |
|--------------------------------|----------------------------------------------|-----------|
| <i>Chapter</i> <b>10</b> ..... | <b>SAS/ACCESS Software</b>                   | <b>35</b> |
| <i>Chapter</i> <b>11</b> ..... | <b>SAS ADX Interface</b>                     | <b>39</b> |
| <i>Chapter</i> <b>12</b> ..... | <b>SAS/ASSIST Software</b>                   | <b>41</b> |
| <i>Chapter</i> <b>13</b> ..... | <b>SAS Component Language</b>                | <b>43</b> |
| <i>Chapter</i> <b>14</b> ..... | <b>SAS/CONNECT Software</b>                  | <b>45</b> |
| <i>Chapter</i> <b>15</b> ..... | <b>SAS/EIS Software</b>                      | <b>47</b> |
| <i>Chapter</i> <b>16</b> ..... | <b>SAS/ETS Software</b>                      | <b>49</b> |
| <i>Chapter</i> <b>17</b> ..... | <b>SAS/GIS Software</b>                      | <b>53</b> |
| <i>Chapter</i> <b>18</b> ..... | <b>SAS/GRAPH Software</b>                    | <b>55</b> |
| <i>Chapter</i> <b>19</b> ..... | <b>SAS/IML Software</b>                      | <b>57</b> |
| <i>Chapter</i> <b>20</b> ..... | <b>SAS Integration Technologies Software</b> | <b>59</b> |
| <i>Chapter</i> <b>21</b> ..... | <b>SAS/IntrNet Software</b>                  | <b>63</b> |
| <i>Chapter</i> <b>22</b> ..... | <b>SAS/MDDb Server Software</b>              | <b>65</b> |
| <i>Chapter</i> <b>23</b> ..... | <b>SAS OLAP Server Software</b>              | <b>67</b> |
| <i>Chapter</i> <b>24</b> ..... | <b>SAS/OR Software</b>                       | <b>69</b> |
| <i>Chapter</i> <b>25</b> ..... | <b>SAS/QC Software</b>                       | <b>71</b> |

*Chapter 26*. . . . . **SAS/SHARE Software** 75

*Chapter 27*. . . . . **SAS/STAT Software** 77

*Chapter 28*. . . . . **SAS/Warehouse Administrator Software** 83



## SAS/ACCESS Software

---

|                                                                       |           |
|-----------------------------------------------------------------------|-----------|
| <i>SAS/ACCESS Software for Relational Databases</i>                   | <b>35</b> |
| <i>In All SAS/ACCESS Interfaces to Relational Databases</i>           | <b>35</b> |
| <i>SAS/ACCESS Interface to DB2 under OS/390</i>                       | <b>36</b> |
| <i>SAS/ACCESS Interface to DB2 under UNIX and PC Hosts</i>            | <b>36</b> |
| <i>SAS/ACCESS Interface to Microsoft SQL Server</i>                   | <b>36</b> |
| <i>SAS/ACCESS Interface to ODBC</i>                                   | <b>36</b> |
| <i>SAS/ACCESS Interface to OLE DB</i>                                 | <b>36</b> |
| <i>SAS/ACCESS Interface to ORACLE</i>                                 | <b>36</b> |
| <i>SAS/ACCESS Interface to SYBASE</i>                                 | <b>36</b> |
| <i>SAS/ACCESS Interface to Teradata</i>                               | <b>37</b> |
| <i>SAS/ACCESS Interface to CA-DATACOM/DB (Available under OS/390)</i> | <b>37</b> |
| <i>SAS/ACCESS Interface to IMS-DL/I (Available under OS/390)</i>      | <b>37</b> |
| <i>SAS/ACCESS Software for PC File Formats</i>                        | <b>37</b> |
| <i>SAS/ACCESS Interface to PeopleSoft</i>                             | <b>37</b> |
| <i>SAS/ACCESS Interface to R/3</i>                                    | <b>37</b> |

---

## SAS/ACCESS Software for Relational Databases

---

### In All SAS/ACCESS Interfaces to Relational Databases

The following changes and enhancements have been made to all of the SAS/ACCESS interfaces to relational databases:

- A new system option, DBSRVTP=, enables you to improve SAS server throughput by specifying whether certain SAS/ACCESS engines put a hold (or block) on the originating client while making performance-critical calls to the database. This option applies when SAS is invoked as a server that is responding to multiple clients.
- The LIBNAME option PRESERVE\_COLUMN\_NAMES= now applies only to table creation. This option is now also available as a data set option.
- SAS/ACCESS can now convert many SAS functions into the SQL of the DBMS, which improves performance by causing the functions to be processed by the DBMS. This enhancement is available for WHERE clauses and PROC SQL when you use the SAS/ACCESS LIBNAME statement.

---

## SAS/ACCESS Interface to DB2 under OS/390

Bulk loading, which uses the DB2 LOAD utility to enable you to rapidly load rows into a DB2 table, is now supported.

---

## SAS/ACCESS Interface to DB2 under UNIX and PC Hosts

- The BIGINT numeric data type, which specifies a big integer, is now supported. Values in a column of this type can range from -9223372036854775808 to +9223372036854775807.
- A new LIBNAME and data set option, INSERTBUFF=, specifies the number of rows in a single DBMS insert.

---

## SAS/ACCESS Interface to Microsoft SQL Server

- A new LIBNAME and data set option, UPDATE SQL=, determines which method is used to update and delete rows in a data source.
- A new LIBNAME and data set option, INSERTBUFF=, specifies how many rows are included in a single DBMS insert.
- The BIGINT numeric data type, which specifies a big integer, is now supported. Values in a column of this type can range from -9223372036854775808 to +9223372036854775807.
- SAS/ACCESS can now pass outer joins between two or more tables, with certain restrictions.

---

## SAS/ACCESS Interface to ODBC

- A new LIBNAME and data set option, UPDATE SQL=, determines which method is used to update and delete rows in a data source.
- A new LIBNAME and data set option, INSERTBUFF=, specifies how many rows are included in a single DBMS insert.
- SAS/ACCESS can now pass outer joins between two or more tables, with certain restrictions.
- The BIGINT numeric data type, which specifies a big integer, is now supported. Values in a column of this type can range from -9223372036854775808 to +9223372036854775807.

---

## SAS/ACCESS Interface to OLE DB

The Microsoft Data Link API is supported through a new LIBNAME engine connection option, UDL\_FILE=.

---

## SAS/ACCESS Interface to ORACLE

SAS/ACCESS can now pass outer joins between two or more tables.

---

## SAS/ACCESS Interface to SYBASE

SAS/ACCESS can now pass outer joins between two or more tables, with certain restrictions.

---

## SAS/ACCESS Interface to Teradata

The TIME and TIMESTAMP date/time data types are now supported.

---

## SAS/ACCESS Interface to CA-DATACOM/DB (Available under OS/390)

- The interface view engine has been enhanced to support the CA-DATACOM/DB SQL-DATE, SQL-TIME, and SQL-STMP date types.
- A new data set option, DDBERLMT=, specifies how many errors are allowed before SAS stops processing and issues a rollback.
- The DDBCOMIT= data set option has been modified to interact with DDBERLMT=.

---

## SAS/ACCESS Interface to IMS-DL/I (Available under OS/390)

The DBCONTENT= argument of the PROC ACCESS ITEM= statement supports two new date values: TFGY2KD4. and TFGY2KN4.

---

## SAS/ACCESS Software for PC File Formats

SAS/ACCESS Software for PC File Formats enables you to read and load Excel 4 and 5 files. Under Windows in Release 8.1 and later, you use the Import/Export Facility to read or load Excel 4, 5, 7 (95), 98, and 2000 files, as well as Microsoft Access files. These same files can also be used in the IMPORT and EXPORT procedures. In Release 8.2, you can use the Import/Export Facility and procedures under Microsoft Windows 2000 as well as other Windows releases.

---

## SAS/ACCESS Interface to PeopleSoft

The new SAS/ACCESS interface to PeopleSoft enables you to extract, browse, and search PeopleSoft metadata. You can query the extracted PeopleSoft metadata to generate views of the PeopleSoft data and save these views as SAS data files and SAS views. You can also save the code that was used to build the query as SAS source code.

For specific information about these features, invoke the SAS/ACCESS interface to PeopleSoft and use the Help buttons on the windows to display the appropriate Help topic.

---

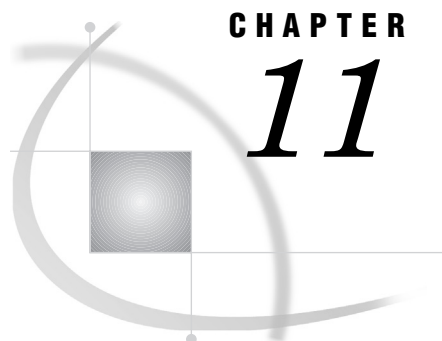
## SAS/ACCESS Interface to R/3

- You can now specify an ODD name when exporting to a new target ODD, or you can select an existing ODD name. Previously you had to use the R/3 table name when exporting to an existing ODD in SAS/Warehouse Administrator.
- In the R/3 Application Setup window, you can now specify three default librefs: Descriptor Library, SAS Data Library, and Source Library. These librefs

correspond to the descriptor information, the location of the SAS data set or view, and the location of the source code.

- You can use the new R/3 BAPI Connector: Logon window to define the R/3 BAPI connection parameters that are required for accessing the R/3 System. When you submit the %CALLRFC function call, the R/3 BAPI Connector: Logon window is automatically displayed unless you use the RFC\_LOGON\_INFO macro string instead.

For specific information about these new features, invoke the SAS/ACCESS interface to R/3 and use the Help buttons on the windows to display the appropriate Help topic.



## SAS ADX Interface

---

|                             |           |
|-----------------------------|-----------|
| <i>Process Variables</i>    | <b>39</b> |
| <i>Import Design Wizard</i> | <b>39</b> |
| <i>Design Details</i>       | <b>39</b> |

---

### Process Variables

The ADX interface now enables you to specify process variables for mixture designs. A process variable is a factor in an experiment that is not part of the mixture but whose levels, when varied, could affect the blending properties of the mixture ingredients.

---

### Import Design Wizard

An import design wizard has been added to enable external design data to be imported into ADX. The external data can be stored in a SAS data set or other file type, including Microsoft Excel. Two-level designs, response surface designs, mixed-level designs, mixture designs, and split-plot designs can be imported.

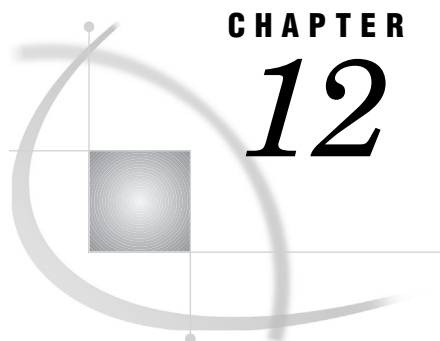
---

### Design Details

The Design Details window now enables you to view two additional details of the basic design. Variance dispersion plots, for Resolution 5 two-level and response surface designs, provide a variance graph to evaluate a design. For mixture designs, the mixture design points can be viewed graphically.







## SAS/ASSIST Software

---

|                                                         |    |
|---------------------------------------------------------|----|
| <i>Improved Performance Through the Use of Metadata</i> | 41 |
| <i>New Data Sources Window</i>                          | 41 |
| <i>New Facility for Defining Your Own Metadata</i>      | 41 |
| <i>New Sample Programs</i>                              | 42 |
| <i>New Table/Column Search</i>                          | 42 |
| <i>For Additional Information</i>                       | 42 |

---

### Improved Performance Through the Use of Metadata

SAS/ASSIST no longer accesses your data sources every time you build a task. Instead, when SAS/ASSIST starts, it examines your data sources and stores metadata about the available tables and columns for the duration of your SAS/ASSIST session. You can also request current information about a data source at any time.

When you are using SAS/ASSIST to access a data warehouse or a DBMS on a remote computer, these enhancements improve performance considerably.

---

### New Data Sources Window

A new Data Sources window displays the data libraries that you have defined. You can use this window to select the libraries that you want to use in your SAS/ASSIST session.

---

### New Facility for Defining Your Own Metadata

You can now define and prepare metadata to be used by all or a selected group of SAS/ASSIST users. This significantly speeds up the process of preparing data information at the beginning of a session, enabling users to get straight to the task at hand.

When a user starts SAS/ASSIST, the program examines all available data. If users have access to very large volumes of data on remote hosts, this can take considerable time, and would in addition have to be performed for each user. With the new facility for predefining metadata, this problem no longer exists. An administrator creates metadata in one central task, using a metadata-generation program—for example, as a batch job in an off-peak period. This metadata is then distributed so that it is immediately available to users when they start their sessions.

The metadata-generation program can also be customized to present information about only a subset of all available data. You can build this logic into an SQL program

that retrieves information about data in the database. (Sample metadata-generation programs are provided with SAS/ASSIST.)

For example, an administrator can specify that only information about tables that have a certain attribute or name prefix is made available to certain groups. Thus, a single metadata-generation process can produce differentiated metadata output. This means that an employee in the finance department views predefined metadata relating only to tables that are relevant for that department, whereas an employee in the personnel department sees a different set of metadata.

---

## New Sample Programs

Sample programs are supplied. These programs extract information from a DBMS and create the metadata that SAS/ASSIST uses. See the previous topic, "New Facility for Defining Your Own Metadata," for more details.

---

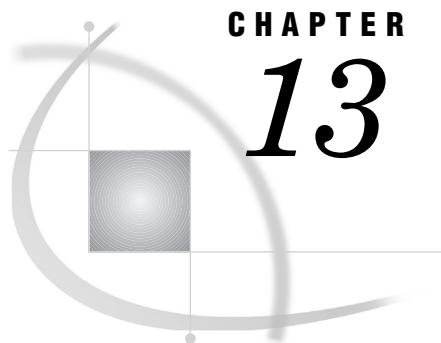
## New Table/Column Search

A new Table/Column search gives you extended search capabilities in case you don't know the name of the table that you are looking for, but you know the name of a column that the table contains. When selecting a table, you can search through all existing column names to find the table that you need.

---

## For Additional Information

For detailed information about the features described here, see *SAS/ASSIST Software: Changes and Enhancements, Release 8.2*.



## SAS Component Language

---

*SCL List Initialization Syntax Supports Brackets* 43

---

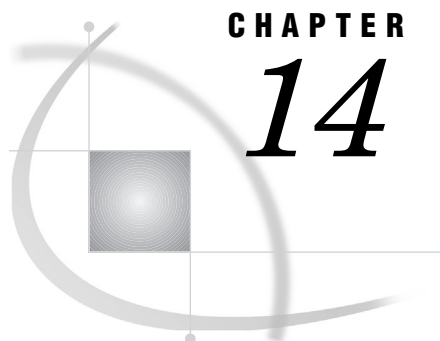
### SCL List Initialization Syntax Supports Brackets

When you use the DECLARE statement to initialize an SCL list, you can now use either braces ( { and } ) or brackets ( [ and ] ) to enclose a series of list items. For example, both of the following list definitions are now valid:

```
dcl list x = {1,2,3};  
dcl list z = [4,5,6];
```

Previously only braces were supported in DECLARE statement syntax.





## SAS/CONNECT Software

---

|                                                                       |    |
|-----------------------------------------------------------------------|----|
| <i>Enhancements to RSUBMIT</i>                                        | 45 |
| <i>Monitoring SAS/CONNECT Tasks</i>                                   | 45 |
| <i>RSUBMIT and Macro Processing</i>                                   | 45 |
| <i>Complete Set of Agent Services</i>                                 | 46 |
| <i>Common Messaging Interface</i>                                     | 46 |
| <i>SAS/CONNECT Spawners</i>                                           | 46 |
| <i>File Compression Improves Performance for Large Data Transfers</i> | 46 |
| <i>CEDA Technology Is Now Part of Base SAS</i>                        | 46 |

---

### Enhancements to RSUBMIT

- For an asynchronous RSUBMIT, you can now use the new LOG= and OUTPUT= options to spool the SAS log or SAS output to the backing store or to a specified file for subsequent retrieval. Also, these options can be used to purge the SAS log or SAS output.
  - With Multi-Process (MP) CONNECT, you can use the new KILLTASK statement to force one or more asynchronous tasks or SAS sessions to terminate immediately.
- 

### Monitoring SAS/CONNECT Tasks

The SAS Explorer now provides a menu selection that enables you to monitor SAS/CONNECT tasks that are executing either synchronously or asynchronously in one or more remote SAS sessions.

---

### RSUBMIT and Macro Processing

Macro processing within a macro-generated RSUBMIT is not always executed in the remote session, as you might expect. To ensure that your RSUBMIT statements are executed in the correct session, you can take advantage of a few simple programming techniques, which are now documented in the “RSUBMIT and MACRO Processing” topic of the SAS/CONNECT Help.

---

## Complete Set of Agent Services

- You can use the CALL Routine Interface to Agent Services to write applications that create, manage, and schedule agents to run on a remote host. These CALL routines are used with SAS data set or SAS macro programming.
- You can also use the new CALL Routine Interface to Agent Services to optionally direct the results of the agent's execution to a specified message queue. In addition to the default SAS messaging service, SAS/CONNECT now supports the external messaging-service providers MQSeries from IBM, MSMQ from Microsoft, and Geneva Message Queuing from Level 8 Systems, all of which require a separate license for SAS Integration Technologies.
- Before you can execute an agent application, you must first prepare the application's execution environment. New documentation about "Setting Up and Running Agent Applications" is in the SAS/CONNECT Help.

---

## Common Messaging Interface

The new Common Messaging Interface is a set of CALL routines that can be used in an application to access a message queue and to store and retrieve messaging information from the local SAS registry or from a distributed LDAP repository. A primary benefit of using this interface is that you can write applications in a consistent manner that is independent of the application messaging transport that is used. The Common Messaging Interface uses the SAS Messaging Service transport by default.

---

## SAS/CONNECT Spawners

See the "SAS/CONNECT Spawners" topic in the SAS/CONNECT Help for updated, comprehensive information about SAS/CONNECT spawners, including

- the benefits of using spawners
- how to start the spawner on the remote host
- how to connect to the spawner from the local host.

---

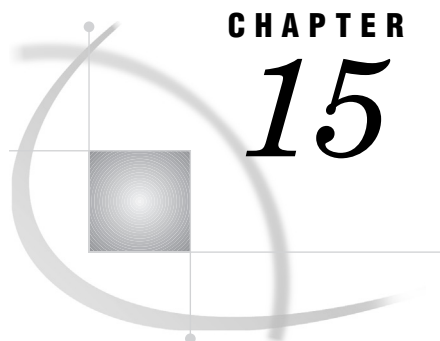
## File Compression Improves Performance for Large Data Transfers

A new file-compression algorithm facilitates faster exchange of large amounts of data. This performance improvement is especially beneficial when you are using PROC UPLOAD and PROC DOWNLOAD to transfer large files.

---

## CEDA Technology Is Now Part of Base SAS

CEDA (Cross-Environment Data Access) technology facilitates data access between users on incompatible host architectures. CEDA functionality, which formerly required a SAS/CONNECT license, is now available in base SAS software.



## SAS/EIS Software

---

|                                               |    |
|-----------------------------------------------|----|
| <i>Distinct Values (NUNIQUE) Window</i>       | 47 |
| <i>Computed Values Window</i>                 | 47 |
| <i>Sorting by Unformatted Category Values</i> | 47 |
| <i>Specifying Labels for Statistics</i>       | 48 |
| <i>Activating Access Control</i>              | 48 |

---

### Distinct Values (NUNIQUE) Window

- The Distinct Values (NUNIQUE) window has been enhanced to enable you to
  - define multiple NUNIQUE columns without closing the window
  - delete multiple NUNIQUE columns
  - define a label and a format for the NUNIQUE column
  - register an NUNIQUE column that was created while you were running a report.
- In addition, the Distinct Values (NUNIQUE) window is now available at run time for the Multidimensional Report object.

---

### Computed Values Window

- The Computed Values window has been enhanced to enable you to
- define a format for the computed value, when the computed value is defined for the analysis dimension of a Multidimensional Report object.
  - register a computed value on the analysis dimension as a computed column in a repository for a Multidimensional Report object.

---

### Sorting by Unformatted Category Values

Sorting by unformatted category values is now available in all multidimensional objects that support category sorts. In Release 8.2, unformatted category values are stored on MDDBs that are created in this release. If the unformatted values are stored, you can specify an unformatted sort.

*Note:* Unformatted category values are not sorted for MDDBs that were created with prior releases. If you specify an unformatted sort for these MDDBs, a message in the Log window tells you that the sort has not been performed. △

---

## Specifying Labels for Statistics

You can now specify customized statistic labels for all statistics that an MDDB supports. The Statistic Labels window is available from the SAS/EIS Setup window or by selecting

Solutions ► Development and Programming ► OLAP Server Administration

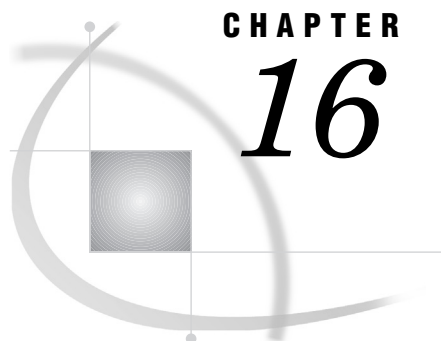
The customized statistic labels are stored in the current parameter catalog entry and are available to any report that is run with the current parameter catalog entry. SASUSER.PARMS.PARMS.SLIST is the default parameter catalog entry. To specify a different parameter catalog entry, use the PARMS= option on the EIS or RUNEIS command.

---

## Activating Access Control

If you used access control for your SAS/EIS applications in a previous release, be sure to activate it for this release, as well. Follow the steps described in the *SAS OLAP Server Administrator's Guide* to activate access control and to point it to the location of your existing access control files.





## SAS/ETS Software

---

|                                       |    |
|---------------------------------------|----|
| <i>ARIMA Procedure</i>                | 49 |
| <i>COUNTREG Procedure</i>             | 49 |
| <i>DATETIME Formats</i>               | 49 |
| <i>Investment Analysis System</i>     | 50 |
| <i>MDC Procedure</i>                  | 50 |
| <i>MODEL Procedure</i>                | 50 |
| <i>QLIM Procedure</i>                 | 50 |
| <i>Time Series Forecasting System</i> | 50 |
| <i>VARMAX Procedure</i>               | 51 |
| <i>X12 Procedure</i>                  | 51 |
| <i>For Additional Information</i>     | 51 |

---

### ARIMA Procedure

The new OUTLIER statement can be used as a diagnostic tool to detect shifts in the mean level of the response series that are not accounted for by the estimated model. The types of changes in the mean level that can be detected are: Additive Outliers, Permanent level shifts, and Temporary level shifts of given durations.

---

### COUNTREG Procedure

The COUNTREG procedure is a new experimental procedure that analyzes count data. Poisson and negative binomial regression as well as zero-inflated Poisson and negative binomial models are supported.

---

### DATETIME Formats

Four new DATETIME formats have been added:

- The DTMONYYw. format writes the date part of a datetime value as the month and year in the form *mmm*yy or *mmm*yyyy.
- The DTWKDATXw. format writes the date part of a datetime value as the day of the week and the date in the form *day-of-week*, *dd mmm* yy (or yyyy).
- The DTYEARw. format writes the date part of a datetime value as the year in the form yy or yyyy.

- The DTTYQCw. format writes the date part of a datetime value as the year and the quarter and separates them with a colon (:).

---

## Investment Analysis System

Investment Analysis now enables you to print computed tables and analysis results to the SAS session print device.

---

## MDC Procedure

The MDC procedure analyzes multinomial discrete choice models. PROC MDC handles most of the commonly used multinomial choice models, such as conditional logit, heteroscedastic extreme value, mixed logit, nested logit, and multinomial probit models. The nested logit model can handle the general level of decision trees.

---

## MODEL Procedure

The MODEL procedure now supports multivariate  $t$  and general likelihood estimation.

---

## QLIM Procedure

The QLIM procedure is an experimental procedure that can be used for analyzing qualitative choice variables and limited dependent variables. PROC QLIM supports a wide variety of models, including Box-Cox regression, logit/probit/tobit with heteroscedasticity, panel data models, and sample selection models. The new version has added a RESTRICT statement, various goodness-of-fit measures, Box-Cox transformation of regressors for logit/probit/tobit models, bivariate probit models, and sample selection models. Most multinomial choice models (conditional logit, mixed logit, heteroscedastic extreme value, value, and nested logit) are supported in the MDC procedure.

---

## Time Series Forecasting System

- You can now add models that contain regressors (including dynamic regressors) to the model selection list. The list can be saved with the project and/or saved as the default list, so that the regressor models are tried when automatic model fitting is performed.
- The FORECAST command now supports the AUDIT=<YES|NO> argument for controlling the audit trail.
- The Forecast Command Builder utility has been improved to make selection of large numbers of series easier. It also provides new types of forecast graphs that are not available elsewhere, including a batch graphing option to graph all the forecasts at once and display them in a framed Web page with thumb-nails. Documentation for this utility is available on the Web. In the Time Series Forecasting System, select

---

## VARMAX Procedure

The VARMAX procedure now includes maximum likelihood estimation for VARMA models and multivariate GARCH models:

- Vector time series can now be estimated by a finite order VAR process with a finite order MA error term. The VARMA processes have infinite order pure VAR and MA representations. Impulse response analysis and forecasting VARMA processes are included.
- Multivariate GARCH models now enable you to specify a functional form for the conditional covariance and also enable you to model the first and second moments jointly.

---

## X12 Procedure

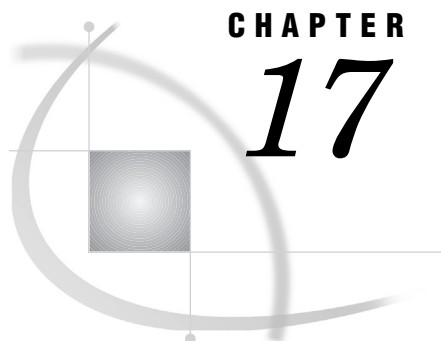
- BY and ID statements have been added.
- The user can control forecasting with the FORECAST statement that has the LEAD option.
- In the TRANSFORM statement, some standard Box-Cox transformations can be requested using the FUNCTION= options, which include the values NONE, LOG, SQRT, INVERSE, LOGISTIC, and AUTO.
- The X11 statement has been greatly expanded with the addition of the OUTFORECAST, SEASONALMA=, and TRENDMA= options.
- The new predefined regression variables available through the REGRESSION PREDEFINED= option are CONSTANT, LOMSTOCK, SEASONAL, TD, TDNOLPYEAR, TD1COEF, and TD1NOPYEAR.
- The following new tables are available:
  - prior-adjustment factors (A2)
  - chi-squared tests for groups of regressors (RegressorGroupChisq)
  - average absolute percentage error in win-sample forecasts (AvgFcstErr)
  - seasonal MA roots (Roots)
  - day of the week trading day component factors (F4).

---

## For Additional Information

For more information about the features described here, see *SAS/ETS Software: Changes and Enhancements, Release 8.2*.





## SAS/GIS Software

---

*PROC GIS MAP ACTION Statement* 53

*Updated Tutorial* 54

---

### PROC GIS MAP ACTION Statement

The PROC GIS MAP ACTION statement has been changed to enable you to create, replace and update actions. In addition, arguments have been added to enable you to create specific action types:

**CREATE**

makes a new action

**REPLACE**

replaces a specified action

**TYPE**

specifies the type of action to be created or updated

**LINK**

specifies a link to an attribute table

**OUT**

specifies an output data table name

**OUTMODE**

specifies how the output data table will be written to

**SCREEN**

specifies a screen catalog entry to be used by FSBROWSE as a result of a BROWSE type action

**FORMULA**

specifies a formula catalog entry to be used by FSVIEW as a result of a VIEW type action

**IMAGEVAR**

specifies the name of the variable in the link data table that contains the image to display for the currently selected feature

**SOURCE**

specifies the location of the source code for a program action

**MAPVAR**

specifies the name of the variable in the link data table containing the name of the map to open when a feature is selected

**COMMAND**

specifies the command or a variable in a linked data table to be executed when either a command action or a system command action is run

**REDISTRICTVAR**

specifies the value in the linked data table that contains the name of the value upon which redistricting is based

**REDISTRICTLAYER**

specifies the name of the polygonal layer to be rethemed by the redistricting action

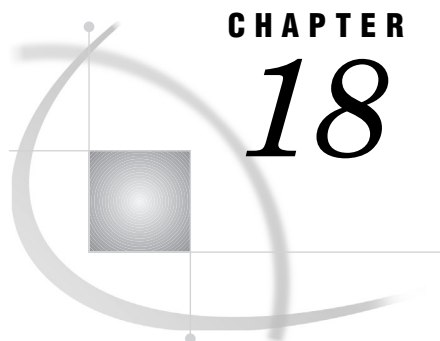
**SUMMARYVAR**

specifies one or more numeric variables to display in the redistricting window when a redistrict action is executed

---

## Updated Tutorial

The SAS/GIS tutorial has been updated and is now run from a Web browser. The data that is created for the tutorial has been changed to reflect the new content. In addition, new menu options enable you to resume work in the tutorial and to delete the tutorial data.



## CHAPTER

## 18

## SAS/GRAPH Software

|                                      |    |
|--------------------------------------|----|
| <i>GCHART Procedure</i>              | 55 |
| <i>GCONTOUR Procedure</i>            | 55 |
| <i>GPLOT Procedure</i>               | 55 |
| <i>GMAP Procedure</i>                | 56 |
| <i>GRADAR Procedure</i>              | 56 |
| <i>GREPLAY Procedure</i>             | 56 |
| <i>Generating Graphs for the Web</i> | 56 |

## GCHART Procedure

The HBAR, HBAR3D, VBAR, and VBAR3D statements support the following new options:

- The RANGE option displays on the axis of the chart the range of numeric data that each bar represents.
- The CREF= and LREF= options enable you to specify the color and style of reference lines that are drawn at the points specified by the REF= option.
- The CAUTOREF= and LAUTOREF= options enable you to specify the color and style of reference lines that are drawn at major tick marks. The tick marks are specified by the AUTOREF= option.

## GCONTOUR Procedure

In the PLOT statement, the new CHREF=, CVREF=, LHREF=, and LVREF= options enable you to specify the line style and line color for individual reference lines.

## GPLOT Procedure

The BUBBLE, BUBBLE2, PLOT, and PLOT2 statements support the following new options:

- The CHREF=, CVREF=, LHREF=, and LVREF= options enable you to specify the color and style of reference lines that are drawn at the points specified by the REF= option.
- The CHAUTOREF=, CVAOUTREF=, LHAUTOREF=, and LVAUTOREF= options enable you to specify the color and style of reference lines that are drawn at major tick marks. The tick marks are specified by the AUTOHREF= and AUTOVREF= options.

---

## GMAP Procedure

In the BLOCK and CHORO statements, the new WOUTLINE= option specifies the width of the outline for all outlined blocks and for the outline of the map areas.

---

## GRADAR Procedure

- The GRADAR procedure now supports the global AXIS statement, which controls the location, values, and appearance of the axes in plots and charts.
- The CHART statement supports the following new options:
  - The MODE= option specifies the display mode for a radar chart.
  - The STARAXIS= option assigns one or more axis definitions to the axes in the radar chart.
  - The HTML= and HTML\_LEGEND= options create links in the HTML files that the Output Delivery System produces.
- By default, tick marks on the spokes are now placed at the minimum value, the maximum value, and at one value in between. Also, the tick marks on the 12 o'clock spoke are now labeled.

---

## GREPLAY Procedure

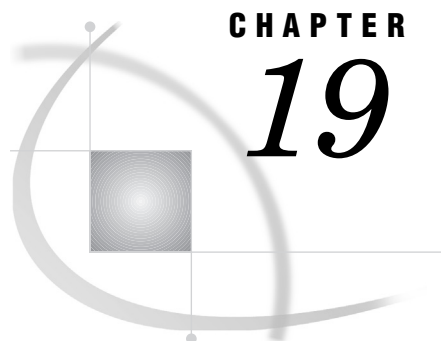
In the TREPLAY statement, the new NAME= and DES= options enable you to differentiate between multiple templates.

---

## Generating Graphs for the Web

Information about generating Web graphs with the JAVA, JAVAMETA, ACTIVEX, HTML, WEBFRAME, GIF, and GIFANIM device drivers has been moved into the SAS/GRAPH Help from its previous location on the SAS Web site.





## SAS/IML Software

---

*Wavelet Analysis* 57

*For Additional Information* 57

---

### Wavelet Analysis

New subroutines for computing, inverting, and displaying wavelet transforms have been added:

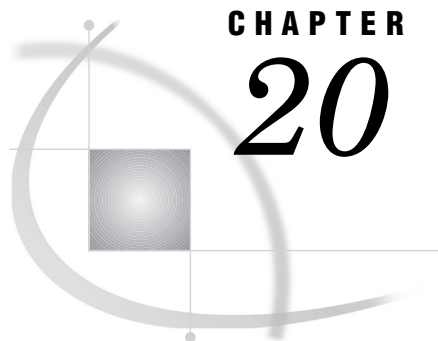
- The WAVFT subroutine computes a specified wavelet transform of one-dimensional data.
- The WAVGET subroutine returns requested information encapsulated in a wavelet transform.
- The WAVIFT subroutine inverts a wavelet transform after applying specified thresholding to the detail coefficients.
- The WAVPRINT subroutine displays requested information encapsulated in a wavelet transform.
- The WAVTHRSH subroutine applies specified thresholding to the detail coefficients of a wavelet transform.

---

### For Additional Information

For more information about the new SAS/IML features, see *SAS/IML Software: Changes and Enhancements, Release 8.2*.





## CHAPTER

## 20

# SAS Integration Technologies Software

|                                                                              |    |
|------------------------------------------------------------------------------|----|
| <i>SAS Business Intelligence Portal</i>                                      | 59 |
| <i>SAS Publishing Framework</i>                                              | 59 |
| <i>Directory Services</i>                                                    | 60 |
| <i>Distributed Objects</i>                                                   | 60 |
| <i>Message Queuing</i>                                                       | 60 |
| <i>Enhancements to SAS Package Retriever</i>                                 | 60 |
| <i>New WebDAV Delivery Type</i>                                              | 60 |
| <i>Additional Properties for the HTML Package Entry Type</i>                 | 61 |
| <i>Enhancements to SAS Publisher</i>                                         | 61 |
| <i>New WebDAV Delivery Type</i>                                              | 61 |
| <i>FTP and HTTP User ID and Password Properties Support Package Delivery</i> | 61 |
| <i>Additional Properties for the E-mail Delivery Type</i>                    | 62 |
| <i>Additional Properties for the ODS Package Entry Type</i>                  | 62 |
| <i>URL Package Entry Type Supports Relative URL Specification</i>            | 62 |
| <i>Data Set Entry Type Supports Data Set Options</i>                         | 62 |
| <i>For Additional Information</i>                                            | 62 |

## SAS Business Intelligence Portal

The SAS Business Intelligence Portal enables you to provide a single, customizable Web interface that aggregates information and applications for users in your enterprise. It uses a role-based security mechanism to deliver only the content that is pertinent to a particular user based on his or her role in the organization. When used in combination with the features of the SAS Publishing Framework, the SAS Business Intelligence Portal becomes the centerpiece of your corporate knowledge-management infrastructure.

## SAS Publishing Framework

- The Web Distributed Authoring and Versioning (WebDAV) delivery type for publishing packages to WebDAV-compliant servers such as Intraspect c-business Server and Microsoft Exchange 2000 is now fully supported. This delivery type has also been improved to enable users to name subdirectories.
- The support for archiving packages has been improved. You can now archive packages to remote file systems, using HTTP and FTP protocols.
- The e-mail delivery type now accepts a data set that contains e-mail recipient addresses. This makes it easier for you to publish to an address list.
- Release 8.1 introduced a very powerful way to publish SAS output by applying a *viewer* that can format result set packages (or specific package entries) into an

HTML-rendered e-mail message. In Release 8.2, you can also create viewers that format packages into e-mail messages that are sent in text format.

---

## Directory Services

You can now use SAS Integration Technologies with Version 3 LDAP servers. (Prior support was based on LDAP Version 2.)

---

## Distributed Objects

- The Integrated Object Model (IOM) interfaces now support versioning. This enables your client applications to query the server to determine what features it supports.
- Release 8.1 added support for using the VisiBroker CORBA ORB on NT servers. In Release 8.2, support for using the IONA Orbix 2000 ORB on NT servers has been added.
- Workspace pooling for Windows clients enables you to share a pool of server workspaces (serially) among multiple clients. This is particularly useful in three-tier deployments like those that use Active Server Pages (ASP).
- Enhancements to the Java Workspace Factory make it easier for you to use the Workspace Pooling feature in your Java clients. In this release, the functions of the WorkspacePoolDriver have been integrated into the Java Workspace Factory. In addition, an LDAP server is no longer required in order to use the Java Workspace Pooling feature.

---

## Message Queuing

- Release 8.1 introduced a set of SAS CALL routines for configuring and accessing message queues on systems that use Geneva Message Queuing (GMQ) software from Level 8 Systems. In Release 8.1, this feature was experimental. In Release 8.2, it is fully supported.
- You can now use the SAS Common Messaging Interface in your SAS applications to access message queues on systems that are running TIBCO TIB/Rendezvous message queuing software.

---

## Enhancements to SAS Package Retriever

---

### New WebDAV Delivery Type

- Web Distributed Authoring and Versioning (WebDAV) is an emerging industry standard that is based on extensions to HTTP 1.1. It enables package publishers, programmers, and package consumers to collaborate on the development of files and collections of files on remote Web servers. It also enables consumers to retrieve packages that have been delivered to a WebDAV-compliant server.
- In order to retrieve a package or a package entry from a secured WebDAV-compliant server, a package consumer must supply an HTTP user ID and an HTTP password.

---

## Additional Properties for the HTML Package Entry Type

- An HTML file package entry can now reference additional HTML files for inclusion in an HTML file package entry. The additional files are called companion files. The Companion File option enables you to retrieve companion files and to specify the output location.
- An HTML file package entry can now be delivered with a property for Character-Set Encoding. When you retrieve the package entry, you can either accept the specified encoding, or you can specify your own encoding value. Encoding is necessary only when the host architectures of the publishing host and the retrieving host are incompatible.

---

## Enhancements to SAS Publisher

---

### New WebDAV Delivery Type

- Web Distributed Authoring and Versioning (WebDAV) is an emerging industry standard that is based on extensions to HTTP 1.1. It enables package publishers, programmers, and package retrievers to collaborate on the development of files and collections of files on remote Web servers. It also enables publishers to publish packages for delivery to a Web-compliant server.
- With the WebDAV delivery type, you can specify several advanced properties. For example, you can specify a collection URL or a parent URL as the storage location for the published package. If you are publishing to a secured server, you must specify an HTTP user ID and an HTTP password. You can specify one or multiple namespaces, or contexts, on a server to which you publish the package. Finally, you can specify a template that contains formatting directives for rendering a specific view of a published package.

---

### FTP and HTTP User ID and Password Properties Support Package Delivery

When you publish packages, you can now include the user ID and password properties that are appropriate to the protocol (FTP or HTTP) for delivery to these delivery types:

- archive
- WebDAV-compliant server
- SAS channel that delivers an archive package by e-mail
- archive package that is published to e-mail
- archive package that is published to a message queue.

Publishers need a user ID and a password in order to publish to a secured server. In addition to the user ID and password attributes, other advanced attributes can be selected to specify package delivery to the WebDAV and e-mail delivery types.

---

## Additional Properties for the E-mail Delivery Type

- Besides rendering package views in HTML format, the viewer processing facility can now render views in text format. A text-formatted view might be necessary if the destination e-mail program does not support the HTML MIME type.
- The package publisher can now pass Reply To and From values to the package recipient's e-mail program. The Reply To value enables recipients to return an e-mail message to the package sender, or package publisher, or to some other designated e-mail address. The From value specifies the e-mail address of the package publisher.
- The e-mail delivery type now supports the specification of an e-mail address list that can be derived from a SAS data set according to a specific variable. SAS options can be specified for opening a data set.

---

## Additional Properties for the ODS Package Entry Type

- An ODS-generated package entry can now reference additional HTML files for inclusion in an ODS package entry. Companion files are typically HTML files and are not ODS-generated files.
- A package publisher can now assign a character-set encoding property to an ODS package entry. The encoding is applied when the package is retrieved. The encoding property is necessary only when the host architectures of the publishing host and the retrieving host are incompatible.

---

## URL Package Entry Type Supports Relative URL Specification

In addition to explicitly specifying the appropriate URL prefix type (http://, ftp://, or file:///), you can now specify a URL that is relative to a base URL. For example, if the base URL is `www.mycorp.com`, then the relative URL `./news/article.html` locates `article.html` in the subdirectory `news`, which is relative to the base URL.

---

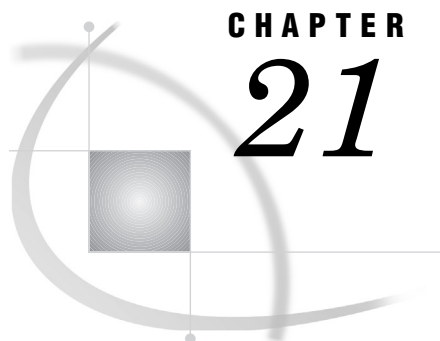
## Data Set Entry Type Supports Data Set Options

SAS options can now be specified for a data set that is being included in a package. The `PW=` option, which enables package consumers to access a read-protected SAS file when they retrieve a package, might be especially useful.

---

## For Additional Information

For more information about these features, please see the Enterprise Integration Community Web site at [www.sas.com/rnd/eai/](http://www.sas.com/rnd/eai/).



## CHAPTER

## 21

## SAS/IntrNet Software

---

*Application Dispatcher*    **63**

*Java Tool*    **64**

*MDDB Report Viewer*    **64**

*New Tool: SAS Design-Time Controls*    **64**

*Xplore Sample Application*    **64**

*For Additional Information*    **64**

---

### Application Dispatcher

- Improved administration interface

When you access the Application Broker URL using no parameters, a default administration page appears in your browser. This page gives you access to administration information, SAS/IntrNet samples, and SAS/IntrNet documentation. Also, additional usage statistics are now available from a Load Manager statistics report.

- Support for new catalog entry types

The replay program feature and the automatic HTTP header generation feature of Application Dispatcher now support several Web-related catalog entry types. These new types include CSS, CSV, PDF, RTF, WML, and XML.

- Support for additional output destinations

The automatic HTTP header generation feature now recognizes most of the Output Delivery System (ODS) output destinations and generates appropriate default Content-type headers. Supported output destinations include HTML, CSV, PDF, POSTSCRIPT, RTF, WML, and XML.

- Support for Application Server and Load Manager as Windows NT services

The Application Server and the Load Manager can now be installed as Windows NT services. This enables you to automatically invoke Application Dispatcher at system boot time on a Windows NT server.

- Improved tools for configuring Application Dispatcher services

The inetcfg utility now generates shared configuration files for all servers in a service. On some hosts, the inetcfg utility generates sample configuration information for the broker.cfg file.

- Improved Load Manager scheduling

Load Manager scheduling for Web applications that use sessions has been improved.

- Customized handling of expired and invalid sessions

A user exit for expired and invalid sessions is now available. You can customize the handling of expired sessions to enable users to re-authenticate and restart a Web application.

□ Improved Application Dispatcher samples

Improved Application Dispatcher samples demonstrate many features of ODS, including generating multiple output formats, building drill-down links into an application, and generating graphical output with the GIF, JAVA, and ACTIVEX drivers.

---

## Java Tool

Version 2.5 of the Java Tools package is now available.

---

## MDDB Report Viewer

The MDDB Report Viewer now supports all of the features of the SAS/EIS Software Access Control system. For more information, see the *SAS/EIS Software Administrator's Guide*.

---

## New Tool: SAS Design-Time Controls

SAS Design-Time Controls are now available for building SAS/IntrNet Web applications. With SAS Design-Time Controls, you can use a WYSIWYG editor on your Windows desktop system to create Web pages that access the power of SAS/IntrNet software.

---

## Xplore Sample Application

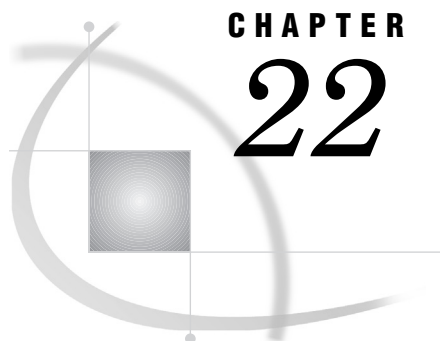
Xplore supports new catalog entry types. These new types include CSS, CSV, PDF, RTF, WML, and XML.

---

## For Additional Information

For more information about SAS/IntrNet software and its components, see the SAS/IntrNet Web site at [www.sas.com/rnd/web/intrnet/](http://www.sas.com/rnd/web/intrnet/).





## CHAPTER

## 22

## SAS/MDDB Server Software

---

*New Methods for the MDDB\_M Class* 65

*Enhanced Method for the MDDB\_M Class* 65

---

### New Methods for the MDDB\_M Class

The MDDB\_M class supports the following new methods:

- `_GET_ORIGINAL_UNFORMATTED_CLASS_VALUES_` returns the unformatted class values, if any, from the MDDB. The unformatted class values are listed in the order in which they were originally stored on the MDDB. Optionally, the method also returns the formatted class values list, so that the correlation between the unformatted and formatted class values can be determined.
- `_GET_CLASS_STORED_STATUS_` retrieves the stored format status for the classifier values. If the classifier stores both the unformatted and formatted values, then the method returns a 1. If the classifier stores only the formatted values, then the method returns a 0.

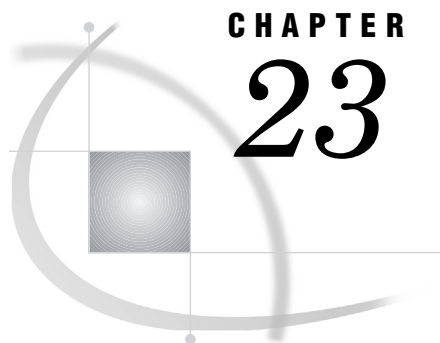
---

### Enhanced Method for the MDDB\_M Class

The MDDB\_M class contains the following updated method:

- `_MAKE_TABLE_` creates a subtable from the main MDDB table. The NUNIQUE list can contain named items for each classifier that is specified. In Release 8.2, each named item corresponds to the name of the analysis variable that is based on the classifier. In previous releases, the name of the analysis variable was generated by SAS. If the named item is omitted from the NUNIQUE list, then SAS creates a name that is compatible with previous releases. The named items on an NUNIQUE list must be unique.





## CHAPTER

## 23

## SAS OLAP Server Software

---

*Support for Multiple Environments* 67  
*Experimental User Interface for Building MDDBs* 67  
*Activating Access Control* 67

---

### Support for Multiple Environments

The SAS OLAP Server is now available in the Windows NT and UNIX environments, in addition to the previously supported OS/390 environment. You can now use the SAS OLAP Server in these operating environments to

- ☐ create MDDBs using PROC MDDB
  - ☐ register OLAP metadata
  - ☐ define and register HOLAP (multidimensional data provider or MDP) data groups
  - ☐ create access control definitions
  - ☐ use model coordination to synchronize multiple EIS viewers, which are displayed side by side on the screen or in an Application Screen Builder object
  - ☐ surface MDDBs and HOLAP data groups using the Open OLAP Server.
- 

### Experimental User Interface for Building MDDBs

A new experimental user interface guides you through the steps of building an MDDB and registering it in a repository. Optionally, you can create crossings for the MDDB.

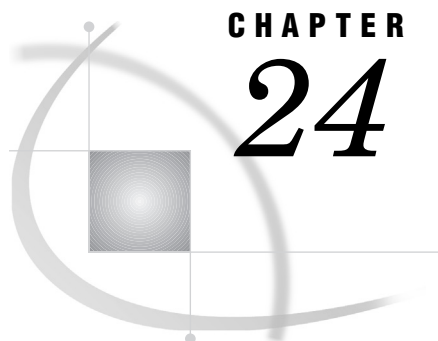
Enter **af c=sashelp.eissrv.startmd.scl** on the command line to start the interface.

---

### Activating Access Control

If you used access control for your SAS OLAP Server environment in a previous release, be sure to activate it for this release, as well. Follow the steps described in the *SAS OLAP Server Administrator's Guide* to activate access control and to point it to the location of your existing access control files.





## SAS/OR Software

---

|                                   |    |
|-----------------------------------|----|
| <i>BOM Procedure</i>              | 69 |
| <i>CPM Procedure</i>              | 69 |
| <i>GANTT Procedure</i>            | 69 |
| <i>INTPOINT Procedure</i>         | 70 |
| <i>PM Procedure</i>               | 70 |
| <i>For Additional Information</i> | 70 |

---

### BOM Procedure

The new BOM procedure performs bill-of-material processing. It composes a series of single-level bills of material into a multilevel, tree-structured bill of material, determines the level of each part in the bill, and represents the multilevel bill-of-material structure in the form of an indented bill of material. PROC BOM can also output a summarized bill of material.

---

### CPM Procedure

- The SETFINISHMILESTONE option has been added to the CPM procedure statement to enable you to specify that milestones (zero duration activities) should have the same start and finish times as the finish time of their predecessor. In other words, this option enables milestones that mark the *end* of the preceding activity to coincide with its finish time.
  - The FIXASTART option has been added to the ACTUAL statement to enable you to request that the actual start and finish times of an activity should be left unchanged even if they coincide with a non-working time. Thus, if the actual start time is specified to be sometime on Sunday, it is left unchanged even if Sunday is a non-working day in the activity's calendar.
- 

### GANTT Procedure

- The milestone color specifications have been made more flexible. The milestone colors are now determined using the same rules as for schedule bars, and consequently, they can also be set individually.
- The WEB= variable enables you to define an HTML reference for each activity. In the past the URL was only associated with the schedule bars for the activity. This association has now been extended to include milestones and ID variables.

---

## INTPOINT Procedure

The new INTPOINT procedure performs optimization using the Interior Point algorithm. This is the same Interior Point algorithm used by the NETFLOW procedure to solve Linear Programming problems and Network problems. In the future, PROC INTPOINT will exploit parallel processing.

---

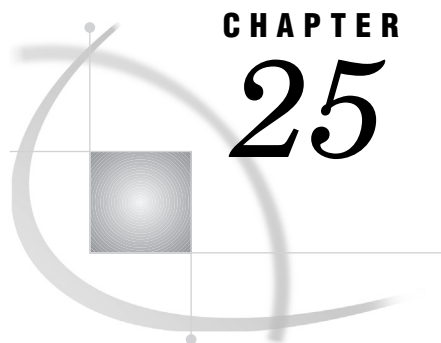
## PM Procedure

- MP2KTOPM is a new SAS macro that converts Microsoft Project 2000 data saved in MDB format to a form that is readable by the PM procedure. The converted data can also be read by the CPM procedure.
- The new options added to the CPM procedure are also available with PROC PM.

---

## For Additional Information

For more information about the features described here, see *SAS/OR Software: Changes and Enhancements, Release 8.2*.



## CHAPTER

## 25

## SAS/QC Software

|                                   |    |
|-----------------------------------|----|
| <i>CAPABILITY Procedure</i>       | 71 |
| <i>RELIABILITY Procedure</i>      | 71 |
| <i>SHEWHART Procedure</i>         | 72 |
| <i>For Additional Information</i> | 73 |

## CAPABILITY Procedure

- ❑ The BARLABEL= option on the HISTOGRAM and COMPHISTOGRAM statements displays labels above histogram bars.
- ❑ The CLIPSPEC= option on the HISTOGRAM and COMPHISTOGRAM statements clips histogram bars at the upper and lower specification limit lines when there are no observations outside the specification limits.
- ❑ The CTEXTSIDE= option on the COMPHISTOGRAM statement specifies the color of row labels.
- ❑ The CTEXTTOP= option on the COMPHISTOGRAM statement specifies the color of column labels.
- ❑ The ENDPOINTS= option on the HISTOGRAM and COMPHISTOGRAM statements specifies histogram interval boundary values and causes the boundaries, rather than interval midpoints, to be labeled.
- ❑ The NENDPOINTS= option on the HISTOGRAM statement specifies the number of histogram interval boundaries and causes the boundaries, rather than interval midpoints, to be labeled.
- ❑ The NMIDPOINTS= option on the HISTOGRAM statement specifies the number of histogram intervals.

## RELIABILITY Procedure

- ❑ The LOGSCALE statement enables the logarithm of the distribution scale parameter to be modeled as a function of explanatory variables.
- ❑ The PLOTSYMBOL= option enables symbols to represent events in a MCF plot.
- ❑ The PLOTCOLOR= option, within the MCFPLOT statement, enables the specification of symbol colors in a MCF plot.
- ❑ The logistic function has been added as an optional transformation of the independent variable in a regression model for the RELATION= option within the MODEL and RELATIONPLOT statements.
- ❑ The CFIT= option, within the PROBPLOT and RELATIONPLOT statements, has been enhanced to allow for multiple colors of fit lines.

- The LFIT= option, within the PROBPLOT and RELATIONPLOT statements, enables the specification of line styles for fit lines and confidence curves in a probability plot.
- The NOPPOS option, within the PROBPLOT and RELATIONPLOT statements, suppresses the plotting of symbols for failures.
- The NPINTERVALS= option, within the PROBPLOT statement, enables the specification of the type of nonparametric confidence interval displayed in a probability plot.
- The PINTERVALS= option, within the PROBPLOT and RELATIONPLOT statements, enables the specification of the type of parametric pointwise confidence interval displayed.
- The POSSYMBOL= option, within the PROBPLOT statement, enables symbols to represent failures on a probability plot.
- The PPOSCOLOR= option, within the PROBPLOT statement, enables the specification of colors of symbols representing failures on a probability plot.
- The SHOWMULTIPLES option, within the PROBPLOT and RELATIONPLOT statements, displays the count for multiple overlaying symbols.
- The CLOTFIT= option, within the RELATIONPLOT statement, enables the specification of colors for percentile lines.
- The FITTYPE= option, within the RELATIONPLOT statement, specifies the method of estimating distribution parameters.
- The LLOTFIT= option, within the RELATIONPLOT statement, enables the specification of line styles for percentile lines.
- The RCENCOLOR= option, within the RELATIONPLOT statement, enables the specification of colors for the symbols representing uncensored, right censored, and left censored observations in a relation plot.
- The RCENSYMBOL= option, within the RELATIONPLOT statement, enables symbols to represent right censored and left censored observations in a relation plot.

---

## SHEWHART Procedure

- The OVERLAY= option specifies variables to be plotted as overlays on the primary control chart.
- The COVERLAY= option specifies the colors used to plot overlay variables on the primary chart.
- The CCOVERLAY= option specifies colors for the line segments connecting points on primary chart overlays.
- The LOVERLAY= option specifies line types for the line segments connecting points on primary chart overlays.
- The OVERLAYHTML= option specifies uniform resource locators (URLs) to be associated with points on primary chart overlays.
- The OVERLAYSYM= option specifies symbols used to plot overlays on the primary chart.
- The OVERLAYSYMHT= option specifies the heights of symbols used to plot overlays on the primary chart.
- The WOVERLAY= option specifies the widths in pixels of the line segments connecting points on primary chart overlays.
- The OVERLAY2= option specifies variables to be plotted as overlays on a secondary control chart.



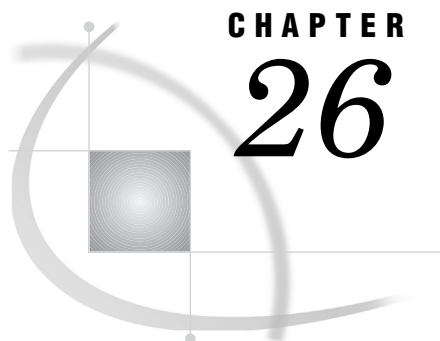
- The COVERLAY2= option specifies the colors used to plot overlay variables on the secondary chart.
- The CCOVERLAY2= option specifies colors for the line segments connecting points on secondary chart overlays.
- The LOVERLAY2= option specifies line types for the line segments connecting points on secondary chart overlays.
- The OVERLAY2HTML= option specifies URLs to be associated with points on secondary chart overlays.
- The OVERLAY2SYM= option specifies symbols used to plot overlays on the secondary chart.
- The OVERLAY2SYMHT= option specifies the heights of symbols used to plot overlays on the secondary chart.
- The WOVERLAY2= option specifies the widths in pixels of the line segments connecting points on secondary chart overlays.
- The OVERLAYLEGLAB= option specifies the label displayed to the left of the overlay legend.
- The COVERLAYCLIP= option specifies the color used to plot clipped values on overlay plots.
- The OVERLAYCLIPSYM= option specifies the symbol used to plot clipped values on overlay plots.
- The OVERLAYCLIPSYMHT= option specifies the height of the symbol used to plot clipped values on overlay plots.
- The NOOVERLAYLEGEND option suppresses the legend for overlay plots.

---

## For Additional Information

For more information about the features described here, see *SAS/QC Software: Changes and Enhancements, Release 8.2*.





## SAS/SHARE Software

---

*Services Support* 75

---

### Services Support

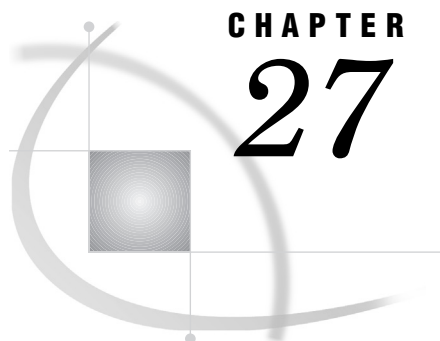
SAS/SHARE servers are now implemented as *services*. This means that they can now respond to system requests such as start and stop commands from the host operating system.

For example, in Windows NT and Windows 2000 environments, you can configure a SAS/SHARE server as a Windows service that starts automatically when the operating system is started. You can also start, stop, and restart your SAS/SHARE server using the Windows Service Control Manager.

*Note:* In order to configure your SAS/SHARE server as a Windows service, you must run the SAS Service Configuration Utility (SSCU). The SSCU is shipped with Base SAS software for Windows. On Windows, more instructions are available in *Configuring a SAS/SHARE Server as a Windows Service* from the table of contents in SAS/SHARE Help.

In an OS/390 operating environment, the SAS/SHARE server also now responds to console STOP commands. This means that you no longer have to use the PROC OPERATE procedure to terminate a SAS/SHARE server.





## SAS/STAT Software

---

|                                   |    |
|-----------------------------------|----|
| <i>BOXPLOT Procedure</i>          | 77 |
| <i>CATMOD Procedure</i>           | 78 |
| <i>FACTOR Procedure</i>           | 78 |
| <i>FREQ Procedure</i>             | 78 |
| <i>GAM Procedure</i>              | 78 |
| <i>LIFEREG Procedure</i>          | 79 |
| <i>LOGISTIC Procedure</i>         | 79 |
| <i>LOESS Procedure</i>            | 79 |
| <i>MI Procedure</i>               | 80 |
| <i>MIANALYZE Procedure</i>        | 80 |
| <i>NPAR1WAY Procedure</i>         | 80 |
| <i>PROBIT Procedure</i>           | 80 |
| <i>REG Procedure</i>              | 81 |
| <i>SURVEYMEANS Procedure</i>      | 81 |
| <i>TRANSREG Procedure</i>         | 81 |
| <i>References</i>                 | 81 |
| <i>For Additional Information</i> | 81 |

---

### BOXPLOT Procedure

The **BOXPLOT** procedure supports several new options in the **PLOT** statement. These options enable you to overlay plots of additional variables on a box plot and to create links in a plot when graphics output is directed into HTML.

- ❑ The **HTML=** option specifies uniform resource locators (URLs) to be associated with box-and-whisker plots.
- ❑ The **COVERLAY=** option specifies the colors used to plot overlay variables.
- ❑ The **CCOVERLAY=** option specifies colors for the line segments connecting points on overlays.
- ❑ The **LOVERLAY=** option specifies line types for the line segments connecting points on overlays.
- ❑ The **NOOVERLAYLEGEND** option suppresses the legend for overlay plots.
- ❑ The **OVERLAY=** option specifies variables to be plotted as overlays on a box plot.
- ❑ The **OVERLAYHTML=** option specifies URLs to be associated with points on overlays.
- ❑ The **OVERLAYLEGLAB=** option specifies the label displayed to the left of the overlay legend.
- ❑ The **OVERLAYSYM=** option specifies symbols used to plot points on overlays.

- The OVERLAYSYMHT= option specifies the heights of symbols used to plot points on overlays.
- The WOVERLAY= option specifies the widths in pixels of the line segments connecting points on overlays.

---

## CATMOD Procedure

You can now produce Wald confidence limits for the parameter estimates with the CLPARM option in the MODEL statement; the ALPHA= option enables you to set the significance level.

---

## FACTOR Procedure

The FLAG=, FUZZ=, and ROUND options are restored and function as they did previously. However, when standard errors or confidence intervals are also requested, these options will have no effect.

---

## FREQ Procedure

The new POINT option in the EXACT statement provides point probabilities for the exact test statistics. In the TABLES statement, there is now a LEVEL= suboption for the BINOMIAL option. The LEVEL= option specifies the variable level for the binomial proportion for one-way tables. The new BINOMIALC option requests the BINOMIAL option statistics, but includes a continuity correction in the asymptotic confidence limits and the asymptotic test. Similarly, the RISKDIFFC option requests the RISKDIFF} option statistics, but includes a continuity correction in the asymptotic confidence limits.

---

## GAM Procedure

The GAM procedure, experimental in Release 8.1, is production in Release 8.2 and includes the following new features:

- In the MODEL statement, a local regression smoother (LOESS) is provided for univariate smoothing.
- Several new distributions are available with the DIST= option in the MODEL statement. These include the gamma, Poisson, and binomial distributions.
- The MAXITER=, MAXITSCORE=, EPSILON=, and EPSSCORE= options are provided for users to control the convergence of the backfitting algorithm and local score algorithm.
- In the OUTPUT statement, the keywords PRED and ADIAG, UCLM, LCLM, RESID, STD, and ALL have been added. They request upper and lower component-wise confidence limits, residual, standard deviation, or all of them to be put into the output data set.
- Multiple SCORE statements are now supported so that users can make predication on multiple data sets.

---

## LIFEREG Procedure

- The MODEL statement now enables you to specify interaction terms among any of the specified explanatory variables. Because of the more complicated models that you can specify, you can only use a single MODEL statement in each PROC LIFEREG invocation.
- The new PROBPLOT statement enables you to construct probability plots for complete or censored data. You can display the fitted distribution and confidence intervals along with nonparametric estimates of the CDF on the probability plots. In addition, you can control graphical features such as plot layout, colors, plotting symbols, line styles, and the text fonts used in the probability plot.
- You can now include models that contain variables listed in the CLASS statement in the OUTEST= data set.
- You can now obtain standardized and Cox-Snell residuals in the OUTPUT= data set.

---

## LOGISTIC Procedure

- You can now fit a generalized logit model (Agresti, 1990) when you have nominal response data. In this multiple-group logit model, each nonreference response category is contrasted against the reference category. This model reduces to the binary logit model when there are only two response categories. Unlike the cumulative logit model (for ordinal response data) that contains one set of slope parameters for the entire model, the generalized logit model contains a set of slope parameters for each response category other than the prespecified reference category.

The generalized logit model is specified by the MODEL statement option LINK=GLOGIT. You can choose a response category as the reference in forming the various logit functions. The default is to use the last response category as the reference.

- Options specific to the response variable can be specified immediately after the response variable in the MODEL statement by enclosing them in a pair of parentheses. These response variable options include the EVENT= option for specifying the event category for the binary response model, the REFERENCE= option for specifying the reference category, the ORDER= option for specifying the order of the response categories, and the DESCENDING option for reversing the order of the response categories.
- For an exact conditional analysis, the new EXACTOPTIONS option NOSORT processes your data without resorting. This enables you to input your data in an order that you think may speed up computations.

---

## LOESS Procedure

The SELECT= option in the MODEL statement now enables you to specify one of three degree of freedom criteria and an associated target value. The model that is selected is the model whose specified degree of freedom criterion is the closest to the specified target value.

---

## MI Procedure

Multiple imputation is a strategy for dealing with data sets with missing values. You replace each missing value with a set of plausible values that represent the uncertainty about the right value to impute. You create multiply imputed data sets, analyze them with standard analyses, and then combine the results. You produce valid statistical inferences that properly reflect the uncertainty due to the missing values.

The MI procedure creates multiple imputed data sets for incomplete  $p$ -dimensional multivariate data. It offers three methods for creating the imputed data sets: the regression method, the propensity score method, and the Markov Chain Monte Carlo (MCMC) method. The procedure creates an output data set containing  $m$  imputed versions of the original data. In each version, the missing values are replaced with imputed values. For the MCMC method, you can specify whether you want a single chain for all  $m$  imputations or a separate chain for each imputation. You can also specify the initial estimates for the MCMC method. After analyzing your imputed data with standard procedures, you use the MIANALYZE procedure to combine the results.

The MI procedure was introduced in Release 8.1 and remains experimental in Release 8.2, with various new options and output displays available. Among others, a new TRANSFORM statement enables you to transform variables before imputation and back-transform these variables before combining inferences and creating output data sets.

---

## MIANALYZE Procedure

The MIANALYZE procedure combines the results of the analyses of  $m$  imputations and generates valid statistical inferences. The procedure uses as input either a specially structured SAS data set that contains the parameter estimates and associated covariance matrix from each imputed data set or a pair of specially structured SAS data sets that contain the parameter estimates and covariance matrices of the parameter estimates, respectively.

The MIANALYZE procedure was introduced in Release 8.1 and remains an experimental procedure in Release 8.2.

---

## NPAR1WAY Procedure

The new KS option in the EXACT statement provides the exact Kolmogorov-Smirnov two-sample test. Also, the new POINT option in the EXACT statement provides point probabilities for the exact test statistics.

---

## PROBIT Procedure

- The MODEL statement now enables you to specify interaction terms among any of the specified explanatory variables. Because of the more complicated models that you can specify, you can only use a single MODEL statement in each PROC PROBIT invocation.
- You can now graphically display results of model parameter estimation. For binary data analysis, you can construct plots of predicted probabilities, inverse predicted



probabilities, and cumulative probabilities. For ordinal multinomial models, you can display plots of predicted probabilities, linear predictors, and cumulative probabilities. In addition, you can control graphical features such as plot layout, colors, plotting symbols, line styles, and the text fonts used in the plots.

- You can now include models that contain variables listed in the CLASS statement and ordinal response multinomial models in the OUTEST= data set.

---

## REG Procedure

The SCORR1 and SCORR2 options in the MODEL statement now enable you to specify TEST and SEQTEST options that request F-tests, *p*-values, and cumulative R-Square values as variables are sequentially added to a model.

---

## SURVEYMEANS Procedure

The RATIO statement requests ratio analysis for means or proportions of analysis variables. A ratio statement names the variables whose means will be used as numerators or denominators in a ratio. There can be any number of analysis variables, either continuous or categorical, in the input data set.

---

## TRANSREG Procedure

The MODEL statement now enables you to specify Box-Cox (1964) transformations of the dependent variables. You can specify a list of power parameters using the LAMBDA= transformation option. The procedure chooses the optimal power parameter using a maximum likelihood criterion (Draper and Smith, 1981). The output contains a table of power parameters, log likelihood, and a confidence interval.

---

## References

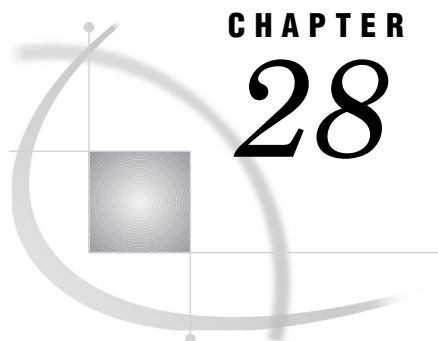
- Agresti, A. (1990), "Categorical Data Analysis," New York: John Wiley & Sons, Inc.
- Box, G.E.P. and Cox, D.R. (1964), "An Analysis of Transformations," *Journal of the Royal Statistics Society*, B-26, 211-252.
- Draper, N.R. and Smith, H. (1981), *Applied Regression Analysis*, Second Edition, New York: John Wiley & Sons, Inc.

---

## For Additional Information

For more information about these features, see *SAS/STAT Software: Changes and Enhancements, Release 8.2*.





## CHAPTER 28

# SAS/Warehouse Administrator Software

---

|                                                    |           |
|----------------------------------------------------|-----------|
| <i>Metadata API Enhancements</i>                   | <b>83</b> |
| <i>LDAP Security Interface Enhancements</i>        | <b>83</b> |
| <i>Closer Integration with Enterprise Reporter</i> | <b>84</b> |
| <i>OLAP Data Store Enhancements</i>                | <b>84</b> |
| <i>Process Library Enhancements</i>                | <b>84</b> |
| <i>Getting Started Tutorial</i>                    | <b>85</b> |

---

## Metadata API Enhancements

You can use the SAS/Warehouse Administrator metadata API to add, update, and delete process objects. For example, you can write a metadata API program that creates a data store and also creates all of the processes that are required to extract, transform, and load information into that data store.

For details about this feature and other new API features in this release, see *SAS/Warehouse Administrator Metadata API Reference, Release 2.2*. You can download a PDF version of this manual from the SAS Web site. To download this manual:

- 1 Display the SAS Web page at [www.sas.com](http://www.sas.com).
  - 2 Select **Service and Support**, then **Online Documentation**.
  - 3 From the Product-Specific Documentation window, select **SAS/Warehouse Administrator**.
  - 4 From the page that is displayed, download *SAS/Warehouse Administrator Metadata API Reference, Release 2.2*.
- 

## LDAP Security Interface Enhancements

The LDAP security interface, which was experimental in Release 8.1 but is fully supported in Release 8.2, enables you to define a security hierarchy for the data stores in a Data Warehouse. This hierarchy can be used to control access to warehouse data stores in much the same way that an operating system uses access control lists (ACLs) to control access to files. New features in this release include:

- the ability to control access to OLAP Groups, OLAP Tables, and OLAP MDDBs
- the ability to control access to a SAS/EIS metabase that contains metadata that was exported from SAS/Warehouse Administrator. This enables LDAP applications to access warehouse data through SAS/EIS.

## Closer Integration with Enterprise Reporter

It is now easier to use Enterprise Reporter to generate reports on groups and data stores that you have selected in the SAS/Warehouse Administrator Explorer. The following menu options are available from the Explorer:

- **Documenting the Data Warehouse** — exports metadata for the current Data Warehouse, as well as the metadata for all ODD Groups in the current Warehouse Environment. The exported metadata is in a form that is optimized for reporting in Enterprise Reporter.

To use this feature, open a Warehouse Environment in the SAS/Warehouse Administrator Explorer and select a group or data store in the Environment. Go to the SAS menu bar and select **Tools**, then **Documenting the Data Warehouse**. The Documenting the Data Warehouse window appears. For details about that window, click its [Help](#) button.

If Enterprise Reporter software is installed locally, you are given the option of creating an InfoFolder for the exported metadata.

- **Open in Enterprise Reporter** — reads the metadata for the selected data store into Enterprise Reporter. Use this option to create ad hoc reports about data stores. You will not see this option unless Enterprise Reporter software is installed locally. To use this feature, right-click a data store and select **Data Utilities**, then **Open in Enterprise Reporter**. The main Enterprise Reporter window appears.

## OLAP Data Store Enhancements

The code that SAS/Warehouse Administrator generates for OLAP Table Load Steps is now more efficient.

- Level 2.0 code generation can now take advantage of recent PROC SUMMARY enhancements to maximize performance for OLAP Tables with multiple crossings. You can specify different sort orders for each class variable. To specify more than 32 class columns in an OLAP Table, use character `_TYPE_` columns (instead of numeric `_TYPE_` columns, which can support a maximum of 32 class columns).
- The Load Options tab for Load Steps has a new option for OLAP Tables: **Single Pass Summarization**. Use this option to either maximize performance or minimize memory use.

*To maximize performance.* If the Load Step for an OLAP Table is generated by SAS/Warehouse Administrator, and the **Single Pass Summarization** option is set to **Yes**, data for all crossings is produced by a single PROC SUMMARY step.

*To minimize memory use.* If the Load Step for an OLAP Table is generated by SAS/Warehouse Administrator, and if the **Single Pass Summarization** option is set to **No**, data for each crossing is produced by a separate PROC SUMMARY step.

- On the Columns tab of the OLAP Table Properties window, you can now assign a role of `_FREQ_` to an OLAP Table column. For an OLAP Table, a `_FREQ_` column gives the number of observations (whether any column values are missing or not) for each crossing and combination of class column values.

## Process Library Enhancements

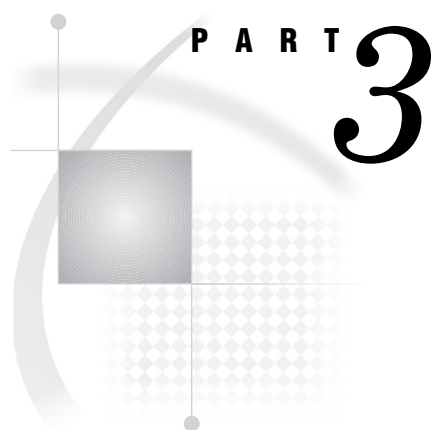
You can now register a Process Library routine as a Post-Load Step. Previously, you could register these routines only as Load Steps.

---

## Getting Started Tutorial

The *Getting Started with SAS/Warehouse Administrator* tutorial is now available from the SAS/Warehouse Administrator desktop. To access the tutorial, select its icon from the desktop.





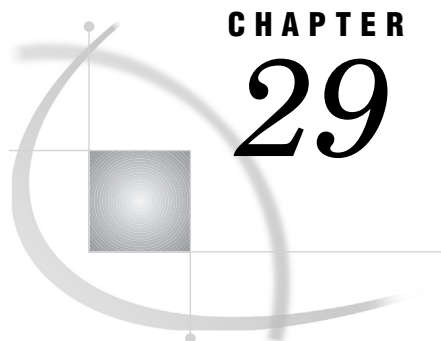
## **New Products**

*Chapter 29*.....**SAS/ACCESS Interface to PeopleSoft** 89

*Chapter 30*.....**SAS Data Quality - Cleanse Software** 91







## CHAPTER

## 29

## SAS/ACCESS Interface to PeopleSoft

---

*Overview*    **89**

*For Additional Information*    **89**

---

### Overview

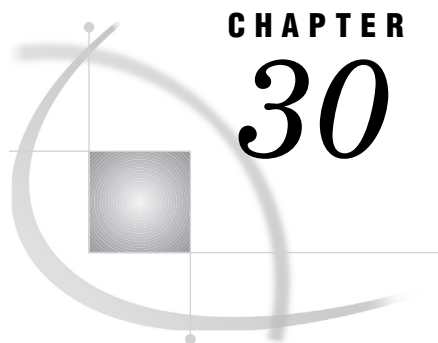
The new SAS/ACCESS interface to PeopleSoft enables you to extract, browse, and search PeopleSoft metadata. You can query the extracted PeopleSoft metadata to generate views of the PeopleSoft data and save these views as SAS data files and SAS views. You can also save the code that was used to build the query as SAS source code.

---

### For Additional Information

For more information about these features, invoke the SAS/ACCESS interface to PeopleSoft and use the Help buttons on the windows to display the appropriate Help topic.





## CHAPTER

## 30

## SAS Data Quality - Cleanse Software

---

*Overview*    91

*SAS Data Quality – Cleanse Features*    91

---

### Overview

SAS Data Quality – Cleanse software enables you to analyze, cleanse, and standardize your data. This enhances the value of your data by reducing duplication and by increasing accuracy and usability. For example, the STATE variable might contain several variations for West Virginia (such as WV, W.V., and W. Virginia). This makes it difficult to construct a query based on the STATE variable that will yield accurate results. You can use SAS Data Quality – Cleanse software's specialized clustering algorithm for state names and abbreviations to group these variations together. Then you can transform the data to conform to a preferred, or standard, form of the state name.

---

### SAS Data Quality – Cleanse Features

- The MATCH procedure enables you to analyze a SAS data set to identify the variations of data that occur for a specified variable and to match similar observations based on criteria that you specify.
- The SCHEME procedure enables you to build standardization schemes from your data and to use those schemes to transform your data.
- The SAS Data Quality – Cleanse functions and CALL routines enable you to perform a wide variety of data analysis, cleansing, and standardization tasks on your data. For example, they enable you to analyze and parse input data and to return standardized values for data such as names and addresses. They also enable you to standardize the capitalization of data values and to determine whether a particular character value represents an individual or an organization.



# Index

---

## A

access control  
     SAS/EIS 48  
     SAS OLAP Server 67  
 ADX interface 39  
 Agent Services 46  
 Application Dispatcher 63  
 Application Response Measurement (ARM)  
     macros 11  
 ARIMA procedure 49  
 ARM macros 11  
 automatic macro variables 11

## B

base SAS software  
     general features 3  
     language elements 7  
     procedures 13  
 BFS (Byte File System) 17  
 bill-of-material processing 69  
 binary formats 8  
 binary informats 11  
 BOM procedure 69  
 BOXPLOT procedure 77  
 Byte File System (BFS) 17

## C

CA-DATACOM/DB interface 37  
 CALL routines 9  
     OS/2 environment 27  
     Windows environment 22  
 CAPABILITY procedure 71  
 CATMOD procedure 78  
 CEDA (Cross-Environment Data Access) 3  
     SAS/CONNECT software 46  
 character-set encoding 11  
 cleansing data 91  
 CMS environment 17  
     Byte File System (BFS) 17  
     commands 18  
     macros 18  
     procedures 18  
     SAS language elements 18  
     sending e-mail with SMTP interface 17  
     statements 18

    system options 18  
     Universal Printing 17  
     writing to SAS log from pipelines 17  
 code-editing windows 5  
 color schemes 5  
 commands 7  
     CMS environment 18  
     OpenVMS environment 25  
     OS/2 environment 27  
     OS/390 environment 29  
     UNIX environments 31  
     Windows environment 22  
 Common Messaging Interface 46  
 compressed files 4, 7, 12  
     SAS/CONNECT software 46  
 Computed Values window  
     SAS/EIS software 47  
 count data 49  
 COUNTREG procedure 49  
 CPM procedure 69  
 Cross-Environment Data Access (CEDA) 3  
     SAS/CONNECT software 46  
 currency conversion 5  
     formats 8  
     informats 11

## D

data set options 7  
 Data Sources window  
     SAS/ASSIST software 41  
 Date and Time formats 8  
 DATETIME formats  
     SAS/ETS software 49  
 DB2 under OS/390, SAS/ACCESS interface 36  
 DB2 under UNIX and PC hosts, SAS/ACCESS  
     interface 36  
 Design Details window  
     ADX interface 39  
 directory services  
     SAS Integration Technologies 60  
 Distinct Values (NUNIQUE) window  
     SAS/EIS software 47  
 distributed objects  
     SAS Integration Technologies 60

## E

e-mail  
     sending with SMTP interface 5, 11  
     sending with SMTP interface, CMS 17  
     sending with SMTP interface, OS/390 29  
 Edit Scheme window 5  
 EFI (External File Interface) 5  
 Enterprise Reporter 84  
 European currency conversion 5  
     formats 8  
     informats 11  
 External File Interface (EFI) 5

## F

FACTOR procedure 78  
 file compression  
     *See* compressed files  
 FILENAME, EMAIL (SMTP) access method 5,  
     11  
     CMS environment 17  
     OS/390 environment 29  
 fonts 12  
 Forecast Command Builder 50  
 formats 8  
     binary 8  
     Date and Time 8  
     European currency conversion 8  
     Unicode 8  
 FREQ procedure 13, 78  
 functions 9  
     OS/2 environment 27  
     Windows environment 22

## G

GAM procedure 78  
 GANTT procedure 69  
 GCHART procedure 55  
 GCONTOUR procedure 55  
 GIS procedure  
     MAP ACTION statement 53  
 GMAP procedure 56  
 GPLOT procedure 55  
 GRADAR procedure 56  
 GREPLAY procedure 56

**I**

I/O enhancements 4  
 import design wizard  
   ADX interface 39  
 IMS-DL/I interface 37  
 informats 10  
   binary 11  
   European currency conversion 11  
   Unicode 10  
 Interior Point algorithm 70  
 INTPOINT procedure 70  
 Investment Analysis System 50

**J**

Java Tools 64

**L**

labels  
   for statistics (SAS/EIS) 48  
 large files  
   SAS/CONNECT software 46  
 LDAP security interface 83  
 LIFEREG procedure 79  
 limited dependent variables 50  
 LOESS procedure 79  
 LOGISTIC procedure 79

**M**

macro facility 11  
 macros  
   ARM macros 11  
   CMS environment 18  
   OS/390 environment 29  
   RSUBMIT and 45  
   Windows environment 22  
 MAP ACTION statement, GIS procedure 53  
 MDC procedure 50  
 MDDDB Report Viewer 64  
 MDDDB\_M class 65  
 message queuing  
   SAS Integration Technologies 60  
 metadata  
   SAS/ASSIST software 41  
 metadata API 83  
 MI procedure 80  
 MIANALYZE procedure 80  
 Microsoft SQL Server interface 36  
 MODEL procedure 50

**N**

new products  
   SAS/ACCESS Interface to PeopleSoft 89  
   SAS Data Quality - Cleanse software 91  
 NPARIWAY procedure 80  
 NUNIQUE columns  
   SAS/EIS software 47

**O**

ODBC interface 36  
 ODS (Output Delivery System) 3  
 OLAP Data Store 84  
 OLE DB interface 36  
 OpenVMS environment 25  
   commands 25  
   statements 25  
   system options 25  
 OPTLOAD procedure 13  
 OPTSAVE procedure 13  
 ORACLE interface 36  
 OS/2 environment 27  
   commands 27  
   functions and CALL routines 27  
   statements 27  
   system options 27  
 OS/390 environment 29  
   commands 29  
   macros 29  
   SAS language elements 29  
   sending e-mail with SMTP interface 29  
   statements 30  
   system options 30  
   Universal Printing 29  
 Output Delivery System (ODS) 3

**P**

PC File Formats interface 37  
 PeopleSoft 89  
 PeopleSoft interface 37  
 performance  
   SAS/ASSIST software 41  
   SAS/CONNECT software 46  
 pipelines  
   writing to SAS log, CMS 17  
 PM procedure 70  
 PRINT procedure 14  
 Printer Setup window 5  
 printing  
   fonts 12  
   Universal Printing 5  
   Universal Printing, CMS 17  
   Universal Printing, OS/390 29  
 PRINTTO procedure 14  
 PROBIT procedure 80  
 procedures 13  
   CMS environment 18  
 Process Library 84  
 process variables 39  
 PRTDEF procedure 14

**Q**

QLM procedure 50  
 qualitative choice variables 50

**R**

R/3 interface 37  
 REG procedure 81

REGISTRY procedure 14  
 RELIABILITY procedure 71  
 REPORT procedure 14  
 RSUBMIT 45  
   macro processing and 45

**S**

sample programs  
   SAS/ASSIST software 42  
 SAS/ACCESS Interface to PeopleSoft 89  
 SAS/ACCESS software for relational  
   databases 35  
   CA-DATACOM/DB 37  
   DB2 under OS/390 36  
   DB2 under UNIX and PC hosts 36  
   IMS-DL/I 37  
   Microsoft SQL Server 36  
   ODBC 36  
   OLE DB 36  
   ORACLE 36  
   PC File Formats 37  
   PeopleSoft 37  
   R/3 37  
   SYBASE 36  
   Teradata 37  
 SAS/ASSIST software 41  
   Data Sources window 41  
   metadata 41  
   performance 41  
   sample programs 42  
   Table/Column search 42  
 SAS Business Intelligence Portal 59  
 SAS Component Language 43  
 SAS/CONNECT software 45  
   Agent Services 46  
   CEDA (Cross-Environment Data Access) 46  
   Common Messaging Interface 46  
   file compression 46  
   large files 46  
   monitoring tasks 45  
   performance 46  
   RSUBMIT 45  
   RSUBMIT and macro processing 45  
   spawners 46  
 SAS Data Quality - Cleanse software 91  
 SAS Design-Time Controls 64  
 SAS/EIS software 47  
   access control, activating 48  
   Computed Values window 47  
   Distinct Values (NUNIQUE) window 47  
   sorting by unformatted category values 47  
   statistic labels 48  
 SAS/ETS software 49  
   ARIMA procedure 49  
   COUNTREG procedure 49  
   DATETIME formats 49  
   Investment Analysis System 50  
   MDC procedure 50  
   MODEL procedure 50  
   QLIM procedure 50  
   Time Series Forecasting System 50  
   VARMAX procedure 51  
   X12 procedure 51  
 SAS/GIS software 53  
   GIS procedure, MAP ACTION statement 53

- tutorial 54
  - SAS/GRAPH software 55
    - GCHART procedure 55
    - GCONTOUR procedure 55
    - GMAP procedure 56
    - GPLOT procedure 55
    - GRADAR procedure 56
    - GREPLAY procedure 56
    - Web graphs 56
  - SAS I/O 4
  - SAS/IML software 57
    - wavelet analysis 57
  - SAS Integration Technologies software 59
    - directory services 60
    - distributed objects 60
    - message queuing 60
    - SAS Business Intelligence Portal 59
    - SAS Package Retriever 60
    - SAS Publisher 61
    - SAS Publishing Framework 59
  - SAS/IntrNet software 63
    - Application Dispatcher 63
    - Java Tools 64
    - MDDB Report Viewer 64
    - SAS Design-Time Controls 64
    - Xplore sample application 64
  - SAS language elements 7
    - CMS environment 18
    - OpenVMS environment 25
    - OS/2 environment 27
    - OS/390 environment 29
    - UNIX environments 31
    - Windows environment 22
  - SAS log
    - writing to, from pipelines (CMS) 17
  - SAS/MDDB Server software 65
    - MDDB\_M class 65
  - SAS OLAP Server software 67
    - access control 67
    - building MDDBs 67
    - multiple environment support 67
  - SAS/OR software 69
    - BOM procedure 69
    - CPM procedure 69
    - GANTT procedure 69
    - INTPOINT procedure 70
    - PM procedure 70
  - SAS output 3
  - SAS Package Retriever 60
    - HTML file package entry 61
    - WebDAV 60
  - SAS Publisher 61
    - data set entry type 62
    - e-mail delivery type 62
    - ODS package entry type 62
    - passwords 61
    - URL package entry type 62
    - user IDs 61
    - WebDAV 61
  - SAS Publishing Framework 59
  - SAS/QC software 71
    - CAPABILITY procedure 71
    - RELIABILITY procedure 71
  - SHEWHART procedure 72
  - SAS resources
    - UNIX environments 32
  - SAS/SHARE software 75
    - services 75
  - SAS/STAT software 77
    - BOXPLOT procedure 77
    - CATMOD procedure 78
    - FACTOR procedure 78
    - FREQ procedure 78
    - GAM procedure 78
    - LIFEREG procedure 79
    - LOESS procedure 79
    - LOGISTIC procedure 79
    - MI procedure 80
    - MIANALYZE procedure 80
    - NPAR1WAY procedure 80
    - PROBIT procedure 80
    - REG procedure 81
    - SURVEYMEANS procedure 81
    - TRANSREG procedure 81
  - SAS System
    - operation in Windows 21
  - SAS/Warehouse Administrator software 83
    - Enterprise Reporter 84
    - LDAP security interface 83
    - metadata API 83
    - OLAP Data Store 84
    - Process Library 84
    - tutorial 85
  - SCL lists
    - initialization 43
  - search capabilities
    - SAS/ASSIST software 42
  - SHEWHART procedure 72
  - SMTP interface
    - sending e-mail 5, 11
    - sending e-mail, CMS 17
    - sending e-mail, OS/390 29
  - sorting
    - by unformatted category values (SAS/EIS) 47
  - spawners
    - SAS/CONNECT software 46
  - SQL procedure 15
  - SQL Server interface 36
  - statements 11
    - CMS environment 18
    - OpenVMS environment 25
    - OS/2 environment 27
    - OS/390 environment 30
    - UNIX environments 31
    - Windows environment 22
  - statistic labels
    - SAS/EIS software 48
  - Statistic Labels window
    - SAS/EIS software 48
  - SURVEYMEANS procedure 81
  - SYBASE interface 36
  - SYNCONFIG command 7
  - system options 12
    - CMS environment 18
    - OpenVMS environment 25
    - OS/2 environment 27
  - OS/390 environment 30
  - UNIX environments 31
  - Windows environment 22
- ## T
- Table/Column search
    - SAS/ASSIST software 42
  - TEMPLATE procedure 4
  - Teradata interface 37
  - Time Series Forecasting System 50
  - TRANSREG procedure 81
- ## U
- Unicode formats 8
  - Unicode informats 10
  - UNIVARIATE procedure 15
  - Universal Printing 5
    - CMS environment 17
    - OS/390 environment 29
  - Universally Unique Identifiers (UUIDs) 12
  - UNIX environments 31
    - commands 31
    - SAS language elements 31
    - SAS resources 32
    - statements 31
    - system options 31
  - UUIDs 12
- ## V
- VARMAX procedure 51
- ## W
- wavelet analysis 57
  - Web Distributed Authoring and Versioning
    - See WebDAV
  - Web graphs 56
  - WebDAV 60
    - SAS Package Retriever 60
    - SAS Publisher 61
  - windowing environment 5
  - Windows environment 21
    - basic operation of SAS System 21
    - commands 22
    - Enhanced Editor 21
    - functions and CALL routines 22
    - macros 22
    - SAS language elements 22
    - statements 22
    - system options 22
- ## X
- X12 procedure 51
  - Xplore sample application 64





# Your Turn

---

If you have comments or suggestions about *What's New in SAS® Software for Release 8.2*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute  
Publications Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [suggest@sas.com](mailto:suggest@sas.com)



Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido

## SAS Publishing Is Easy to Reach

Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)

You will find product and service details, including

- sample chapters
- tables of contents
- author biographies
- book reviews

Learn about

- regional user-group conferences
- trade-show sites and dates
- authoring opportunities
- custom textbooks

**Explore all the services that SAS Publishing has to offer!**

### Your Listserv Subscription Automatically Brings the News to You

Do you want to be among the first to learn about the latest books and services available from SAS Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS® release(s) each book addresses.

To subscribe,

1. Send an e-mail message to [listserv@vm.sas.com](mailto:listserv@vm.sas.com).
2. Leave the "Subject" line blank.
3. Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-531-7447. See the SAS Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-531-7447.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute Inc.  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.



---

SAS Publishing

# **SAS/ASSIST® Software: Changes and Enhancements, Release 8.2**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ASSIST® Software: Changes and Enhancements, Release 8.2*, Cary, NC: SAS Institute Inc., 2001.

**SAS/ASSIST® Software: Changes and Enhancements, Release 8.2**

Copyright © 2001 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-869-7

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2001

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

IBM® and all other International Business Machines Corporation product or service names are registered trademarks or trademarks of International Business Machines Corporation in the USA and other countries.

Oracle® and all other Oracle Corporation product or service names are registered trademarks or trademarks of Oracle Corporation in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

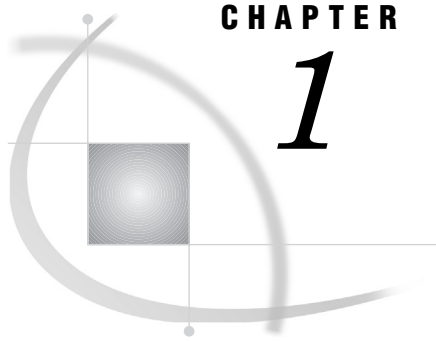
---

# Contents

|                                                                 |          |                                                   |          |
|-----------------------------------------------------------------|----------|---------------------------------------------------|----------|
| <b>Chapter 1</b>                                                | <b>△</b> | <b>Changes to the Workflow and User Interface</b> | <b>1</b> |
| Overview                                                        |          |                                                   | 1        |
| SAS/ASSIST Now Metadata Driven                                  |          |                                                   | 1        |
| Batch-Generated Metadata                                        |          |                                                   | 2        |
| Startup Sequence for a New User                                 |          |                                                   | 2        |
| Startup Sequence for an Existing User                           |          |                                                   | 4        |
| Changing the Settings for Metadata Creation in the User Profile |          |                                                   | 4        |
| Selecting a Table                                               |          |                                                   | 5        |
| Searching for Data                                              |          |                                                   | 6        |
| <b>Chapter 2</b>                                                | <b>△</b> | <b>Administrator Tasks — Creating Metadata</b>    | <b>9</b> |
| Overview                                                        |          |                                                   | 9        |
| Metadata Feature                                                |          |                                                   | 9        |
| Generating Metadata in Batch                                    |          |                                                   | 10       |
| Sample Metadata-Generating Program                              |          |                                                   | 11       |







## CHAPTER

## 1

# Changes to the Workflow and User Interface

---

|                                                                        |   |
|------------------------------------------------------------------------|---|
| <i>Overview</i>                                                        | 1 |
| <i>SAS/ASSIST Now Metadata Driven</i>                                  | 1 |
| <i>Batch-Generated Metadata</i>                                        | 2 |
| <i>Startup Sequence for a New User</i>                                 | 2 |
| <i>Data Sources Window</i>                                             | 2 |
| <i>Data Source Types</i>                                               | 2 |
| <i>The WORK library</i>                                                | 3 |
| <i>Optimizing Performance</i>                                          | 3 |
| <i>Creating Metadata</i>                                               | 3 |
| <i>Updating Metadata</i>                                               | 3 |
| <i>Startup Sequence for an Existing User</i>                           | 4 |
| <i>Start Window</i>                                                    | 4 |
| <i>Changing the Settings for Metadata Creation in the User Profile</i> | 4 |
| <i>Selecting a Table</i>                                               | 5 |
| <i>Searching for Data</i>                                              | 6 |
| <i>Search Using Table or Column Names</i>                              | 6 |

---

## Overview

In SAS/ASSIST software, Release 8.2, you can access your data in quicker, more efficient ways. Your administrator can set up batch programs to create metadata and make it available from within a SAS/ASSIST session. This can be customized globally for an entire site, or locally for a particular division or user. You can then control which data is accessible to your session when you use SAS/ASSIST. The existence of locally stored metadata now makes it possible for SAS/ASSIST to instantly display comprehensive selection lists, enabling users to search across all tables or columns.

## SAS/ASSIST Now Metadata Driven

In order to improve performance, SAS/ASSIST Software is now *metadata driven*. This means that, instead of examining each data source and extracting information each time you invoke SAS/ASSIST software, the software examines permanently saved, locally stored files that contain this information. This information, which includes table names, number of observations, column names and attributes, and more, is called *metadata*.

The advantages of the new method are

- ☐ An organized view of your data is always instantly available.
- ☐ All available tables or columns can be shown in a selection list.

- You can search for a table using a column name that is contained in the table.
- Performance when creating tasks based on remote data is improved.
- Performance when creating tasks based on data sources with many tables is improved.
- Irrelevant data sources can be excluded when creating metadata.

---

## Batch-Generated Metadata

Information about data sources you frequently use can be prepared for you through a batch job in advance, and made available to you when you invoke SAS/ASSIST. This enables your administrator to define a data environment that is custom built for your needs.

---

## Startup Sequence for a New User

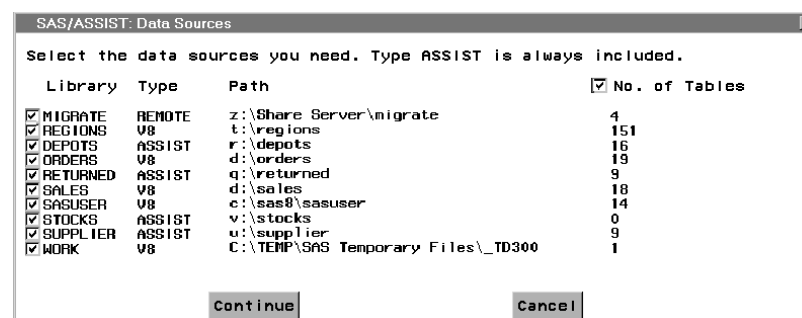
---

### Data Sources Window

When you invoke SAS/ASSIST for the first time, the Welcome window is displayed. When you select **Continue**, the program searches in libraries that are declared for the current session for all available data sources. The results of this search are displayed in the Data Sources window, as shown in the following display. This window can be accessed in subsequent sessions from the Start window, described in “Start Window” on page 4, or from within a session by selecting

**Tools** ► **Examine Data Sources**

**Display 1.1** Data Sources Window



By default, all the data source checkboxes are selected in the Data Sources window. When you select **Continue**, SAS/ASSIST examines these data sources and establishes metadata for all data it finds. Selecting only the data sources relevant to your project improves performance and helps you to better understand the data.

---

### Data Source Types

Data sources can contain metadata that was created for you in batch mode. The data source type **ASSIST** indicates that the metadata is batch generated. Not all data sources have batch-generated metadata.

Data sources can be the following types:

|               |                                                                                                                                                                          |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ASSIST</b> | Metadata was generated in batch for these data sources. The actual sources could be on any number of engines.                                                            |
| <b>REMOTE</b> | Library has been assigned using a libname statement with a Remote attribute. The source could be a SAS data source or any engine that SAS is able to access on a server. |
| Other         | Any SAS/ACCESS engine type may appear as a data type. This includes <b>V8</b> , <b>V6</b> , <b>Oracle</b> , <b>DB2</b> , <b>Sybase</b> , and <b>ODBC</b> .               |

---

## The WORK library

The **Library** list includes the **WORK** library. The metadata is automatically updated when you create temporary tables in the **WORK** library, even if you have cleared the check box for **WORK** in the Data Sources window.

---

## Optimizing Performance

The Data Sources window also contains a **No.of Tables** check box. Selecting this check box causes the Data Sources window to display the number of tables that are present in the data source. This gives you an indication of how long the process of metadata creation will take. Metadata creation time increases as the number of tables in a data source increases. Calculating the number of tables in a data source is a dynamic operation and can be time consuming.

---

## Creating Metadata

When you select Continue, SAS/ASSIST software examines the selected data sources and gathers information about the data, including table names, column names, and the location of the data. This information is saved in two tables, **\_ASTDCOL** and **\_ASTDTAB**, in the **SASUSER** library. The batch-generated metadata associated with the **ASSIST**-type data sources is also stored in the **SASUSER** library. See “Generating Metadata in Batch” on page 10 for more information on generating metadata in batch mode.

---

## Updating Metadata

Stored metadata can be reused in its current configuration, modified, updated at the beginning of a session, or refreshed from within a session. The parameters for these options can be set in the User Profile window, which is described in “Changing the Settings for Metadata Creation in the User Profile” on page 4.

When you select **Continue** in the Data Sources window, metadata is created and the Start window appears. For a description of the Start window, see “Start Window” on page 4.

---

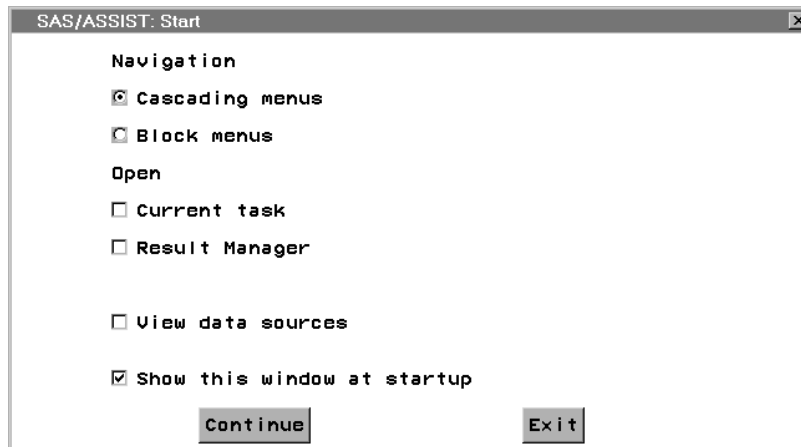
## Startup Sequence for an Existing User

---

### Start Window

When you invoke SAS/ASSIST software, the Start window is displayed, as shown in the following display.

Display 1.2 Start Window



You can choose between the cascading or block menus to navigate the software. If you select **Cascading menus**, select the **Current task** check box to open the task you were working with when you ended the previous session. Select the **Result Manager** check box to view the **Result Manager**. If you select **Block menus**, these two options are not available.

If you select the **View data sources** check box, the Data Sources window is displayed after you select [Continue](#). See “Data Sources Window” on page 2 for more information.

If you do not want the Start window to appear every time you invoke SAS/ASSIST software, clear the check box **Show this window at startup**. You can change the **Show startup dialog** option to **Yes** in the User Profile window. For more information about the User Profile window, see “Changing the Settings for Metadata Creation in the User Profile” on page 4.

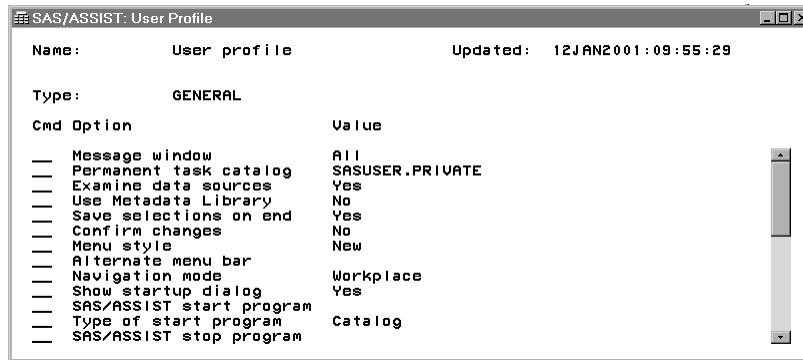
Select [Continue](#) to start your SAS/ASSIST session.

---

## Changing the Settings for Metadata Creation in the User Profile

---

You can open the User Profile window by selecting **User Profile** from the **Tools** menu on the main menu bar. The User Profile window appears as shown in the following display.

**Display 1.3**    User Profile    Window

In the User Profile window, you can change the value for different options that affect how you navigate and manage SAS/ASSIST software. The following is a description of selected options that relate to metadata management:

**Examine data sources:**

- ☐ **Yes:** Data sources are automatically examined each time you invoke SAS/ASSIST software. This is advantageous if your data structure is dynamic.
- ☐ **No:** Metadata that is saved in your SASUSER library is reused. This is advantageous to users whose data structure remains static.

*Note:* The raw data is not accessed at this stage, so any changes to the data itself are irrelevant for this process. 

**Use Metadata Library:**

- ☐ **Yes:** Metadata prepared in batch mode is copied and appended to metadata created at startup. The data sources associated with the batch-generated metadata are displayed in the Data Sources window as data type **ASSIST**. A library named METADATA must be assigned for the individual user, and it must point to the location where the metadata generated in the batch process is stored.
- ☐ **No:** Batch-generated metadata is not included. This option may be restricted to **Yes**, depending on the settings your administrator specifies.

**Show startup dialog:**

- ☐ **Yes:** The Start window displays when you invoke SAS/ASSIST software.
- ☐ **No:** The Start window does not display when you invoke SAS/ASSIST software.

*Note:* If this option is set to **No**, you can still view your data sources and update your metadata from within a session. To do so, select **Examine Data Sources** from the **Tools** menu. Modify the list of data sources, and select Continue to update your metadata. 

For information on other options listed in the User Profile window, see *SAS/ASSIST Software: Changes and Enhancements, Version 6*.

---

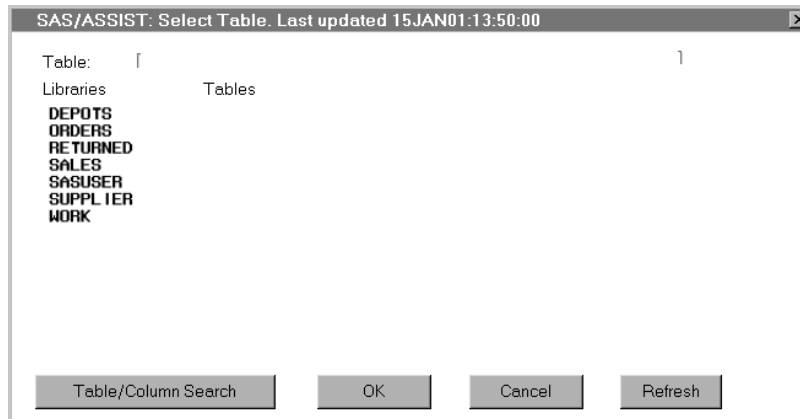
## Selecting a Table

To create a task, select the table that you want to use. For example, you can create a table listing by selecting

Tasks ► Report Writing ► Listing

The List a Table window appears. When you select **Table**, the Select Table window is displayed, as shown in the following display.

**Display 1.4** Select Table Window



By default, the **Table** field and **Tables** list in this window are empty. Select one of the libraries on the **Libraries** list to populate the **Tables** list. When you select a table, the table name is displayed in the **Table** field. When you select **OK**, the Select Table window closes and the selected table name appears in the List a Table window.

Select **Refresh** to open the Data Sources window. In this window, you can update your view of available data.

---

## Searching for Data

Searches are carried out using metadata stored locally in the SASUSER library. In SAS/ASSIST Software Release 8.2, searches using column names are also possible.

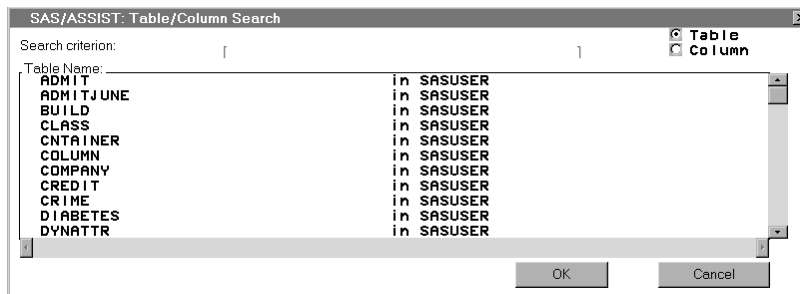
---

### Search Using Table or Column Names

In the Select Table window, select **Table/Column Search** to search using table or column names.

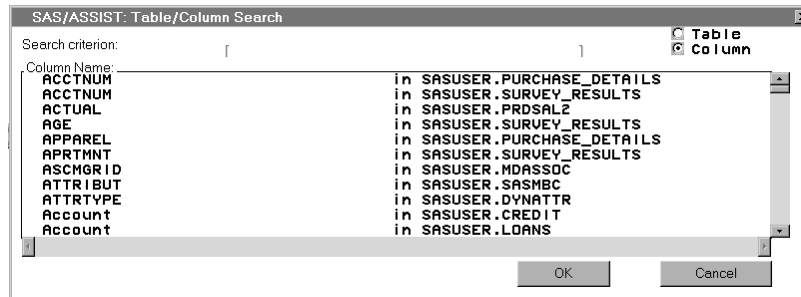
If you select **Table** in the Table/Column Search window, the window shown in the following display appears.

**Display 1.5** Table/Column Search Window



If you select the **Column** radio button in the Table/Column Search window, the window shown in the following display appears. The **Column Name** list contains a list of all the available columns and the table in which they are located.

**Display 1.6** Table/Column Search Window



The Table/Column Search window displays an alphabetical list of all the available tables or columns. Both table and column names starting with an uppercase character take a higher alphabetical precedence than those starting with a lowercase character, so **z** has a higher precedence than **a**.

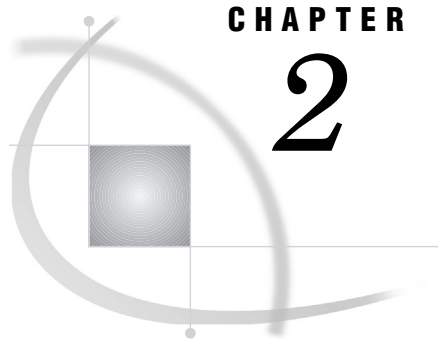
You can subset the **Table Name** and **Column Name** lists by entering a character (or string of characters) in the **Search Criterion** field. Subsetting results in a list of tables or columns whose initial characters match the search criterion characters. At this time, wildcard searches are not supported.

*Note:* You must select a table or column from the subset list, even if only one table or column is found. 

After you have selected a table or column, select **OK**. The Select Table window appears and, if you selected a table, the table name is displayed in the **Table** field.







## CHAPTER

## 2

# Administrator Tasks — Creating Metadata

|                                           |    |
|-------------------------------------------|----|
| <i>Overview</i>                           | 9  |
| <i>Metadata Feature</i>                   | 9  |
| <i>Generating Metadata in Batch</i>       | 10 |
| <i>Sample Metadata-Generating Program</i> | 11 |

## Overview

The way in which SAS/ASSIST software generates metadata and the way in which the metadata is made available to the user has been changed in order to optimize performance at runtime.

## Metadata Feature

In SAS/ASSIST software, Release 8.2, metadata is either generated at startup or generated prior to startup in batch mode. When users invoke the software, the batch-generated metadata is immediately available and can be appended to the metadata generated at startup by the user.

Metadata is created in batch mode by using an SQL program. A sample program is provided with SAS/ASSIST software, Release 8.2. It is located in the SASHELP library at SASHELP.QASSIST.METADICT.SOURCE. The batch program must also contain a list of LIBNAME statements. When a library is both defined in the batch program and assigned in a user session, metadata for data sources located in this library is made available to the user. Conversely, if a library is defined in the batch program but not assigned in an individual user session, data sources located in this library are not visible to this particular user.

The batch-generated metadata and the user-generated metadata are stored together in two SAS tables, \_ASTDCOL and \_ASTDTAB, which are automatically created in the SASUSER library the first time a user invokes the software.

The advantages of creating metadata in batch mode are

- Use of resources is reduced when users invoke SAS/ASSIST software: If metadata is created at startup by the individual user, the software accesses all available data in order to gather information. Creating metadata in batch mode avoids the need for each user to access remote hosts in order to examine the data.
- Metadata can be customized to suit the needs of individual users and groups: You can subset metadata generation in the sample program provided with Release 8.2 so that users only see metadata for the information they normally work with. This simplifies the process of searching for tables or columns.

- Processing time is significantly reduced when users update metadata during a session: Creating metadata at startup can be time consuming, especially when a large number of tables are accessed or when data resides on remote hosts. Batch-generated metadata, however, is stored in two tables in a shared location. These tables are appended to the user-generated metadata.

---

## Generating Metadata in Batch

The sample SQL metadata-generating program supplied with this release can be modified to suit the needs of individual organizations. This section explains how to modify the original program to suit your needs. The original program is included in “Sample Metadata-Generating Program” on page 11 of this document.

To create metadata:

- 1 Invoke SAS. Create a program defining the library in which metadata will be stored and the libraries that point to the underlying data sources. The METADATA library must be assigned the name METADATA.

```
libname metadata <shared location>;
libname library1 <data source 1>;
libname library2 <data source 2>;
...
```

Depending on which DBMS you are assigning libraries to, you may want to define additional LIBNAME options that are specific to that system.

- 2 Copy the sample metadata-generating program to your text editor in SAS. Issue the following command in the command line:

```
copy sashelp.qassist.metadict.source
```

- 3 Define the libraries to be included in the metadata. In the program code, you will find this line:

```
%let sellib=('INFORMIX', SASDATA);
```

Change the libraries in the brackets to the libraries you defined in the LIBNAME statements in step 1. For example, in the following code, the program is modified to include four libnames: DEPOTS, RETURNED, STOCKS, and SUPPLIER.

```
%let sellib=('DEPOTS,RETURNED,STOCKS,SUPPLIER');
```

- 4 Further modify the code to customize the task of metadata creation, as shown in the following code:

```
create table _astdcol as
  select *
  from dictionary.columns
  where (libname=DEPOTS and memname in (ROSSLAIGHRE,SHANNON,COVENTRY,LUTON))
  or (libname=SUPPLIER and memname in(MATERIALS,UTILITIES,PACKAGING)) or
  libname in (RETURNED,STOCKS)
  order by libname, memname, name;

create table _astdtab as
  select *
  from dictionary.tables
  where (libname=DEPOTS and memname in (ROSSLAIGHRE,SHANNON,COVENTRY,LUTON&))
  or (libname=SUPPLIER and memname in(MATERIALS,UTILITIES,PACKAGING)) or
  libname in (RETURNED,STOCKS)
```



```

/* Output library for query manager tables */

%let sellib=('INFORMIX','SASDATA');      /* <--- THIS MUST BE
CHANGED TO THE ACTUAL LIBNAMES */
/* Libraries to be excluded in Dictionary tables */
/* To select all: */
/* %let sellib=('*'); */

%macro upc(macvar,value);
  %global &macvar
  %let &macvar=%upcase(&value);
%mend;

/* upcase the value of selllib */
%upc(sellib,&sellib);
proc sql;
  create table work.__log
    (db2rc   char(5)  label='Return Code',
     db2msg1 char(70) label='Message-1',
     db2msg2 char(70) label='Message-2',
     db2msg3 char(70) label='Message-3',
     db2msg4 char(70) label='Message-4');
  quit;

proc sql;
  drop table _astdcol;
  drop table _astdtab;

  create table _astdcol as
  select *
  from dictionary.columns
  where libname in &sellib
  order by libname, memname, name;

  create table _astdtab as
  select *
  from dictionary.tables
  where libname in &sellib
  order by libname, memname;

  alter table _astdtab
    modify crdate format=DATETIME19.,
    modate format=DATETIME19.
  ;

/* data &dictlib._astdtab(index=(libname memname) */
data &dictlib._astdtab(label="ASSIST Dictionary tables: Table info");

  set _astdtab;
run;

/*data &dictlib._astdcol(index=(libname memname name f_lib f_mem) compress=NO */
data &dictlib._astdcol(compress=NO label="ASSIST Dictionary tables: Column info");

```

```
set _astdcol;
attrib
index length= $1 format=$1. label="Index"
f_lib length= $1 format=$1. label="First library"
f_mem length= $1 format=$1. label="First member"
;
by libname memname;
f_lib=put(first.libname,1.);
f_mem=put(first.memname,1.);
if idxusage eq ' ' then index=' ';
else index='*';
if label=' ' then label=name;
run;
```



# Your Turn

---

If you have comments or suggestions about *SAS/ASSIST® Software: Changes and Enhancements, Release 8.2*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing  
SAS Campus Drive  
Cary, NC 27513  
**email:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute Inc.  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**email:** [suggest@sas.com](mailto:suggest@sas.com)





*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS® release(s) each book addresses.

To subscribe,

- 1.** Send an e-mail message to **listserv@vm.sas.com**.
- 2.** Leave the "Subject" line blank.
- 3.** Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-531-7447. See the SAS Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-531-7447.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 20% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute Inc.  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS business, call 919-677-8000\*

**\* Note:** Customers outside the U.S. should contact their local SAS office.



SAS Publishing

TECHNICAL REPORT

# SAS/ETS® Software: Changes and Enhancements, Release 8.2



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ETS® Software: Changes and Enhancements, Release 8.2*, Cary, NC: SAS Institute Inc., 2001

**SAS/ETS® Software: Changes and Enhancements, Release 8.2**  
Copyright © 2001 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-849-2

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2001

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

|                                                    |                |
|----------------------------------------------------|----------------|
| <b>Part 1. General Information</b>                 | <b>1</b>       |
| Chapter 1. The SASECRSP Interface Engine . . . . . | 3              |
| Chapter 2. The SASEFAME Interface Engine . . . . . | 29             |
| <br><b>Part 2. Procedure Reference</b>             | <br><b>51</b>  |
| Chapter 3. The ARIMA Procedure . . . . .           | 53             |
| Chapter 4. The MODEL Procedure . . . . .           | 67             |
| Chapter 5. The VARMAX Procedure . . . . .          | 91             |
| Chapter 6. The X12 Procedure . . . . .             | 97             |
| <br><b>Subject Index</b>                           | <br><b>109</b> |
| <br><b>Syntax Index</b>                            | <br><b>111</b> |



# Part 1

## General Information

### Table of Contents

---

|                                                    |    |
|----------------------------------------------------|----|
| Chapter 1. The SASECRSP Interface Engine . . . . . | 3  |
| Chapter 2. The SASEFAME Interface Engine . . . . . | 29 |

*Part 1. General Information*



# Chapter 1

## The SASECRSP Interface Engine

### Chapter Table of Contents

---

|                                                                                |    |
|--------------------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                                      | 5  |
| <b>GETTING STARTED</b> . . . . .                                               | 5  |
| Structure of a SAS Data Set Containing Time Series Data . . . . .              | 5  |
| Reading CRSP Data Files . . . . .                                              | 5  |
| Using the SAS DATA Step . . . . .                                              | 6  |
| Using SAS Procedures . . . . .                                                 | 6  |
| Using CRSP Date Formats, Informats, and Functions . . . . .                    | 6  |
| <b>SYNTAX</b> . . . . .                                                        | 6  |
| The LIBNAME <i>libref</i> SASECRSP Statement . . . . .                         | 7  |
| <b>DETAILS</b> . . . . .                                                       | 9  |
| The SAS Output Data Set . . . . .                                              | 9  |
| Available Data Sets . . . . .                                                  | 9  |
| Understanding CRSP Date Formats, Informats, and Functions . . . . .            | 14 |
| <b>EXAMPLES</b> . . . . .                                                      | 18 |
| Example 1.1 Extracting a List of PERMNOs Using a RANGE . . . . .               | 18 |
| Example 1.2 Extracting All PERMNOs Using a RANGE . . . . .                     | 20 |
| Example 1.3 Extracting One PERMNO Using No RANGE, Wide Open . . . . .          | 22 |
| Example 1.4 Extracting Selected PERMNOs in Range Using INSET= Option . . . . . | 24 |
| Example 1.5 Converting Dates Using the CRSP Date Functions . . . . .           | 26 |
| <b>REFERENCES</b> . . . . .                                                    | 28 |

*Part 1. General Information*

# Chapter 1

## The SASECRSP Interface Engine

---

### Overview

The SASECRSP interface engine enables SAS users to access and process time series data residing in CRSPAccess data files and provides a seamless interface between CRSP and SAS data processing.

The SASECRSP engine uses the LIBNAME statement to enable you to specify which time series you would like to read from the CRSPAccess data files, and how you would like to perform selection on the CRSP set you choose to access. You choose the daily CRSP data by specifying SETID=10, or the monthly CRSP data by specifying SETID=20. You can select which securities you wish to access by specifying PERMNO=*number*, and specify your range of dates for the selected time series by using RANGE=*'begdt-enddt'*, or specify an input SAS data set named *setname* as input for issues with the INSET=*'setname'* option. The SAS Data step can then be used to perform further subsetting and to store the resulting time series into a SAS data set. You can perform more analysis if desired either in the same SAS session or in another session at a later time. Since CRSP and SAS use three different date representations, you can use the engine-provided CRSP date formats, informats, and functions for flexible seamless programming.

---

### Getting Started

---

#### Structure of a SAS Data Set Containing Time Series Data

SAS requires time series data to be in a specific form that is recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as a time ID variable. The SASECRSP engine provides a time ID variable called CALDT.

---

#### Reading CRSP Data Files

The SASECRSP engine supports reading time series, events, and portfolios from CRSPAccess data files. Only the series specified by the CRSP *setid* is written to the SAS data set. The CRSP environment variable CRSPDB\_SASCAL must be defined to allow the SASECRSP engine access to the CRSPAccess database calendars.

---

## Using the SAS DATA Step

If desired, you can store the selected series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. Once the data is stored in a SAS data set you can use it as you would any other SAS data set.

---

## Using SAS Procedures

You can print the output SAS data set by using the PRINT procedure and report information concerning the contents of your data set by using the CONTENTS procedure. You can create a view of the CRSPAccess data base by using the SQL procedure to create your view using the SASECRSP engine in your *libref*, along with the USING clause.

---

## Using CRSP Date Formats, Informats, and Functions

CRSP has historically used two different methods to represent dates, and SAS has used a third. The SASECRSP engine provides 22 functions, 15 informats, and 10 formats to enable you to easily translate the dates from one internal representation to another. See the section “Understanding CRSP Date Formats, Informats, and Functions” on page 14 for details.

---

## Syntax

The SASECRSP engine uses standard engine syntax. Options used by SASECRSP are summarized in the table below.

| Description                                                                                      | Statement                      | Option  |
|--------------------------------------------------------------------------------------------------|--------------------------------|---------|
| specify a CRSPAccess set id where 10 is daily and 20 monthly. This is a required option.         | LIBNAME <i>libref</i> SASECRSP | SETID=  |
| specify the PERMNO of a security to be kept in the SAS data set.                                 | LIBNAME <i>libref</i> SASECRSP | PERMNO= |
| specify the range of data in format YYYYMMDD using the <i>crsp_begdt</i> and <i>crsp_enddt</i> . | LIBNAME <i>libref</i> SASECRSP | RANGE=  |
| use a SAS data set named <i>setname</i> as input for issues.                                     | LIBNAME <i>libref</i> SASECRSP | INSET=  |

---

## The LIBNAME *libref* SASECRSP Statement

**LIBNAME** *libref* **SASECRSP** '*physical name*' *options*;

The CRSP environment variable CRSPDB\_SASCAL must be defined to allow the SASECRSP engine access to the CRSPAccess database calendars. Often your CRSPDB\_SASCAL will be pointing to the same path as your *physical name*. You must use a fully qualified pathname. Your *physical name* must end with either a forward slash if you are on a UNIX system, or a backward slash if you are on WINNT.

For a more complete description of SASECRSP set names, key fields, and date fields, see Table 1.1.

The following options can be used in the LIBNAME *libref* SASECRSP statement:

### **SETID=***crsp\_setidnumber*

specifies the CRSP set ID number where 10 is the daily data set ID and 20 is the monthly data set ID number. Two possible values for *crsp\_setidnumber* are 10 or 20. The SETID limits the frequency selection of time series that are included in the SAS data set. SETID is a required option for the SASECRSP engine. Depending on your host system, both should end in either a forward slash (UNIX) or a backward slash (WINNT). For more details about the CRSPAccess *crsp\_setidnumber*, refer to “Using Multiple Products” in the *CRSPAccess Database Format Installation Guide*. As an example, to access monthly data, you would use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'
SETID=20;
```

### **PERMNO=***crsp\_permnumber*

By default, the SASECRSP engine reads all PERMNOs in the CRSPAccess database that you name in your SASECRSP *libref*. You can limit the time series read from the CRSP database by specifying the PERMNO= option on your LIBNAME statement. From a performance standpoint, the PERMNO= option does random access and reads only the data for the PERMNOs listed. There is no limit to the number of *crsp\_permnumber* options that you can use. As an example, to access monthly data for Microsoft Corporation and for International Business Machine Corporation, you could use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'
SETID=20
PERMNO=10107
PERMNO=12490;
```

### **RANGE=***'crsp\_begdt-crsp\_enddt'*

To limit the time range of data read from the CRSPAccess database, specify the RANGE= option in your SASECRSP *libref*, where *crsp\_begdt* is the beginning date in YYYYMMDD format and *crsp\_enddt* is the ending date of the range in

## Part 1. General Information

YYYYMMDD format. From a performance standpoint, the engine reads all the data for a company and then restricts the data range before passing it back.

As an example, to access monthly data for Microsoft Corporation and for International Business Machine Corporation for the first quarter of 2000, you could use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'
SETID=20
PERMNO=10107
PERMNO=12490
RANGE='19990101-19990331';
```

**INSET**=*'setname[,keyfieldname][,keyfieldtype][,date1field][,date2field]'*

When you specify a SAS data set named *setname* as input for issues, the SASECRSP engine assumes a default PERMNO field containing selected CRSP PERMNOs is present in the data set. If optional parameters are included, the first one is the CRSPAccess supported key name, and the second is the key type. The third optional parameter is the beginning date or event date, and the fourth is the ending date in YYYYMMDD format. For more details about the CRSPAccess keyfields and keyfieldtypes, refer to “Stock and Indices Data Structures” in the *CRSP Data Description Guide*.

The following example might be used to extract the data for a portfolio of four companies to access monthly data where the range of dates is left open:

```
data testin1;
  permno = 10107; output;
  permno = 12490; output;
  permno = 14322; output;
  permno = 25788; output;
run;

LIBNAME mstk sasecrsp 'physical-name'
SETID=20
INSET='testin1';

data a;
  set mstk.prc;
run;

proc print data=a;
run;
```

Suppose you want to restrict the dates for each PERMNO specifically. The example below shows how to specify the INSET= option with the 'setname[,keyfieldname][[,keyfieldtype][[,date1field][[,date2field]]' parameters:

```
data testin2;
  permno = 10107; date1 = 19990101; date2 = 19991231; output;
  permno = 12490; date1 = 19970101; date2 = 19971231; output;
  permno = 14322; date1 = 19990901; date2 = 19991231; output;
  permno = 25788; date1 = 19950101; date2 = 19950331; output;
run;

LIBNAME mstk2 sasecrsp 'physical-name'
SETID=20
INSET='testin2,PERMNO,PERMNO,DATE1,DATE2';

data b;
  set mstk2.prc;
run;

proc print data=b;
run;
```

---

## Details

---

### The SAS Output Data Set

You can use the SAS DATA step to write the selected CRSP data to a SAS data set. This enables the user to easily analyze the data using SAS. The name of the output data set is specified by you on the DATA statement. This causes the engine supervisor to create a SAS data set using the specified name in either the SAS WORK library or, if specified, the USER library.

The contents of the SAS data set include the DATE of each observation, the series names of each series read from the CRSPAccess database, event variables, and the label or description of each series/event.

---

### Available Data Sets

Table 1.1 shows the available data sets provided by the SASECRSP interface. Missing values are represented as '.' in the SAS data set. You can see the available data sets in the SAS LIBNAME window of the SAS Display Manager by selecting the SASECRSP libref in the LIBNAME window that you have previously used in your libname statement. You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. You can view your SAS output observations by double clicking on the desired output data set libref in the libname window of the Display Manager. You can use PROC SQL along with the SASECRSP engine to create a view of your SAS data set.

Part 1. General Information

**Table 1.1.** Data Sets Available

| Dataset                                              | Fields    | Label                                      | Type      |
|------------------------------------------------------|-----------|--------------------------------------------|-----------|
| STKHEAD<br>Header Identification<br>and Summary Data | PERMNO    | PERMNO                                     | Numeric   |
|                                                      | PERMCO    | PERMCO                                     | Numeric   |
|                                                      | COMPNO    | Nasdaq Company Number                      | Numeric   |
|                                                      | ISSUNO    | Nasdaq Issue Number                        | Numeric   |
|                                                      | HEXCD     | Exchange Code Header                       | Numeric   |
|                                                      | HSHRCD    | Share Code Header                          | Numeric   |
|                                                      | HSICCD    | Standard Industrial<br>Classification Code | Numeric   |
|                                                      | BEGDT     | Begin of Stock Data                        | Numeric   |
|                                                      | ENDDT     | End of Stock Data                          | Numeric   |
|                                                      | DLSTCD    | Delisting Code Header                      | Numeric   |
|                                                      | HCUSIP    | CUSIP Header                               | Character |
|                                                      | HTICK     | Ticker Symbol Header                       | Character |
|                                                      | HCOMNAM   | Company Name Header                        | Character |
| NAMES<br>Name History Array                          | PERMNO    | PERMNO                                     | Numeric   |
|                                                      | NAMEDT    | Names Date                                 | Numeric   |
|                                                      | NAMEENDDT | Names Ending Date                          | Numeric   |
|                                                      | SHRCD     | Share Code                                 | Numeric   |
|                                                      | EXCHCD    | Exchange Code                              | Numeric   |
|                                                      | SICCD     | Standard Industrial<br>Classification Code | Numeric   |
|                                                      | NCUSIP    | CUSIP                                      | Numeric   |
|                                                      | TICKER    | Ticker Symbol                              | Character |
|                                                      | COMNAM    | Company Name                               | Character |
|                                                      | SHRCLS    | Share Class                                | Numeric   |
| DISTS<br>Distribution Event<br>Array                 | PERMNO    | PERMNO                                     | Numeric   |
|                                                      | DISTCD    | Distribution Code                          | Numeric   |
|                                                      | DIVAMT    | Dividend Cash Amount                       | Numeric   |
|                                                      | FACPR     | Factor to Adjust Price                     | Numeric   |
|                                                      | FACSHR    | Factor to Adjust Share                     | Numeric   |
|                                                      | DCLRDT    | Distribution<br>Declaration Date           | Numeric   |
|                                                      | EXDT      | Ex-Distribution Date                       | Numeric   |
|                                                      | RCRDDT    | Record Date                                | Numeric   |
|                                                      | PAYDT     | Payment Date                               | Numeric   |
|                                                      | ACPERM    | Acquiring PERMNO                           | Numeric   |
|                                                      | ACCOMP    | Acquiring PERMCO                           | Numeric   |
| SHARES<br>Shares Outstanding<br>Observation Array    | PERMNO    | PERMNO                                     | Numeric   |
|                                                      | SHROUT    | Shares Outstanding                         | Numeric   |
|                                                      | SHRSDT    | Shares Outstanding<br>Observation Date     | Numeric   |
|                                                      | SHRENDT   | Shares Outstanding<br>Observation End Date | Numeric   |
|                                                      | SHRFLG    | Shares Outstanding<br>Observation Flag     | Numeric   |
|                                                      |           |                                            |           |



Table 1.1. (continued)

| Dataset                                             | Fields    | Label                                 | Type    |
|-----------------------------------------------------|-----------|---------------------------------------|---------|
| DELIST<br>Delisting<br>History Array                | PERMNO    | PERMNO                                | Numeric |
|                                                     | DLSTDT    | Delisting Date                        | Numeric |
|                                                     | DLSTCD    | Delisting Code                        | Numeric |
|                                                     | NWPERM    | New PERMNO                            | Numeric |
|                                                     | NWCOMP    | New PERMCO                            | Numeric |
|                                                     | NEXTDT    | Delisting<br>Next Price Date          | Numeric |
|                                                     | DLAMT     | Delisting Amount                      | Numeric |
|                                                     | DLRETX    | Delisting Return<br>Without Dividends | Numeric |
|                                                     | DLPRC     | Delisting Price                       | Numeric |
|                                                     | DLPDT     | Delisting Amount Date                 | Numeric |
|                                                     | DLRET     | Delisting Return                      | Numeric |
| NASDIN<br>Nasdaq Information<br>Array               | PERMNO    | PERMNO                                | Numeric |
|                                                     | TRTSCD    | Nasdaq Traits Code                    | Numeric |
|                                                     | TRTSDT    | Nasdaq Traits Date                    | Numeric |
|                                                     | TRTSENDDT | Nasdaq Traits End Date                | Numeric |
|                                                     | NMSIND    | Nasdaq National Market<br>Indicator   | Numeric |
|                                                     | MMCNT     | Market Maker Count                    | Numeric |
| PRC<br>Price or Bid/Ask<br>Average Time Series      | NSDINX    | Nasd Index Code                       | Numeric |
|                                                     | PERMNO    | PERMNO                                | Numeric |
|                                                     | CALDT     | Calendar Trading Date                 | Numeric |
| RET<br>Returns Time Series                          | PRC       | Price or Bid/Ask Aver                 | Numeric |
|                                                     | PERMNO    | PERMNO                                | Numeric |
|                                                     | CALDT     | Calendar Trading Date                 | Numeric |
| ASKHI<br>Ask or High<br>Time Series                 | RET       | Returns                               | Numeric |
|                                                     | PERMNO    | PERMNO                                | Numeric |
|                                                     | CALDT     | Calendar Trading Date                 | Numeric |
| BIDLO<br>Bid or Low<br>Time Series                  | ASKHI     | Ask or High                           | Numeric |
|                                                     | PERMNO    | PERMNO                                | Numeric |
|                                                     | CALDT     | Calendar Trading Date                 | Numeric |
| BID<br>Bid Time Series                              | BIDLO     | Bid or Low                            | Numeric |
|                                                     | PERMNO    | PERMNO                                | Numeric |
|                                                     | CALDT     | Calendar Trading Date                 | Numeric |
| ASK<br>Ask Time Series                              | BID       | Bid                                   | Numeric |
|                                                     | PERMNO    | PERMNO                                | Numeric |
|                                                     | CALDT     | Calendar Trading Date                 | Numeric |
| RETX<br>Returns without<br>Dividends Time Series    | ASK       | Ask                                   | Numeric |
|                                                     | PERMNO    | PERMNO                                | Numeric |
|                                                     | CALDT     | Calendar Trading Date                 | Numeric |
| SPREAD<br>Spread Between Bid<br>and Ask Time Series | RETX      | Returns w/o Dividends                 | Numeric |
|                                                     | PERMNO    | PERMNO                                | Numeric |
|                                                     | CALDT     | Calendar Trading Date                 | Numeric |
|                                                     | SPREAD    | Spread Between Bid Ask                | Numeric |

Part 1. General Information

**Table 1.1.** (continued)

| Dataset              | Fields   | Label                 | Type    |
|----------------------|----------|-----------------------|---------|
| ALTPRC               | PERMNO   | PERMNO                | Numeric |
| Price Alternate      | CALDT    | Calendar Trading Date | Numeric |
| Time Series          | ALTPRC   | Price Alternate       | Numeric |
| VOL                  | PERMNO   | PERMNO                | Numeric |
| Volume Time Series   | CALDT    | Calendar Trading Date | Numeric |
|                      | VOL      | Volume                | Numeric |
| NUMTRD               | PERMNO   | PERMNO                | Numeric |
| Number of Trades     | CALDT    | Calendar Trading Date | Numeric |
| Time Series          | NUMTRD   | Number of Trades      | Numeric |
| ALTPRCDT             | PERMNO   | PERMNO                | Numeric |
| Alternate Price Date | CALDT    | Calendar Trading Date | Numeric |
| Time Series          | ALTPRCDT | Alternate Price Date  | Numeric |
| PORT1                | PERMNO   | PERMNO                | Numeric |
| Portfolio Data for   | CALDT    | Calendar Trading Date | Numeric |
| Portfolio Type 1     | PORT1    | Portfolio Assignment  | Numeric |
|                      |          | for Portfolio Type 1  | Numeric |
|                      | STAT1    | Portfolio Statistic   | Numeric |
|                      |          | for Portfolio Type 1  | Numeric |
| PORT2                | PERMNO   | PERMNO                | Numeric |
| Portfolio Data for   | CALDT    | Calendar Trading Date | Numeric |
| Portfolio Type 2     | PORT2    | Portfolio Assignment  | Numeric |
|                      |          | for Portfolio Type 2  | Numeric |
|                      | STAT2    | Portfolio Statistic   | Numeric |
|                      |          | for Portfolio Type 2  | Numeric |
| PORT3                | PERMNO   | PERMNO                | Numeric |
| Portfolio Data for   | CALDT    | Calendar Trading Date | Numeric |
| Portfolio Type 3     | PORT3    | Portfolio Assignment  | Numeric |
|                      |          | for Portfolio Type 3  | Numeric |
|                      | STAT3    | Portfolio Statistic   | Numeric |
|                      |          | for Portfolio Type 3  | Numeric |
| PORT4                | PERMNO   | PERMNO                | Numeric |
| Portfolio Data for   | CALDT    | Calendar Trading Date | Numeric |
| Portfolio Type 4     | PORT4    | Portfolio Assignment  | Numeric |
|                      |          | for Portfolio Type 4  | Numeric |
|                      | STAT4    | Portfolio Statistic   | Numeric |
|                      |          | for Portfolio Type 4  | Numeric |
| PORT5                | PERMNO   | PERMNO                | Numeric |
| Portfolio Data for   | CALDT    | Calendar Trading Date | Numeric |
| Portfolio Type 5     | PORT5    | Portfolio Assignment  | Numeric |
|                      |          | for Portfolio Type 5  | Numeric |
|                      | STAT5    | Portfolio Statistic   | Numeric |
|                      |          | for Portfolio Type 5  | Numeric |
| PORT6                | PERMNO   | PERMNO                | Numeric |
| Portfolio Data for   | CALDT    | Calendar Trading Date | Numeric |
| Portfolio Type 6     | PORT6    | Portfolio Assignment  | Numeric |

**Table 1.1.** (continued)

| Dataset                                         | Fields                                | Label                                                                                                                          | Type                                                           |
|-------------------------------------------------|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
|                                                 | STAT6                                 | for Portfolio Type 6<br>Portfolio Statistic<br>for Portfolio Type 6                                                            | Numeric<br>Numeric<br>Numeric                                  |
| PORT7<br>Portfolio Data for<br>Portfolio Type 7 | PERMNO<br>CALDT<br>PORT7<br><br>STAT7 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 7<br>Portfolio Statistic<br>for Portfolio Type 7 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |
| PORT8<br>Portfolio Data for<br>Portfolio Type 8 | PERMNO<br>CALDT<br>PORT8<br><br>STAT8 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 8<br>Portfolio Statistic<br>for Portfolio Type 8 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |
| PORT9<br>Portfolio Data for<br>Portfolio Type 9 | PERMNO<br>CALDT<br>PORT9<br><br>STAT9 | PERMNO<br>Calendar Trading Date<br>Portfolio Assignment<br>for Portfolio Type 9<br>Portfolio Statistic<br>for Portfolio Type 9 | Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric<br>Numeric |

## Understanding CRSP Date Formats, Informats, and Functions

CRSP has historically used two different methods to represent dates, while SAS has used a third. The three formats are SAS Dates, CRSP Dates, and Integer Dates. The SASECRSP engine provides 22 functions, 15 informats, and 10 formats to enable you to easily translate the dates from one internal representation to another. A SASECRSP libname assign must be active to use these date access methods.

SAS dates are internally stored as the number of days since January 1, 1960. The SAS method is an industry standard and provides a great deal of flexibility, including a wide variety of informats, formats, and functions.

CRSP dates are designed to ease time series storage and access. Internally the dates are stored as an offset into an array of trading days. Note that there are five different CRSP trading day calendars: Annual, Quarterly, Monthly, Weekly, and Daily. The CRSP method provides fewer missing values and makes trading period calculations very easy. However, there are also many valid calendar dates that are not available in the CRSP trading calendars, and care must be taken when using other dates.

Integer dates are a way to represent dates that are platform independent and maintain the correct sort order. However, the distance between dates is not maintained.

The best way to illustrate these formats is with some sample data. The table below only shows CRSP Daily and Monthly dates.

**Table 1.2.** Date Representations for Daily and Monthly Data

| Date              | SAS Date | CRSP Date<br>(Daily) | CRSP Date<br>(Monthly) | Integer Date |
|-------------------|----------|----------------------|------------------------|--------------|
| July 31, 1962     | 942      | 21                   | 440                    | 19620731     |
| August 31, 1962   | 973      | 44                   | 441                    | 19620831     |
| December 30, 1998 | 14,243   | 9190                 | NA*                    | 19981230     |
| December 31, 1998 | 14,244   | 9191                 | 877                    | 19981231     |

\* Not available if an exact match is requested.

Having an understanding of the internal differences in representing SAS dates, CRSP dates, and CRSP integer dates will help you use the SASECRSP formats, informats, and functions effectively. Always keep in mind the frequency of the CRSP calendar that you are accessing when you specify a CRSP date.

### The CRSP Date Formats

There are two types of formats for CRSP dates, and five frequencies are available for each of the two types. The two types are exact dates (CRSPDT\*) and range dates (CRSPDR\*), where \* can be A for annual, Q for quarterly, M for monthly, W for weekly, or D for daily. The ten formats are: CRSPDTA, CRSPDTQ, CRSPDTM, CRSPDTW, CRSPDTD, CRSPDRA, CRSPDRQ, CRSPDRM, CRSPDRW, and CRSPDRD.

Here are some samples using the monthly and daily calendar as examples. The Annual (CRSPDTA and CRSPDRA), Quarterly (CRSPDTQ and CRSPDRQ), and the Weekly (CRSPDTW and CRSPDRW) work analogously.

**Table 1.3.** Sample CRSPDT Formats for Daily and Monthly Data

| Date                | CRSP Date<br>Daily ,<br>Monthly | CRSPDTD8.<br>Daily Date | CRSPDRD8.<br>Daily<br>Range | CRSPDTM8.<br>Monthly<br>Date | CRSPDRM8.<br>Monthly Range |
|---------------------|---------------------------------|-------------------------|-----------------------------|------------------------------|----------------------------|
| July 31,1962        | 21 , 440                        | 19620731                | 19620731 +                  | 19620731                     | 19620630-<br>19620731      |
| August31,1962       | 44 , 441                        | 19620831                | 19620831 +                  | 19620831                     | 19620801-<br>19620831      |
| December30,<br>1998 | 9190 , NA *                     | 19981230                | 19981230 +                  | NA*                          | NA*                        |
| December31,<br>1998 | 9191 , 877                      | 19981231                | 19981231 +                  | 19981231                     | 19981201-<br>19981231      |

+ Daily ranges will look similar to Monthly Ranges if they are Mondays or immediately following a trading holiday.

\* When working with exact matches, no CRSP monthly date exists for December 30, 1998.

### ***The @CRSP Date Informats***

There are three types of informats for CRSP dates, and five frequencies are available for each of the three types. The three types are exact (@CRSPDT\*), range (@CRSPDR\*), and backward (@CRSPDB\*) dates, where \* can be A for annual, Q for quarterly, M for monthly, W for weekly, or D for daily. The fifteen formats are: @CRSPDTA, @CRSPDTQ, @CRSPDTM, @CRSPDTW, @CRSPDTD, @CRSPDRA, @CRSPDRQ, @CRSPDRM, @CRSPDRW, @CRSPDRD, @CRSPDBA, @CRSPDBQ, @CRSPDBM, @CRSPDBW, and @CRSPDBD.

The five CRSPDT\* informats find exact matches only. The five CRSPDR\* informats look for an exact match, and if it is not found it goes forward, matching the CRSPDR\* formats. The five CRSPDB\* informats look for an exact match, and if it is not found it goes backward. Here is a sample using only the CRSP monthly calendar as an example, but the daily, weekly, quarterly, and annual frequencies work analogously.

## Part 1. General Information

**Table 1.4.** Sample @CRSP Date Informat Using Monthly Data

| Input Date<br>(Integer Date) | CRSP Date<br>CRSPDTM | CRSP Date<br>CRSPDRM | CRSP Date<br>CRSPDBM | CRSPDTM8.<br>Monthly<br>Date | CRSPDRM8.<br>Monthly<br>Range |
|------------------------------|----------------------|----------------------|----------------------|------------------------------|-------------------------------|
| 19620731                     | 440                  | 440                  | 440                  | 19620731                     | 19620630-<br>19620731         |
| 19620815                     | .(missing)           | 441                  | 440                  | See below+                   | See below*                    |
| 19620831                     | 441                  | 441                  | 441                  | 19620831                     | 19620801-<br>19620831         |

+ If missing, then missing. If 441, then 19620831. If 440, then 19620731.

\* If missing, then missing. If 441, then 19620801-19620831. If 440, then 19620630-19620731.

### The CRSP Date Functions

There are 22 date functions provided with the SASECRSP engine. These functions are used internally by the engine, but also are available to the end users. There are six groups of functions. The first four have five functions each, one for each CRSP calendar frequency.

**Table 1.5.** CRSP Date Functions

| Function Group                                                                          | Function Name | Argument One | Argument Two | Return Value |
|-----------------------------------------------------------------------------------------|---------------|--------------|--------------|--------------|
| <b>CRSP dates to Integer dates for December 31, 1998</b>                                |               |              |              |              |
| Annual                                                                                  | crspdcia      | 74           | None         | 19981231     |
| Quarterly                                                                               | crspdcicq     | 293          | None         | 19981231     |
| Monthly                                                                                 | crspdcim      | 877          | None         | 19981231     |
| Weekly                                                                                  | crspdcicw     | 1905         | None         | 19981231     |
| Daily                                                                                   | crspdcid      | 9191         | None         | 19981231     |
| <b>CRSP dates to SAS dates for December 31, 1998</b>                                    |               |              |              |              |
| Annual                                                                                  | crspdcsa      | 74           | None         | 14,244       |
| Quarterly                                                                               | crspdcscq     | 293          | None         | 14,244       |
| Monthly                                                                                 | crspdcsm      | 877          | None         | 14,244       |
| Weekly                                                                                  | crspdcsw      | 1905         | None         | 14,244       |
| Daily                                                                                   | crspdcscd     | 9191         | None         | 14,244       |
| <b>Integer dates to CRSP dates exact is illustrated, but can be forward or backward</b> |               |              |              |              |
| Annual                                                                                  | crspdica      | 19981231     | 0            | 74           |
| Quarterly                                                                               | crspdicq      | 19981231     | 0            | 293          |
| Monthly                                                                                 | crspdicm      | 19981231     | 0            | 877          |
| Weekly                                                                                  | crspdicw      | 19981231     | 0            | 1905         |
| Daily                                                                                   | crspdicd      | 19981231     | 0            | 9191         |
| <b>SAS dates to CRSP dates exact is illustrated, but can be forward or backward</b>     |               |              |              |              |
| Annual                                                                                  | crspdsca      | 14,244       | 0            | 74           |
| Quarterly                                                                               | crspdsqc      | 14,244       | 0            | 293          |
| Monthly                                                                                 | crspdsclm     | 14,244       | 0            | 877          |
| Weekly                                                                                  | crspdsclw     | 14,244       | 0            | 1905         |
| Daily                                                                                   | crspdsclcd    | 14,244       | 0            | 9191         |

**Table 1.5.** (continued)

| Function Group                                          | Function Name | Argument One | Argument Two | Return Value |
|---------------------------------------------------------|---------------|--------------|--------------|--------------|
| <b>Integer dates to SAS dates for December 31, 1998</b> |               |              |              |              |
| Integer to SAS                                          | crspdi2s      | 19981231     | None         | 14,244       |
| <b>SAS dates to Integer dates for December 31, 1998</b> |               |              |              |              |
| SAS to Integer                                          | crspds2i      | 14,244       | None         | 19981231     |

---

## Examples

---

### Example 1.1. Extracting a List of PERMNOs Using a RANGE

This example specifies a list of PERMNOs that are desired from the monthly set of CRSPAccess data. Suppose you want the range of data starting January 1, 1995, and ending June 30, 1996.

```
title2 'Define a range inside the data range ';
title3 'Valid trading dates (19890131--19981231)';
title4 'My range is ( 19950101-19960630 )';

libname testit1 sasecrsp '/mydata/crspeng/m_sampdata/'
    setid=20
    permno=81871
    permno=82200
    permno=82224
    permno=83435
    permno=83696
    permno=83776
    permno=84788
    range='19950101-19960630';

data a;
    set testit1.ask;
run;

proc print data=a;
run;
```



**Output 1.1.1.** Printout of the ASK Monthly Time Series Data with RANGE

| The SAS System                           |        |          |           | 1 |
|------------------------------------------|--------|----------|-----------|---|
| Define a range inside the data range     |        |          |           |   |
| Valid trading dates (19890131--19981231) |        |          |           |   |
| My range is ( 19950101-19960630 )        |        |          |           |   |
| Obs                                      | PERMNO | CALDT    | ASK       |   |
| 1                                        | 81871  | 19950731 | 18.25000  |   |
| 2                                        | 81871  | 19950831 | 19.25000  |   |
| 3                                        | 81871  | 19950929 | 26.00000  |   |
| 4                                        | 81871  | 19951031 | 26.00000  |   |
| 5                                        | 81871  | 19951130 | 25.50000  |   |
| 6                                        | 81871  | 19951229 | 24.25000  |   |
| 7                                        | 81871  | 19960131 | 22.00000  |   |
| 8                                        | 81871  | 19960229 | 32.50000  |   |
| 9                                        | 81871  | 19960329 | 30.25000  |   |
| 10                                       | 81871  | 19960430 | 33.75000  |   |
| 11                                       | 81871  | 19960531 | 27.50000  |   |
| 12                                       | 81871  | 19960628 | 30.50000  |   |
| 13                                       | 82200  | 19950831 | 49.50000  |   |
| 14                                       | 82200  | 19950929 | 62.75000  |   |
| 15                                       | 82200  | 19951031 | 88.00000  |   |
| 16                                       | 82200  | 19951130 | 138.50000 |   |
| 17                                       | 82200  | 19951229 | 139.25000 |   |
| 18                                       | 82200  | 19960131 | 164.25000 |   |
| 19                                       | 82200  | 19960229 | 51.00000  |   |
| 20                                       | 82200  | 19960329 | 41.62500  |   |
| 21                                       | 82200  | 19960430 | 61.25000  |   |
| 22                                       | 82200  | 19960531 | .         |   |
| 23                                       | 82200  | 19960628 | 62.50000  |   |
| 24                                       | 82224  | 19950929 | 46.50000  |   |
| 25                                       | 82224  | 19951031 | 48.50000  |   |
| 26                                       | 82224  | 19951130 | 47.75000  |   |
| 27                                       | 82224  | 19951229 | 49.75000  |   |
| 28                                       | 82224  | 19960131 | 49.00000  |   |
| 29                                       | 82224  | 19960229 | 47.00000  |   |
| 30                                       | 82224  | 19960329 | 53.00000  |   |
| 31                                       | 82224  | 19960430 | 55.50000  |   |
| 32                                       | 82224  | 19960531 | .         |   |
| 33                                       | 82224  | 19960628 | 51.00000  |   |
| 34                                       | 83435  | 19960430 | 30.25000  |   |
| 35                                       | 83435  | 19960531 | .         |   |
| 36                                       | 83435  | 19960628 | 21.00000  |   |
| 37                                       | 83696  | 19960628 | 19.12500  |   |

---

## Example 1.2. Extracting All PERMNOs Using a RANGE

If you want all PERMNOs extracted for the ASK time series from the monthly data set, then you would not specify the PERMNO= option.

```
title2 'Define a range inside the data range ';
title3 'Valid trading dates (19890131--19981231)';
title4 'My range is ( 19950101-19950228 )';

libname testit2 sasecrsp '/mydata/crspeng/m_sampdata/'
    setid=20
    range='19950101-19950228';

data b;
    set testit2.ask;
run;

proc print data=b;
run;
```

## Output 1.2.1. Printout of All PERMNOs of ASK Monthly with RANGE

| The SAS System                           |        |          |           | 1 |
|------------------------------------------|--------|----------|-----------|---|
| Define a range inside the data range     |        |          |           |   |
| Valid trading dates (19890131--19981231) |        |          |           |   |
| My range is ( 19950101-19950228 )        |        |          |           |   |
| Obs                                      | PERMNO | CALDT    | ASK       |   |
| 1                                        | 10078  | 19950131 | 32.75000  |   |
| 2                                        | 10078  | 19950228 | 32.12500  |   |
| 3                                        | 10104  | 19950131 | 42.87500  |   |
| 4                                        | 10104  | 19950228 | 31.50000  |   |
| 5                                        | 10107  | 19950131 | 59.37500  |   |
| 6                                        | 10107  | 19950228 | 63.00000  |   |
| 7                                        | 10155  | 19950131 | 2.43750   |   |
| 8                                        | 10155  | 19950228 | 3.00000   |   |
| 9                                        | 10837  | 19950131 | 3.12500   |   |
| 10                                       | 10837  | 19950228 | 3.50000   |   |
| 11                                       | 11042  | 19950131 | 21.87500  |   |
| 12                                       | 11042  | 19950228 | 23.50000  |   |
| 13                                       | 11081  | 19950131 | 42.62500  |   |
| 14                                       | 11081  | 19950228 | 41.62500  |   |
| 15                                       | 14593  | 19950131 | 40.50000  |   |
| 16                                       | 14593  | 19950228 | 39.62500  |   |
| 17                                       | 40484  | 19950131 | 35.75000  |   |
| 18                                       | 40484  | 19950228 | 38.50000  |   |
| 19                                       | 44194  | 19950131 | 14.25000  |   |
| 20                                       | 44194  | 19950228 | 15.25000  |   |
| 21                                       | 49606  | 19950131 | 139.00000 |   |
| 22                                       | 49606  | 19950228 | 129.50000 |   |
| 23                                       | 50156  | 19950131 | 44.75000  |   |
| 24                                       | 50156  | 19950228 | 41.75000  |   |
| 25                                       | 50404  | 19950131 | 18.37500  |   |
| 26                                       | 50404  | 19950228 | 20.12500  |   |
| 27                                       | 59248  | 19950131 | 16.75000  |   |
| 28                                       | 59248  | 19950228 | 16.25000  |   |
| 29                                       | 59328  | 19950131 | 69.37500  |   |
| 30                                       | 59328  | 19950228 | 79.75000  |   |
| 31                                       | 75030  | 19950131 | 21.25000  |   |
| 32                                       | 75030  | 19950228 | 22.87500  |   |
| 33                                       | 76535  | 19950131 | 2.75000   |   |
| 34                                       | 76535  | 19950228 | 2.75000   |   |
| 35                                       | 76829  | 19950131 | 21.50000  |   |
| 36                                       | 76829  | 19950228 | 25.00000  |   |
| 37                                       | 77352  | 19950131 | 15.25000  |   |
| 38                                       | 77352  | 19950228 | 16.00000  |   |
| 39                                       | 77418  | 19950131 | 54.50000  |   |
| 40                                       | 77418  | 19950228 | 71.25000  |   |
| 41                                       | 77882  | 19950131 | 6.50000   |   |
| 42                                       | 77882  | 19950228 | 7.12500   |   |
| 43                                       | 78083  | 19950131 | 32.75000  |   |
| 44                                       | 78083  | 19950228 | 34.75000  |   |
| 45                                       | 78117  | 19950131 | 39.25000  |   |
| 46                                       | 78117  | 19950228 | 39.00000  |   |
| 47                                       | 78987  | 19950131 | 22.50000  |   |
| 48                                       | 78987  | 19950228 | 25.25000  |   |
| 49                                       | 81181  | 19950131 | 3.75000   |   |
| 50                                       | 81181  | 19950228 | 3.62500   |   |
| 51                                       | 91708  | 19950131 | .         |   |
| 52                                       | 91708  | 19950228 | .         |   |

---

### Example 1.3. Extracting One PERMNO Using No RANGE, Wide Open

If you want the entire range of available data for one particular PERMNO extracted from the monthly data set, then you would not specify the RANGE= option.

```
title2 'Select only PERMNO = 81871';
title3 'Valid trading dates (19890131--19981231)';
title4 'No range option, leave wide open';

libname testit3 sasecrsp '/mydata/crspeng/m_sampdata/'
    setid=20
    permno=81871;

data c;
    set testit3.ask;
run;

proc print data=c;
run;
```

**Output 1.3.1.** Printout of PERMNO = 81871 of ASK Monthly without RANGE

| The SAS System                           |        |          |          | 1 |
|------------------------------------------|--------|----------|----------|---|
| Select only PERMNO = 81871               |        |          |          |   |
| Valid trading dates (19890131--19981231) |        |          |          |   |
| No range option, leave wide open         |        |          |          |   |
| Obs                                      | PERMNO | CALDT    | ASK      |   |
| 1                                        | 81871  | 19950731 | 18.25000 |   |
| 2                                        | 81871  | 19950831 | 19.25000 |   |
| 3                                        | 81871  | 19950929 | 26.00000 |   |
| 4                                        | 81871  | 19951031 | 26.00000 |   |
| 5                                        | 81871  | 19951130 | 25.50000 |   |
| 6                                        | 81871  | 19951229 | 24.25000 |   |
| 7                                        | 81871  | 19960131 | 22.00000 |   |
| 8                                        | 81871  | 19960229 | 32.50000 |   |
| 9                                        | 81871  | 19960329 | 30.25000 |   |
| 10                                       | 81871  | 19960430 | 33.75000 |   |
| 11                                       | 81871  | 19960531 | 27.50000 |   |
| 12                                       | 81871  | 19960628 | 30.50000 |   |
| 13                                       | 81871  | 19960731 | 26.12500 |   |
| 14                                       | 81871  | 19960830 | 19.12500 |   |
| 15                                       | 81871  | 19960930 | 19.50000 |   |
| 16                                       | 81871  | 19961031 | 14.00000 |   |
| 17                                       | 81871  | 19961129 | 18.75000 |   |
| 18                                       | 81871  | 19961231 | 24.25000 |   |
| 19                                       | 81871  | 19970131 | 29.75000 |   |
| 20                                       | 81871  | 19970228 | 24.37500 |   |
| 21                                       | 81871  | 19970331 | 15.00000 |   |
| 22                                       | 81871  | 19970430 | 18.25000 |   |
| 23                                       | 81871  | 19970530 | 25.12500 |   |
| 24                                       | 81871  | 19970630 | 31.12500 |   |
| 25                                       | 81871  | 19970731 | 35.00000 |   |
| 26                                       | 81871  | 19970829 | 33.00000 |   |
| 27                                       | 81871  | 19970930 | 26.81250 |   |
| 28                                       | 81871  | 19971031 | 18.37500 |   |
| 29                                       | 81871  | 19971128 | 16.50000 |   |
| 30                                       | 81871  | 19971231 | 16.25000 |   |
| 31                                       | 81871  | 19980130 | 22.75000 |   |
| 32                                       | 81871  | 19980227 | 21.00000 |   |
| 33                                       | 81871  | 19980331 | 22.50000 |   |
| 34                                       | 81871  | 19980430 | 16.12500 |   |
| 35                                       | 81871  | 19980529 | 11.12500 |   |
| 36                                       | 81871  | 19980630 | 13.43750 |   |
| 37                                       | 81871  | 19980731 | 22.87500 |   |
| 38                                       | 81871  | 19980831 | 17.75000 |   |
| 39                                       | 81871  | 19980930 | 24.25000 |   |
| 40                                       | 81871  | 19981030 | 26.00000 |   |

---

### Example 1.4. Extracting Selected PERMNOs in Range Using INSET= Option

You can select the PERMNOs you want to extract and specify the range of data to extract by defining an INSET such as testin2.

```
title2 'INSET=testin2 uses date ranges along with PERMNOs:';
title3 '10107, 12490, 14322, 25788';
title4 'Begin dates and end dates for each permno are used in the INSET';

data testin2;
    permno = 10107; date1 = 19980731; date2 = 19981231; output;
    permno = 12490; date1 = 19970101; date2 = 19971231; output;
    permno = 14322; date1 = 19950731; date2 = 19960131; output;
    permno = 25778; date1 = 19950101; date2 = 19950331; output;
run;

libname mstk2 sasecrsp '/mydata/crspeng/m_sampdata/' setid=20
    inset='testin2,PERMNO,PERMNO,DATE1,DATE2';

data b;
    set mstk2.prc;
run;

proc print data=b;
run;
```

**Output 1.4.1.** Printout of PRC Monthly Time Series Using INSET= Option

| The SAS System                                                  |        |          |           | 1 |
|-----------------------------------------------------------------|--------|----------|-----------|---|
| Second INSET=testin2 uses date ranges along with PERMNOs:       |        |          |           |   |
| 10107, 12490, 14322, 25788                                      |        |          |           |   |
| Begin dates and end dates for each permno are used in the INSET |        |          |           |   |
| Obs                                                             | PERMNO | CALDT    | PRC       |   |
| 1                                                               | 10107  | 19980731 | 109.93750 |   |
| 2                                                               | 10107  | 19980831 | 95.93750  |   |
| 3                                                               | 10107  | 19980930 | 110.06250 |   |
| 4                                                               | 10107  | 19981030 | 105.87500 |   |
| 5                                                               | 10107  | 19981130 | 122.00000 |   |
| 6                                                               | 10107  | 19981231 | 138.68750 |   |
| 7                                                               | 12490  | 19970131 | 156.87500 |   |
| 8                                                               | 12490  | 19970228 | 143.75000 |   |
| 9                                                               | 12490  | 19970331 | 137.25000 |   |
| 10                                                              | 12490  | 19970430 | 160.50000 |   |
| 11                                                              | 12490  | 19970530 | 86.50000  |   |
| 12                                                              | 12490  | 19970630 | 90.25000  |   |
| 13                                                              | 12490  | 19970731 | 105.75000 |   |
| 14                                                              | 12490  | 19970829 | 101.37500 |   |
| 15                                                              | 12490  | 19970930 | 106.00000 |   |
| 16                                                              | 12490  | 19971031 | 98.50000  |   |
| 17                                                              | 12490  | 19971128 | 109.50000 |   |
| 18                                                              | 12490  | 19971231 | 104.62500 |   |
| 19                                                              | 14322  | 19950731 | 32.62500  |   |
| 20                                                              | 14322  | 19950831 | 32.37500  |   |
| 21                                                              | 14322  | 19950929 | 36.87500  |   |
| 22                                                              | 14322  | 19951031 | 34.00000  |   |
| 23                                                              | 14322  | 19951130 | 39.37500  |   |
| 24                                                              | 14322  | 19951229 | 39.00000  |   |
| 25                                                              | 14322  | 19960131 | 41.50000  |   |
| 26                                                              | 25778  | 19950131 | 49.87500  |   |
| 27                                                              | 25778  | 19950228 | 57.25000  |   |
| 28                                                              | 25778  | 19950331 | 59.37500  |   |

---

## Example 1.5. Converting Dates Using the CRSP Date Functions

This example shows how to use the CRSP date functions and formats. The CRSPDTD formats are used for all the crspdt variables, while the YYMMDD format is used for the sasdt variables.

```
title2 'OUT= Data Set';
title3 'CRSP Functions for sasecrsp';
title4 'Always assign the libname sasecrsp first';

libname mstk sasecrsp '/mydata/crspeng/m_sampdata/' setid=20;

data a (keep = crspdt crspdt2 crspdt3
           sasdt sasdt2 sasdt3
           intdt intdt2 intdt3);
  format crspdt crspdt2 crspdt3 crspdt8.;
  format sasdt sasdt2 sasdt3 yymmdd6.;
  format intdt intdt2 intdt3 8.;
  format exact 2.;
  crspdt = 1;
  sasdt = '2jul1962'd;
  intdt = 19620702;
  exact = 0;

/* Call the CRSP date to Integer function*/
  intdt2 = crspdcid(crspdt);

/* Call the SAS date to Integer function*/
  intdt3 = crspds2i(sasdt);

/* Call the Integer to Crsp date function*/
  crspdt2 = crspdicd(intdt,exact);

/* Call the Sas date to Crsp date conversion function*/
  crspdt3 = crspdsd(sasdt,exact);

/* Call the CRSP date to SAS date conversion function*/
  sasdt2 = crspdcsd(crspdt);

/* Call the Integer to Sas date conversion function*/
  sasdt3 = crspdi2s(intdt);
run;

proc print;
run;

title2 'Proc CONTENTS showing formats for sasecrsp';
proc contents data=a;
run;
```



**Output 1.5.1.** Printout of Date Conversions Using the CRSP Date Functions

```

                                The SAS System                                1
                                OUT= Data Set
                                CRSP Functions for sasecrsp
                                Always assign the libname sasecrsp first

Obs crspdt   crspdt2  crspdt3   sasdt sasdt2 sasdt3   intdt   intdt2   intdt3
1 19620702 19620702 19620702 620702 620702 620702 19620702 19620702 19620702

Proc CONTENTS showing formats for sasecrsp
CRSP Functions for sasecrsp
Always assign the libname sasecrsp first

The CONTENTS Procedure

Data Set Name: WORK.A                      Observations:      1
Member Type:  DATA                      Variables:         9
Engine:       V8                        Indexes:           0
Created:      13:15 Monday, November 27, 2000  Observation Length: 72
Last Modified: 13:15 Monday, November 27, 2000 Deleted Observations: 0
Protection:                                     Compressed:       NO
Data Set Type:                                   Sorted:          NO
Label:

-----Engine/Host Dependent Information-----

Data Set Page Size:      8192
Number of Data Set Pages: 1
First Data Page:        1
Max Obs per Page:       113
Obs in First Data Page: 1
Number of Data Set Repairs: 0
File Name:               /tmp/SAS_work67560000352F/a.sas7bdat
Release Created:         8.0202M0
Host Created:            HP-UX
Inode Number:            414
Access Permission:       rw-r--r--
Owner Name:              saskff
File Size (bytes):       16384

-----Alphabetic List of Variables and Attributes-----

#   Variable   Type   Len   Pos   Format
-----
1   crspdt     Num    8     0   CRSPD8.
2   crspdt2    Num    8     8   CRSPD8.
3   crspdt3    Num    8    16   CRSPD8.
7   intdt      Num    8    48   8.
8   intdt2     Num    8    56   8.
9   intdt3     Num    8    64   8.
4   sasdt      Num    8    24   YYMDD6.
5   sasdt2     Num    8    32   YYMDD6.
6   sasdt3     Num    8    40   YYMDD6.

```

---

## References

- Center for Research in Security Prices (2000), *CRSP Data Description Guide*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/stock\\_ind\\_data\\_descriptions.pdf](http://www.crsp.uchicago.edu/file_guides/stock_ind_data_descriptions.pdf)].
- Center for Research in Security Prices (2000), *CRSP Programmer's Guide*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/stock\\_ind\\_programming.pdf](http://www.crsp.uchicago.edu/file_guides/stock_ind_programming.pdf)].
- Center for Research in Security Prices (2000), *CRSP Access Database Format Release Notes*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/ca\\_release\\_notes.pdf](http://www.crsp.uchicago.edu/file_guides/ca_release_notes.pdf)].
- Center for Research in Security Prices (2000), *CRSP Utilities Guide*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/stock\\_ind\\_utilities.pdf](http://www.crsp.uchicago.edu/file_guides/stock_ind_utilities.pdf)].
- Center for Research in Security Prices (2000), *CRSP SFA Guide*, Chicago: The University of Chicago Graduate School of Business, [[http://www.crsp.uchicago.edu/file\\_guides/stock\\_ind\\_sfa.pdf](http://www.crsp.uchicago.edu/file_guides/stock_ind_sfa.pdf)].

## Chapter 2

# The SASEFAME Interface Engine

### Chapter Table of Contents

---

|                                                             |    |
|-------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                   | 31 |
| <b>GETTING STARTED</b> . . . . .                            | 31 |
| Using Remote FAME Data Access . . . . .                     | 31 |
| <b>SYNTAX</b> . . . . .                                     | 32 |
| The LIBNAME <i>libref</i> SASEFAME Statement . . . . .      | 32 |
| <b>EXAMPLES</b> . . . . .                                   | 33 |
| Example 2.1 Remote FAME Access, Using SAS CONNECT . . . . . | 33 |
| Example 2.2 Remote FAME Access, Using FAME CHLI . . . . .   | 43 |
| <b>REFERENCES</b> . . . . .                                 | 49 |



## Chapter 2

# The SASEFAME Interface Engine

---

### Overview

The SASEFAME interface engine enables SAS users to access and process time series data residing in a FAME database, and provides a seamless interface between FAME and SAS data processing.

The SASEFAME engine uses the LIBNAME statement to enable you to specify which time series you would like to read from the FAME database, and how you would like to convert the selected time series to the same time scale. The SAS DATA step can then be used to perform further subsetting and to store the resulting time series into a SAS data set. You can perform more analysis if desired either in the same SAS session or in another session at a later time.

---

### Getting Started

---

#### Using Remote FAME Data Access

There are two ways to access your remote FAME data bases. The first way is to use SAS/CONNECT to submit a remote SAS session to your FAME server, and then use PROC DOWNLOAD to bring your results back to your SAS client. For an illustration, see Example 2.1. You can also refer to *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software* for more about SAS/CONNECT.

The second way uses an experimental feature of the SASEFAME interface where the FAME CHLI communicates with your remote FAME server using the frdb\_m port number and the node name of your FAME master server in your *libref*. For an illustration, see Example 2.2. You can also refer to “Starting the Master Server” in the *Guide to FAME Database Servers*.

---

## Syntax

The SASEFAME engine uses standard engine syntax. Options used by SASEFAME are summarized in the following table.

| Description                                                                                                                                                            | Statement                      | Option    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|-----------|
| specifies the FAME frequency and the FAME technique.                                                                                                                   | LIBNAME <i>libref</i> SASEFAME | CONVERT=  |
| specifies a FAME wildcard to match data object series names within the FAME database, which limits the selection of time series that are included in the SAS data set. | LIBNAME <i>libref</i> SASEFAME | WILDCARD= |

---

### The LIBNAME *libref* SASEFAME Statement

**LIBNAME** *libref* SASEFAME 'physical name' *options*;

If you are accessing a remote FAME database using FAME CHLI, you can use the following syntax for *physical name* :

'#port number\@hostname physical path name'

---

## Examples

---

### Example 2.1. Remote FAME Access, Using SAS CONNECT

Suppose you are running SAS in a client/server environment and have SAS/CONNECT capability allowing you access to your FAME server. You could access your FAME remote data by signing on to your FAME server from your client session and doing a remote submit to access your FAME data. You could then use PROC DOWNLOAD to bring your remote data into the local client SAS session.

If you start your program by signing on to the remote server and checking your FAME environment variable definitions, your code might look like this:

```
options validvarname=any;
/* VALIDVARNAME set to ANY allows
   glue character in the time series names */

%let remnode=mysunbox.unx.sas.com;
/* name the sastcpd service for the connection */
signon remnode.shr2;
rsubmit;

%let FAME=%sysget(FAME);
%put(&FAME);

options validvarname=any;
libname lib1 sasefame "/usr/local/famelib/util"
              convert=(frequency=annual technique=constant)
              wildcard="?";
```

Here is the corresponding segment of the SAS log showing the remote sign-on process and the definition of the FAME environment variable on the remote host, as well as the warning generated when you use the VALIDVARNAME option allowing special characters in your SAS variable names:

**Output 2.1.1. SAS Log of the SAS/CONNECT Remote Access to FAME Data**

```

1          The SAS System
NOTE: Copyright (c) 1999-2000 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Release 8.2 (TS02.02M0D05092000)
      Licensed to SAS Institute Inc., Site 0000000001.
NOTE: This session is executing on the HP-UX B.10.20 platform.

      options validvarname=any;
WARNING: VALIDVARNAME=ANY is an experimental option. The DATA Step and
the SQL procedure have been tested for use with this
option. Other use of this option may cause undetected errors.
2
3          %let remnode=mysunbox.unx.sas.com;
4
5
6          /* use this one on the AIX box
7          filename rlink '/users/myuser/connect/sunbox.scr';
8          */
9          /* use this one on H8X */
10         filename rlink '/u/myuser/connect/sunbox.scr';
11
12         /* options remote=remnode comamid=tcp; */
13         signon remnode.shr2;
NOTE: Remote signon to REMNODE.SHR2 commencing (SAS Release 8.02.02M0D050900).
SESSION ESTABLISHED
TRACE: Stop;
NOTE: End of script file trace.
NOTE: Copyright (c) 1999-2000 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Release 8.1 (TS01.01M0P05082000)
      Licensed to SAS Institute Y2K Testing, Site 0000000001.
NOTE: This session is executing on the SunOS 5.6 platform.
This message is contained in the SAS news file, and is presented upon
initialization. Edit the files "news" in the "misc/base" directory to
display site-specific news and information in the program log.
The command line option "-nonews" will prevent this display.
NOTE: SAS initialization used:
      real time          0.26 seconds
      cpu time           0.25 seconds

NOTE: Remote signon to REMNODE.SHR2 complete.
14         rsubmit;

NOTE: Remote submit to REMNODE.SHR2 commencing.
1  %let FAME=%sysget(FAME);
2  %put(&FAME);
(/usr/local/famelib80)
3
4  options mprint validvarname=any;
WARNING: VALIDVARNAME=ANY is an experimental option. The DATA Step and
the SQL procedure have been tested for use with this option.
Other use of this option may cause undetected errors.
5
NOTE: Libref LIB1 was successfully assigned as follows:
Engine:          FAMECHLI
Physical Name:   /usr/local/famelib/util

```

Next, you would use your libname assignment to access the remote data like this:

```

data oecd1;
    set lib1.oecd1;
run;

title2 'OECD1: PRINT Procedure';

```



```

proc print data=oeed1(obs=10);
run;

title2 'OECD1: CONTENTS Procedure';
proc contents data=oeed1;
run;

proc sort data=oeed1;
  by date;
run;

title2 'OECD1: MEANS Procedure';
proc means data=oeed1 sum;
  by date;
  var 'fin.herd'n;
run;

proc download inlib=work /* remote SASWORK */
  outlib=work; /* local SASWORK */
run;

proc datasets lib=work;
  contents data=_ALL_;
run;

endrsubmit;
signoff;

/* NOW the local work has the fame data */
proc datasets lib=work;
  contents data=_ALL_;
run;

```

Here is the remainder of the segment of the SAS log showing the download process and the remote sign-off along with the PROC DATASETS work directory results:

Output 2.1.2. SAS Log of the SAS/CONNECT Remote Access to FAME Data

NOTE: Download in progress from data=WORK.OECD1 to out=WORK.OECD1  
 NOTE: 2744 bytes were transferred at 52406 bytes/second.  
 NOTE: The data set WORK.OECD1 has 7 observations and 49 variables.  
 NOTE: Downloaded 7 observations of 49 variables.  
 NOTE: The data set WORK.OECD1 has 7 observations and 49 variables.

NOTE: PROCEDURE DOWNLOAD used:  
       real time              2.71 seconds  
       cpu time               0.04 seconds

275 proc datasets lib=work;

```

      -----Directory-----
Libref:          WORK
Engine:          V8
Physical Name:    /usr/tmp/SAS_work3EEB00002CC4_sunbox
File Name:        /usr/tmp/SAS_work3EEB00002CC4_sunbox
Inode Number:     212060504
Access Permission: rwxrwxrwx
Owner Name:       myuser

```

```

      -----Directory-----
File Size (bytes): 398

```

|   |       |         | File  |                    |
|---|-------|---------|-------|--------------------|
| # | Name  | Memtype | Size  | Last Modified      |
| 1 | OECD1 | DATA    | 40960 | 10MAY2000:13:19:58 |

277 contents data=\_ALL\_;  
 278 run;

NOTE: Remote submit to REMNODE.SHR2 complete.  
 15 signoff;  
 NOTE: Remote signoff from REMNODE.SHR2 commencing.  
 NOTE: The PROCEDURE DATASETS printed pages 43-48.  
 NOTE: PROCEDURE DATASETS used:  
       real time              0.17 seconds  
       cpu time               0.09 seconds

NOTE: Remote signoff from REMNODE.SHR2 complete.

```

16
17 proc datasets lib=work;
      -----Directory-----
Libref:          WORK
Engine:          V8

```

```

      -----Directory-----
Physical Name:    /tmp/SAS_workDF8500003D1D
File Name:        /tmp/SAS_workDF8500003D1D
Inode Number:     1089
Access Permission: rwxr-xr-x
Owner Name:       myuser
File Size (bytes): 1024

```

|   |       |         | File  |                    |
|---|-------|---------|-------|--------------------|
| # | Name  | Memtype | Size  | Last Modified      |
| 1 | OECD1 | DATA    | 40960 | 10MAY2000:13:20:03 |

/\* NOW the local work has the fame data \*/

```

18
19 contents data=_ALL_;
20 run;

21
22 endsas;

```

Output 2.1.3. Printout of the SAS/CONNECT Remote Access to FAME Data

```

***fameproc: Using FAME with a multitude of SAS Procs***
OECD1: PRINT Procedure
1

```

| Obs | DATE | AUS.    |          | AUT.    |          | BEL.    |          | CAN.    |          |
|-----|------|---------|----------|---------|----------|---------|----------|---------|----------|
|     |      | DIRDES  | AUS.HERD | DIRDES  | AUT.HERD | DIRDES  | BEL.HERD | DIRDES  | CAN.HERD |
| 1   | 1985 | .       | .        | 360.900 | 5990.60  | 334.700 | 14936.00 | 1345.90 | 1642     |
| 2   | 1986 | 680.400 | 881.70   | .       | .        | 341.300 | 15459.80 | 1435.30 | 1755     |
| 3   | 1987 | 725.800 | 983.80   | .       | .        | 369.200 | 16614.30 | 1486.50 | 1849     |
| 4   | 1988 | 750.000 | 1072.90  | .       | .        | 374.000 | 16572.70 | 1589.60 | 2006     |
| 5   | 1989 | .       | .        | .       | .        | .       | 18310.70 | 1737.00 | 2214     |
| 6   | 1990 | .       | .        | .       | .        | .       | 18874.20 | 1859.20 | 2347     |
| 7   | 1991 | .       | .        | .       | .        | .       | .        | 1959.60 | 2488     |

| Obs | CHE.    |          | DEU.    |          | DNK.    |          | ESP.    |          |
|-----|---------|----------|---------|----------|---------|----------|---------|----------|
|     | DIRDES  | CHE.HERD | DIRDES  | DEU.HERD | DIRDES  | DNK.HERD | DIRDES  | ESP.HERD |
| 1   | .       | .        | 2702.10 | 6695.70  | 191.300 | 1873.70  | 335.500 | 31987.0  |
| 2   | 366.600 | 900      | .       | 7120.00  | 211.700 | 2123.00  | 354.800 | 36778.0  |
| 3   | .       | .        | 3365.60 | 8338.50  | 232.700 | 2372.00  | 409.900 | 43667.0  |
| 4   | 632.100 | 1532     | 3538.60 | 8780.00  | 258.100 | 2662.00  | 508.200 | 55365.5  |
| 5   | .       | 1648     | 3777.20 | 9226.60  | 284.800 | 2951.00  | 623.600 | 69270.5  |
| 6   | .       | .        | 2953.30 | 9700.00  | .       | .        | 723.600 | 78848.0  |
| 7   | .       | .        | .       | .        | .       | .        | .       | 89908.0  |

| Obs | FIN.    |          | FRA.    |          | GBR.    |          | GRC.    |          |
|-----|---------|----------|---------|----------|---------|----------|---------|----------|
|     | DIRDES  | FIN.HERD | DIRDES  | FRA.HERD | DIRDES  | GBR.HERD | DIRDES  | GRC.HERD |
| 1   | 183.700 | 1097.00  | 2191.60 | 15931.00 | 2068.40 | 1174.00  | .       | .        |
| 2   | 202.000 | 1234.00  | 2272.90 | 17035.00 | 2226.10 | 1281.00  | 44.600  | 3961.00  |
| 3   | 224.300 | 1401.30  | 2428.70 | 18193.00 | 2381.10 | 1397.20  | .       | .        |
| 4   | 247.700 | 1602.00  | 2573.50 | 19272.00 | 2627.00 | 1592.00  | 60.600  | 6674.50  |
| 5   | 259.700 | 1725.50  | 2856.50 | 21347.80 | 2844.10 | 1774.20  | 119.800 | 14485.20 |
| 6   | 271.000 | 1839.00  | 3005.20 | 22240.00 | .       | .        | .       | .        |
| 7   | .       | .        | .       | .        | .       | .        | .       | .        |

| Obs | IRL.    |          | ISL.    |          | ITA.    |          | JPN.     |          |
|-----|---------|----------|---------|----------|---------|----------|----------|----------|
|     | DIRDES  | IRL.HERD | DIRDES  | ISL.HERD | DIRDES  | ITA.HERD | DIRDES   | JPN.HERD |
| 1   | 39.2000 | 28.3707  | 7.1000  | 268.300  | 1344.90 | 1751008  | 8065.70  | 1789780  |
| 2   | 46.5000 | 35.0000  | .       | .        | 1460.60 | 2004453  | 8290.10  | 1832575  |
| 3   | 48.1000 | 36.0380  | 7.8000  | 420.400  | 1674.40 | 2362102  | 9120.80  | 1957921  |
| 4   | 49.6000 | 37.0730  | .       | .        | 1861.50 | 2699927  | 9657.20  | 2014073  |
| 5   | 50.2000 | 39.0130  | 10.3000 | 786.762  | 1968.00 | 2923504  | 10405.90 | 2129372  |
| 6   | 51.7000 | .        | 11.0000 | 902.498  | 2075.00 | 3183071  | .        | 2296992  |
| 7   | .       | .        | 11.8000 | 990.865  | 2137.80 | 3374000  | .        | .        |

| Obs | NLD.    |          | NOR.    |          | NZL.    |          | PRT.    |          |
|-----|---------|----------|---------|----------|---------|----------|---------|----------|
|     | DIRDES  | NLD.HERD | DIRDES  | NOR.HERD | DIRDES  | NZL.HERD | DIRDES  | PRT.HERD |
| 1   | 798.400 | 2032     | 209.000 | 1802.50  | 59.2000 | 80.100   | .       | .        |
| 2   | 836.000 | 2093     | .       | .        | .       | .        | 76.700  | 5988.90  |
| 3   | 886.500 | 2146     | 250.000 | 2165.80  | .       | .        | .       | .        |
| 4   | 883.000 | 2105     | .       | .        | .       | .        | 111.500 | 10158.20 |
| 5   | 945.000 | 2202     | 308.900 | 2771.40  | 78.7000 | 143.800  | .       | .        |
| 6   | .       | .        | .       | .        | .       | .        | .       | .        |
| 7   | .       | .        | 352.000 | 3100.00  | .       | .        | .       | .        |

| Obs | SWE.    |          | TUR.    |          | USA.     |          | YUG.    |          |
|-----|---------|----------|---------|----------|----------|----------|---------|----------|
|     | DIRDES  | SWE.HERD | DIRDES  | TUR.HERD | DIRDES   | USA.HERD | DIRDES  | YUG.HERD |
| 1   | 840.10  | 6844     | 144.800 | 22196    | 14786.00 | 14786.00 | 175.900 | 1.87     |
| 2   | .       | .        | 136.400 | 26957    | 16566.90 | 16566.90 | 208.600 | 4.10     |
| 3   | 1016.90 | 8821     | 121.900 | 32309    | 18326.10 | 18326.10 | 237.000 | 10.21    |
| 4   | .       | .        | 174.400 | 74474    | 20246.20 | 20246.20 | 233.000 | 29.81    |
| 5   | 1076.00 | 11104    | 212.300 | 143951   | 22159.50 | 22159.50 | 205.100 | 375.22   |
| 6   | .       | .        | .       | .        | 23556.10 | 23556.10 | .       | 2588.50  |
| 7   | .       | .        | .       | .        | 24953.80 | 24953.80 | .       | .        |

**Output 2.1.4.** PROC CONTENTS of the Remote FAME Data on the SUN Server Node

```
***fameproc: Using FAME on Remote Server ***
      OECD1: CONTENTS Procedure
      The CONTENTS Procedure

Data Set Name: WORK.OECD1              Observations:      7
Member Type:   DATA                   Variables:        49
Engine:        V8                      Indexes:          0
Created:       13:19 Wednesday, May 10, 2000 Observation Length: 392
Last Modified: 13:19 Wednesday, May 10, 2000 Deleted Observations: 0
Protection:                               Compressed:      NO
Data Set Type:                               Sorted:        NO
Label:

      -----Engine/Host Dependent Information-----

Data Set Page Size:      32768
Number of Data Set Pages: 1
First Data Page:        1
Max Obs per Page:       83
Obs in First Data Page: 7
Number of Data Set Repairs: 0
File Name:               /usr/tmp/SAS_work3EEB00002CC4_sunbox/oecd1.sas7bdat
Release Created:         8.0101M0
Host Created:            SunOS
Inode Number:            211927040
Access Permission:       rw-rw-rw-
Owner Name:              myuser
File Size (bytes):       40960
```

**Output 2.1.5.** PROC CONTENTS of the Remote FAME Data on the SUN Server Node

| ***fameproc: Using FAME on Remote Server ***          |            |      |     |     |        |          |                     |
|-------------------------------------------------------|------------|------|-----|-----|--------|----------|---------------------|
| OECD1: CONTENTS Procedure                             |            |      |     |     |        |          |                     |
| The CONTENTS Procedure                                |            |      |     |     |        |          |                     |
| -----Alphabetic List of Variables and Attributes----- |            |      |     |     |        |          |                     |
| #                                                     | Variable   | Type | Len | Pos | Format | Informat | Label               |
| 2                                                     | AUS.DIRDES | Num  | 8   | 8   |        |          |                     |
| 3                                                     | AUS.HERD   | Num  | 8   | 16  |        |          |                     |
| 4                                                     | AUT.DIRDES | Num  | 8   | 24  |        |          |                     |
| 5                                                     | AUT.HERD   | Num  | 8   | 32  |        |          |                     |
| 6                                                     | BEL.DIRDES | Num  | 8   | 40  |        |          |                     |
| 7                                                     | BEL.HERD   | Num  | 8   | 48  |        |          |                     |
| 8                                                     | CAN.DIRDES | Num  | 8   | 56  |        |          |                     |
| 9                                                     | CAN.HERD   | Num  | 8   | 64  |        |          |                     |
| 10                                                    | CHE.DIRDES | Num  | 8   | 72  |        |          |                     |
| 11                                                    | CHE.HERD   | Num  | 8   | 80  |        |          |                     |
| 1                                                     | DATE       | Num  | 8   | 0   | YEAR4. | 4.       | Date of Observation |
| 12                                                    | DEU.DIRDES | Num  | 8   | 88  |        |          |                     |
| 13                                                    | DEU.HERD   | Num  | 8   | 96  |        |          |                     |
| 14                                                    | DNK.DIRDES | Num  | 8   | 104 |        |          |                     |
| 15                                                    | DNK.HERD   | Num  | 8   | 112 |        |          |                     |
| 16                                                    | ESP.DIRDES | Num  | 8   | 120 |        |          |                     |
| 17                                                    | ESP.HERD   | Num  | 8   | 128 |        |          |                     |
| 18                                                    | FIN.DIRDES | Num  | 8   | 136 |        |          |                     |
| 19                                                    | FIN.HERD   | Num  | 8   | 144 |        |          |                     |
| 20                                                    | FRA.DIRDES | Num  | 8   | 152 |        |          |                     |
| 21                                                    | FRA.HERD   | Num  | 8   | 160 |        |          |                     |
| 22                                                    | GBR.DIRDES | Num  | 8   | 168 |        |          |                     |
| 23                                                    | GBR.HERD   | Num  | 8   | 176 |        |          |                     |
| 24                                                    | GRC.DIRDES | Num  | 8   | 184 |        |          |                     |
| 25                                                    | GRC.HERD   | Num  | 8   | 192 |        |          |                     |
| 26                                                    | IRL.DIRDES | Num  | 8   | 200 |        |          |                     |
| 27                                                    | IRL.HERD   | Num  | 8   | 208 |        |          |                     |
| 28                                                    | ISL.DIRDES | Num  | 8   | 216 |        |          |                     |
| 29                                                    | ISL.HERD   | Num  | 8   | 224 |        |          |                     |
| 30                                                    | ITA.DIRDES | Num  | 8   | 232 |        |          |                     |
| 31                                                    | ITA.HERD   | Num  | 8   | 240 |        |          |                     |
| 32                                                    | JPN.DIRDES | Num  | 8   | 248 |        |          |                     |
| 33                                                    | JPN.HERD   | Num  | 8   | 256 |        |          |                     |
| 34                                                    | NLD.DIRDES | Num  | 8   | 264 |        |          |                     |
| 35                                                    | NLD.HERD   | Num  | 8   | 272 |        |          |                     |
| 36                                                    | NOR.DIRDES | Num  | 8   | 280 |        |          |                     |
| 37                                                    | NOR.HERD   | Num  | 8   | 288 |        |          |                     |
| 38                                                    | NZL.DIRDES | Num  | 8   | 296 |        |          |                     |
| 39                                                    | NZL.HERD   | Num  | 8   | 304 |        |          |                     |
| 40                                                    | PRT.DIRDES | Num  | 8   | 312 |        |          |                     |
| 41                                                    | PRT.HERD   | Num  | 8   | 320 |        |          |                     |
| 42                                                    | SWE.DIRDES | Num  | 8   | 328 |        |          |                     |
| 43                                                    | SWE.HERD   | Num  | 8   | 336 |        |          |                     |
| 44                                                    | TUR.DIRDES | Num  | 8   | 344 |        |          |                     |
| 45                                                    | TUR.HERD   | Num  | 8   | 352 |        |          |                     |
| 46                                                    | USA.DIRDES | Num  | 8   | 360 |        |          |                     |
| 47                                                    | USA.HERD   | Num  | 8   | 368 |        |          |                     |
| 48                                                    | YUG.DIRDES | Num  | 8   | 376 |        |          |                     |
| 49                                                    | YUG.HERD   | Num  | 8   | 384 |        |          |                     |

**Output 2.1.6.** PROC MEANS and PROC TABULATE of Remote FAME Data

|                                                          |  |
|----------------------------------------------------------|--|
| ***fameproc: Using FAME with a multitude of SAS Procs*** |  |
| OECD1: MEANS Procedure                                   |  |
| ----- Date of Observation=1985 -----                     |  |
| The MEANS Procedure                                      |  |
| Analysis Variable : FIN.HERD                             |  |
| Sum                                                      |  |
| -----                                                    |  |
| 1097.00                                                  |  |
| -----                                                    |  |
| ----- Date of Observation=1986 -----                     |  |
| Analysis Variable : FIN.HERD                             |  |
| Sum                                                      |  |
| -----                                                    |  |
| 1234.00                                                  |  |
| -----                                                    |  |
| ----- Date of Observation=1987 -----                     |  |
| Analysis Variable : FIN.HERD                             |  |
| Sum                                                      |  |
| -----                                                    |  |
| 1401.30                                                  |  |
| -----                                                    |  |
| ----- Date of Observation=1988 -----                     |  |
| Analysis Variable : FIN.HERD                             |  |
| Sum                                                      |  |
| -----                                                    |  |
| 1602.00                                                  |  |
| -----                                                    |  |
| ----- Date of Observation=1989 -----                     |  |
| Analysis Variable : FIN.HERD                             |  |
| Sum                                                      |  |
| -----                                                    |  |
| 1725.50                                                  |  |
| -----                                                    |  |
| ----- Date of Observation=1990 -----                     |  |
| The MEANS Procedure                                      |  |
| Analysis Variable : FIN.HERD                             |  |
| Sum                                                      |  |
| -----                                                    |  |
| 1839.00                                                  |  |
| -----                                                    |  |
| ----- Date of Observation=1991 -----                     |  |
| Analysis Variable : FIN.HERD                             |  |
| Sum                                                      |  |
| -----                                                    |  |
| .                                                        |  |
| -----                                                    |  |
| OECD1: TABULATE Procedure                                |  |
| -----                                                    |  |
| FIN.HERD                                                 |  |
| -----                                                    |  |
| Sum                                                      |  |
| -----                                                    |  |
| 8898.80                                                  |  |
| -----                                                    |  |

**Output 2.1.7.** PROC CONTENTS of the Remote FAME Data on the HP-UX Client Node

| The DATASETS Procedure                      |                                          |                       |     |
|---------------------------------------------|------------------------------------------|-----------------------|-----|
| Data Set Name:                              | WORK.OECD1                               | Observations:         | 7   |
| Member Type:                                | DATA                                     | Variables:            | 49  |
| Engine:                                     | V8                                       | Indexes:              | 0   |
| Created:                                    | 13:20 Wednesday, May 10, 2000            | Observation Length:   | 392 |
| Last Modified:                              | 13:20 Wednesday, May 10, 2000            | Deleted Observations: | 0   |
| Protection:                                 |                                          | Compressed:           | NO  |
| Data Set Type:                              |                                          | Sorted:               | YES |
| Label:                                      |                                          |                       |     |
| -----Engine/Host Dependent Information----- |                                          |                       |     |
| Data Set Page Size:                         | 32768                                    |                       |     |
| Number of Data Set Pages:                   | 1                                        |                       |     |
| First Data Page:                            | 1                                        |                       |     |
| Max Obs per Page:                           | 83                                       |                       |     |
| Obs in First Data Page:                     | 7                                        |                       |     |
| Number of Data Set Repairs:                 | 0                                        |                       |     |
| File Name:                                  | /tmp/SAS_workDF8500003D1D/oecd1.sas7bdat |                       |     |
| Release Created:                            | 8.0202M0                                 |                       |     |
| Host Created:                               | HP-UX                                    |                       |     |
| Inode Number:                               | 1452                                     |                       |     |
| Access Permission:                          | rw-r--r--                                |                       |     |
| Owner Name:                                 | myuser                                   |                       |     |
| File Size (bytes):                          | 40960                                    |                       |     |

Part 1. General Information

**Output 2.1.8.** PROC CONTENTS of the Remote FAME Data on the HP-UX Client Node

| ***fameproc: Using FAME with a multitude of SAS Procs*** |            |      |     |                |        |          | 1                   |
|----------------------------------------------------------|------------|------|-----|----------------|--------|----------|---------------------|
| OECD1: CONTENTS Procedure                                |            |      |     |                |        |          |                     |
| The CONTENTS Procedure                                   |            |      |     |                |        |          |                     |
| -----Alphabetic List of Variables and Attributes-----    |            |      |     |                |        |          |                     |
| #                                                        | Variable   | Type | Len | Pos            | Format | Informat | Label               |
| 2                                                        | AUS.DIRDES | Num  | 8   | 8              |        |          |                     |
| 3                                                        | AUS.HERD   | Num  | 8   | 16             |        |          |                     |
| 4                                                        | AUT.DIRDES | Num  | 8   | 24             |        |          |                     |
| 5                                                        | AUT.HERD   | Num  | 8   | 32             |        |          |                     |
| 6                                                        | BEL.DIRDES | Num  | 8   | 40             |        |          |                     |
| 7                                                        | BEL.HERD   | Num  | 8   | 48             |        |          |                     |
| 8                                                        | CAN.DIRDES | Num  | 8   | 56             |        |          |                     |
| 9                                                        | CAN.HERD   | Num  | 8   | 64             |        |          |                     |
| 10                                                       | CHE.DIRDES | Num  | 8   | 72             |        |          |                     |
| 11                                                       | CHE.HERD   | Num  | 8   | 80             |        |          |                     |
| 1                                                        | DATE       | Num  | 8   | 0              | YEAR4. | 4.       | Date of Observation |
| 12                                                       | DEU.DIRDES | Num  | 8   | 88             |        |          |                     |
| 13                                                       | DEU.HERD   | Num  | 8   | 96             |        |          |                     |
| 14                                                       | DNK.DIRDES | Num  | 8   | 104            |        |          |                     |
| 15                                                       | DNK.HERD   | Num  | 8   | 112            |        |          |                     |
| 16                                                       | ESP.DIRDES | Num  | 8   | 120            |        |          |                     |
| 17                                                       | ESP.HERD   | Num  | 8   | 128            |        |          |                     |
| 18                                                       | FIN.DIRDES | Num  | 8   | 136            |        |          |                     |
| 19                                                       | FIN.HERD   | Num  | 8   | 144            |        |          |                     |
| 20                                                       | FRA.DIRDES | Num  | 8   | 152            |        |          |                     |
| 21                                                       | FRA.HERD   | Num  | 8   | 160            |        |          |                     |
| 22                                                       | GBR.DIRDES | Num  | 8   | 168            |        |          |                     |
| 23                                                       | GBR.HERD   | Num  | 8   | 176            |        |          |                     |
| 24                                                       | GRC.DIRDES | Num  | 8   | 184            |        |          |                     |
| 25                                                       | GRC.HERD   | Num  | 8   | 192            |        |          |                     |
| 26                                                       | IRL.DIRDES | Num  | 8   | 200            |        |          |                     |
| 27                                                       | IRL.HERD   | Num  | 8   | 208            |        |          |                     |
| 28                                                       | ISL.DIRDES | Num  | 8   | 216            |        |          |                     |
| 29                                                       | ISL.HERD   | Num  | 8   | 224            |        |          |                     |
| 30                                                       | ITA.DIRDES | Num  | 8   | 232            |        |          |                     |
| 31                                                       | ITA.HERD   | Num  | 8   | 240            |        |          |                     |
| 32                                                       | JPN.DIRDES | Num  | 8   | 248            |        |          |                     |
| 33                                                       | JPN.HERD   | Num  | 8   | 256            |        |          |                     |
| 34                                                       | NLD.DIRDES | Num  | 8   | 264            |        |          |                     |
| 35                                                       | NLD.HERD   | Num  | 8   | 272            |        |          |                     |
| 36                                                       | NOR.DIRDES | Num  | 8   | 280            |        |          |                     |
| 37                                                       | NOR.HERD   | Num  | 8   | 288            |        |          |                     |
| 38                                                       | NZL.DIRDES | Num  | 8   | 296            |        |          |                     |
| 39                                                       | NZL.HERD   | Num  | 8   | 304            |        |          |                     |
| 40                                                       | PRT.DIRDES | Num  | 8   | 312            |        |          |                     |
| 41                                                       | PRT.HERD   | Num  | 8   | 320            |        |          |                     |
| 42                                                       | SWE.DIRDES | Num  | 8   | 328            |        |          |                     |
| 43                                                       | SWE.HERD   | Num  | 8   | 336            |        |          |                     |
| 44                                                       | TUR.DIRDES | Num  | 8   | 344            |        |          |                     |
| 45                                                       | TUR.HERD   | Num  | 8   | 352            |        |          |                     |
| 46                                                       | USA.DIRDES | Num  | 8   | 360            |        |          |                     |
| 47                                                       | USA.HERD   | Num  | 8   | 368            |        |          |                     |
| 48                                                       | YUG.DIRDES | Num  | 8   | 376            |        |          |                     |
| 49                                                       | YUG.HERD   | Num  | 8   | 384            |        |          |                     |
| -----Sort Information-----                               |            |      |     |                |        |          |                     |
|                                                          |            |      |     | Sortedby:      | DATE   |          |                     |
|                                                          |            |      |     | Validated:     | YES    |          |                     |
|                                                          |            |      |     | Character Set: | ASCII  |          |                     |



---

## Example 2.2. Remote FAME Access, Using FAME CHLI

Suppose you are running FAME in a client/server environment and have FAME CHLI capability allowing you access to your FAME server. You could access your FAME remote data by specifying the port number of the tcpip service that is defined for your frdb\_m and the node name of your FAME master server in your physical path. In the following example, the FAME server node name is booker, and the port number is 5555, which was designated in the FAME master command. Refer to “Starting the Master Server” in the *Guide to FAME Database Servers* for more about starting your FAME master server.

```
/* DRIECON Database, Using FAME with REMOTE ACCESS VIA CHLI */
/*****

libname test1 sasefame '#5555\@booker \ $FAME/util';
data a;
    set test1.driecon;
    run;

proc contents data=a;
    run;
proc means data=a n;
    run;
```

Your log will show the FAME databases that are read in for conversion:

## Output 2.2.1. SAS Log Segment of the FAME CHLI Remote Access to FAME Data

```

22          /* DRIECON Database, Using FAME with REMOTE ACCESS VIA HLI */
23          /*****
24
25          libname test1 sasefame '#5555\@booker \$FAME/util';
WARNING: Remote access using the FAME CHLI is an experimental feature of the
        FAME Engine. Please make sure your port number is defined correctly
        on your frdb_m master server.
NOTE: Libref TEST1 was successfully assigned as follows:
      Engine:          FAMECHLI
      Physical Name: #5555\@booker \$FAME/util
26          data a;
27              set test1.driecon;
$N -- SERIES (PRECISION by MONTHLY)
$N copied to work data base as $N.
BOPMERCH -- SERIES (PRECISION by QUARTERLY)
BOPMERCH copied to work data base as BOPMERCH.
CUSA0 -- SERIES (PRECISION by MONTHLY)
CUSA0 copied to work data base as CUSA0.
CUSA0NS -- SERIES (PRECISION by MONTHLY)
CUSA0NS copied to work data base as CUSA0NS.
DBTGFNS -- SERIES (PRECISION by MONTHLY)
DBTGFNS copied to work data base as DBTGFNS.
NOTE: DESC -- SERIES (STRING by CASE) rejected for conversion.
DJ30C -- SERIES (PRECISION by BUSINESS)
DJ30C copied to work data base as DJ30C.
DJ65CMC -- SERIES (PRECISION by BUSINESS)
DJ65CMC copied to work data base as DJ65CMC.
FBL3Y -- SERIES (PRECISION by BUSINESS)
FBL3Y copied to work data base as FBL3Y.
FCN30YY -- SERIES (PRECISION by BUSINESS)
FCN30YY copied to work data base as FCN30YY.
FIP1Q -- SERIES (PRECISION by BUSINESS)
FIP1Q copied to work data base as FIP1Q.
FIP30Q -- SERIES (PRECISION by BUSINESS)
FIP30Q copied to work data base as FIP30Q.
FSCD30Y -- SERIES (PRECISION by BUSINESS)
FSCD30Y copied to work data base as FSCD30Y.
GDP -- SERIES (PRECISION by QUARTERLY)
GDP copied to work data base as GDP.
GDP92C -- SERIES (PRECISION by QUARTERLY)
GDP92C copied to work data base as GDP92C.
GICV -- SERIES (PRECISION by MONTHLY)
GICV copied to work data base as GICV.
GNP -- SERIES (PRECISION by QUARTERLY)
GNP copied to work data base as GNP.
GNP92C -- SERIES (PRECISION by QUARTERLY)
GNP92C copied to work data base as GNP92C.
.
.
JLAG -- SERIES (PRECISION by MONTHLY)
JLAG copied to work data base as JLAG.
JLEAD -- SERIES (PRECISION by MONTHLY)
JLEAD copied to work data base as JLEAD.
JQIND -- SERIES (PRECISION by MONTHLY)
JQIND copied to work data base as JQIND.
.
.
NOTE: NAMES -- SERIES (STRING by CASE) rejected for conversion.
ZA.CP -- SERIES (PRECISION by ANNUAL)
ZB.CP -- SERIES (PRECISION by ANNUAL)
28          run;
NOTE: There were 53 observations read from the data set TEST1.DRIECON.
NOTE: The data set WORK.A has 53 observations and 53 variables.

```

## Output 2.2.2. Printout of the FAME CHLI Remote Access to FAME Data

The SAS System

1

The CONTENTS Procedure

|                |                                 |                       |     |
|----------------|---------------------------------|-----------------------|-----|
| Data Set Name: | WORK.A                          | Observations:         | 53  |
| Member Type:   | DATA                            | Variables:            | 53  |
| Engine:        | V8                              | Indexes:              | 0   |
| Created:       | 16:49 Friday, November 17, 2000 | Observation Length:   | 424 |
| Last Modified: | 16:49 Friday, November 17, 2000 | Deleted Observations: | 0   |
| Protection:    |                                 | Compressed:           | NO  |
| Data Set Type: |                                 | Sorted:               | NO  |
| Label:         |                                 |                       |     |

-----Engine/Host Dependent Information-----

|                             |                                      |
|-----------------------------|--------------------------------------|
| Data Set Page Size:         | 40960                                |
| Number of Data Set Pages:   | 1                                    |
| First Data Page:            | 1                                    |
| Max Obs per Page:           | 96                                   |
| Obs in First Data Page:     | 53                                   |
| Number of Data Set Repairs: | 0                                    |
| File Name:                  | /tmp/SAS_work30D40000397D/a.sas7bdat |
| Release Created:            | 8.0202M0                             |
| Host Created:               | HP-UX                                |
| Inode Number:               | 3076                                 |
| Access Permission:          | rw-r--r--                            |
| Owner Name:                 | myuser                               |
| File Size (bytes):          | 49152                                |

-----Alphabetic List of Variables and Attributes-----

| #  | Variable | Type | Len | Pos | Format | Informat | Label                                                                 |
|----|----------|------|-----|-----|--------|----------|-----------------------------------------------------------------------|
| 2  | \$N      | Num  | 8   | 8   |        |          | POPULATION INCLUDING ARMED<br>FORCES OVERSEAS (P25E)                  |
| 3  | BOPMERCH | Num  | 8   | 16  |        |          | U.S. BALANCE ON MERCHANDISE<br>TRADE (BOP)                            |
| 4  | CUSA0    | Num  | 8   | 24  |        |          | CPI (ALL URBAN) - ALL ITEMS                                           |
| 5  | CUSA0NS  | Num  | 8   | 32  |        |          | CPIU - All items                                                      |
| 1  | DATE     | Num  | 8   | 0   | YEAR4. | 4.       | Date of Observation                                                   |
| 6  | DBTGFNS  | Num  | 8   | 40  |        |          |                                                                       |
| 7  | DJ30C    | Num  | 8   | 48  |        |          | DOW JONES: 30 INDUSTRIAL<br>AVERAGE - CLOSE                           |
| 8  | DJ65CMC  | Num  | 8   | 56  |        |          | DOW JONES: 65 COMPOSITE AVERAGE                                       |
| 9  | FBL3Y    | Num  | 8   | 64  |        |          | TREASURY BILL: SECONDARY, 3-MONTH<br>BOND-EQUIVALENT YIELD (H15) - US |
| 10 | FCN30YY  | Num  | 8   | 72  |        |          | GOVT ISSUE: CONSTANT MATURITY,<br>30-YR (H15) - US                    |
| 11 | FIP1Q    | Num  | 8   | 80  |        |          | COMMERCIAL PAPER: NON-FINAN,<br>1-DAY QUOTED YIELD - US               |
| 12 | FIP30Q   | Num  | 8   | 88  |        |          | COMMERCIAL PAPER: NON-FINAN,<br>1-MO QUOTED YIELD - US                |
| 13 | FSCD30Y  | Num  | 8   | 96  |        |          | CD: SECONDARY MKT, 1-MO YIELD - US                                    |
| 14 | GDP      | Num  | 8   | 104 |        |          | GROSS DOMESTIC PRODUCT                                                |
| 15 | GDP92C   | Num  | 8   | 112 |        |          | GROSS DOMESTIC PRODUCT (CHAINED)                                      |
| 16 | GICV     | Num  | 8   | 120 |        |          | NEW CONSTRUCTION PUT IN PLACE<br>- PUBLIC TOTAL (C30)                 |
| 17 | GNP      | Num  | 8   | 128 |        |          | GROSS NATIONAL PRODUCT                                                |
| 18 | GNP92C   | Num  | 8   | 136 |        |          | GROSS NATIONAL PRODUCT                                                |
| 19 | HUCMPNC  | Num  | 8   | 144 |        |          | HOUSING COMPLETIONS, PRIVATE<br>- NORTH CENTRAL (C22)                 |
| 20 | HUCMPNE  | Num  | 8   | 152 |        |          | HOUSING COMPLETIONS, PRIVATE<br>- NORTHEAST (C22)                     |
| 21 | HUCMPSO  | Num  | 8   | 160 |        |          | HOUSING COMPLETIONS,<br>PRIVATE - SOUTH (C22)                         |
| 22 | HUCMPWT  | Num  | 8   | 168 |        |          | HOUSING COMPLETIONS, PRIVATE-WEST(C22)                                |

## Output 2.2.3. PROC CONTENTS of the Remote FAME Data

| The SAS System                                        |                 |      |     |     |        | 2        |                                                                         |
|-------------------------------------------------------|-----------------|------|-----|-----|--------|----------|-------------------------------------------------------------------------|
| The CONTENTS Procedure                                |                 |      |     |     |        |          |                                                                         |
| -----Alphabetic List of Variables and Attributes----- |                 |      |     |     |        |          |                                                                         |
| #                                                     | Variable        | Type | Len | Pos | Format | Informat | Label                                                                   |
| 23                                                    | HUSTS           | Num  | 8   | 176 |        |          | HOUSING STARTS, PRIVATE INCLUDING FARM - TOTAL (C20)                    |
| 24                                                    | HUSTS1          | Num  | 8   | 184 |        |          | HOUSING STARTS, PRIVATE INCL FARM - ONE UNIT (C20)                      |
| 25                                                    | HUSTS1NS        | Num  | 8   | 192 |        |          | HOUSING STARTS, PRIVATE INCL FARM - ONE UNIT (C20)                      |
| 26                                                    | I               | Num  | 8   | 200 |        |          | GROSS PRIVATE DOMESTIC INVESTMENT                                       |
| 27                                                    | I92C            | Num  | 8   | 208 |        |          | GROSS PRIVATE DOMESTIC INVESTMENT (CHAINED)                             |
| 28                                                    | ICV_G           | Num  | 8   | 216 |        |          | NEW CONSTRUCTION PUT IN PLACE - TOTAL (C30)                             |
| 29                                                    | JCOIN%LAG       | Num  | 8   | 224 |        |          | RATIO, COINCIDENT INDEX TO LAGGING INDEX (BCI)                          |
| 30                                                    | JLAG            | Num  | 8   | 232 |        |          | LAGGING INDICATORS COMPOSITE INDEX (BCI)                                |
| 31                                                    | JLEAD           | Num  | 8   | 240 |        |          | LEADING INDICATORS COMPOSITE INDEX (BCI)                                |
| 32                                                    | JQIND           | Num  | 8   | 248 |        |          | INDUSTRIAL PROD INDEX - TOTAL INDEX (G17)                               |
| 33                                                    | JQIND20         | Num  | 8   | 256 |        |          | INDUSTRIAL PROD INDEX - FOODS (G17)                                     |
| 35                                                    | JQIND21         | Num  | 8   | 272 |        |          | INDUSTRIAL PROD INDEX - TOBACCO PRODUCTS (G17)                          |
| 36                                                    | JQIND26         | Num  | 8   | 280 |        |          | INDUSTRIAL PROD INDEX - PAPER & PRODUCTS (G17)                          |
| 37                                                    | JQIND28         | Num  | 8   | 288 |        |          | INDUSTRIAL PROD INDEX - CHEMICALS & PRODUCTS (G17)                      |
| 38                                                    | JQIND37         | Num  | 8   | 296 |        |          | INDUSTRIAL PROD INDEX - TRANSPORTATION EQUIPMENT (G17)                  |
| 39                                                    | JQIND39         | Num  | 8   | 304 |        |          | INDUSTRIAL PROD INDEX - MISC MANUFACTURES (G17)                         |
| 34                                                    | JQIND208        | Num  | 8   | 264 |        |          | INDUSTRIAL PROD INDEX - BEVERAGES (G17)                                 |
| 40                                                    | JQINDEQPB       | Num  | 8   | 312 |        |          | INDUSTRIAL PROD INDEX - BUSINESS EQUIPMENT (G17)                        |
| 41                                                    | MNY1            | Num  | 8   | 320 |        |          | MONEY SUPPLY - CURRENCY, DEMAND DEPOSITS, OTHER CHECKABLE DEPOSITS (H6) |
| 42                                                    | MNY2            | Num  | 8   | 328 |        |          | MONEY SUPPLY - M2 (H6)                                                  |
| 43                                                    | PIDGNP          | Num  | 8   | 336 |        |          | IMPLICIT PRICE DEFLATOR - GROSS NATIONAL PRODUCT                        |
| 44                                                    | RUC             | Num  | 8   | 344 |        |          | UNEMPLOYMENT RATE - CIVILIAN (ESIT)                                     |
| 45                                                    | RXC132%<br>USNS | Num  | 8   | 352 |        |          | EXCHANGE RATE IN NEW YORK - FRENCH FRANC PER U.S. DOLLAR (G5)           |
| 46                                                    | RXC134%<br>USNS | Num  | 8   | 360 |        |          | EXCHANGE RATE IN NEW YORK - GERMAN MARK PER U.S. DOLLAR (G5)            |
| 47                                                    | RXC158%<br>USNS | Num  | 8   | 368 |        |          | EXCHANGE RATE IN NEW YORK - JAPANESE YEN PER U.S. DOLLAR (G5)           |
| 48                                                    | RXUS%<br>C112NS | Num  | 8   | 376 |        |          | EXCHANGE RATE IN NEW YORK - U.S. CENTS PER BRITISH POUND (G5)           |
| 49                                                    | STR             | Num  | 8   | 384 |        |          | RETAIL SALES -TOTAL (RTR)                                               |
| 50                                                    | WPINS           | Num  | 8   | 392 |        |          | PRODUCER PRICE INDEX - ALL COMMODITIES (PPI)                            |
| 51                                                    | YP              | Num  | 8   | 400 |        |          | PERSONAL INCOME                                                         |
| 52                                                    | ZA              | Num  | 8   | 408 |        |          | CORPORATE PROFITS AFTER TAX EXCLUDING IVA                               |
| 53                                                    | ZB              | Num  | 8   | 416 |        |          | CORPORATE PROFITS BEFORE TAX EXCLUDING IVA                              |

Output 2.2.4. PROC MEANS of the Remote Fame Data

|                     |   |
|---------------------|---|
| The SAS System      | 4 |
| The MEANS Procedure |   |
| Variable            |   |
| -----               |   |
| DATE                |   |
| \$N                 |   |
| BOPMERCH            |   |
| CUSA0               |   |
| CUSA0NS             |   |
| DBTGfNS             |   |
| DJ30C               |   |
| DJ65CMC             |   |
| FBL3Y               |   |
| FCN30YY             |   |
| FIP1Q               |   |
| FIP30Q              |   |
| FSCD30Y             |   |
| GDP                 |   |
| GDP92C              |   |
| GICV                |   |
| GNP                 |   |
| GNP92C              |   |
| HUCMPNC             |   |
| HUCMPNE             |   |
| HUCMPSO             |   |
| HUCMPWT             |   |
| HUSTS               |   |
| HUSTS1              |   |
| HUSTS1NS            |   |
| I                   |   |
| I92C                |   |
| ICV_G               |   |
| JCOIN%LAG           |   |
| JLAG                |   |
| JLEAD               |   |
| JQIND               |   |
| JQIND20             |   |
| JQIND208            |   |
| JQIND21             |   |
| JQIND26             |   |
| JQIND28             |   |
| JQIND37             |   |
| JQIND39             |   |
| JQINDEQPB           |   |
| MNY1                |   |
| MNY2                |   |
| PIDGNP              |   |
| RUC                 |   |
| RXC132%USNS         |   |
| RXC134%USNS         |   |
| RXC158%USNS         |   |
| RXUS%C112NS         |   |
| STR                 |   |
| WPINS               |   |
| YP                  |   |
| ZA                  |   |
| ZB                  |   |
| -----               |   |

## Output 2.2.5. PROC MEANS of the Remote FAME Data

| The SAS System                                                          |  | 5  |
|-------------------------------------------------------------------------|--|----|
| The MEANS Procedure                                                     |  |    |
| Label                                                                   |  | N  |
| -----                                                                   |  |    |
| Date of Observation                                                     |  | 53 |
| POPULATION INCLUDING ARMED FORCES OVERSEAS (P25E)                       |  | 49 |
| U.S. BALANCE ON MERCHANDISE TRADE (BOP)                                 |  | 39 |
| CPI (ALL URBAN) - ALL ITEMS                                             |  | 52 |
| CPIU - All items                                                        |  | 52 |
|                                                                         |  | 41 |
| DOW JONES: 30 INDUSTRIAL AVERAGE - CLOSE                                |  | 19 |
| DOW JONES: 65 COMPOSITE AVERAGE                                         |  | 19 |
| TREASURY BILL: SECONDARY, 3-MONTH BOND-EQUIVALENT YIELD (H15) - US      |  | 20 |
| GOVT ISSUE: CONSTANT MATURITY, 30-YR (H15) - US                         |  | 22 |
| COMMERCIAL PAPER: NON-FINAN, 1-DAY QUOTED YIELD - US                    |  | 1  |
| COMMERCIAL PAPER: NON-FINAN, 1-MO QUOTED YIELD - US                     |  | 20 |
| CD: SECONDARY MKT, 1-MO YIELD - US                                      |  | 23 |
| GROSS DOMESTIC PRODUCT                                                  |  | 53 |
| GROSS DOMESTIC PRODUCT (CHAINED)                                        |  | 52 |
| NEW CONSTRUCTION PUT IN PLACE - PUBLIC TOTAL (C30)                      |  | 35 |
| GROSS NATIONAL PRODUCT                                                  |  | 53 |
| GROSS NATIONAL PRODUCT                                                  |  | 52 |
| HOUSING COMPLETIONS, PRIVATE - NORTH CENTRAL (C22)                      |  | 20 |
| HOUSING COMPLETIONS, PRIVATE - NORTHEAST (C22)                          |  | 20 |
| HOUSING COMPLETIONS, PRIVATE - SOUTH (C22)                              |  | 20 |
| HOUSING COMPLETIONS, PRIVATE - WEST (C22)                               |  | 20 |
| HOUSING STARTS, PRIVATE INCLUDING FARM - TOTAL (C20)                    |  | 52 |
| HOUSING STARTS, PRIVATE INCL FARM - ONE UNIT (C20)                      |  | 40 |
| HOUSING STARTS, PRIVATE INCL FARM - ONE UNIT (C20)                      |  | 40 |
| GROSS PRIVATE DOMESTIC INVESTMENT                                       |  | 53 |
| GROSS PRIVATE DOMESTIC INVESTMENT (CHAINED)                             |  | 52 |
| NEW CONSTRUCTION PUT IN PLACE - TOTAL (C30)                             |  | 35 |
| RATIO, COINCIDENT INDEX TO LAGGING INDEX (BCI)                          |  | 40 |
| LAGGING INDICATORS COMPOSITE INDEX (BCI)                                |  | 40 |
| LEADING INDICATORS COMPOSITE INDEX (BCI)                                |  | 40 |
| INDUSTRIAL PROD INDEX - TOTAL INDEX (G17)                               |  | 52 |
| INDUSTRIAL PROD INDEX - FOODS (G17)                                     |  | 52 |
| INDUSTRIAL PROD INDEX - BEVERAGES (G17)                                 |  | 45 |
| INDUSTRIAL PROD INDEX - TOBACCO PRODUCTS (G17)                          |  | 52 |
| INDUSTRIAL PROD INDEX - PAPER & PRODUCTS (G17)                          |  | 52 |
| INDUSTRIAL PROD INDEX - CHEMICALS & PRODUCTS (G17)                      |  | 52 |
| INDUSTRIAL PROD INDEX - TRANSPORTATION EQUIPMENT (G17)                  |  | 52 |
| INDUSTRIAL PROD INDEX - MISC MANUFACTURES (G17)                         |  | 52 |
| INDUSTRIAL PROD INDEX - BUSINESS EQUIPMENT (G17)                        |  | 52 |
| MONEY SUPPLY - CURRENCY, DEMAND DEPOSITS, OTHER CHECKABLE DEPOSITS (H6) |  | 40 |
| MONEY SUPPLY - M2 (H6)                                                  |  | 40 |
| IMPLICIT PRICE DEFLATOR - GROSS NATIONAL PRODUCT                        |  | 52 |
| UNEMPLOYMENT RATE - CIVILIAN (ESIT)                                     |  | 9  |
| EXCHANGE RATE IN NEW YORK - FRENCH FRANC PER U.S. DOLLAR (G5)           |  | 32 |
| EXCHANGE RATE IN NEW YORK - GERMAN MARK PER U.S. DOLLAR (G5)            |  | 32 |
| EXCHANGE RATE IN NEW YORK - JAPANESE YEN PER U.S. DOLLAR (G5)           |  | 32 |
| EXCHANGE RATE IN NEW YORK - U.S. CENTS PER BRITISH POUND (G5)           |  | 32 |
| RETAIL SALES -TOTAL (RTR)                                               |  | 32 |
| PRODUCER PRICE INDEX - ALL COMMODITIES (PPI)                            |  | 52 |
| PERSONAL INCOME                                                         |  | 53 |
| CORPORATE PROFITS AFTER TAX EXCLUDING IVA                               |  | 53 |
| CORPORATE PROFITS BEFORE TAX EXCLUDING IVA                              |  | 53 |
| -----                                                                   |  |    |

---

## References

- DRI/McGraw-Hill (1997), *DataLink*, Lexington, MA.
- DRI/McGraw-Hill Data Search and Retrieval for Windows (1996), *DRIPRO User's Guide*, Lexington, MA.
- FAME Information Services (1995), *Command Reference for Release 7.6, Vols. 1 and 2*, Ann Arbor, MI.
- FAME Information Services (1998), *Guide to FAME Database Servers*, New York, NY, [<http://www.fame.com>].
- FAME Information Services (1995), *Reference Guide to Seamless C HLI*, Ann Arbor, MI.
- FAME Information Services (1995), *User's Guide to FAME*, Ann Arbor, MI.
- Organization for Economic Cooperation and Development (1992), *Annual National Accounts: Volume I. Main Aggregates Content Documentation for Magnetic Tape Subscription*, Paris, France.
- Organization for Economic Cooperation and Development (1992), *Annual National Accounts: Volume II. Detailed Tables Technical Documentation for Magnetic Tape Subscription*, Paris, France.
- Organization for Economic Cooperation and Development (1992), *Main Economic Indicators Database Note*, Paris, France.
- Organization for Economic Cooperation and Development (1992), *Main Economic Indicators Inventory*, Paris, France.
- Organization for Economic Cooperation and Development (1992), *Main Economic Indicators OECD Statistics on Magnetic Tape Document*, Paris, France.
- Organization for Economic Cooperation and Development (1992), *OECD Statistical Information Research and Inquiry System Magnetic Tape Format Documentation*, Paris, France.
- Organization for Economic Cooperation and Development (1992), *Quarterly National Accounts Inventory of Series Codes*, Paris, France.
- Organization for Economic Cooperation and Development (1992), *Quarterly National Accounts Technical Documentation*, Paris, France.
- SAS Institute Inc. (1996), *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*, Cary, NC: SAS Institute, Inc.





# Part 2

## Procedure Reference

### Table of Contents

---

|                                           |    |
|-------------------------------------------|----|
| Chapter 3. The ARIMA Procedure . . . . .  | 53 |
| Chapter 4. The MODEL Procedure . . . . .  | 67 |
| Chapter 5. The VARMAX Procedure . . . . . | 91 |
| Chapter 6. The X12 Procedure . . . . .    | 97 |



# Chapter 3

## The ARIMA Procedure

### Chapter Table of Contents

---

|                                                |    |
|------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                      | 55 |
| <b>SYNTAX</b> . . . . .                        | 55 |
| OUTLIER Statement . . . . .                    | 55 |
| <b>DETAILS</b> . . . . .                       | 56 |
| Displayed Output . . . . .                     | 58 |
| Modeling in the Presence of Outliers . . . . . | 58 |
| <b>EXAMPLES</b> . . . . .                      | 59 |
| Example 3.1 Nile Data . . . . .                | 59 |
| Example 3.2 Airline Data . . . . .             | 62 |
| <b>REFERENCES</b> . . . . .                    | 65 |



## Chapter 3

# The ARIMA Procedure

---

### Overview

The OUTLIER statement is a new enhancement to the ARIMA procedure. This statement is used as a diagnostic tool to detect shifts in the mean level of the response series that are not accounted for by the model in the preceding ESTIMATE statement. You can control different aspects of this detection process, such as the type and number of changes to detect, the significance level of the underlying tests, etc., by setting various options. Currently three types of changes in the mean level can be detected. These types are

- **Additive Outlier (AO)**, which represents a spurious change in one of the measurements
- **Level Shift (LS)**, which stands for a permanent shift in the level starting at some point in the observation period
- **Temporary Change (TC( [  $d$  ] ))**, which represents a temporary change in the level of duration  $d$

---

### Syntax

---

#### OUTLIER Statement

**OUTLIER** *options*;

The OUTLIER statement applies to the model fitted in the preceding ESTIMATE statement. The following options are used in the OUTLIER statement:

**TYPE = ADDITIVE | SHIFT | TEMP**(  $d_1, \dots, d_k$  )

The TYPE= option specifies the types of level shifts to search for. The default is TYPE= (ADDITIVE SHIFT), which requests searching for additive outliers and permanent level shifts. The option TEMP(  $d_1, \dots, d_k$  ) requests searching for temporary changes in the level of durations  $d_1, \dots, d_k$ . These options can also be abbreviated as AO, LS, and TC.

**ALPHA=** *significance-level*

The ALPHA= option specifies the significance level for tests in the OUTLIER statement. The default is 0.05.

**SIGMA= ROBUST | MSE**

The statistical tests performed during the outlier detection require an estimate of error

variance. Using the SIGMA= option you can choose between two types of error variance estimates. SIGMA= MSE corresponds to the usual mean squared error (MSE) estimate, and SIGMA= ROBUST corresponds to a robust estimate of the error variance. The default is SIGMA= ROBUST.

**MAXNUM=** *number*

This option is used to limit the number of outliers to search. The default is MAXNUM= 5.

**MAXPCT=** *number*

This option is similar to MAXNUM= option. In the MAXPCT= option you can limit the number of outliers to search to a percentage of the series length. The default is MAXPCT= 2. When both of these options are specified, the smaller of the two search numbers is used.

The following examples illustrate the various possibilities for the OUTLIER statement.

```
outlier;
```

This sets all the options to their default values, that is, it is equivalent to

```
outlier type=(ao ls) alpha=0.05 sigma=robust  
maxnum=5 maxpct=2;
```

The following statement requests a search for permanent level shifts and for temporary level changes of durations 6 and 12. The search is limited to at most three changes, and the significance level of the underlying tests is 0.001. MSE is used as the estimate of error variance.

```
outlier type=(ls tc(6 12)) alpha=0.001 sigma=mse maxnum=3;
```

---

## Details

You can use the OUTLIER statement to detect changes in the level of the response series that are not accounted for by the currently estimated model. The types of changes considered are Additive Outliers (AO), Level Shifts (LS), and Temporary Changes (TC).

Let  $\eta_t$  be a regression variable describing some type of change in the mean response. In time series literature  $\eta_t$  is called a shock signature. An additive outlier at some time point  $i$  corresponds to a shock signature  $\eta_t$  such that  $\eta_i = 1.0$  and  $\eta_t$  is 0.0 at all other points. Similarly a permanent level shift originating at time  $i$  has a shock signature such that  $\eta_t$  is 0.0 for  $t < i$  and 1.0 for  $t \geq i$ . A temporary level shift of duration  $d$  originating at time  $i$  will have  $\eta_t$  equal to 1.0 between  $i$  and  $i + d$  and 0.0 otherwise.

Suppose that the preceding ESTIMATE statement has the ARIMA model

$$D(B)Y_t = \mu_t + \frac{\theta(B)}{\phi(B)}a_t$$

where  $Y_t$  is the response series,  $D(B)$  is the differencing polynomial in the backward shift operator  $B$  (possibly identity),  $\mu_t$  is the transfer function input,  $\phi(B)$  and  $\theta(B)$  are the AR and MA polynomials, and  $a_t$  is the Gaussian white noise series.

The problem of detection of level shifts in the OUTLIER statement is formulated as a problem of sequential selection of shock signatures that improve the model in the ESTIMATE statement. This is similar to the forward selection process in the stepwise regression procedure. The selection process starts with considering shock signatures of the type specified in the TYPE= option, originating at each non-missing measurement. This involves testing  $H_0: \beta = 0$  versus  $H_a: \beta \neq 0$  in the model

$$D(B)(Y_t - \beta\eta_t) = \mu_t + \frac{\theta(B)}{\phi(B)}a_t$$

for each of these shock signatures. The most significant shock signature, if it also satisfies the significance criterion in ALPHA= option, is included in the model. If no significant shock signature is found then the outlier detection process stops, otherwise this augmented model, which incorporates the selected shock signature in its transfer function input, becomes the null model for the subsequent selection process. This iterative process stops if at any stage no more significant shock signatures are found or if the number of iterations exceed the maximum search number resulting due to the MAXNUM= and MAXPCT= settings. In all these iterations the parameters of the ARIMA model in the ESTIMATE statement are held fixed.

The precise details of the testing procedure for a given shock signature  $\eta_t$  are as follows:

The preceding testing problem is equivalent to testing  $H_0: \beta = 0$  versus  $H_a: \beta \neq 0$  in the following “regression with ARMA errors” model

$$N_t = \beta\zeta_t + \frac{\theta(B)}{\phi(B)}a_t$$

where  $N_t = (D(B)Y_t - \mu_t)$  is the “noise” process and  $\zeta_t = D(B)\eta_t$  is the “effective” shock signature.

In this setting, under  $H_0$ ,  $N = (N_1, N_2, \dots, N_n)^T$  is a mean zero Gaussian vector with variance covariance matrix  $\sigma^2\Sigma$ . Here  $\sigma^2$  is the variance of the white noise process  $a_t$  and  $\Sigma$  is the variance covariance matrix associated with the ARMA model. Moreover, under  $H_a$ ,  $N$  has  $\beta\zeta$  as the mean vector where  $\zeta = (\zeta_1, \zeta_2, \dots, \zeta_n)^T$ . Additionally, the generalized least squares estimate of  $\beta$  and its variance is given by

$$\begin{aligned}\hat{\beta} &= \delta/\kappa \\ \text{Var}(\hat{\beta}) &= \sigma^2/\kappa\end{aligned}$$

where  $\delta = \zeta^T\Sigma^{-1}N$  and  $\kappa = \zeta^T\Sigma^{-1}\zeta$ . The test statistic  $\tau^2 = \delta^2/(\sigma^2\kappa)$  is used to test the significance of  $\beta$ , which has an approximate chi-squared distribution with 1 degree of freedom under  $H_0$ . The type of estimate of  $\sigma^2$  used in the calculation of  $\tau^2$  can be specified by the SIGMA= option. The default setting is SIGMA=ROBUST

that corresponds to a robust estimate suggested in an outlier detection procedure in X-12-ARIMA, the Census Bureau's time series analysis program; refer to Findley et al. (1998) for additional information. The setting SIGMA=MSE corresponds to the usual mean squared error estimate (MSE) computed the same way as in the ESTIMATE statement with the NODF option. The robust estimate of  $\sigma^2$  is computed by the formula

$$\hat{\sigma}^2 = (1.49 \times \text{Median}(|\hat{a}_t|))^2$$

where  $\hat{a}_t$  are the standardized residuals of the null ARIMA model.

The quantities  $\delta$  and  $\kappa$  are efficiently computed by a method described in de Jong and Penzer (1998); refer also to Kohn and Ansley (1985).

---

## Displayed Output

The output of the OUTLIER statement has two parts. The first part, the Outlier Summary, contains the information about the maximum number of outliers searched, the number of outliers actually detected, and the significance level used in the outlier detection. The second part, the Outlier Table, tabulates the results of the outlier detection process. The outliers are listed in the order in which they are found. This table contains five columns:

- The Obs column contains the observation number of the start of the level shift.
- The Type column lists the type of the level shift.
- The Estimate column contains  $\hat{\beta}$ , the estimate of the regression coefficient of the shock signature.
- The Chi-Square column lists the value of the test statistic  $\tau^2$ .
- The Approx Prob > ChiSq column lists the approximate  $p$ -value of the test statistic.

---

## Modeling in the Presence of Outliers

In practice, modeling and forecasting time series data in the presence of outliers is a difficult problem for several reasons. The presence of outliers can adversely affect the model identification and estimation steps. Their presence close to the end of the observation period can have a serious impact on the forecasting performance of the model. In some cases level shifts are associated with changes in the mechanism driving the observation process, and separate models may be appropriate to different sections of the data. In view of all these difficulties, diagnostic tools such as outlier detection and residual analysis are essential in any modeling process.

The following modeling strategy, which incorporates level shift detection in the familiar Box-Jenkins modeling methodology, seems to work in many cases:

1. Proceed with model identification and estimation as usual. Suppose this results in a tentative ARIMA model, say M.



2. Check for suspected level shifts unaccounted for by model M using the OUTLIER statement. In this step, unless there is evidence to justify it, the number of level shifts searched should be kept small.
3. Augment the original dataset with the regression variables corresponding to the detected outliers.
4. Include the first few of these regression variables in M, and call this model M1. Re-estimate all the parameters of M1. It is important not to include too many of these outlier variables in the model in order to avoid the danger of over-fitting.
5. Check the adequacy of M1 by examining the parameter estimates, residual analysis, and outlier detection. Refine it more if necessary.

---

## Examples

The following examples illustrate the syntax and usage of the OUTLIER statement in some simple situations.

---

### Example 3.1. Nile Data

This example is discussed in de Jong and Penzer (1998). The data consist of readings of the annual flow volume of the Nile River at Aswan from 1871 to 1970. These data have also been studied by Cobb (1978). These studies indicate that levels in the years 1877 and 1913, that is, the 7th and 43rd measurements, are strong candidates for additive outliers, and that there was a shift in the flow levels starting from the year 1899. The year 1899 corresponds to the 29th observation. This shift in 1899 is attributed partly to the weather changes and partly to the start of construction work for a new dam at Aswan.

```
data nile;
  input level @@;
  datalines;
1120 1160 963 1210 1160 1160 813 1230 1370 1140
995 935 1110 994 1020 960 1180 799 958 1140
1100 1210 1150 1250 1260 1220 1030 1100 774 840
874 694 940 833 701 916 692 1020 1050 969
831 726 456 824 702 1120 1100 832 764 821
768 845 864 862 698 845 744 796 1040 759
781 865 845 944 984 897 822 1010 771 676
649 846 812 742 801 1040 860 874 848 890
744 749 838 1050 918 986 797 923 975 815
1020 906 901 1170 912 746 919 718 714 740
;
```

You can start the modeling process with the ARIMA(0, 1, 1) model, an ARIMA model close to the Structural model suggested in de Jong and Penzer (1998), and examine the parameter estimates, the residual autocorrelations, and the outliers.

```
proc arima data=nile;
  identify var=level(1) noprint;
  estimate q=1 noint method=ml plot;
  outlier maxnum=5;
run;
```

A portion of the estimation and the outlier detection output is shown in Output 3.1.1.

**Output 3.1.1.** Output from Fitting ARIMA(0, 1, 1) Model

| The ARIMA Procedure           |          |                |            |                    |     |
|-------------------------------|----------|----------------|------------|--------------------|-----|
| Maximum Likelihood Estimation |          |                |            |                    |     |
| Parameter                     | Estimate | Standard Error | t Value    | Approx Pr >  t     | Lag |
| MA1,1                         | 0.73271  | 0.07132        | 10.27      | <.0001             | 1   |
| Variance Estimate             |          |                | 20810.22   |                    |     |
| Std Error Estimate            |          |                | 144.2575   |                    |     |
| AIC                           |          |                | 1267.091   |                    |     |
| SBC                           |          |                | 1269.686   |                    |     |
| Number of Residuals           |          |                | 99         |                    |     |
| Model for variable level      |          |                |            |                    |     |
| Period(s) of Differencing     |          |                | 1          |                    |     |
| No mean term in this model.   |          |                |            |                    |     |
| Outlier Summary               |          |                |            |                    |     |
| Maximum number searched       |          |                | 5          |                    |     |
| Number found                  |          |                | 5          |                    |     |
| Significance used             |          |                | 0.05       |                    |     |
| Outlier Table                 |          |                |            |                    |     |
| Obs                           | Type     | Estimate       | Chi-Square | Approx Prob> ChiSq |     |
| 29                            | Shift    | -315.75346     | 13.13      | 0.0003             |     |
| 43                            | Additive | -403.97105     | 11.83      | 0.0006             |     |
| 7                             | Additive | -335.49351     | 7.69       | 0.0055             |     |
| 94                            | Additive | 305.03568      | 6.16       | 0.0131             |     |
| 18                            | Additive | -287.81484     | 6.00       | 0.0143             |     |

Note that the first three outliers detected are indeed the ones discussed earlier. You can include the shock signatures corresponding to these three outliers in the Nile data set.

```
data nile;
  set nile;
  if _n_ = 7 then A07 = 1.0;
  else A07 = 0.0;
  if _n_ = 43 then A043 = 1.0;
  else A043 = 0.0;
  if _n_ >= 29 then LS29 = 1.0;
  else LS29 = 0.0;
run;
```

Now you can refine the earlier model by including these outliers. After examining the parameter estimates and residuals (not shown) of the ARIMA(0, 1, 1) model with these regressors, the following stationary MA1 model (with regressors) appears to fit the data well.

```
proc arima data=nile;
  identify var=level crosscorr=( A07 A043 LS29 ) noprint;
  estimate q=1 noint
    input=(A07 A043 LS29) method=ml plot;
  outlier maxnum=5 alpha=0.01;
run;
```

The relevant estimation output is shown in Output 3.1.2. No outliers, at significance level 0.01, were detected.

**Output 3.1.2.** MA1 Model with Outliers

| The ARIMA Procedure                |            |                |            |                            |        |          |        |        |        |
|------------------------------------|------------|----------------|------------|----------------------------|--------|----------|--------|--------|--------|
| Maximum Likelihood Estimation      |            |                |            |                            |        |          |        |        |        |
| Parameter                          | Estimate   | Standard Error | t Value    | Pr >  t                    | Lag    | Variable | Shift  |        |        |
| MU                                 | 1109.9     | 26.65625       | 41.64      | <.0001                     | 0      | level    |        | 0      |        |
| MA1,1                              | -0.19584   | 0.10148        | -1.93      | 0.0536                     | 1      | level    |        | 0      |        |
| NUM1                               | -319.63689 | 116.50628      | -2.74      | 0.0061                     | 0      | A07      |        | 0      |        |
| NUM2                               | -376.48708 | 116.11662      | -3.24      | 0.0012                     | 0      | A043     |        | 0      |        |
| NUM3                               | -255.20007 | 31.32609       | -8.15      | <.0001                     | 0      | LS29     |        | 0      |        |
| Constant Estimate                  |            |                |            | 1109.861                   |        |          |        |        |        |
| Variance Estimate                  |            |                |            | 13761.25                   |        |          |        |        |        |
| Std Error Estimate                 |            |                |            | 117.3083                   |        |          |        |        |        |
| AIC                                |            |                |            | 1241.659                   |        |          |        |        |        |
| SBC                                |            |                |            | 1254.685                   |        |          |        |        |        |
| Number of Residuals                |            |                |            | 100                        |        |          |        |        |        |
| Autocorrelation Check of Residuals |            |                |            |                            |        |          |        |        |        |
| To Lag                             | Chi-Square | DF             | Pr > ChiSq | -----Autocorrelations----- |        |          |        |        |        |
| 6                                  | 3.43       | 5              | 0.6334     | -0.000                     | 0.007  | 0.074    | -0.100 | -0.068 | -0.109 |
| 12                                 | 9.73       | 11             | 0.5552     | -0.070                     | 0.100  | -0.107   | -0.123 | -0.083 | -0.087 |
| 18                                 | 14.36      | 17             | 0.6417     | 0.079                      | -0.071 | -0.025   | 0.132  | -0.007 | 0.094  |
| 24                                 | 17.22      | 23             | 0.7983     | 0.008                      | -0.058 | 0.055    | -0.049 | -0.073 | -0.087 |
| Outlier Summary                    |            |                |            |                            |        |          |        |        |        |
| Maximum number searched            |            |                |            | 5                          |        |          |        |        |        |
| Number found                       |            |                |            | 0                          |        |          |        |        |        |
| Significance used                  |            |                |            | 0.01                       |        |          |        |        |        |

---

### Example 3.2. Airline Data

This is a test example where an additive outlier at observation number 50 and a level shift at observation number 100 are artificially introduced in the well-known airline passenger data, the Series G in Box and Jenkins (1976).

```
data airline;
  set sashelp.air;
  logair = log(air);
  if _n_ = 50 then logair = logair - 0.25;
  if _n_ >= 100 then logair = logair + 0.5;
run;
```

The “Airline model,”  $ARIMA(0, 1, 1) \times (0, 1, 1)_{12}$ , is known to be a good fit to the unmodified log-transformed airline passenger series. The preliminary identification steps (not shown) again suggest the Airline model for the modified data because they exhibit a strong trend and seasonal behavior, and the Airline model has wide applicability in such cases.

```
proc arima data=airline;
  identify var=logair( 1, 12 ) noprint;
  estimate q=(1)(12) noint method=ml;
  outlier maxnum=3 alpha=0.01;
run;
```

A portion of the estimation and outlier detection output is shown in Output 3.2.1.

Output 3.2.1. Output of Airline Model

| The ARIMA Procedure           |          |                |            |                    |     |
|-------------------------------|----------|----------------|------------|--------------------|-----|
| Maximum Likelihood Estimation |          |                |            |                    |     |
| Parameter                     | Estimate | Standard Error | t Value    | Approx Pr >  t     | Lag |
| MA1,1                         | 0.20390  | 0.08309        | 2.45       | 0.0141             | 1   |
| MA2,1                         | 0.76150  | 0.07875        | 9.67       | <.0001             | 12  |
| Variance Estimate             |          |                | 0.004689   |                    |     |
| Std Error Estimate            |          |                | 0.068479   |                    |     |
| AIC                           |          |                | -318.305   |                    |     |
| SBC                           |          |                | -312.555   |                    |     |
| Number of Residuals           |          |                | 131        |                    |     |
| Model for variable logair     |          |                |            |                    |     |
| Period(s) of Differencing     |          |                | 1,12       |                    |     |
| No mean term in this model.   |          |                |            |                    |     |
| Outlier Summary               |          |                |            |                    |     |
| Maximum number searched       |          |                | 3          |                    |     |
| Number found                  |          |                | 3          |                    |     |
| Significance used             |          |                | 0.01       |                    |     |
| Outlier Table                 |          |                |            |                    |     |
| Obs                           | Type     | Estimate       | Chi-Square | Approx Prob> ChiSq |     |
| 100                           | Shift    | 0.49325        | 199.36     | <.0001             |     |
| 50                            | Additive | -0.27508       | 104.78     | <.0001             |     |
| 135                           | Additive | -0.10488       | 13.08      | 0.0003             |     |

Clearly the level shift at observation number 100 and the additive outlier at observation number 50 are the dominant outliers. Moreover, the corresponding regression coefficients seem to correctly estimate the size and sign of the change. You can augment the airline data with these two regressors.

```
data airline;
  set airline;
  if _n_ = 50 then A0 = 1;
  else A0 = 0.0;
  if _n_ >= 100 then LS = 1;
  else LS = 0.0;
run;
```

You can now refine the previous model by including these regressors. Note that the differencing order of the dependent series is matched to the differencing orders of the outlier regressors to get the correct “effective” outlier signatures.

```
proc arima data=airline;
  identify var=logair(1, 12)
    crosscorr=( A0(1, 12) LS(1, 12) ) noprint;
  estimate q=(1)(12) noint
    input=(A0 LS) method=ml plot;
  outlier maxnum=3 alpha=0.01;
run;
```

The estimation and outlier detection results are shown in Output 3.2.2.

**Output 3.2.2.** Airline Model with Outliers

| The ARIMA Procedure           |          |                |            |                    |      |          |       |
|-------------------------------|----------|----------------|------------|--------------------|------|----------|-------|
| Maximum Likelihood Estimation |          |                |            |                    |      |          |       |
| Parameter                     | Estimate | Standard Error | t Value    | Approx Pr >  t     | Lag  | Variable | Shift |
| MA1,1                         | 0.39529  | 0.08117        | 4.87       | <.0001             | 1    | logair   | 0     |
| MA2,1                         | 0.55870  | 0.08678        | 6.44       | <.0001             | 12   | logair   | 0     |
| NUM1                          | -0.27421 | 0.02761        | -9.93      | <.0001             | 0    | A0       | 0     |
| NUM2                          | 0.49913  | 0.02985        | 16.72      | <.0001             | 0    | LS       | 0     |
| Variance Estimate             |          |                |            | 0.001382           |      |          |       |
| Std Error Estimate            |          |                |            | 0.037173           |      |          |       |
| AIC                           |          |                |            | -482.191           |      |          |       |
| SBC                           |          |                |            | -470.69            |      |          |       |
| Number of Residuals           |          |                |            | 131                |      |          |       |
| No mean term in this model.   |          |                |            |                    |      |          |       |
| Outlier Summary               |          |                |            |                    |      |          |       |
| Maximum number searched       |          |                |            |                    | 3    |          |       |
| Number found                  |          |                |            |                    | 3    |          |       |
| Significance used             |          |                |            |                    | 0.01 |          |       |
| Outlier Table                 |          |                |            |                    |      |          |       |
| Obs                           | Type     | Estimate       | Chi-Square | Approx Prob> ChiSq |      |          |       |
| 135                           | Additive | -0.10310       | 12.63      | 0.0004             |      |          |       |
| 62                            | Additive | -0.08872       | 12.33      | 0.0004             |      |          |       |
| 29                            | Additive | 0.08686        | 11.66      | 0.0006             |      |          |       |

The output shows that a few outliers still remain to be accounted for and that the model could be refined further.

---

## References

- Box, G.E.P. and Jenkins, G.M. (1976), *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day.
- Cobb, G.W. (1978), "The Problem of the Nile: Conditional Solution to a Change Point Problem," *Biometrika*, 65, 243-251.
- de Jong, P. and Penzer, J. (1998), "Diagnosing Shocks in Time Series," *Journal of the American Statistical Association*, Vol. 93, No. 442.
- Fidley, D.F., Monsell, B.C., Bell, W.R., Otto, M.C. and Chen, B.C. (1998), "New Capabilities and Methods of the X-12-ARIMA Seasonal Adjustment Program," *Journal of Business and Economic Statistics*, 16, 127-177.
- Kohn, R. and Ansley, C. (1985), "Efficient Estimation and Prediction in Time Series Regression Models," *Biometrika*, 72, 3, 694-697.





# Chapter 4

## The MODEL Procedure

### Chapter Table of Contents

---

|                                                                  |    |
|------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                        | 69 |
| <b>SYNTAX</b> . . . . .                                          | 70 |
| ERRORMODEL Statement . . . . .                                   | 70 |
| New FIT and SOLVE Statement Syntax . . . . .                     | 71 |
| <b>DETAILS</b> . . . . .                                         | 72 |
| Multivariate t-Distribution Estimation . . . . .                 | 72 |
| Multivariate t-Distribution Simulation . . . . .                 | 73 |
| General Likelihood Estimation and Simulation . . . . .           | 75 |
| Alternate Distribution Simulation . . . . .                      | 77 |
| Mixtures of Distributions - Copulas . . . . .                    | 79 |
| <b>EXAMPLES</b> . . . . .                                        | 80 |
| Example 4.1 Cauchy Distribution Estimation . . . . .             | 80 |
| Example 4.2 Switching Regression Example . . . . .               | 82 |
| Example 4.3 Simulating from a Mixture of Distributions . . . . . | 86 |
| <b>REFERENCES</b> . . . . .                                      | 89 |



# Chapter 4

## The MODEL Procedure

---

### Overview

The MODEL procedure estimates and simulates systems of variables with correlated residuals. In prior releases, these residuals were restricted to have normal distributions with mean zero and non-constant variance. The following new distributions are now supported for multivariate estimation:

- The centered  $t$ -distribution with estimated degrees of freedom and non-constant variance
- A user-specified likelihood function

The following distributions are now supported for multivariate Monte Carlo simulation:

- Cauchy
- Chi-squared
- Empirical
- $F$
- Poisson
- $t$
- Uniform

A transformation technique is used to create a covariance matrix for generating the correct innovations in a Monte Carlo simulation.

---

## Syntax

---

### ERRORMODEL Statement

---

**ERRORMODEL** *equation-name* ~ *distribution* [CDF=( *CDF options* )];

The ERRORMODEL statement is the mechanism for specifying the distribution of the residuals. You must specify the dependent/endogenous variables or general form model name, a tilde (~), and then a distribution with its parameters. The following options are used in the ERRORMODEL statement:

#### Options to Specify the Distribution

##### **CAUCHY**( <location, scale> )

specifies the Cauchy distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

##### **CHISQUARED** ( *df* < , *nc*> )

specifies the  $\chi^2$  distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

##### **GENERAL**( *Likelihood* <, *parm1*, *parm2*, ... *parmn*> )

specifies a general log-likelihood function that you construct using SAS programming statements. *parm1*, *parm2*, ... *parmn* are optional parameters for this distribution and are used for documentation purposes only.

##### **F**( *ndf*, *ddf* < , *nc*> )

specifies the *F* distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

##### **NORMAL**( *v*<sub>1</sub> *v*<sub>2</sub> ... *v*<sub>*n*</sub> )

specifies a multivariate normal (Gaussian) distribution with mean 0 and variances *v*<sub>1</sub> through *v*<sub>*n*</sub>.

##### **POISSON**( *mean* )

specifies the Poisson distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

##### **T**( *v*<sub>1</sub> *v*<sub>2</sub> ... *v*<sub>*n*</sub>, *df* )

specifies a multivariate *t*-distribution with noncentrality 0, variance *v*<sub>1</sub> through *v*<sub>*n*</sub>, and common degrees of freedom *df*.

##### **UNIFORM**( <*left*, *right*> )

specifies the uniform distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

**Options to Specify the CDF for Simulation****CDF=( CDF(options) )**

specifies the univariate distribution that is used for simulation so that the estimation can be done for one set of distributional assumptions and the simulation for another. The CDF can be any of the distributions from the previous section with the exception of the General Likelihood. In addition, you can specify the empirical distribution of the residuals.

**EMPIRICAL= ( <TAILS=(options)> )**

uses the sorted residual data to create an empirical CDF.

**TAILS=( tail options )**

specifies how to handle the tails in computing the inverse CDF from an empirical distribution, where *Tail options* are:

NORMAL specifies the normal distribution to extrapolate the tails.

T( *df* ) specifies the *t*-distribution to extrapolate the tails.

PERCENT= *p* specifies the percent of the observations to use in constructing each tail. The default for the PERCENT= option is 10. A normal distribution or a *t*-distribution is used to extrapolate the tails to infinity. The variance for the tail distributions is obtained from the data so that the empirical CDF is continuous.

---

**New FIT and SOLVE Statement Syntax**

The following option is now available on the FIT statement and is used to output a residual covariance matrix for non-normal distributions. This covariance matrix should be used in a Monte Carlo simulation and is created by converting the residuals to a standard normal distribution and using these converted residuals to compute a covariance matrix.

**OUTSN= sas data-set**

specifies a data set that the normalized residual covariance matrix will be written to.

The following options are now available on the SOLVE statement and apply only when Monte Carlo simulation is selected.

**RESIDDATA= sas data-set**

specifies a data set that contains the residuals that are to be used in the empirical distribution. This data set can be created using the OUT= option on the FIT statement.

**WISHART=df**

specifies that a Wishart distribution with degrees of freedom *df* be used in place of the normal error covariance matrix. This option is used to model the variance of the error covariance matrix when Monte Carlo simulation is selected.

## Details

### Multivariate t-Distribution Estimation

The multivariate  $t$ -distribution is specified using the `ERRORMODEL` statement with the `T` option. Other method specifications ( `FIML` and `OLS`, for example ) are ignored when the `ERRORMODEL` statement is used for a distribution other than normal.

The probability density function for the multivariate  $t$ -distribution is

$$P_q = \frac{\Gamma(\frac{df+m}{2})}{(\pi * df)^{\frac{m}{2}} * \Gamma(\frac{df}{2}) |\Sigma(\sigma)|^{\frac{1}{2}}} * \left( 1 + \frac{\mathbf{q}'(\mathbf{y}_t, \mathbf{x}_t, \theta) \Sigma(\sigma)^{-1} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{df} \right)^{-\frac{df+m}{2}}$$

where  $m$  is the number of equations and  $df$  is the degrees of freedom.

The maximum likelihood estimators of  $\theta$  and  $\sigma$  are the  $\hat{\theta}$  and  $\hat{\sigma}$  that minimize the negative log-likelihood function:

$$\begin{aligned} l_n(\theta, \sigma) = & - \sum_{t=1}^n \ln \left( \frac{\Gamma(\frac{df+m}{2})}{(\pi * df)^{\frac{m}{2}} * \Gamma(\frac{df}{2})} * \left( 1 + \frac{q'_t \Sigma^{-1} q_t}{df} \right)^{-\frac{df+m}{2}} \right) \\ & + \frac{n}{2} * \ln(|\Sigma|) - \sum_{t=1}^n \ln \left( \left| \frac{\partial q_t}{\partial y'_t} \right| \right) \end{aligned}$$

The `ERRORMODEL` statement is used to request the  $t$ -distribution maximum likelihood estimation. An `OLS` estimation is done to obtain initial parameter estimates and `MSE.var` estimates. Use `NOOLS` to turn off this initial estimation. If the errors are distributed normally,  $t$ -distribution estimation will produce results similar to `FIML`.

The multivariate model has a single shared degrees of freedom parameter, which is estimated. The degrees of freedom parameter can also be set to a fixed value. The negative log-likelihood value and the  $l_2$  norm of the gradient of the negative log-likelihood function are shown in the estimation summary.

#### *t-Distribution Details*

Since a variance term is explicitly specified using the `ERRORMODEL` statement,  $\Sigma(\theta)$  is estimated as a correlation matrix and  $\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)$  is normalized by the variance.

The gradient of the negative log-likelihood function with respect to the degrees of freedom is

$$\begin{aligned} \frac{\partial l_n}{\partial df} = & \frac{nm}{2 df} - \frac{n}{2} \Gamma'(\frac{df+m}{2}) + \frac{n}{2} \Gamma'(\frac{df}{2}) + \\ & 0.5 \log(1 + \frac{\mathbf{q}' \Sigma^{-1} \mathbf{q}}{df}) - \frac{0.5(df+m)}{(1 + \frac{\mathbf{q}' \Sigma^{-1} \mathbf{q}}{df})} \frac{\mathbf{q}' \Sigma^{-1} \mathbf{q}}{df^2} \end{aligned}$$

The gradient of the negative log-likelihood function with respect to the parameters is

$$\frac{\partial l_n}{\partial \theta_i} = \frac{0.5(df + m)}{(1 + \mathbf{q}'\Sigma^{-1}\mathbf{q}/df)} \left[ \frac{(2\mathbf{q}'\Sigma^{-1}\frac{\partial \mathbf{q}}{\partial \theta_i})}{df} + \mathbf{q}'\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_i}\Sigma^{-1}\mathbf{q} \right] - \frac{n}{2}\text{trace}(\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_i})$$

where

$$\frac{\partial \Sigma(\theta)}{\partial \theta_i} = \frac{2}{n} \sum_{t=1}^n \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)'}{\partial \theta_i}$$

and

$$\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) = \frac{\epsilon(\theta)}{\sqrt{h(\theta)}} \in R^{m \times n}$$

The estimator of the variance-covariance of  $\hat{\theta}$  (COVB) for the  $t$ -distribution is the inverse of the likelihood Hessian. The gradient is computed analytically and the Hessian is computed numerically.

---

## Multivariate t-Distribution Simulation

To perform a Monte Carlo analysis of models that have residuals distributed as a multivariate  $t$ , use the `ERRORMODEL` statement with either the `~ t(variance, df)` option or with the `CDF=t(variance, df)` option. The `CDF=` option specifies the distribution that is used for simulation so that the estimation can be done for one set of distributional assumptions and the simulation for another.

The following is an example of estimating and simulating a system of equations with  $t$ -distributed errors using the `ERRORMODEL` statement:

```

/* generate simulation data set */
data five;
set xfrate end=last;
if last then do;
  todate = date +5;
  do date = date to todate;
    output;
  end;
end;
end;
```

The preceding `DATA` step generates the data set to request a five-days-ahead forecast. The following statements estimate and forecast the three forward-rate models of the form

$$\begin{aligned} rate_t &= rate_{t-1} + \mu * rate_{t-1} + \nu \\ \nu &= \sigma * rate_{t-1} * \epsilon \\ \epsilon &\sim N(0, 1) \end{aligned}$$

```

Title "Daily Multivariate Geometric Brownian Motion Model "
      "of D-Mark/USDollar Forward Rates";

proc model data=xfrate;

    parms df 15;          /* Give initial value to df */

    demusd1m = lag(demusd1m) + mu1m * lag(demusd1m);
    var_demusd1m = sigma1m ** 2 * lag(demusd1m **2);
    demusd3m = lag(demusd3m) + mu3m * lag(demusd3m);
    var_demusd3m = sigma3m ** 2 * lag(demusd3m ** 2);
    demusd6m = lag(demusd6m) + mu6m * lag(demusd6m);
    var_demusd6m = sigma6m ** 2 * lag(demusd6m ** 2);

    /* Specify the error distribution */
    errormodel demusd1m demusd3m demusd6m
        ~ t( var_demusd1m var_demusd3m var_demusd6m, df );

    /* output normalized S matrix */
    fit demusd1m demusd3m demusd6m / outsn=s;
run;

    /* forecast five days in advance */
    solve demusd1m demusd3m demusd6m /
        data=five sdata=s random=1500 out=monte;
    id date;
run;

    /* select out the last date -----*/
    data monte; set monte;
    if date = '10dec95'd then output;
run;

title "Distribution of demusd1m Five Days Ahead";
proc univariate data=monte noprint;
    var demusd1m ;
    histogram demusd1m / normal(noprint color=red)
        kernel(noprint color=blue) cfill=ligr;
run;

```

The Monte Carlo simulation specified in the preceding example draws from a multivariate  $t$ -distribution with constant degrees of freedom and forecasted variance and computes future states of DEMUSD1M, DEMUSD3M, and DEMUSD6M. The OUTSN= option on the FIT statement is used to specify the data set for the normalized  $\Sigma$  matrix. That is the  $\Sigma$  matrix is created by crossing the normally distributed residuals. The normally distributed residuals are created from the  $t$ -distributed residuals using the normal inverse CDF and the  $t$  CDF.

The distribution of DEMUSD1M on the fifth day is shown in Figure 4.1. The two curves overlayed on the graph are a kernel density estimation and a normal distribution fit to the results.



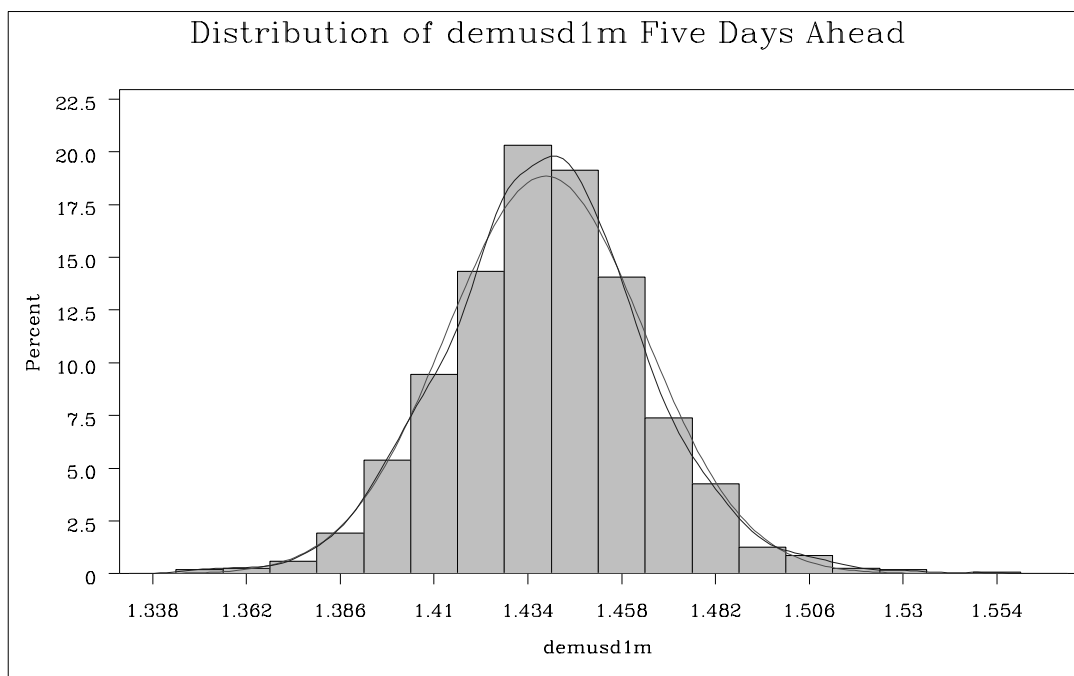


**Figure 4.1.** Distribution of DEMUSD1M

---

## General Likelihood Estimation and Simulation

The `GENERAL` option on the `ERRORMODEL` statement can be used to estimate distributions other than normal and  $t$ . The `GENERAL` option on the `ERRORMODEL` statement expects as its first argument a log-likelihood function.

When a general log-likelihood function is specified, a quasi-Newton method is used to find the minimum of the function. The standard errors are computed using the Hessian of the log-likelihood function at the solution. It is important to include all scale terms in the likelihood if correct standard errors are desired. The `OUTS=` data set is created from the cross of the residuals normalized according to the `VARDEF=` option. This `S` matrix is not normalized by any implied variance terms in the specified likelihood. This is important to consider if the `S` matrix is used in a Monte Carlo simulation.

For simulation and forecasting, the likelihood function is ignored. If Monte Carlo is requested, a CDF function can be specified using the `CDF=` option. If no CDF is specified, an empirical distribution with normal tails is used.

### ***Empirical Distribution Estimation and Simulation***

For simulation, if the CDF for the model is not built in to the procedure, you can use the `CDF=EMPIRICAL()` option. This uses the sorted residual data to create an empirical CDF. For computing the inverse CDF the program needs to know how to handle the tails. For continuous data, the tail distribution is generally poorly determined. To counter this, the `PERCENT=` option specifies the percent of the observations to use in constructing each tail. The default for the `PERCENT=` option is 10. A normal

distribution or a *t*-distribution is used to extrapolate the tails to infinity. The standard errors for this extrapolation are obtained from the data so that the empirical CDF is continuous.

The following SAS statements fit a model using least squares as the likelihood function, but represent the distribution of the residuals with an empirical CDF. The plot of the empirical probability distribution is shown in Figure 4.2.

```
data t; /* Sum of two normals */
  format date monyy.;
  do t=0 to 3 by 0.1;
    date = intnx( 'month', '1jun90'd, (t*10)-1);
    y = (0.1 * rannor(123)-10) +
        ( .5 * rannor(456)+10);
    output;
  end;
run;

proc model data=t time=t itprint;
  dependent y;
  parm a 5 ;
  y = a;
  obj = resid.y * resid.y;
  errormodel y ~ general( obj )
    cdf=(empirical=( tails=( t(15) percent= 5)));

  fit y / outns=s out=r;
  id date;
  solve y / data=t(where=(date='1jun95'd ))
    residdata=r sdata=s random=200 seed=6789 out=monte;
run;

/* Generate the pdf -----*/
proc kde data =monte out=density;
  var y;
run;

symbol1 value=none interpol=join;
proc gplot data=density;
  plot density*y ;
run;
```

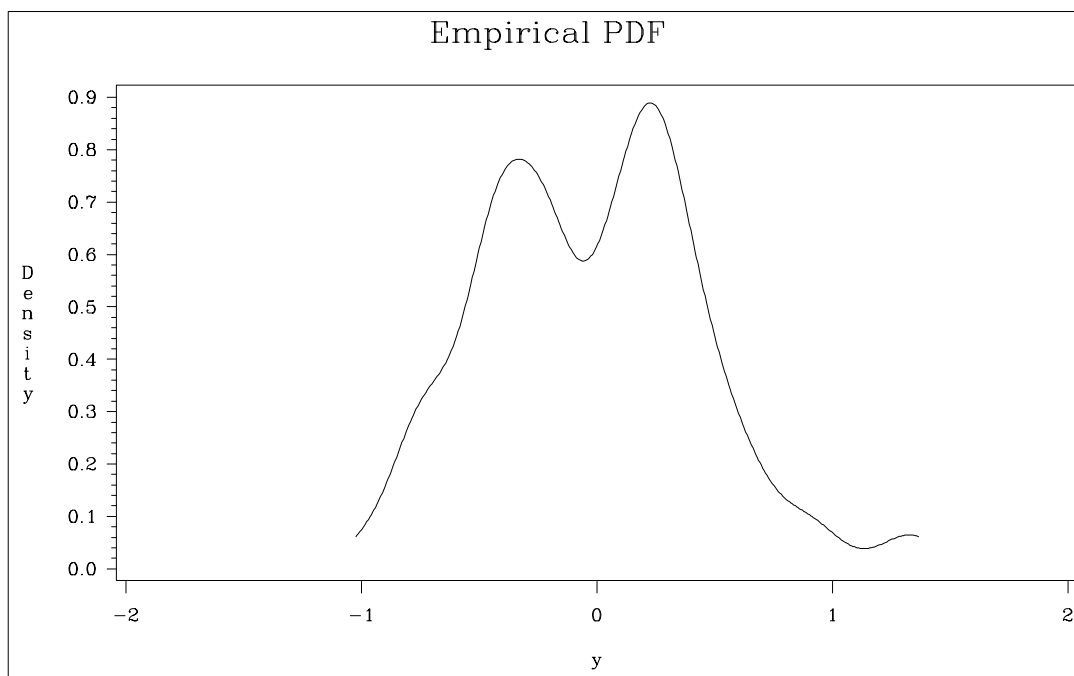


**Figure 4.2.** Empirical PDF Plot

---

## Alternate Distribution Simulation

As an alternate to the normal distribution, the `ERRORMODEL` statement can be used in a simulation to specify other distributions. The distributions available for simulation are Cauchy, Chi-squared,  $F$ , Poisson,  $t$ , and Uniform. An empirical distribution can also be used if the residuals are specified using the `RESIDDATA=` option on the `SOLVE` statement.

Except for the  $t$ , all of these alternate distributions are univariate but can be used together in a multivariate simulation. The `ERRORMODEL` statement applies to solved for equations only. That is, the normal form or general form equation referred to by the `ERRORMODEL` statement must be one of the equations you have selected in the `SOLVE` statement.

In the following example, two Poisson distributed variables are used to simulate the calls arriving and leaving a call center.

```
data s;          /* Covariance between arriving and leaving */
  arriving = 1; leaving = 0.7; _name_ = "arriving";
  output;
  arriving = 0.7; leaving = 1.0; _name_ = "leaving";
  output;
run;

data calls;
  date = '20mar2001'd;
  output;
run;
```

The first DATA step generates a data set containing a covariance matrix for the AR-RIVING and LEAVING variables. The covariance is

$$\begin{vmatrix} 1 & .7 \\ .7 & 1 \end{vmatrix}$$

```
proc model data=calls;
  arriving = 10;
  errormodel arriving ~ poisson( 10 );
  /* Have four people answering the phone */
  leaving = 4;
  errormodel leaving ~ poisson( 11 );

  waiting = arriving - leaving;
  solve arriving leaving / random=500 sdata=s out=sim;
run;

title "Distribution of Clients Waiting";
proc univariate data=sim noprint;
  var waiting ;
  histogram waiting / cfill=ligr;
run;
```

The distribution of number of waiting clients is shown in Figure 4.3.

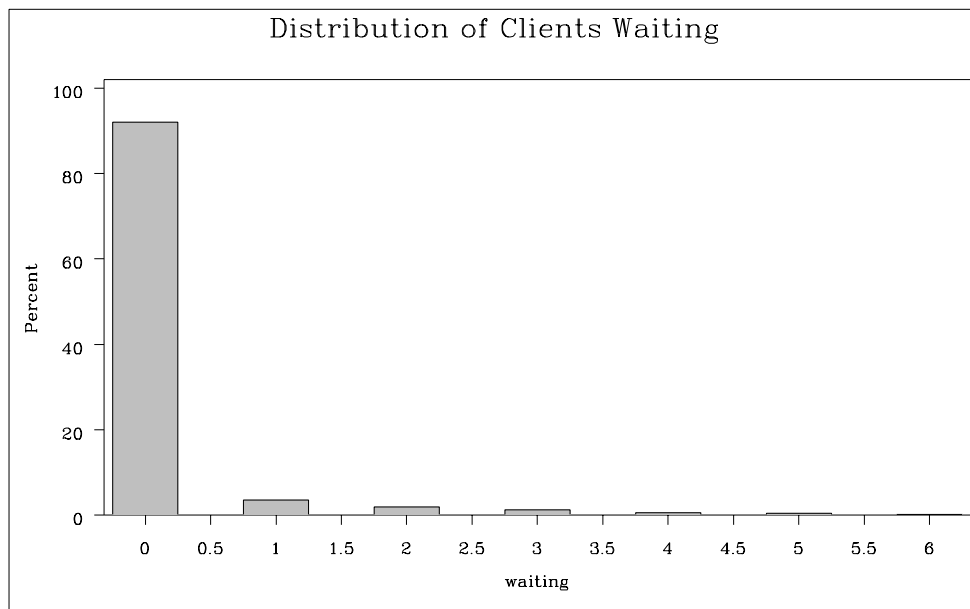


**Figure 4.3.** Distribution of Number of Clients Waiting

---

## Mixtures of Distributions - Copulas

The theory of copulas is what enables the MODEL procedure to combine and simulate multivariate distributions with different marginals. This section provides a brief overview of copulas.

Modeling a system of variables accurately is a difficult task. The underlying, ideal, distributional assumptions for each variable are usually different from each other. An individual variable may be best modeled as a  $t$ -distribution or as a Poisson process. The correlation of the various variables are very important to estimate as well. A joint estimation of a set of variables would make it possible to estimate a correlation structure but would restrict the modeling to single, simple multivariate distribution (for example, the normal). Even with a simple multivariate distribution, the joint estimation would be computationally difficult and would have to deal with issues of missing data.

Using the MODEL procedure ERRORMODEL statement you can combine and simulate from models of different distributions. The covariance matrix for the combined model is constructed using the copula induced by the multivariate normal distribution. A copula is a function that couples joint distributions to their marginal distributions.

The copula used by the model procedure is based on the multivariate normal. This particular multivariate normal has zero mean and covariance matrix  $R$ . The user provides  $R$ , which can be created using the following steps

1. Each model is estimated separately and their residuals saved.
2. The residuals for each model are converted to a normal distribution using their CDFs,  $F_i(\cdot)$ , using the relationship  $\Phi^{-1}(F(\epsilon_{it}))$ .
3. Cross these normal residuals, to create a covariance matrix  $R$ .

If the model of interest can be estimated jointly, such as multivariate T, then the OUTSN= option can be used to generate the correct covariance matrix.

A draw from this mixture of distributions is created using the following steps that are performed automatically by the MODEL procedure.

1. Independent  $N(0, 1)$  variables are generated.
2. These variables are transformed to a correlated set using the covariance matrix  $R$ .
3. These correlated normals are transformed to a uniform using  $\Phi(\cdot)$ .
4.  $F^{-1}(\cdot)$  is used to compute the final sample value.

---

## Examples

---

### Example 4.1. Cauchy Distribution Estimation

---

In this example a nonlinear model is estimated using the Cauchy distribution. Then a simulation is done for one observation in the data.

The following DATA step creates the data for the model.

```

/* Generate a Cauchy distributed Y */
data c;
  format date monyy.;
  call streaminit(156789);
  do t=0 to 20 by 0.1;
    date=intnx('month', '01jun90'd, (t*10)-1);
    x=rand('normal');
    e=rand('cauchy') + 10 ;
    y=exp(4*x)+e;
    output;
  end;
run;
```

The model to be estimated is

$$y = e^{-a x} + \epsilon$$

$$\epsilon \sim \text{Cauchy}(nc)$$

That is, the residuals of the model are distributed as a Cauchy distribution with non-centrality parameter  $nc$ .

The log likelihood for the Cauchy distribution is

$$like = -\log(1 + (x - nc)^2 * \pi)$$

The following SAS statements specify the model and the log-likelihood function.

```

title2 'Cauchy Distribution';

proc model data=c ;
  dependent y;

  parm a -2 nc 4;

  y=exp(-a*x);

  /* Likelihood function for the residuals */
  obj = log(1+(-resid.y-nc)**2 * 3.1415926);
```

```

errormodel y ~ general(obj) cdf=cauchy(nc);

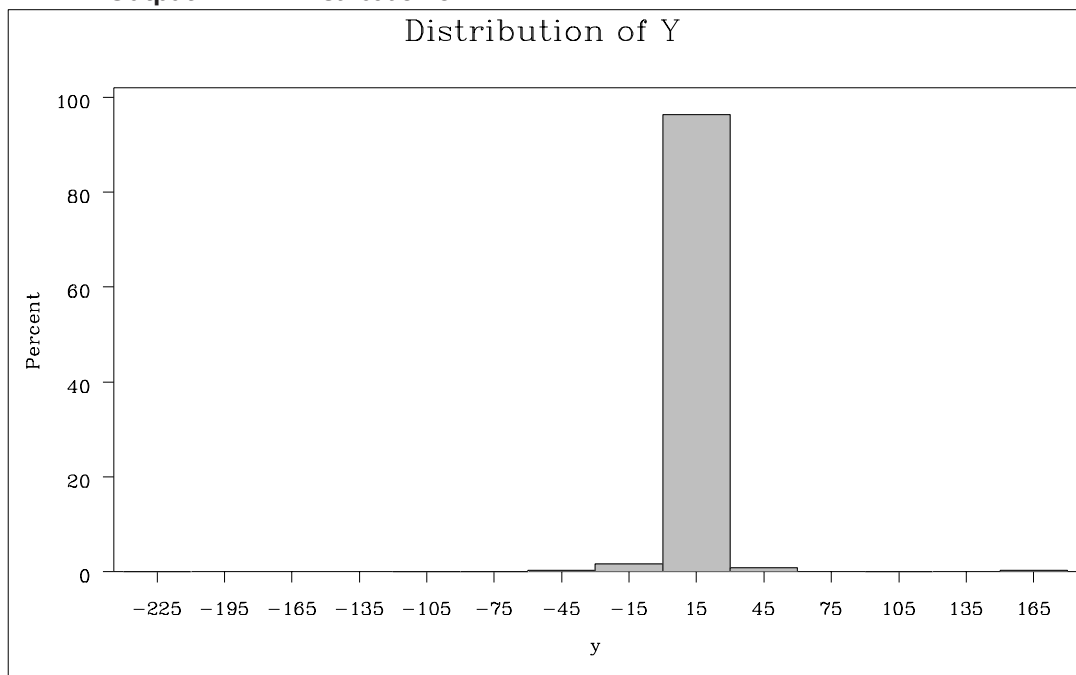
fit y / outsn=s1 method=marquardt;
solve y / sdata=s1 data=c(obs=1) random=1000
      seed=256789 out=out1;
run;

```

The FIT statement uses the OUTSN= option to put out the  $\Sigma$  matrix for residuals from the normal distribution. The  $\Sigma$  matrix is  $1 \times 1$  and has value 4.65824. The OUTS= matrix is the scalar 2989.0. Because the distribution is univariate (no covariances), the OUTS= would produce the same simulation results. The simulation is performed using the SOLVE statement.

The distribution of  $y$  is shown in Output 4.1.1.

**Output 4.1.1.** Distribution of Y



---

**Example 4.2. Switching Regression Example**

Take the usual linear regression problem

$$y = X\beta + u$$

where  $Y$  denotes the  $n$  column vector of the dependent variable,  $X$  denotes the  $(n \times k)$  matrix of independent variables,  $\beta$  denotes the  $k$  column vector of coefficients to be estimated,  $n$  denotes the number of observations ( $i=1,2,\dots,n$ ), and  $k$  denotes the number of independent variables.

You can take this basic equation and split it into two regimes, where the  $i$ th observation on  $y$  is generated by one regime or the other.

$$y_i = \sum_{j=1}^k \beta_{1j} X_{ji} + u_{1i} = x_i' \beta_1 + u_{1i}$$

$$y_i = \sum_{j=1}^k \beta_{2j} X_{ji} + u_{2i} = x_i' \beta_2 + u_{2i}$$

where  $x_{hi}$  and  $x_{hj}$  are the  $i$ th and  $j$ th observations, respectively, on  $x_h$ . The errors,  $u_{1i}$  and  $u_{2i}$ , are assumed to be distributed normally and independently, with mean zero and constant variance. The variance for the first regime is  $\sigma_1^2$ , and the variance for the second regime is  $\sigma_2^2$ . If  $\sigma_1^2 \neq \sigma_2^2$  and  $\beta_1 \neq \beta_2$ , the regression system given previously is thought to be switching between the two regimes.

The problem is to estimate  $\beta_1$ ,  $\beta_2$ ,  $\sigma_1$ , and  $\sigma_2$  without knowing *a priori* which of the  $n$  values of the dependent variable,  $y$ , was generated by which regime. If it is known *a priori* which observations belong to which regime, a simple Chow test can be used to test  $\sigma_1^2 = \sigma_2^2$  and  $\beta_1 = \beta_2$ .

Using Goldfeld and Quandt's D-method for switching regression, you can solve this problem. Assume that there exists observations on some exogenous variables  $z_{1i}, z_{2i}, \dots, z_{pi}$ , where  $z$  determines whether the  $i$ th observation is generated from one equation or the other.

$$y_i = x_i' \beta_1 + u_{1i} \quad \text{if } \sum_{j=1}^p \pi_j z_{ji} \leq 0$$

$$y_i = x_i' \beta_2 + u_{2i} \quad \text{if } \sum_{j=1}^p \pi_j z_{ji} > 0$$

where  $\pi_j$  are unknown coefficients to be estimated. Define  $d(z_i)$  as a continuous approximation to a step function. Replacing the unit step function with a continuous



approximation using the cumulative normal integral enables a more practical method that produces consistent estimates.

$$d(z_i) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\sum \pi_j z_{ji}} \exp \left[ -\frac{1}{2} \frac{\xi^2}{\sigma^2} \right] d\xi$$

$D$  is the  $n$  dimensional diagonal matrix consisting of  $d(z_i)$ .

$$D = \begin{bmatrix} d(z_1) & 0 & 0 & 0 \\ 0 & d(z_2) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & d(z_n) \end{bmatrix}$$

The parameters to estimate are now the  $k$   $\beta_1$ 's, the  $k$   $\beta_2$ 's,  $\sigma_1^2$ ,  $\sigma_2^2$ ,  $p$   $\pi$ 's, and the  $\sigma$  introduced in the  $d(z_i)$  equation. The  $\sigma$  can be considered as given *a priori*, or it can be estimated, in which the estimated magnitude provides an estimate of the success in discriminating between the two regimes (Goldfeld and Quandt 1976).

$$Y = (I - D)X\beta_1 + DX\beta_2 + W$$

where  $W = (I - D)U_1 + DU_2$ , and  $W$  is a vector of unobservable and heteroscedastic error terms. The covariance matrix of  $W$  is denoted by  $\Omega$ , where  $\Omega = (I - D)^2\sigma_1^2 + D^2\sigma_2^2$ . The maximum likelihood parameter estimates maximize the following log-likelihood function.

$$\begin{aligned} \log L = & -\frac{n}{2} \log 2\pi - \frac{1}{2} \log |\Omega| - \\ & \frac{1}{2} * [[Y - (I - D)X\beta_1 - DX\beta_2]' \Omega^{-1} [Y - (I - D)X\beta_1 - DX\beta_2]] \end{aligned}$$

As an example, you now can use this switching regression likelihood to develop a model of housing starts as a function of changes in mortgage interest rates. The data for this example is from the U.S. Census Bureau and covers the period from January 1973 to March 1999. The hypothesis is that there will be different coefficients on your model based on whether the interest rates are going up or down.

So the model for  $z_i$  will be the following

$$z_i = p * (rate_i - rate_{i-1})$$

where  $rate_i$  is the mortgage interest rate at time  $i$  and  $p$  is a scale parameter to be estimated.

The regression model will be the following

$$\begin{aligned} starts_i &= intercept_1 + ar1 * starts_{i-1} + djf1 * decjanfeb & z_i < 0 \\ starts_i &= intercept_2 + ar2 * starts_{i-1} + djf2 * decjanfeb & z_i \geq 0 \end{aligned}$$

where  $starts_i$  is the number of housing starts at month  $i$  and  $decjanfeb$  is a dummy variable indicating that the current month is one of December, January, or February.

This model is written using the following SAS statements.

```
proc model data=switch;
  parms sig1=10 sig2=10 int1 b11 b13 int2 b21 b23 p;
  bounds 0.0001 < sig1 sig2;

  a = p*dif(rate);          /* Upper bound of integral */
  d = probnorm(a);          /* Normal CDF as an approx of switch */

                                /* Regime 1 */
  y1 = int1 + zlag(starts)*b11 + decjanfeb *b13 ;
                                /* Regime 2 */
  y2 = int2 + zlag(starts)*b21 + decjanfeb *b23 ;
                                /* Composite regression equation */
  starts = (1 - d)*y1 + d*y2;

                                /* Resulting log-likelihood function */
  logL = (1/2)*( (318*log(2*3.1415)) +
    log( (sig1**2)*((1-d)**2)+(sig2**2)*(d**2) )
    + (resid.starts*( 1/( (sig1**2)*((1-d)**2)+
    (sig2**2)*(d**2) ) )*resid.starts) ) ;

  errormodel starts ~ general(logL);

fit starts / method=marquardt converge=1.0e-5;

  /* Test for significant differences in the parms */
test int1 = int2 ,/ lm;
test b11 = b21 ,/ lm;
test b13 = b23 ,/ lm;
test sig1 = sig2 ,/ lm;

run;
```

Four TEST statements were added to test the hypothesis that the parameters were the same in both regimes. The parameter estimates and ANOVA table from this run are shown in Output 4.2.1.

**Output 4.2.1.** Parameter Estimates from the Switching Regression

| Nonlinear Likelihood Summary of Residual Errors |             |                   |         |                   |          |          |             |
|-------------------------------------------------|-------------|-------------------|---------|-------------------|----------|----------|-------------|
| Equation                                        | DF<br>Model | DF<br>Error       | SSE     | MSE               | Root MSE | R-Square | Adj<br>R-Sq |
| starts                                          | 9           | 304               | 85877.9 | 282.5             | 16.8075  | 0.7806   | 0.7748      |
| Nonlinear Likelihood Parameter Estimates        |             |                   |         |                   |          |          |             |
| Parameter                                       | Estimate    | Approx<br>Std Err | t Value | Approx<br>Pr >  t |          |          |             |
| sig1                                            | 15.47451    | 0.9475            | 16.33   | <.0001            |          |          |             |
| sig2                                            | 19.77797    | 1.2710            | 15.56   | <.0001            |          |          |             |
| int1                                            | 32.82232    | 5.9070            | 5.56    | <.0001            |          |          |             |
| b11                                             | 0.739529    | 0.0444            | 16.65   | <.0001            |          |          |             |
| b13                                             | -15.456     | 3.1909            | -4.84   | <.0001            |          |          |             |
| int2                                            | 42.73243    | 6.8153            | 6.27    | <.0001            |          |          |             |
| b21                                             | 0.734112    | 0.0477            | 15.37   | <.0001            |          |          |             |
| b23                                             | -22.5178    | 4.2979            | -5.24   | <.0001            |          |          |             |
| p                                               | 25.94332    | 8.5181            | 3.05    | 0.0025            |          |          |             |

The test results shown in Output 4.2.2 suggest that the variance of the housing starts, SIG1 and SIG2, are significantly different in the two regimes. The tests also show a significant difference in the AR term on the housing starts.

**Output 4.2.2.** Parameter Estimates from the Switching Regression

| Test Results |      |           |            |             |
|--------------|------|-----------|------------|-------------|
| Test         | Type | Statistic | Pr > ChiSq | Label       |
| Test0        | L.M. | 0.02      | 0.8810     | int1 = int2 |
| Test1        | L.M. | 240001    | <.0001     | b11 = b21   |
| Test2        | L.M. | 0.02      | 0.8933     | b13 = b23   |
| Test3        | L.M. | 319354    | <.0001     | sig1 = sig2 |

### Example 4.3. Simulating from a Mixture of Distributions

This example illustrates how to perform a multivariate simulation using models that have different error distributions. Three models are used. The first model has *t*-distributed errors. The second model is a GARCH(1,1) model with normally distributed errors. The third model has a non-central Cauchy distribution.

The following SAS statements generate the data for this example. The T and the CAUCHY data sets use a common seed so that those two series will be correlated.

```
%let df = 7.5;
%let sig1 = .5;
%let var2 = 2.5;

data t;
  format date monyy.;
  do date='1jun2001'd to '1nov2002'd;
    /* t-distribution with df,sig1 */
    t = .05 * date + 5000 + &sig1*tinv(ranuni(1234),&df);
    output;
  end;
run;

data normal;
  format date monyy.;
  le = &var2;
  lv = &var2;
  do date='1jun2001'd to '1nov2002'd;
    /* Normal with GARCH error structure */
    v = 0.0001 + 0.2 * le**2 + .75 * lv;
    e = sqrt( v ) * rannor(12345) ;
    normal = 25 + e;
    le = e;
    lv = v;
    output;
  end;
run;

data cauchy;
  format date monyy.;
  PI = 3.1415926;
  do date='1jun2001'd to '1nov2002'd;
    cauchy = -4 + tan((ranuni(1234) - 0.5) * PI);
    output;
  end;
run;
```

Since the multivariate joint likelihood is unknown, the models must be estimated separately. The residuals for each model are saved using the OUT= option. Also, each model is saved using the OUTMODEL= option. The ID statement is used to provide a variable in the residual data set to merge by. The XLAG function is used

to model the GARCH(1,1) process. The XLAG function returns the lag of the first argument if it is nonmissing, otherwise it returns the second argument.

```
proc model data=t outmod=t;
parms df 10 vt 4;
    t = a * date + c;
    errormodel t ~ t( vt, df );
fit t / out=tresid;
id date;
run;

proc model data=normal outmod=normal;
normal = b0 ;
h.normal = arch0 + arch1 * xlag(resid.normal **2 , mse.normal) +
    GARCH1 * xlag(h.normal, mse.normal);

fit normal /fiml out=nresid;
id date;
run;

proc model data= cauchy outmod=cauchy;
parms nc = 1;
    /* nc is noncentrality parm to Cauchy dist */
    cauchy = nc;
    obj = log(1+resid.cauchy**2 * 3.1415926);
    errormodel cauchy ~ general(obj) cdf=cauchy(nc);

fit cauchy / out=cresid;
id date;
run;
```

The simulation requires a covariance matrix created from normal residuals. The following Data Step code uses the inverse CDFs of the  $t$  and Cauchy distributions to convert the residuals to the normal distribution. The CORR procedure is used to create a correlation matrix using the converted residuals.

```
    /* Merge and normalize the 3 residual data sets */
data c; merge tresid nresid cresid; by date;
t = probit(cdf("T", t/sqrt(0.2789), 16.58 ));
cauchy = probit(cdf("CAUCHY", cauchy, -4.0623));
run;

proc corr data=c out=s;
var t normal cauchy;
run;
```

Now the models can be simulated together using the MODEL procedure SOLVE statement. The data set created by the CORR procedure is used as the correlation matrix. Note that the errormodel statement is not saved with the model and must be restated for this simulation.

```

/* Create one observation driver data set */
data sim; merge t normal cauchy; by date;
data sim; set sim(firstobs = 519 );

proc model data=sim model=( t normal cauchy);
  errormodel t ~ t( vt, df );
  errormodel cauchy ~ cauchy(nc);
solve t cauchy normal / random=2000 seed=1962 out=monte
  sdata=s(where=(_type_="CORR"));
run;

```

An estimation of the joint density of the  $t$  and Cauchy distribution is created using the KDE procedure. Bounds were placed on the Cauchy dimension because of its fat tail behavior. The joint PDF is shown in Output 4.3.1.

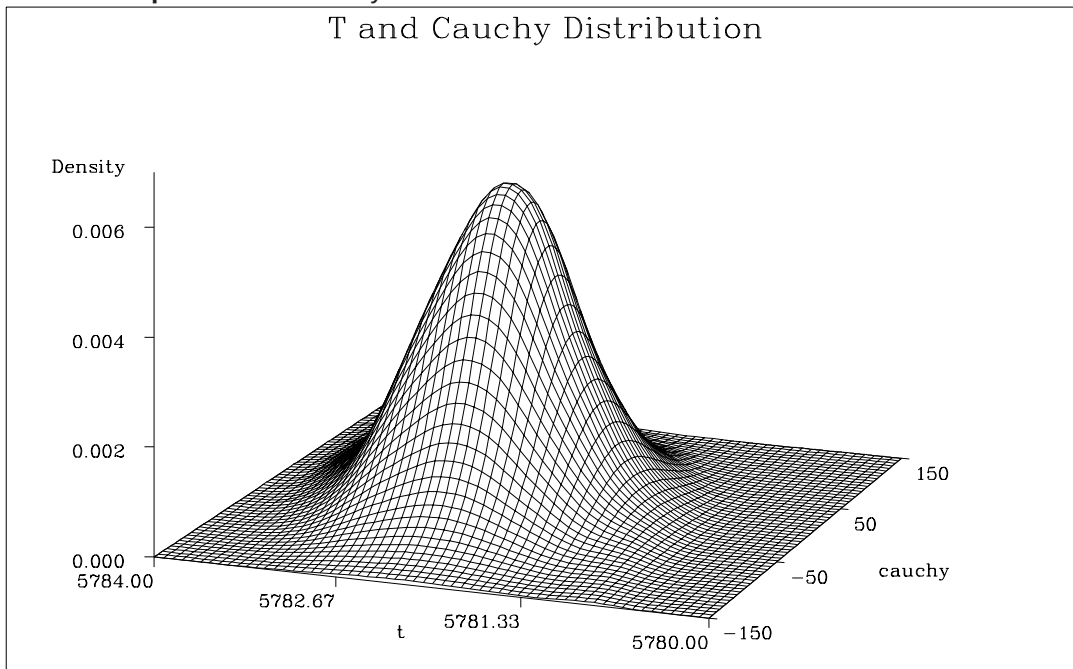
```

proc kde gridl=5780,-150 gridu=5784,150 data=monte out=density;
var t cauchy;
run;

title "T and Cauchy Distribution";
proc g3d data=density;
  plot t*cauchy=density;
run;

```

**Output 4.3.1.** Density of T and CAUCHY Truncated in the CAUCHY dimension



---

## References

- Goldfeld, S.M. and Quandt, R.E. (1972), *Nonlinear Methods in Econometrics*, Amsterdam: North-Holland Publishing Company.
- Goldfeld, S.M. and Quandt, R.E. (1973), "A Markov Model for Switching Regressions," *Journal of Econometrics*, 3-16.
- Goldfeld, S.M. and Quandt, R.E. (1973), "The Estimation of Structural Shifts by Switching Regressions," *Annals of Economic and Social Measurement*, 2/4.
- Goldfeld, S.M. and Quandt, R.E. (1976), *Studies in Nonlinear Estimation*, Cambridge, MA: Ballinger Publishing Company.
- Nelsen, Roger B. (1999), *Introduction to Copulas*, New York, New York: Springer-Verlag.





# Chapter 5

## The VARMAX Procedure

### Chapter Table of Contents

---

|                              |    |
|------------------------------|----|
| <b>OVERVIEW</b> . . . . .    | 93 |
| <b>SYNTAX</b> . . . . .      | 93 |
| MODEL Statement . . . . .    | 93 |
| RESTRICT Statement . . . . . | 94 |
| OUTEST= Data Set . . . . .   | 94 |



# Chapter 5

## The VARMAX Procedure

---

### Overview

The VARMAX procedure now includes maximum likelihood estimation for VARMA models and multivariate GARCH models:

Vector time series can now be estimated by a finite order VAR process with a finite order MA error term. The VARMA processes have infinite order pure VAR and MA representations. Impulse response analysis and forecasting VARMA processes are included.

Multivariate GARCH models now enable you to specify a functional form for the conditional covariance and also enable you to model the first and second moments jointly.

---

### Syntax

---

#### MODEL Statement

##### *Lag Specification Options*

**Q=** *number*

**Q=** (*number-list*)

specifies the order of the moving-average error process. Subset models of moving-average orders can be specified as, for example, Q=(1,5). The default is Q=0.

##### *GARCH Model Estimation Options*

**GARCH=**(**Q=***number* **P=***number* **FORM=** *value* **MEAN** )

specifies a GARCH-type multivariate conditional heteroscedasticity model. The GARCH= option in the MODEL statement specifies the family of GARCH models to be estimated. See the “??” section on page ?? for details.

The following options can be used in the GARCH= option.

**FORM=** *value*

specifies the representation for a GARCH model. Valid values are as follows:

**BEKK** specifies a *BEKK* representation. This is the default.

**BEW** specifies a vectorized representation.

**DIAG** specifies a diagonal representation.

**MEAN**

specifies the GARCH-M model.

**P=***number*

**P=(number-list)**

specifies the order of the process or the subset of GARCH terms to be fitted.  
By default, P=0.

**Q=number**

**Q=(number-list)**

specifies the order of the process or the subset of ARCH terms to be fitted. This option is required in the GARCH= option.

### Printing Options

**PRINT=(ROOTS)**

In addition, when the GARCH= option is specified, this option prints the roots of the GARCH characteristic polynomials to check covariance stationarity for the GARCH process.

---

## RESTRICT Statement

The following keyword is for the fitted moving-average model:

$MA(l, i, j)$  is the moving-average parameter of the previous lag  $l$  value of the  $j$ th error process,  $\epsilon_{j,t-l}$

The following keywords are for the fitted GARCH model. The indexes  $i$  and  $j$  refer to the position of the element in the coefficient matrix.

- $GCHM(i, j)$  is the GARCH-M parameter of  $\text{vech}(H_t)$  for  $i = 1, \dots, k$  and  $j = 1, \dots, k(k+1)/2$
- $GCHC(i, j)$  is the constant parameter of the covariance matrix,  $H_t$ , for  $1 \leq i \leq j \leq k$
- $ACH(l, i, j)$  is the ARCH parameter of the previous lag  $l$  value of  $\epsilon_t \epsilon_t'$
- $GCH(l, i, j)$  is the GARCH parameter of the previous lag  $l$  value of covariance matrix,  $H_t$

The indexes  $i$  and  $j$  for  $ACH(l, i, j)$  and  $GCH(l, i, j)$  have different ranges according to the GARCH model representation. For example,  $i, j = 1, \dots, k$  for a *BEKK* representation,  $i, j = 1, \dots, k(k+1)/2$  for a *BEW* representation, and  $i = j = 1, \dots, k(k+1)/2$  for a *diagonal* representation.

---

## OUTEST= Data Set

The following keywords are for the fitted GARCH model:

- $GCHM\_i$  are numeric variables containing the estimates of the GARCH-M parameters of the covariance matrix and their standard errors, where  $i = 1, \dots, k(k+1)/2$ ,  $k$  is the number of endogenous variables.

- $ACHl_i$  are numeric variables containing the estimates of the ARCH parameters of the covariance matrix and their standard errors, where  $l$  is the lag  $l$ th coefficient matrix and  $i = 1, \dots, k$  for a *BEKK* representation, or  $i = 1, \dots, k(k+1)/2$  for other representations.  $k$  is the number of endogenous variables.
- $GCHl_i$  are numeric variables containing the estimates of the GARCH parameters of the covariance matrix and their standard errors, where  $l$  is the lag  $l$ th coefficient matrix and  $i = 1, \dots, k$  for a *BEKK* representation, or  $i = 1, \dots, k(k+1)/2$  for other representations.  $k$  is the number of endogenous variables.
- $GCHC_i$  are numeric variables containing the estimates of the constant parameters of the covariance matrix and their standard errors, where  $i = 1, \dots, k$  for a *BEKK* representation,  $k$  is the number of endogenous variables.  $i = 1$  for other representations.



# Chapter 6

## The X12 Procedure

### Chapter Table of Contents

---

|                               |     |
|-------------------------------|-----|
| <b>OVERVIEW</b> . . . . .     | 99  |
| <b>SYNTAX</b> . . . . .       | 99  |
| BY Statement . . . . .        | 99  |
| ID Statement . . . . .        | 99  |
| FORECAST Statement . . . . .  | 100 |
| TRANSFORM Statement . . . . . | 102 |
| X11 Statement . . . . .       | 104 |
| <b>REFERENCES</b> . . . . .   | 107 |





# Chapter 6

## The X12 Procedure

---

### Overview

The X12 procedure has several new and enhanced statements, as well as some new predefined regression variables and tables:

BY and ID statements have been added. The user can control forecasting with the FORECAST statement that has the LEAD option. In the TRANSFORM statement, some standard Box-Cox transformations can be requested using the FUNCTION= options, which include the values NONE, LOG, SQRT, INVERSE, LOGISTIC, and AUTO. The X11 statement has been greatly expanded with the addition of the OUT-FORECAST, SEASONALMA=, and TRENDMA= options.

The new predefined regression variables available through the REGRESSION PREDEFINED= option are CONSTANT, LOMSTOCK, SEASONAL, TD, TD-NOLPYEAR, TD1COEF, and TD1NOPYEAR.

The following new tables are available: prior-adjustment factors (A2), chi-squared tests for groups of regressors (RegressorGroupChisq), average absolute percentage error in win-sample forecasts (AvgFcstErr), seasonal MA roots (Roots), and day of the week trading day component factors (F4).

---

### Syntax

---

#### BY Statement

*BY variables;*

A BY statement can be used with PROC X12 to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input DATA= data set to be sorted in order of the BY variables.

---

#### ID Statement

*ID variables;*

If you are creating an output data set, use the ID statement to put values of the ID variables, in addition to the table values, into the output data set. The ID statement has no effect when an output data set is not created. If the DATE= variable is specified in the MONTHLY or QUARTERLY statement, this variable is included automatically

in the OUTPUT data set. If no DATE= variable is specified, the variable \_DATE\_ is added.

The date variable (or \_DATE\_ ) values outside the range of the actual data (from forecasting) are extrapolated, while all other ID variables are missing.

---

## FORECAST Statement

### **FORECAST** *options;*

The FORECAST statement is used to forecast the time series using the estimated model. The output contains point forecast and forecast statistics for the transformed and original series.

The following option can appear in the FORECAST statement.

#### **LEAD=** *value*

Specifies the number of periods ahead to forecast. The default is the number of periods in a year (4 or 12) and the maximum is 60.

Forecasts and Standard Errors Tables are displayed in association with the FORECAST statement. Confidence limits are also included. If the data are transformed, then two tables will be displayed, one table for the original data, and one table for the transformed data.

### **REGRESSION** *options;*

The REGRESSION statement includes regression variables in a regARIMA model or specifies regression variables whose effects are to be removed by the IDENTIFY statement to aid in ARIMA model identification. Predefined regression variables are selected with the PREDEFINED option. Table A6 provides information related to trading-day effects. Table A8 provides information related to outlier factors. You should note that missing values in the input series automatically create missing value regressors. Combining your model with additional predefined regression variables may result in a singularity problem. If a singularity occurs, then you may need to alter either the model or the choices of the predefined regressors in order to successfully perform the regression.

The following options can appear in the REGRESSION statement.

**PREDEFINED= CONSTANT**  
**PREDEFINED= LOM**  
**PREDEFINED= LOMSTOCK**  
**PREDEFINED= LOQ**  
**PREDEFINED= LPYEAR**  
**PREDEFINED= SEASONAL**  
**PREDEFINED= TD**  
**PREDEFINED= TDNOLPYEAR**  
**PREDEFINED= TD1COEF**  
**PREDEFINED= TD1NOLPYEAR**

lists the predefined regression variables to be included in the model. Data values for these variables are calculated by the program, mostly as functions of the calendar. The values LOM and LOQ are actually equivalent: the actual regression is controlled by the PROC X12 SEASONS= option. Multiple predefined regression variables can be used. The syntax for using both a length-of-month and a seasonal regression could be in one of the following forms:

```
regression predefined=lom seasonal;
```

```
regression predefined=(lom seasonal);
```

```
regression predefined=lom predefined=seasonal;
```

Certain restrictions apply when using more than one predefined regression variable. Only one of TD, TDNOLPYEAR, TD1COEF, or TD1NOLPYEAR may be specified. LPYEAR cannot be used with TD, TD1COEF, LOM, LOMSTOCK, or LOQ. LOM or LOQ cannot be used with TD or TD1COEF.

**Table 6.1.** Predefined Regression Variables in X-12-ARIMA

| Regression Effect                                   | Variable Definitions                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Trend Constant<br>CONSTANT                          | $(1 - B)^{-d}(1 - B^s)^{-D}I(t \geq 1)$ ,<br>where $I(t \geq 1) = \begin{cases} 1 & \text{for } t \geq 1 \\ 0 & \text{for } t < 1 \end{cases}$                                                                                                                                                                                                                                                                                                                                  |
| Length-of-Month<br>(monthly flow)<br>LOM            | $m_t - \bar{m}$ where $m_t$ = length of month $t$ (in days)<br>and $\bar{m} = 30.4375$ (average length of month)                                                                                                                                                                                                                                                                                                                                                                |
| Stock Length-of-Month<br>LOMSTOCK                   | $SLOM_t = \begin{cases} m_t - \bar{m} - \mu(l) & \text{for } t = 1 \\ SLOM_{t-1} + m_t - \bar{m} & \text{otherwise} \end{cases}$<br>where $\bar{m}$ and $m_t$ are defined in LOM and<br>$\mu(l) = \begin{cases} 0.375 & \text{when 1st February in series is a leap year} \\ 0.125 & \text{when 2nd February in series is a leap year} \\ -0.125 & \text{when 3rd February in series is a leap year} \\ -0.375 & \text{when 4th February in series is a leap year} \end{cases}$ |
| Length-of-Quarter<br>(quarterly flow)<br>LOQ        | $q_t - \bar{q}$ where $q_t$ = length of quarter $t$ (in days)<br>and $\bar{q} = 91.3125$ (average length of quarter)                                                                                                                                                                                                                                                                                                                                                            |
| Leap Year<br>(monthly and quarterly flow)<br>LPYEAR | $LY_t = \begin{cases} 0.75 & \text{in leap year February (first quarter)} \\ -0.25 & \text{in other Februaries (first quarter)} \\ 0 & \text{otherwise} \end{cases}$                                                                                                                                                                                                                                                                                                            |
| Fixed Seasonal<br>SEASONAL                          | $M_{1,t} = \begin{cases} 1 & \text{in January} \\ -1 & \text{in December, ..., } M_{11,t} = \begin{cases} 1 & \text{in November} \\ -1 & \text{in December} \\ 0 & \text{otherwise} \end{cases} \\ 0 & \text{otherwise} \end{cases}$                                                                                                                                                                                                                                            |
| Trading Day<br>TD, TDNOLPYEAR                       | $T_{1,t} = (\text{no. of Mondays}) - (\text{no. of Sundays}), \dots$<br>$T_{6,t} = (\text{no. of Saturdays}) - (\text{no. of Sundays})$                                                                                                                                                                                                                                                                                                                                         |
| One Coefficient Trading Day<br>TD1COEF, TD1NOLPYEAR | $(\text{no. of weekdays}) - \frac{5}{2}(\text{no. of Saturdays and Sundays})$                                                                                                                                                                                                                                                                                                                                                                                                   |

## TRANSFORM Statement

### TRANSFORM options;

The TRANSFORM statement transforms or adjusts the series prior to estimating a regARIMA model. With this statement, the series can be Box-Cox (power) transformed. The Prior Adjustment Factors table is associated with the TRANSFORM statement.

Only one of the following options can appear in the TRANSFORM statement.

#### POWER= value

Transform the input series  $Y_t$  using a Box-Cox power transformation,

$$Y_t \rightarrow y_t = \begin{cases} \log(Y_t) & \lambda = 0 \\ \lambda^2 + (Y_t^\lambda - 1)/\lambda & \lambda \neq 0 \end{cases}$$

The power  $\lambda$  must be specified (for example, `POWER=.33`). The default is no transformation ( $\lambda = 1$ ); that is, `POWER=1`. The log transformation (`POWER=0`), square root transformation (`POWER=.5`), and the inverse transformation (`POWER=-1`) are equivalent to the corresponding `FUNCTION=` option.

**Table 6.2.** Power Values Related to the Census Bureau Function Argument

| <b>function=</b> | <b>transformation</b>     | <b>range for <math>Y_t</math></b> | <b>equivalent power argument</b> |
|------------------|---------------------------|-----------------------------------|----------------------------------|
| none             | $Y_t$                     | all values                        | <i>power</i> = 1                 |
| log              | $\log(Y_t)$               | $Y_t > 0$ for all $t$             | <i>power</i> = 0                 |
| sqrt             | $2(\sqrt{Y_t} - 0.875)$   | $Y_t \geq 0$ for all $t$          | <i>power</i> = 0.5               |
| inverse          | $2 - \frac{1}{Y_t}$       | $Y_t \neq 0$ for all $t$          | <i>power</i> = -1                |
| logistic         | $\log(\frac{Y_t}{1-Y_t})$ | $0 < Y_t < 1$ for all $t$         | <i>none equivalent</i>           |

**FUNCTION=NONE**

**FUNCTION=LOG**

**FUNCTION=SQRT**

**FUNCTION=INVERSE**

**FUNCTION=LOGISTIC**

**FUNCTION=AUTO**

The transformation used by `FUNCTION=NONE`, `LOG`, `SQRT`, `INVERSE`, and `LOGISTIC` is related to the `POWER=` option as shown in Table 6.2. `FUNCTION=AUTO` uses an AIC-based selection to decide between a log transformation and no transformation. The default is `FUNCTION=NONE`.

However, the `FUNCTION=` and `POWER=` options are not completely equivalent. In some cases, using the `FUNCTION=` option causes the program to automatically select other options. For instance, `FUNCTION=NONE` causes the default mode to be `MODE=ADD` in the `X11` statement. Also, the choice of transformation invoked by the `FUNCTION=AUTO` option may impact the default mode of the `X11` statement.

Note that there are restrictions on the value used in the `POWER` and `FUNCTION` options when preadjustment factors for seasonal adjustment are generated from a `regARIMA` model. When seasonal adjustment is requested with the `X11` statement, any value of the `POWER` option can be used for the purpose of forecasting the series with a `regARIMA` model. However, this is not the case when factors generated from the regression coefficients are used to adjust either the original series or the final seasonally adjusted series. In this case, the only accepted transformations are the log transformation, which can be specified as `POWER=0` (for multiplicative or log-additive seasonal adjustments) and no transformation, which can be specified as `POWER=1` (for additive seasonal adjustments). If no seasonal adjustment is performed, any `POWER` transformation can be used. The preceding restrictions also apply to `FUNCTION=NONE` and `FUNCTION=LOG`.

---

## X11 Statement

### **X11 options;**

The X11 statement is an optional statement for invoking seasonal adjustment by an enhanced version of the methodology of the Census Bureau X-11 and X-11Q programs. You can control the type of seasonal adjustment decomposition calculated with the `MODE=` option. The output includes the final tables and diagnostics for the X-11 seasonal adjustment method listed in Table 6.3.

**Table 6.3.** Tables Related to X11 Seasonal Adjustment

| Table Name | Description                                                               |
|------------|---------------------------------------------------------------------------|
| B1         | original series, adjusted for prior effects and forecast extended         |
| C17        | final weights for the irregular component                                 |
| D8         | final unmodified SI ratios (differences)                                  |
| D8A        | <i>F</i> tests for stable and moving seasonality, D8                      |
| D9         | final replacement values for extreme SI ratios (differences), D iteration |
| D9A        | moving seasonality ratios for each period                                 |
| D10        | final seasonal factors                                                    |
| D10D       | final seasonal difference                                                 |
| D11        | final seasonally adjusted series                                          |
| D12        | final trend-cycle                                                         |
| D13        | final irregular component                                                 |
| D16        | combined seasonal and trading day factors                                 |
| D16B       | final adjustment differences                                              |
| D18        | combined calendar adjustment factors                                      |
| E4         | ratio of yearly totals of original and seasonally adjusted series         |
| E5         | percent changes (differences) in original series                          |
| E6         | percent changes (differences) in seasonally adjusted series               |
| E7         | percent changes (differences) in final trend component series             |
| F2A - F2I  | X11 diagnostic summary                                                    |
| F3         | monitoring and quality assessment statistics                              |
| F4         | day of the week trading day component factors                             |
| G          | spectral plots                                                            |

For more details on the X-11 seasonal adjustment diagnostics, refer to Shiskin, Young, and Musgrave (1967), Lothian and Morry (1978), and Ladiray and Quenneville (1999).

The following options can appear in the X11 statement.

**MODE= ADD**

**MODE= MULT**

**MODE= LOGADD**

**MODE= PSEUDOADD**

determines the mode of the seasonal adjustment decomposition to be performed. There are four choices: multiplicative (MODE=MULT), additive (MODE=ADD), pseudo-additive (MODE=PSEUDOADD), and log-additive (MODE=LOGADD) decomposition. If this option is omitted, the procedure performs multiplicative adjustments. Table 6.4 shows the values of the MODE= option and the corresponding models for the original (O) and the seasonally adjusted (SA) series.

**Table 6.4.** Modes of Seasonal Adjustment and Their Models

| Value of Mode Option | Name            | Model for $O$              | Model for $SA$    |
|----------------------|-----------------|----------------------------|-------------------|
| mult                 | Multiplicative  | $O = C \times S \times I$  | $SA = C \times I$ |
| add                  | Additive        | $O = C + S + I$            | $SA = C + I$      |
| pseudoadd            | Pseudo-Additive | $O = C \times [S + I - 1]$ | $SA = C \times I$ |
| logadd               | Log-Additive    | $Log(O) = C + S + I$       | $SA = exp(C + I)$ |

**OUTFCST**

**OUTFORECAST**

determines if forecasts will be included in certain tables sent to the output data set. If OUTFORECAST is specified, then forecasted values will be included in the output data set for tables A6, A8, A16, B1, D10, and D16. The default is not to include forecasts.

**SEASONALMA=S3X1**

**SEASONALMA=S3X3**

**SEASONALMA=S3X5**

**SEASONALMA=S3X9**

**SEASONALMA=S3X15**

**SEASONALMA=STABLE**

**SEASONALMA=X11DEFAULT**

**SEASONALMA=MSR**

specifies which seasonal moving average (also called seasonal “filter”) will be used to estimate the seasonal factors. These seasonal moving averages are  $n \times m$  moving averages, meaning that an  $n$ -term simple average is taken of a sequence of consecutive  $m$ -term simple averages. X11DEFAULT is the method used by the U.S. Census Bureau’s X-11-ARIMA program. The default for PROC X12 is SEASONALMA=MSR, which is the methodology of Statistic Canada’s X-11-ARIMA/88 program.

The following seasonal filters can be selected for the entire series:

**Table 6.5.** X-12-ARIMA Seasonal Filter Options and Descriptions

| Filter Name | Description of Filter                                                                                                                                                                                                            |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| S3X1        | A $3 \times 1$ moving average.                                                                                                                                                                                                   |
| S3X3        | A $3 \times 3$ moving average.                                                                                                                                                                                                   |
| S3X5        | A $3 \times 5$ moving average.                                                                                                                                                                                                   |
| S3X9        | A $3 \times 9$ moving average.                                                                                                                                                                                                   |
| S3X15       | A $3 \times 15$ moving average.                                                                                                                                                                                                  |
| STABLE      | Stable seasonal filter. As single seasonal factor for each calendar month or quarter is generated by calculating the simple average of all the values for each month or quarter (taken after detrending and outlier adjustment). |
| X11DEFAULT  | A $3 \times 3$ moving average is used to calculate the initial seasonal factors in each iteration, and a $3 \times 5$ moving average to calculate the final seasonal factors.                                                    |
| MSR         | Choose filter automatically using moving seasonality ratio of X-11-ARIMA/88 (Dagum 1988).                                                                                                                                        |

**TRENDMA=** *value*

specifies which Henderson moving average will be used to estimate the final trend-cycle. Any odd number greater than one and less than or equal to 101 can be specified. Example: trendma=23. If no selection is made the program will select a trend moving average based on statistical characteristics of the data. For monthly series, either a 9-, 13-, or 23-term Henderson moving average will be selected. For quarterly series, the program will choose either a 5- or a 7-term Henderson moving average.



---

## References

- Ladiray, D. and Quenneville B. (2001), "Seasonal Adjustment with the X-11 Method," New York: Springer-Verlag.
- Lothian, J. and Morry M., (1978), "A Test of Quality Control Statistics for the X-11-ARIMA Seasonal Adjustment Program," Research Paper, Seasonal Adjustment and Times Series Staff, Statistics Canada.
- Shiskin, J., Young, A. H., and Musgrave, J. C. (1967), "The X-11 Variant of the Census Method II Seasonal Adjustment Program," Technical Paper No. 15, U.S. Department of Commerce, Bureau of the Census.



# Subject Index

## @

- @CRSPDB Date Informats
  - SASECRSP engine, 15
- @CRSPDR Date Informats
  - SASECRSP engine, 15
- @CRSPDT Date Informats
  - SASECRSP engine, 15

## B

- BY groups
  - X12 procedure, 99

## C

- Cauchy distribution estimation
  - example, 80
- CONTENTS procedure
  - SASECRSP engine, 6
- creating a FAME view
  - See SASEFAME engine
- CRSP and SAS Dates
  - SASECRSP engine, 14
- CRSP Date Formats
  - SASECRSP engine, 14
- CRSP Date Functions
  - SASECRSP engine, 14
- CRSP Date Informats
  - SASECRSP engine, 15
- CRSP Integer Date Format
  - SASECRSP engine, 14
- CRSPDB\_SASCAL environment variable
  - SASECRSP engine, 5
- CRSPDCI Date Functions
  - SASECRSP engine, 16
- CRSPDCS Date Functions
  - SASECRSP engine, 16
- CRSPDI2S Date Function
  - SASECRSP engine, 16
- CRSPDIC Date Functions
  - SASECRSP engine, 16
- CRSPDS2I Date Function
  - SASECRSP engine, 16
- CRSPDSC Date Functions
  - SASECRSP engine, 16
- CRSPDT Date Formats
  - SASECRSP engine, 14

## D

- DRI data files in FAME.db
  - See SASEFAME engine

- DRI/McGraw-Hill data files in FAME.db
  - See SASEFAME engine

## E

- Environment variable, CRSPDB\_SASCAL
  - SASECRSP engine, 5
- example
  - Cauchy distribution estimation, 80
  - Multivariate Mixture of Distributions, 86
  - Simulating from a Mixture of Distributions, 86
  - Switching Regression, 82
  - The D-method, 82

## F

- FAME data files
  - See SASEFAME engine
- FAME Information Services Databases
  - See SASEFAME engine

## G

- Gaussian distribution
  - MODEL procedure, 70

## I

- ID variables
  - X12 procedure, 99

## L

- LIBNAME interface engine for FAME database
  - See SASEFAME engine
- libname statement
  - SASECRSP engine, 5
- LIBNAME statement
  - SASEFAME engine, 31

## M

- main economic indicators (OECD) data files in FAME.db
  - See SASEFAME engine
- MODEL procedure
  - Gaussian distribution, 70
  - multivariate  $t$ -distribution, 72
  - normal distribution, 70
- multivariate  $t$ -distribution
  - MODEL procedure, 72
- Multivariate Mixture of Distributions
  - example, 86

**N**

national accounts data files (OECD) in FAME.db  
     See SASEFAME engine  
 negative log-likelihood function, 72  
 Nonlinear time series analysis  
     VARMAX procedure, 93  
 normal distribution  
     MODEL procedure, 70

**O**

OECD data files in FAME.db  
     See SASEFAME engine  
 Organization for Economic Cooperation and Development data files in FAME.db  
     See SASEFAME engine

**R**

reading from CRSP data files  
     SASECRSP engine, 5  
 Remote FAME Access, using FAME CHLI  
     SASEFAME engine, 31  
 Remote FAME Access, using SAS CONNECT  
     SASEFAME engine, 31

**S**

SAS and CRSP Dates  
     SASECRSP engine, 14  
 SAS DATA step  
     SASECRSP engine, 6  
 SAS Date Format  
     SASECRSP engine, 14  
 SAS output data set  
     SASECRSP engine, 9  
 SASECRSP engine  
     @CRSPDB Date Informats, 15  
     @CRSPDR Date Informats, 15  
     @CRSPDT Date Informats, 15  
     CONTENTS procedure, 6  
     CRSP and SAS Dates, 14  
     CRSP Date Formats, 14  
     CRSP Date Functions, 14  
     CRSP Date Informats, 15  
     CRSP Integer Date Format, 14  
     CRSPDB\_SASCAL environment variable, 5  
     CRSPDCI Date Functions, 16  
     CRSPDCS Date Functions, 16  
     CRSPDI2S Date Function, 16  
     CRSPDIC Date Functions, 16  
     CRSPDS2I Date Function, 16  
     CRSPDSC Date Functions, 16  
     CRSPDT Date Formats, 14  
     Environment variable, CRSPDB\_SASCAL, 5  
     libname statement, 5  
     reading from CRSP data files, 5  
     SAS and CRSP Dates, 14  
     SAS DATA step, 6  
     SAS Date Format, 14  
     SAS output data set, 9  
     setid option, 5

SQL procedure, creating a view, 6  
 SQL procedure, using clause, 6

**SASEFAME engine**

creating a FAME view, 31  
 DRI data files in FAME.db, 31  
 DRI/McGraw-Hill data files in FAME.db, 31  
 FAME data files, 31  
 FAME Information Services Databases, 31  
 LIBNAME interface engine for FAME databases, 31  
 LIBNAME statement, 31  
 main economic indicators (OECD) data files in FAME.db, 31  
 national accounts data files (OECD) in FAME.db, 31  
 OECD data files in FAME.db, 31  
 Organization for Economic Cooperation and Development data files in FAME.db, 31  
 Remote FAME Access, using FAME CHLI, 31  
 Remote FAME Access, using SAS CONNECT, 31  
 viewing a FAME database, 31

**setid option**

SASECRSP engine, 5

Simulating from a Mixture of Distributions  
 example, 86

SQL procedure, creating a view

SASECRSP engine, 6

SQL procedure, using clause

SASECRSP engine, 6

Switching Regression

example, 82

**T**

The D-method  
 example, 82

**V**

VARMAX procedure

Nonlinear time series analysis, 93

viewing a FAME database

See SASEFAME engine

**X**

X12 procedure

BY groups, 99

ID variables, 99

# Syntax Index

## A

ALPHA= option  
OUTLIER statement (ARIMA), 55

## B

BY statement  
X12 procedure, 99

## C

CAUCHY option  
ERRORMODEL statement (MODEL), 70  
CDF= option  
ERRORMODEL statement (MODEL), 71  
CHISQUARED option  
ERRORMODEL statement (MODEL), 70

## E

EMPIRICAL= option  
ERRORMODEL statement (MODEL), 71

## F

F option  
ERRORMODEL statement (MODEL), 70  
FORECAST statement  
X12 procedure, 100  
FUNCTION= option  
TRANSFORM statement (X12), 103

## G

GARCH option  
MODEL statement, 93  
GENERAL= option  
ERRORMODEL statement (MODEL), 70

## I

ID statement  
X12 procedure, 99  
IDENTIFY statement  
ARIMA procedure, 55  
INSET= option  
LIBNAME statement (SASECRSP), 8

## L

LEAD= option  
FORECAST statement (X12), 100  
LIBNAME *libref* SASEFAME statement, 32  
LIBNAME *libref* SASECRSP statement, 7

## M

MAXNUM= option  
OUTLIER statement (ARIMA), 56  
MAXPCT= option  
OUTLIER statement (ARIMA), 56  
MODE= option  
X11 statement (X12), 105

## N

NORMAL option  
ERRORMODEL statement (MODEL), 70

## O

OUTFORECAST option  
X11 statement (X12), 105

## P

PERMNO= option  
LIBNAME statement (SASECRSP), 7  
POISSON option  
ERRORMODEL statement (MODEL), 70  
POWER= option  
TRANSFORM statement (X12), 102  
PREDEFINED= option  
PREDEFINED statement (X12), 101

## Q

Q= option  
MODEL statement (VARMAX), 93

## R

RANGE= option  
LIBNAME statement (SASECRSP), 7  
REGRESSION statement  
X12 procedure, 100  
Remote FAME data access  
physical name using #port number, 32

## S

SEASONALMA= option  
X11 statement (X12), 105  
SETID= option  
LIBNAME statement (SASECRSP), 7  
SIGMA= option  
OUTLIER statement (ARIMA), 55

## **T**

T option

    ERRORMODEL statement (MODEL), 70

t option

    ERRORMODEL statement (MODEL), 72

TRANSFORM statement

    X12 procedure, 102

TRENDMA= option

    X11 statement (X12), 106

TYPE= option

    OUTLIER statement (ARIMA), 55

## **U**

U option

    ERRORMODEL statement (MODEL), 70

## **X**

X11 statement

    X12 procedure, 104

# Your Turn

---

If you have comments or suggestions about *SAS/ETS® Software: Changes and Enhancements, Release 8.2*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute Inc.  
SAS Publishing  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute Inc.  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [suggest@sas.com](mailto:suggest@sas.com)





*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS® Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

- 1.** Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
- 2.** Leave the "Subject" line blank.
- 3.** Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.



SAS Publishing

TECHNICAL REPORT

# **SAS/IML<sup>®</sup> Software: Changes and Enhancements, Release 8.2**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/IML® Software: Changes and Enhancements, Release 8.2*, Cary, NC: SAS Institute Inc., 2001

**SAS/IML® Software: Changes and Enhancements, Release 8.2**  
Copyright © 2001 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-867-0

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2001

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

|                                                                   |    |
|-------------------------------------------------------------------|----|
| Chapter 1. Wavelet Analysis . . . . .                             | 1  |
| Chapter 2. Fractionally Integrated Time Series Analysis . . . . . | 35 |
| Subject Index . . . . .                                           | 47 |
| Syntax Index . . . . .                                            | 49 |



# Chapter 1

## Wavelet Analysis

### Chapter Table of Contents

---

|                                                                    |    |
|--------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                          | 3  |
| Some Brief Mathematical Preliminaries . . . . .                    | 3  |
| <b>GETTING STARTED</b> . . . . .                                   | 5  |
| Creating the Wavelet Decomposition . . . . .                       | 7  |
| Wavelet Coefficient Plots . . . . .                                | 10 |
| Multiresolution Approximation Plots . . . . .                      | 13 |
| Multiresolution Decomposition Plots . . . . .                      | 16 |
| Wavelet Scalograms . . . . .                                       | 17 |
| Reconstructing the Signal from the Wavelet Decomposition . . . . . | 20 |
| <b>SYNTAX</b> . . . . .                                            | 22 |
| Wavelet Analysis Calls . . . . .                                   | 22 |
| WAVFT Call . . . . .                                               | 22 |
| WAVGET Call . . . . .                                              | 24 |
| WAVIFT Call . . . . .                                              | 26 |
| WAVPRINT Call . . . . .                                            | 28 |
| WAVTHRSH Call . . . . .                                            | 29 |
| <b>DETAILS</b> . . . . .                                           | 30 |
| Using Symbolic Names . . . . .                                     | 30 |
| Obtaining Help for the Wavelet Macros and Modules . . . . .        | 32 |
| <b>REFERENCES</b> . . . . .                                        | 32 |





# Chapter 1

## Wavelet Analysis

---

### Overview

Wavelets are a versatile tool for understanding and analyzing data, with important applications in nonparametric modeling, pattern recognition, feature identification, data compression, and image analysis. Wavelets provide a description of your data that localizes information at a range of scales and positions. Moreover, they can be computed very efficiently, and there is an intuitive and elegant mathematical theory to guide you in applying them.

---

### Some Brief Mathematical Preliminaries

The discrete wavelet transform decomposes a function as a sum of basis functions called wavelets. These basis functions have the property that they can be obtained by dilating and translating two basic types of wavelets known as the *scaling function* or *father wavelet*  $\phi$ , and the *mother wavelet*  $\psi$ . These translates and dilations are defined as follows:

$$\begin{aligned}\phi_{j,k}(x) &= 2^{j/2} \phi(2^j x - k) \\ \psi_{j,k}(x) &= 2^{j/2} \psi(2^j x - k)\end{aligned}$$

The index  $j$  defines the dilation or *level* while the index  $k$  defines the translate. Loosely speaking, sums of the  $\phi_{j,k}(x)$  capture low frequencies and sums of the  $\psi_{j,k}(x)$  represent high frequencies in the data. More precisely, for any suitable function  $f(x)$  and for any  $j_0$ ,

$$f(x) = \sum_k c_k^{j_0} \phi_{j_0,k}(x) + \sum_{j \geq j_0} \sum_k d_k^j \psi_{j,k}(x)$$

where the  $c_k^j$  and  $d_k^j$  are known as the scaling coefficients and the detail coefficients respectively. For orthonormal wavelet families these coefficients can be computed by

$$\begin{aligned}c_k^j &= \int f(x) \phi_{j,k}(x) dx \\ d_k^j &= \int f(x) \psi_{j,k}(x) dx\end{aligned}$$

The key to obtaining fast numerical algorithms for computing the detail and scaling coefficients for a given function  $f(x)$  is that there are simple recurrence relationships

that enable you to compute the coefficients at level  $j - 1$  from the values of the scaling coefficients at level  $j$ . These formulae are

$$\begin{aligned} c_k^{j-1} &= \sum_i h_{i-2k} c_i^j \\ d_k^{j-1} &= \sum_i g_{i-2k} c_i^j \end{aligned}$$

The coefficients  $h_k$  and  $g_k$  that appear in these formulae are called *filter coefficients*. The  $h_k$  are determined by the father wavelet and they form a low-pass filter;  $g_k = (-1)^k h_{1-k}$  and form a high-pass filter. The preceding sums are formally over the entire (infinite) range of integers. However, for wavelets that are zero except on a finite interval, only finitely many of the filter coefficients are non-zero and so in this case the sums in the recurrence relationships for the detail and scaling coefficients are finite.

Conversely, if you know the detail and scaling coefficients at level  $j - 1$  then you can obtain the scaling coefficients at level  $j$  using the relationship

$$c_k^j = \sum_i h_{k-2i} c_i^{j-1} + \sum_i g_{k-2i} d_i^{j-1}$$

Suppose that you have data values

$$y_k = f(x_k), \quad k = 0, 1, 2, \dots, N - 1$$

at  $N = 2^J$  equally spaced points  $x_k$ . It turns out that the values  $2^{-J/2} y_k$  are good approximations of the scaling coefficients  $c_k^J$ . Then using the recurrence formula you can find  $c_k^{J-1}$  and  $d_k^{J-1}$ ,  $k = 0, 1, 2, \dots, N/2 - 1$ . The discrete wavelet transform of the  $y_k$  at level  $J - 1$  consists of the  $N/2$  scaling and  $N/2$  detail coefficients at level  $J - 1$ . A technical point that arises is that in applying the recurrence relationships to finite data, a few values of the  $c_k^J$  for  $k < 0$  or  $k \geq N$  may be needed. One way to cope with this difficulty is to extend the sequence  $c_k^J$  to the left and right using some specified boundary treatment.

Continuing by replacing the scaling coefficients at any level  $j$  by the scaling and detail coefficients at level  $j - 1$  yields a sequence of  $N$  coefficients

$$\{c_0^0, d_0^0, d_0^1, d_1^1, d_0^2, d_1^2, d_2^2, d_3^2, \dots, d_7^2, \dots, d_0^{J-1}, \dots, d_{N/2-1}^{J-1}\}$$

This sequence is the finite discrete wavelet transform of the input data  $\{y_k\}$ . At any level  $j_0$  the finite dimensional approximation of the function  $f(x)$  is

$$f(x) \approx \sum_k c_k^{j_0} \phi_{j_0,k}(x) + \sum_{j=j_0}^{J-1} \sum_k d_k^j \psi_{j,k}(x)$$

## Getting Started

Fourier Transform Infrared (FT-IR) spectroscopy is an important tool in analytic chemistry. This example demonstrates wavelet analysis applied to an FT-IR spectrum of quartz (Sullivan 2000). The following DATA step creates a data set containing the spectrum, expressed as an absorbance value for each of 850 wave numbers.

```
data quartzInfraredSpectrum;
    WaveNumber=4000.6167786 - _N_ *4.00084378;
    input Absorbance @@;
datalines;
4783 4426 4419 4652 4764 4764 4621 4475 4430 4618
4735 4735 4655 4538 4431 4714 4738 4707 4627 4523
4512 4708 4802 4811 4769 4506 4642 4799 4811 4732
4583 4676 4856 4868 4796 4849 4829 4677 4962 4994
4924 4673 4737 5078 5094 4987 4632 4636 5010 5166
5166 4864 4547 4682 5161 5291 5143 4684 4662 5221
5640 5640 5244 4791 4832 5629 5766 5723 5121 4690
5513 6023 6023 5503 4675 5031 6071 6426 6426 5723
5198 5943 6961 7135 6729 5828 6511 7500 7960 7960
7299 6484 7257 8180 8542 8537 7154 7255 8262 8898
8898 8263 7319 7638 8645 8991 8991 8292 7309 8005
9024 9024 8565 7520 7858 8652 8966 8966 8323 7513
8130 8744 8879 8516 7722 8099 8602 8729 8726 8238
7885 8350 8600 8603 8487 7995 8194 8613 8613 8408
7953 8236 8696 8696 8552 8102 7852 8570 8818 8818
8339 7682 8535 9038 9038 8503 7669 7794 8864 9163
9115 8221 7275 8012 9317 9317 8512 7295 7623 9021
9409 9338 8116 6860 7873 9282 9490 9191 7012 7392
9001 9483 9457 8107 6642 7695 9269 9532 9246 7641
6547 8886 9457 9457 8089 6535 7537 9092 9406 9178
7591 6470 7838 9156 9222 7974 6506 7360 8746 9057
8877 7455 6504 7605 8698 8794 8439 7057 7202 8240
8505 8392 7287 6634 7418 8186 8229 7944 6920 6829
7499 7949 7831 7057 6866 7262 7626 7626 7403 6791
7062 7289 7397 7397 7063 6985 7221 7221 7199 6977
7088 7380 7380 7195 6957 6847 7426 7570 7508 6952
6833 7489 7721 7718 7254 6855 7132 7914 8040 7880
7198 6864 7575 8270 8229 7545 7036 7637 8470 8570
8364 7591 7413 8195 8878 8878 8115 7681 8313 9102
9185 8981 8283 8197 8932 9511 9511 9101 8510 8670
9686 9709 9504 8944 8926 9504 9964 9964 9627 9212
9366 9889 10100 9939 9540 9512 9860 10121 10121 9828
9567 9513 9782 9890 9851 9510 9385 9339 9451 9451
9181 9076 9015 8960 9014 8957 8760 8760 8602 8584
8584 8459 8469 8373 8279 8327 8282 8341 8341 8155
8260 8260 8250 8350 8245 8358 8403 8355 8490 8490
8439 8689 8689 8621 8680 8661 8897 9028 8900 8873
8873 9187 9377 9377 9078 9002 9147 9635 9687 9535
9127 9242 9824 9928 9775 9200 9047 9572 10102 10102
9631 9024 9209 10020 10271 9830 9062 9234 10154 10483
10453 9582 9011 9713 10643 10701 10372 9368 9857 10865
10936 10572 9574 9691 10820 11452 11452 10623 9903 10787
```

```

11931 12094 11302 10604 11458 12608 12808 12589 11629 11795
12863 13575 13575 12968 12498 13268 14469 14469 13971 13727
14441 15334 15515 15410 14986 15458 16208 16722 16722 16618
17061 17661 18089 18089 18184 18617 19015 19467 19633 19830
20334 20655 20947 21347 21756 22350 22584 22736 22986 23412
24126 24498 24501 24598 24986 25729 26356 26356 26271 26754
27624 28162 28162 28028 28305 29223 30073 30219 30185 30308
31831 32699 32819 32793 33320 34466 35600 36038 36086 36518
37517 38765 39462 39681 40209 41243 42274 42772 42876 43172
43929 44842 45351 45395 45551 46035 46774 47353 47353 47362
47908 48539 48936 48978 49057 49497 50101 50670 50914 51134
51603 52276 53007 53399 53769 54281 54815 54914 55365 55874
56180 56272 56669 57076 57422 57458 57525 57681 57679 57318
57318 57181 57417 57409 57144 57047 56377 56551 56483 56098
56034 55598 55364 55364 55146 54904 54990 55501 55533 55362
54387 55340 55240 54748 53710 55346 55795 55795 55060 55945
55945 55753 56759 56859 57509 56741 56273 56961 58566 58566
58104 59275 59275 59051 59090 59461 60362 60560 61103 61272
61380 61878 62067 62237 62214 61182 61532 62173 62253 60473
61346 63143 63378 61519 61753 63078 63841 63841 62115 61227
63237 63237 61338 63951 63951 63604 63633 64625 65135 64976
63630 63494 63834 63338 63218 62324 64131 64234 65122 64551
64127 64415 64621 64621 63142 65344 65585 65476 65074 64714
63803 65085 65085 65646 65646 64851 65390 65390 64997 65541
65587 65682 65952 65952 65390 65702 65846 65734 65734 65628
65509 65571 65636 65636 65620 65487 65544 65547 65738 65758
65711 65360 65362 65362 65231 65333 65453 65473 65435 65302
65412 65412 65351 65242 65242 65170 65221 65297 65297 65202
65177 65183 65184 65179 65209 65209 65144 65134 65113 65009
64919 64945 64988 64988 64856 64686 64529 64370 64282 64233
64169 63869 63685 63480 63373 63349 63307 63131 63017 62885
62736 62736 62706 62666 62622 62671 62781 62853 62950 63106
63135 63141 63220 63263 63489 63807 63966 64132 64294 64612
64841 64985 65159 65204 65259 65540 65707 65749 65732 65719
65820 65895 65925 65925 65888 65937 66059 66109 66109 66078
66007 65897 65897 65747 65490 64947 64598 64363 64140 63801
63571 63395 63333 63442 63442 63339 63196 62911 62118 61795
61454 61456 61607 62025 62190 62190 62023 61780 61502 61482
61458 61320 61015 60852 60708 60684 60522 60488 60506 60640
60797 60995 61141 61141 61036 60664 60522 60017 59681 59129
58605 58035 57192 56137 54995 53586 52037 50283 48565 45419
43341 41111 36131 35377 34431 31679 29237 26898 24655 22417
19876 17244 15176 12575 10532 8180 6040 4059 2210 575
;

```

The following statements produce the line plot of these data displayed in Figure 1.1.

```

symbol1 c=black i=join v=none;
proc gplot data=quartzInfraredSpectrum;
  plot Absorbance*WaveNumber/
    hminor = 0    vminor = 0
    vaxis = axis1
    hreverse frame;

```

```
axis1 label = ( r=0 a=90 );
run;
```

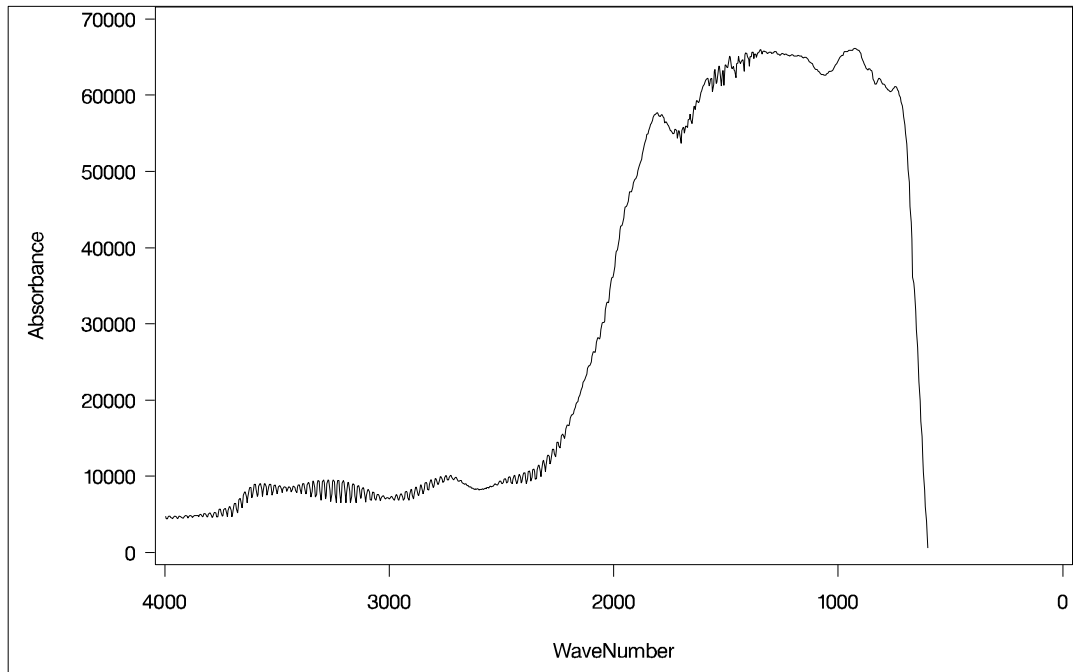


**Figure 1.1.** FT-IR Spectrum of Quartz

This data contains information at two distinct scales, namely a low frequency underlying curve superimposed with a high frequency oscillation. Notice that the oscillation is not uniform but that it occurs in several distinct bands. Wavelet analysis is an appropriate tool for providing insight into this type of data as it enables you to identify the frequencies present in the absorbance data as the wave number changes. This property of wavelets is known as “time frequency localization”; in this case the role of time is played by WaveNumber. Also note that the dependent variable Absorbance is measured at equally spaced values of the independent variable WaveNumber. This condition is necessary for the direct use of the discrete wavelet transform that is implemented in the SAS/IML wavelet functions.

---

## Creating the Wavelet Decomposition

The following SAS code starts the wavelet analysis:

```
%wavginit;
proc iml;
  %wavinit;
```

Notice that the previous code segment includes two SAS macro calls. You can use the IML wavelet functions without using the WAVGINIT and WAVINIT macros. The macros are called to initialize and load IML modules that you can use to produce several standard wavelet diagnostic plots. These macros have been provided as autocall macros that you can invoke directly in your SAS code.

The WAVGINIT macro must be called prior to invoking PROC IML. This macro defines several macro variables that are used to adjust the size, aspect ratio, and font size for the plots produced by the wavelet plot modules. This macro can also take several optional arguments that control the positioning and size of the wavelet diagnostic plots. See “Obtaining Help for Wavelet Modules and Macros” on page 32 for details on getting help about this macro call.

The WAVINIT macro must be called from within PROC IML. It loads the IML modules that you can use to produce wavelet diagnostic plots. This macro also defines symbolic macro variables that you can use to improve the readability of your code.

The following statements read the absorbance variable into an IML vector:

```
use quartzInfraredSpectrum;  
read all var{Absorbance} into absorbance;
```

You are now in a position to begin the wavelet analysis. The first step is to set up the options vector that specifies which wavelet and what boundary handling you want to use. You do this as follows:

```
optn                = &waveSpec;    /* optn=j(1,4,.); */  
optn[&family]      = &daubechies; /* optn[3] = 1; */  
optn[&member]      = 3;           /* optn[4] = 3; */  
optn[&boundary]    = &polynomial; /* optn[1] = 3; */  
optn[&degree]      = &linear;     /* optn[2] = 1; */
```

These statements use macro variables that are defined in the WAVINIT macro. The equivalent code without using these macro variables is given in the adjacent comments. As indicated by the suggestive macro variable names, this options vector specifies that the wavelet to be used is the third member of the Daubechies wavelet family and that boundaries are to be handled by extending the signal as a linear polynomial at each endpoint.

The next step is to create the wavelet decomposition with the following call:

```
call wavft(decomp,absorbance,optn);
```

This call computes the wavelet transform specified by the vector `optn` of the input vector `absorbance`. The specified transform is encapsulated in the vector `decomp`. This vector is not intended to be used directly. Rather you use this vector as an argument to other IML wavelet subroutines and plot modules. For example, you use the WAVPRINT subroutine to print the information encapsulated in a wavelet decomposition. The following code produces output in Figure 1.2.

```
call wavprint(decomp,&summary);  
call wavprint(decomp,&detailCoeffs,1,4);
```

| Decomposition Summary                  |             |                            |             |             |
|----------------------------------------|-------------|----------------------------|-------------|-------------|
| Decomposition Name                     |             | DECOMP                     |             |             |
| Wavelet Family                         |             | Daubechies Extremal Phase  |             |             |
| Family Member                          |             | 3                          |             |             |
| Boundary Treatment                     |             | Recursive Linear Extension |             |             |
| Number of Data Points                  |             | 850                        |             |             |
| Start Level                            |             | 0                          |             |             |
| Wavelet Detail Coefficients for DECOMP |             |                            |             |             |
| Translate                              | Level 1     | Level 2                    | Level 3     | Level 4     |
| 0                                      | -1.70985E-9 | 1.31649E-10                | -8.6402E-12 | 5.10454E-11 |
| 1                                      | 1340085.30  | -128245.70                 | 191.084707  | 4501.36     |
| 2                                      |             | 62636.70                   | 6160.27     | -1358.23    |
| 3                                      |             | -238445.36                 | -54836.56   | -797.724143 |
| 4                                      |             |                            | 39866.95    | 676.034389  |
| 5                                      |             |                            | -28836.85   | -5166.59    |
| 6                                      |             |                            | 223421.00   | -6088.99    |
| 7                                      |             |                            |             | -5794.67    |
| 8                                      |             |                            |             | 30144.74    |
| 9                                      |             |                            |             | -3903.53    |
| 10                                     |             |                            |             | 638.063264  |
| 11                                     |             |                            |             | -10803.45   |
| 12                                     |             |                            |             | 33616.35    |
| 13                                     |             |                            |             | -50790.30   |

**Figure 1.2.** Output of WAVPRINT CALLS

Usually such displayed output is of limited use. More frequently you will want to represent the transformed data graphically or use the results in further computational routines. As an example, you can estimate the noise level of the data using a robust measure of the standard deviation of the highest level detail coefficients, as demonstrated in the following statements:

```
call wavget(tLevel,decomp,&topLevel);
call wavget(noiseCoeffs,decomp,&detailCoeffs,tLevel-1);

noiseScale=mad(noiseCoeffs,"nmad");
print "Noise scale = " noiseScale;
```

The result is shown in Figure 1.3;

| NOISESCALE    |           |
|---------------|-----------|
| Noise scale = | 169.18717 |

**Figure 1.3.** Scale of Noise in the Absorbance Data

The first WAVGET call is used to obtain the top level number in the wavelet decomposition `decomp`. The highest level of detail coefficients are defined at one level below the top level in the decomposition. The second WAVGET call returns these coefficients in the vector `noiseCoeffs`. Finally, the MAD function computes a robust estimate of the standard deviation of these coefficients.

## Wavelet Coefficient Plots

Diagnostic plots greatly facilitate the interpretation of a wavelet decomposition. One standard plot is the detail coefficients arranged by level. Using a module included by the WAVINIT macro call, you can produce the plot shown in Figure 1.5 as follows:

```
call coefficientPlot(decomp, , , , "Quartz Spectrum");
```

The first argument specifies the wavelet decomposition and is required. All other arguments are optional and need not be specified. You can use the WAVHELP macro to obtain a description of the arguments of this and other wavelet plot modules. The WAVHELP macro is defined in autocall the WAVINIT macro. For example, invoking the WAVHELP macro as follows writes the calling information shown in Figure 1.4 to the SAS log.

```
%wavhelp(coefficientPlot);
```

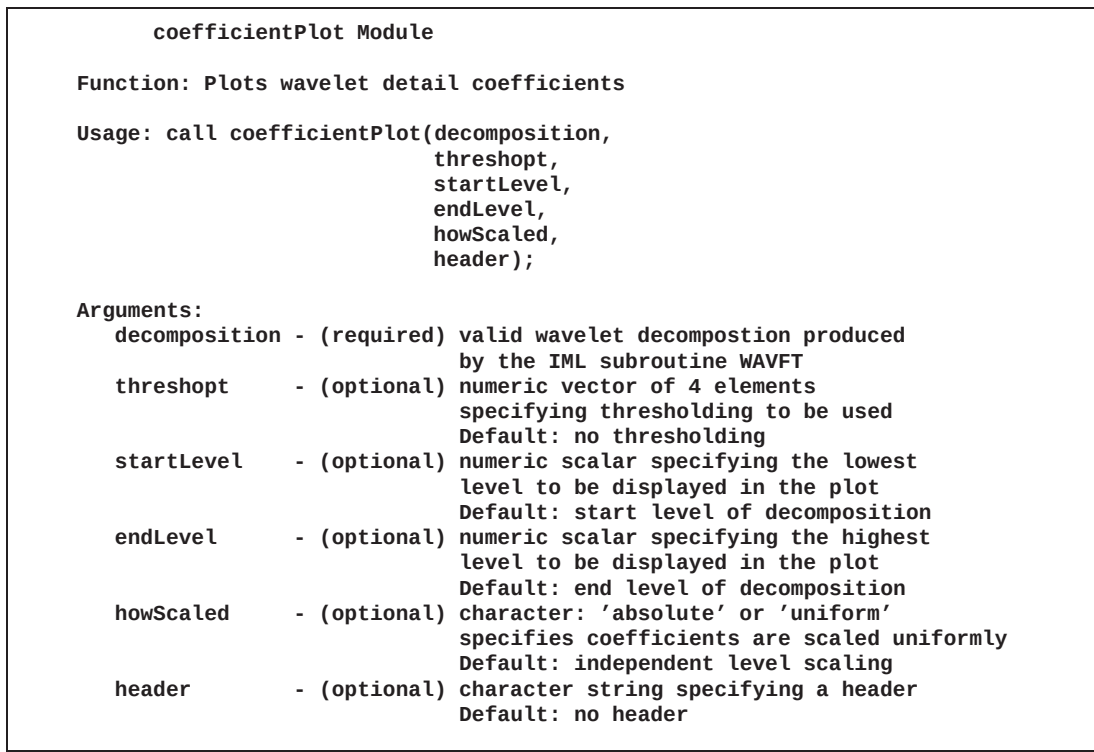


**Figure 1.4.** Log Output Produced by %wavhelp(coefficientPlot) Call



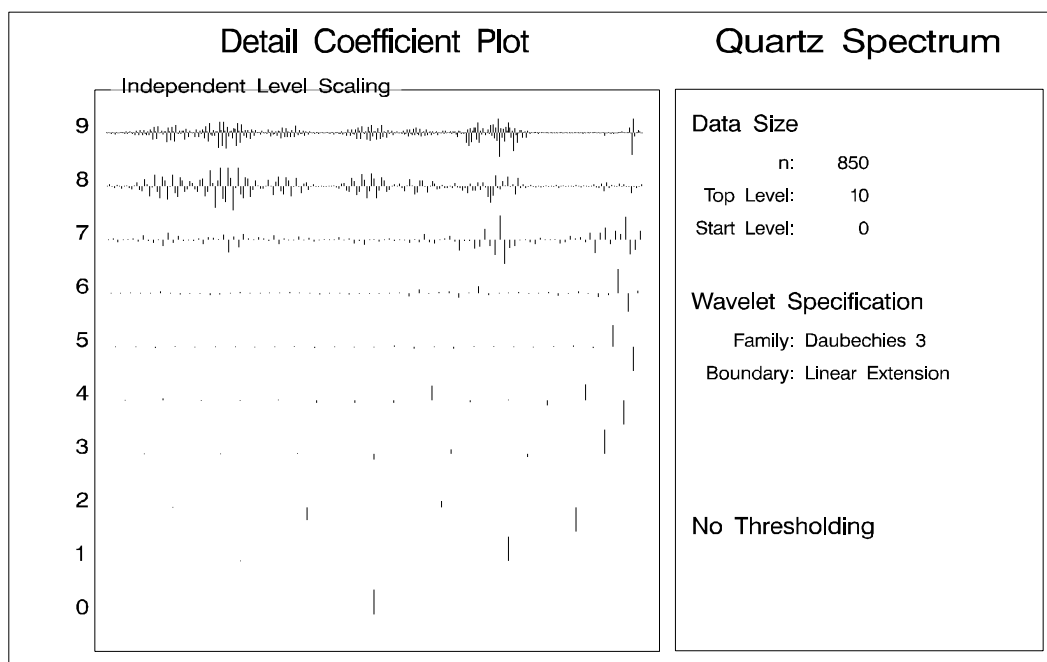


**Figure 1.5.** Detail Coefficients Scaled by Level

In this plot the detail coefficients at each level are scaled independently. The oscillations present in the absorbance data are captured in the detail coefficients at levels 7, 8, and 9. The following statement produces a coefficient plot of just these higher level detail coefficients and shows them scaled uniformly.

```
call coefficientPlot(decomp, ,7, ,
                    'uniform',"Quartz Spectrum");
```

The plot is shown in Figure 1.6.

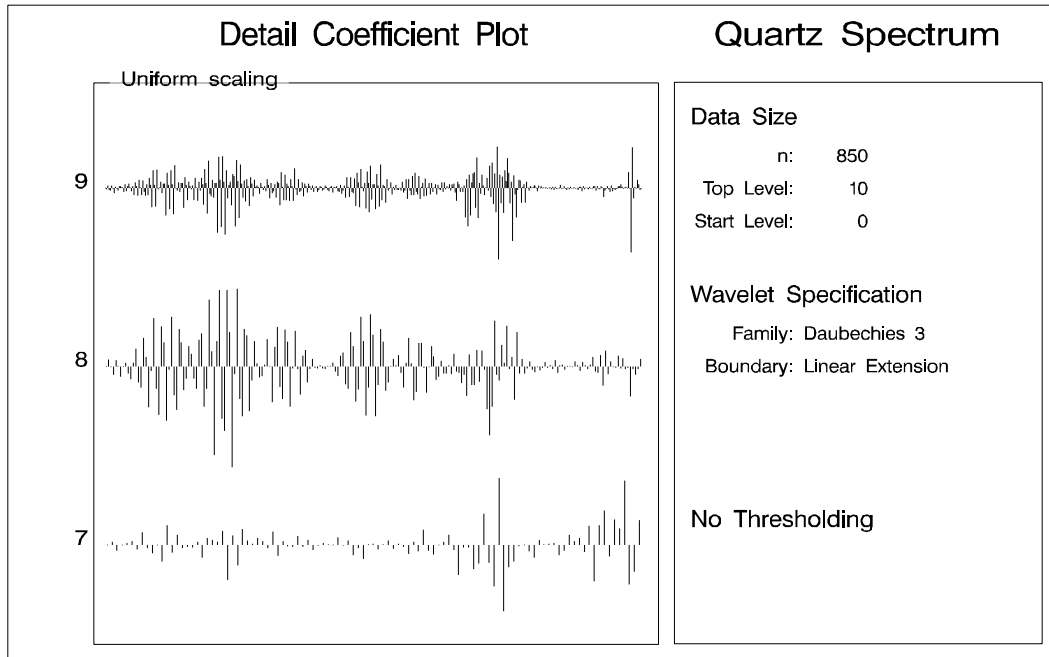


**Figure 1.6.** Uniformly Scaled Detail Coefficients

As noted earlier, noise in the data is captured in the detail coefficients, particularly in the small coefficients at higher levels in the decomposition. By zeroing or shrinking these coefficients, you can get smoother reconstructions of the input data. This is done by specifying a threshold value for each level of detail coefficients and then zeroing or shrinking all the detail coefficients below this threshold value. The IML wavelet functions and modules support several policies for how this thresholding is performed as well as for selecting the thresholding value at each level. See the “WAVIFT Call” on page 26 for details.

An options vector is used to specify the desired thresholding; several standard choices are predefined as macro variables in the WAVINIT module. The following statements produce the detail coefficient plot with the “SureShrink” thresholding algorithm of Donoho and Johnstone (1995).

```
call coefficientPlot(decomp,&SureShrink,6,, ,
                    "Quartz Spectrum");
```

The plot is shown in Figure 1.7.

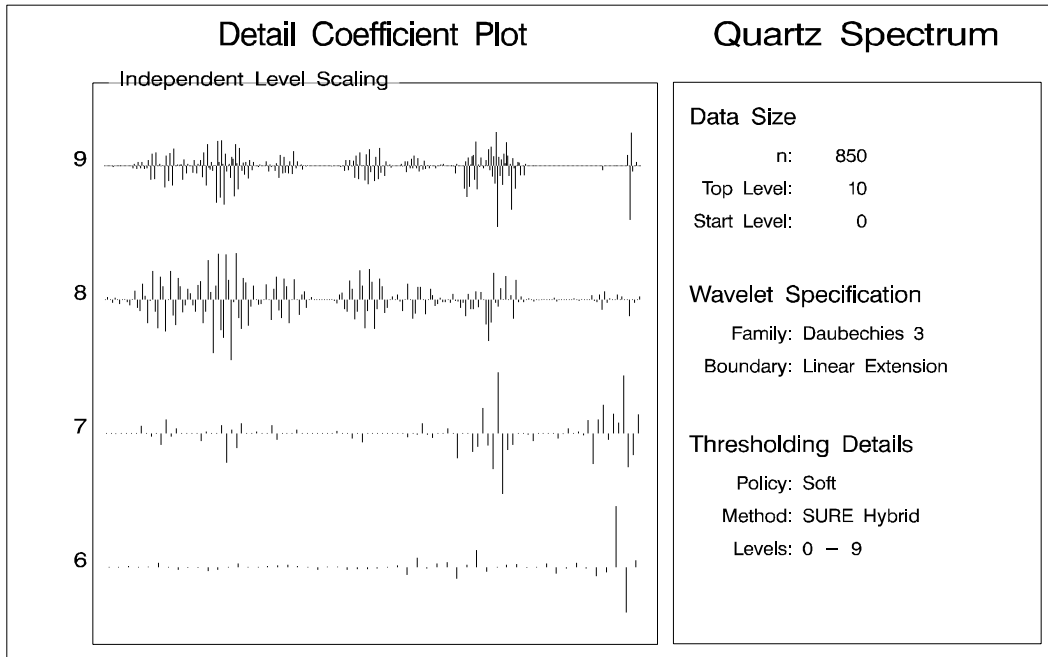


**Figure 1.7.** Thresholded Detail Coefficients

You can see that “SureShrink” thresholding has zeroed some of the detail coefficients at the higher levels but the larger coefficients that capture the oscillation in the data are still present. Consequently, reconstructions of the the input signal using the thresholded detail coefficients will still capture the essential features of the data, but will be smoother as much of the very fine scale detail has been eliminated.

---

## Multiresolution Approximation Plots

One way of presenting reconstructions is in a multiresolution approximation plot. In this plot reconstructions of the input data are shown by level. At any level the reconstruction at that level uses only the detail and scaling coefficients defined below that level.

The following statement produces such a plot, starting at level 3:

```
call mraApprox(decomp, ,3, ,"Quartz Spectrum");
```

The results are shown in Figure 1.8.

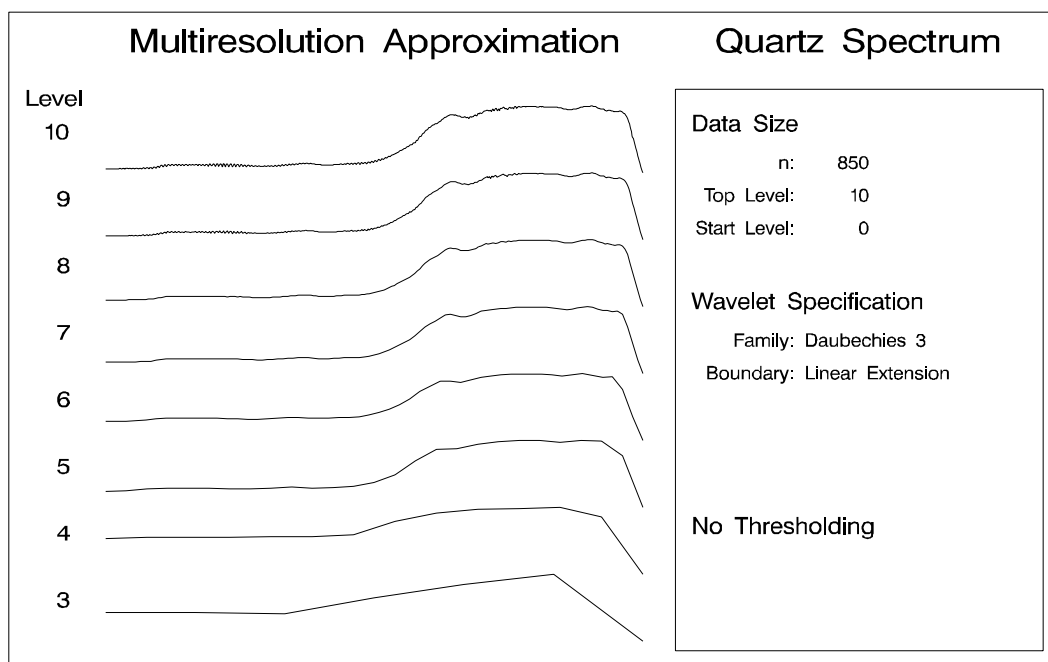


**Figure 1.8.** Multiresolution Approximation

You can see that even at level 3, the basic form of the input signal has been captured. As noted earlier, the oscillation present in the absorbance data is captured in the detail coefficients above level 7. Thus, the reconstructions at level 7 and below are largely free of these oscillation since they do not use any of the higher detail coefficients. You can confirm this observation by plotting just this level in the multiresolution analysis as follows:

```
call mraApprox(decomp, ,7,7,"Quartz Spectrum");
```

The results are shown in Figure 1.9.

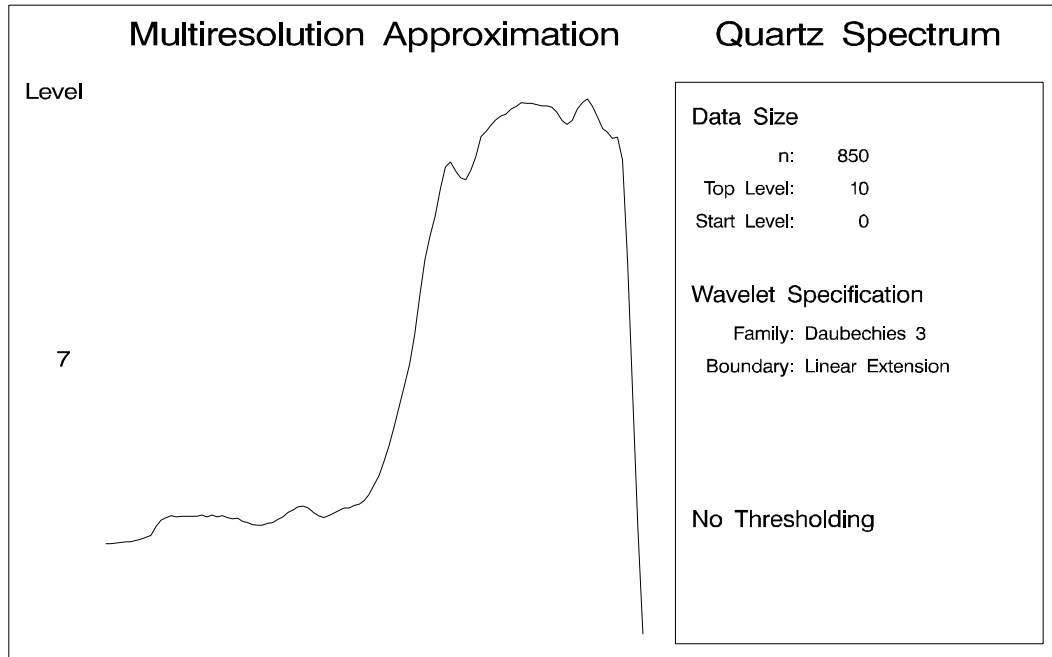


**Figure 1.9.** Level 7 of the Multiresolution Approximation

You can also plot the multiresolution approximations obtained with thresholded detail coefficients. For example, the following statement plots the top level reconstruction obtained using the “SureShrink” threshold:

```
call mraApprox(decomp,&SureShrink,10,10,
               "Quartz Spectrum");
```

The results are shown in Figure 1.10.

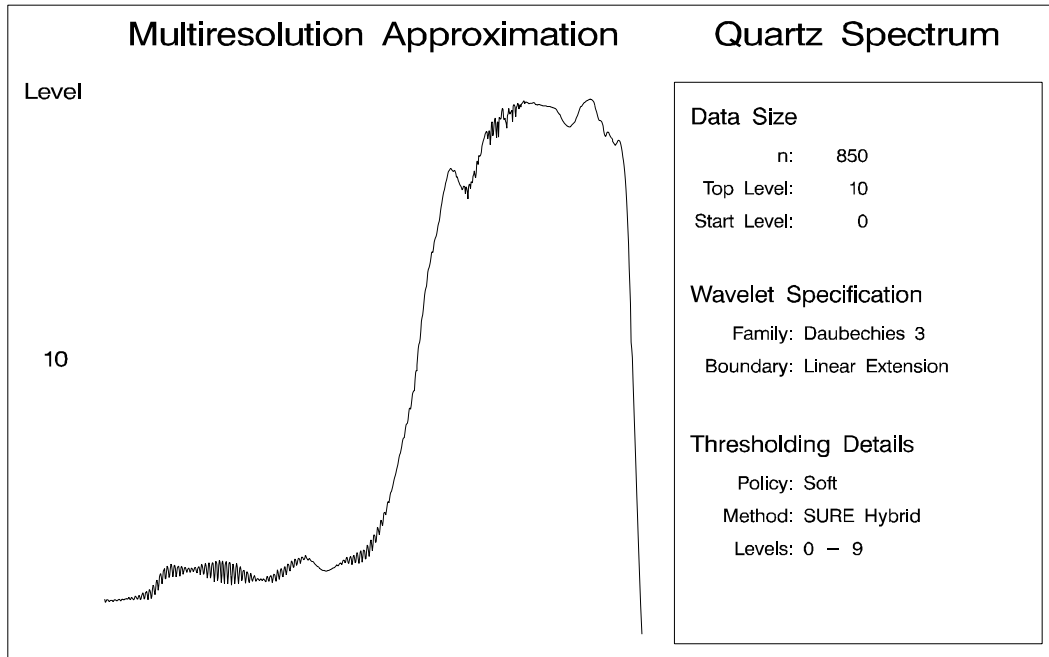


**Figure 1.10.** Top Level of Multiresolution Approximation with SureShrink Thresholding Applied

Note that the high frequency oscillation is still present in the reconstruction even with “SureShrink” thresholding applied.

## Multiresolution Decomposition Plots

A related plot is the multiresolution decomposition plot, which shows the detail coefficients at each level. For convenience, the starting level reconstruction at the lowest level of the plot and the reconstruction at the highest level the plot are also included. Adding suitably scaled versions of all the detail levels to the starting level reconstruction recovers the final reconstruction. The following statement produces such a plot, yielding the results shown in Figure 1.11.

```
call mraDecomp(decomp, ,5, , , "Quartz Spectrum");
```

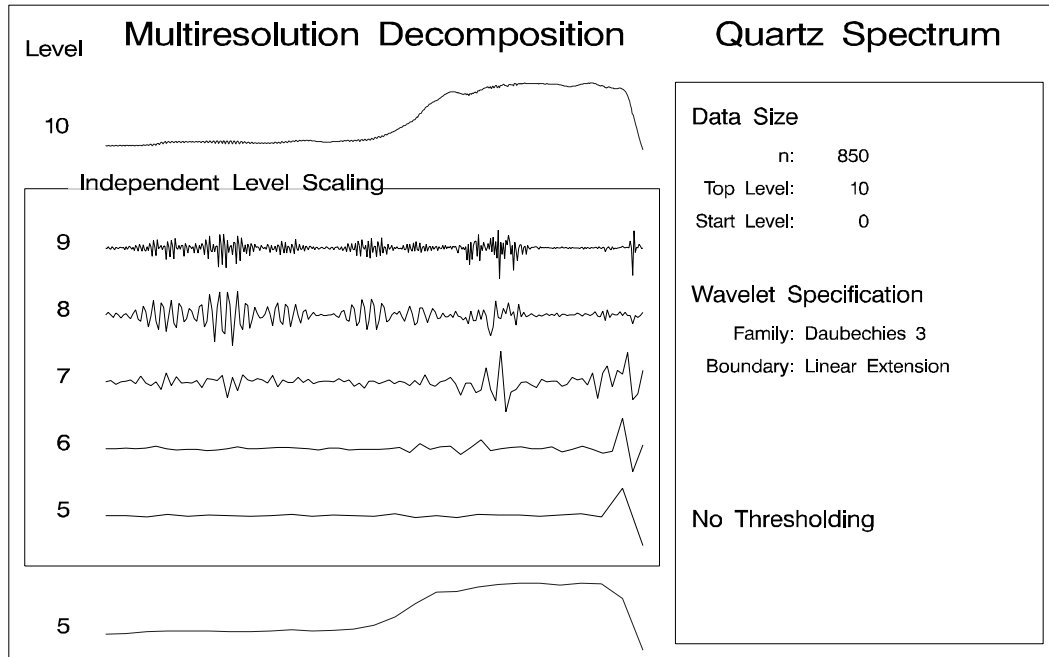


Figure 1.11. Multiresolution Decomposition

## Wavelet Scalograms

Wavelet scalograms communicate the time frequency localization property of the discrete wavelet transform. In this plot each detail coefficient is plotted as a filled rectangle whose color corresponds to the magnitude of the coefficient. The location and size of the rectangle are related to the time interval and the frequency range for this coefficient. Coefficients at low levels are plotted as wide and short rectangles to indicate that they localize a wide time interval but a narrow range of frequencies in the data. In contrast, rectangles for coefficients at high levels are plotted thin and tall to indicate that they localize small time ranges but large frequency ranges in the data. The heights of the rectangles grow as a power of 2 as the level increases. If you include all levels of coefficients in such a plot, the heights of the rectangles at the lowest levels are so small that they will not be visible. You can use an option to plot the heights of the rectangles on a logarithmic scale. This results in rectangles of uniform height but requires that you interpret the frequency localization of the coefficients with care.

The following statement produces a scalogram plot of all levels with “SureShrink” thresholding applied:

```
call scalogram(decomp,&SureShrink, , ,0.25,
               'log','Quartz Spectrum');
```

The sixth argument specifies that the rectangle heights are to be plotted on a logarithmic scale. The role of the fifth argument (0.25) is to amplify the magnitude of the small detail coefficients. This is necessary since the detail coefficients at the lower

levels are orders of magnitude larger than those at the higher levels. The amplification is done by first scaling the magnitudes of all detail coefficients to lie in the interval  $[0, 1]$  and then raising these scaled magnitudes to the power 0.25. Note that smaller powers yield larger amplification of the small detail coefficient magnitudes. The default amplification is  $1/3$ .

The results are shown in Figure 1.12.

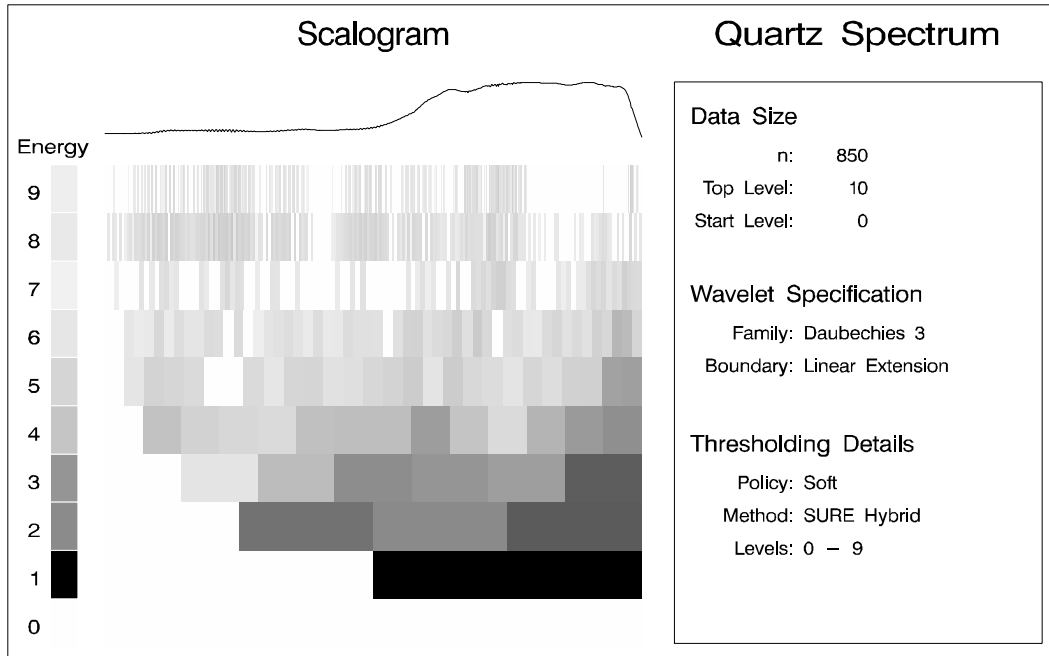


**Figure 1.12.** Scalogram Showing All Levels

The bar on the left-hand side of the scalogram plot indicates the overall energy of each level. This energy is defined as the sum of the squares of the detail coefficients for each level. These energies are amplified using the same algorithm for amplifying the detail coefficient magnitudes. The energy bar in Figure 1.12 shows that higher energies occur at the lower levels whose coefficients capture the gross features of the data. In order to interpret the finer-scale details of the data it is helpful to focus on just the higher levels. The following statement produces a scalogram for levels 6 and above without using a logarithmic scale for the rectangle heights, and using the default coefficient amplification.

```
call scalogram(decomp,&SureShrink,6, , , ,
               "Quartz Spectrum");
```

The result is shown in Figure 1.13.



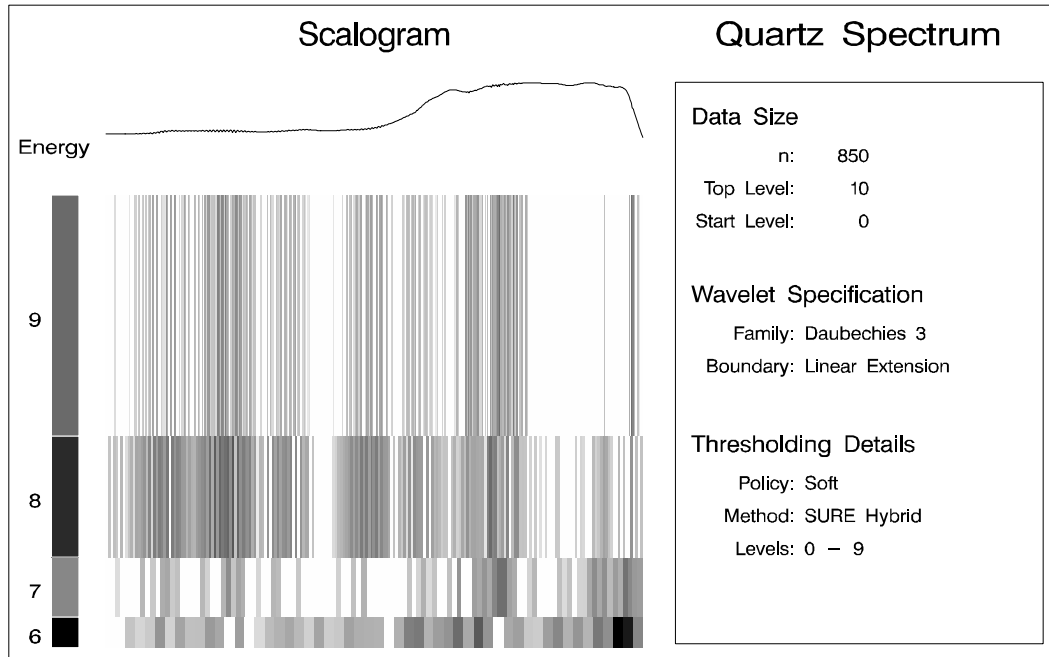


**Figure 1.13.** Scalogram of Levels 6 and Above Using SureShrink Thresholding

The scalogram in Figure 1.13 reveals that most of the energy of the oscillation in the data is captured in the detail coefficients at level 8. Also note that many of the coefficients at the higher levels are set to zero by “SureShrink” thresholding. You can verify this by comparing Figure 1.13 with Figure 1.14, which shows the corresponding scalogram except that no thresholding is done. The following statement produces Figure 1.14:

```
call scalogram(decomp, ,6, , , , "Quartz Spectrum");
```

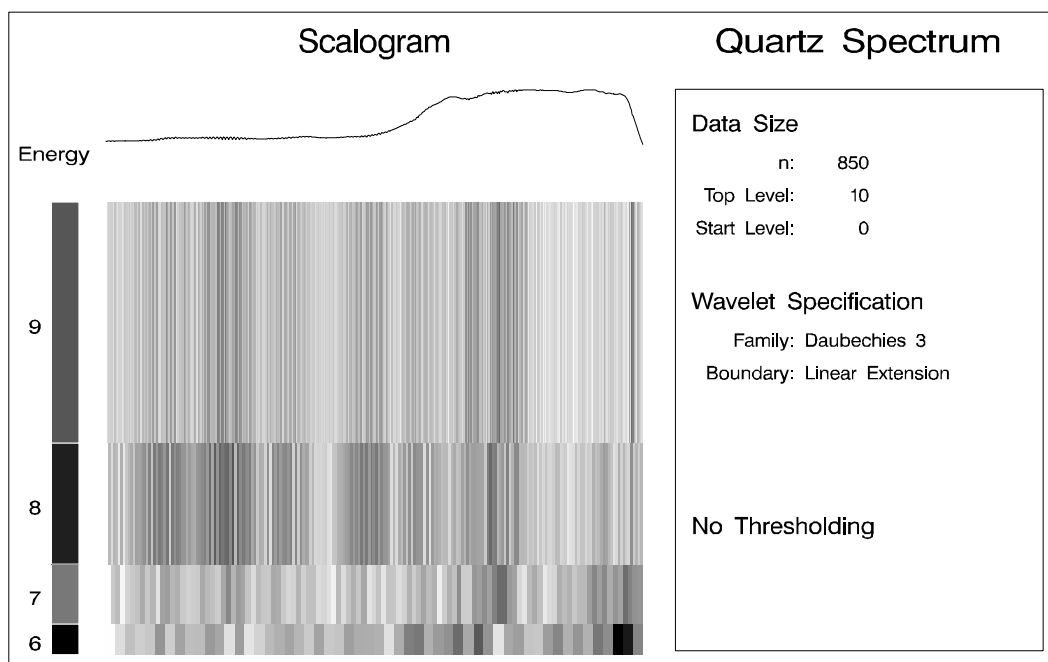


**Figure 1.14.** Scalogram of Levels 6 and Above Using No Thresholding

## Reconstructing the Signal from the Wavelet Decomposition

You can use the WAVIFT subroutine to invert a wavelet transformation computed using the WAVFT subroutine. If no thresholding is specified, then up to numerical rounding error this inversion is exact. The following statements provide an illustration of this:

```
call wavift(reconstructedAbsorbance,decomp);
errorSS=ssq(absorbance-reconstructedAbsorbance);
print "The reconstruction error sum of squares = " errorSS;
```

The output is shown in Figure 1.15.

```
ERRORSS
The reconstruction error sum of squares = 1.288E-16
```

**Figure 1.15.** Exact Reconstruction Property of WAVIFT

Usually you use the WAVIFT subroutine with thresholding specified. This yields a smoothed reconstruction of the input data. You can use the following statements to create a smoothed reconstruction of absorbance and add this variable to the Quartz-InfraredSpectrum data set.

```
call wavift(smoothedAbsorbance,decomp,&SureShrink);
create temp from smoothedAbsorbance[colname='smoothedAbsorbance'];
append from smoothedAbsorbance;
```

```

    close temp;
quit;

data quartzInfraredSpectrum;
    set quartzInfraredSpectrum;
    set temp;
run;

```

The following statements produce the line plot of the smoothed absorbance data shown in Figure 1.16:

```

symbol1 c=black i=join v=None;
proc gplot data=quartzInfraredSpectrum;
    plot smoothedAbsorbance*WaveNumber/
        hminor = 0    vminor = 0
        vaxis = axis1
        hreverse frame;
        axis1 label = ( r=0 a=90 );
run;

```

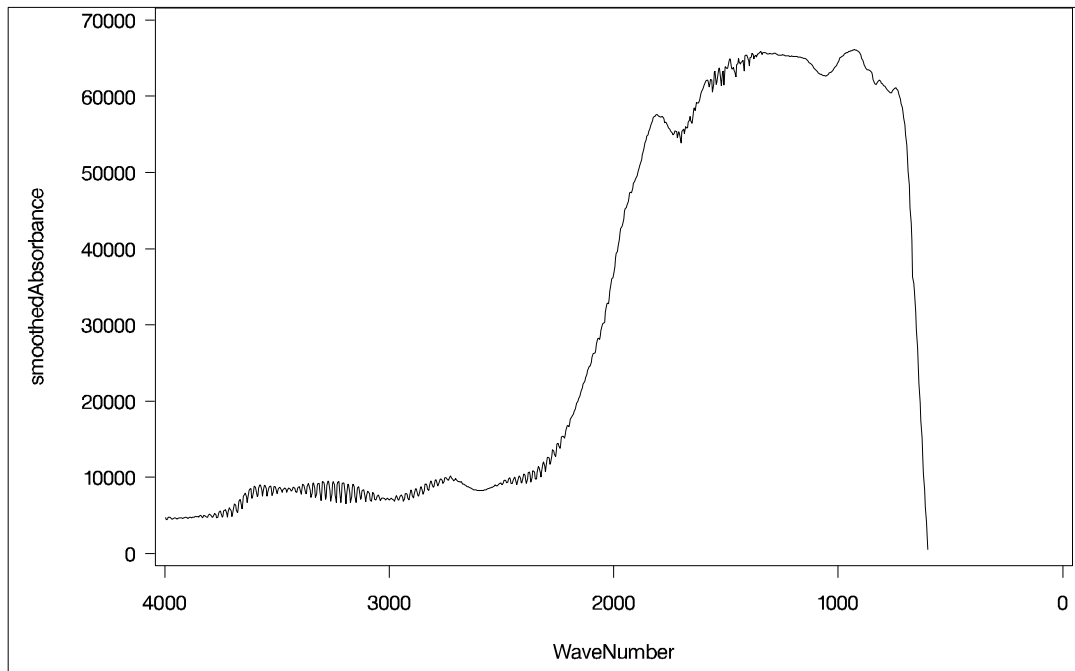


**Figure 1.16.** Smoothed FT-IR Spectrum of Quartz

You can see by comparing Figure 1.1 with Figure 1.16 that the wavelet smooth of the absorbance data has preserved all the essential features of this data.

---

## Syntax

---

### Wavelet Analysis Calls

---

|               |                                                                                              |
|---------------|----------------------------------------------------------------------------------------------|
| WAVFT Call    | computes a specified wavelet transform of one-dimensional data                               |
| WAVGET Call   | returns requested information encapsulated in a wavelet transform                            |
| WAVIFT Call   | inverts a wavelet transform after applying specified thresholding to the detail coefficients |
| WAVPRINT Call | displays requested information encapsulated in a wavelet transform                           |
| WAVTHRSH Call | applies specified thresholding to the detail coefficients of a wavelet transform             |

---

### WAVFT Call

**computes fast wavelet transform**

**CALL WAVFT(*decomp*, *data*, *opt* <, *levels*> );**

The Fast Wavelet Transform (WAVFT) subroutine computes a specified discrete wavelet transform of the input data, using the algorithm of Mallat (1989). This transform decomposes the input data into sets of detail and scaling coefficients defined at a number of scales or “levels.”

The input data are used as scaling coefficients at the top level in the decomposition. The fast wavelet transform then recursively computes a set of detail and a set of scaling coefficients at the next lower level by respectively applying “low pass” and “high pass” conjugate mirror filters to the scaling coefficients at the current level. The number of coefficients in each of these new sets is approximately half the number of scaling coefficients at the level above them. Depending on the filters being used, a number of additional scaling coefficients, known as *boundary coefficients*, may be involved. These boundary coefficients are obtained by extending the sequence of interior scaling coefficients using a specified method.

Details of the discrete wavelet transform and the fast wavelet transformation algorithm are available in many references, including Mallat (1989), Daubechies (1992), and Ogden (1997).

The inputs to the WAVFT subroutine are as follows:

|                |                                                                                                                             |
|----------------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>data</i>    | specifies the data to transform. This data must be either a row or column vector.                                           |
| <i>opt</i>     | refers to an options vector with the following components:                                                                  |
| <i>opt</i> [1] | specifies the boundary handling used in computing the wavelet transform. At each level of the wavelet decomposition, neces- |

sary boundary scaling coefficients are obtained by extending the interior scaling coefficients at that level as follows:

- opt*[1]=0 specifies extension by zero.
- opt*[1]=1 specifies periodic extension.
- opt*[1]=2 specifies polynomial extension.
- opt*[1]=3 specifies extension by reflection.
- opt*[1]=4 specifies extension by anti-symmetric reflection.
- opt*[2] specifies the polynomial degree that is used for polynomial extension. The value of *opt*[2] is ignored if *opt*[1] ≠ 2.
  - opt*[2]=0 specifies constant extension.
  - opt*[2]=1 specifies linear extension.
  - opt*[2]=2 specifies quadratic extension.
- opt*[3] specifies the wavelet family.
  - opt*[3]=1 specifies the Daubechies Extremal phase family (Daubechies 1992).
  - opt*[3]=2 specifies the Daubechies Least Asymmetric family (also known as the Symmlet family) (Daubechies 1992).
- opt*[4] specifies the wavelet family member. Valid values are
  - opt*[4]=1 through 10, if *opt*[3]=1
  - opt*[4]=4 through 10, if *opt*[3]=2

Some examples of wavelet specifications are

- opt*={1 1 1}; specifies the first member (more commonly known as the Haar system) of the Daubechies extremal phase family with periodic boundary handling.
- opt*={2 1 2 5}; specifies the fifth member of the Symmlet family with linear extension boundary handling.

*levels* is an optional scalar argument that specifies the number of levels from the top level to be computed in the decomposition. If you do not specify this argument, then the decomposition terminates at level 0. Usually, you will not need to specify this optional argument. You use this option to avoid unneeded computations in situations where you are interested in the detail and scaling coefficients at only higher levels.

The WAVFT subroutine returns

*decomp* a row vector that encapsulates the specified wavelet transform. The information that is encoded in this vector includes:

- the options specified for computing the transform
- the number of detail coefficients at each level of the decomposition
- all detail coefficients
- the scaling coefficients at the bottom level of the decomposition

- boundary scaling coefficients at all levels of the decomposition

**Note:** *decomp* is a private representation of the specified wavelet transform and is not intended to be interpreted in its raw form. Rather, you should use this vector as an input argument to the WAVIFT, WAVPRINT, WAVGET, and WAVTHRSH subroutines.

---

## WAVGET Call

**extracts wavelet information**

**CALL WAVGET(*result*, *decomp*, *request* <, *options*> );**

The WAVGET subroutine is used to return information that is encoded in a wavelet decomposition.

The required inputs are

*decomp* specifies a wavelet decomposition that has been computed using a call to the WAVFT subroutine.

*request* specifies a scalar indicating what information is to be returned.

You can specify different optional arguments depending on the value of *request*:

*request*=1 requests the number of points in the input data vector.

*result* returns as a scalar containing this number.

*request*=2 requests the detail coefficients at a specified level. Valid syntax is

**CALL WAVGET(*result*, *decomp*, 2, *level* <, *opt*> );**

where the argument

*level* is the level at which the detail coefficients are requested.

*opt* is an optional vector that specifies the thresholding to be applied to the returned detail coefficients. See the WAVIFT subroutine call for details. If you omit this argument, no thresholding is applied.

*result* returns as a column vector containing the specified detail coefficients.

*request*=3 requests the scaling coefficients at a specified level. Valid syntax is

**CALL WAVGET(*result*, *decomp*, 3, *level* <, *opt*> );**

where the argument

- level* is the level at which the scaling coefficients are requested.
- opt* is an optional vector that specifies the thresholding to be applied. See the WAVIFT subroutine call for a description of this vector. The scaling coefficients at the requested level are obtained by using the inverse wavelet transform, after applying the specified thresholding. If you omit this argument, no thresholding is applied.
- result* returns as a column vector containing the specified scaling coefficients.
- request=4* requests the thresholding status of the detail coefficients in *decomp*.
- result* returns as a scalar whose value is
- 0, if the detail coefficients have not been thresholded.
  - 1, otherwise.
- request=5* requests the wavelet options vector that you specified in the WAVFT subroutine call to compute *decomp*.
- result* returns as a column vector with four elements containing the specified options vector. See the WAVFT subroutine call for the interpretation of the vector entries.
- request=6* requests the index of the top level in *decomp*.
- result* returns as a scalar containing this number.
- request=7* requests the index of the lowest level in *decomp*.
- result* returns as a scalar containing this number.
- request=8* requests a vector evaluating the father wavelet used in *decomp*, at an equally spaced grid spanning the support of the father wavelet. The number of points in the grid is specified as a power of 2 times the support width of the father wavelet. For wavelets in the Daubechies extremal phase and least asymmetric families, the support width of the father wavelet is  $2m - 1$ , where  $m$  is the family member. Valid syntax is

**CALL WAVGET(*result*, *decomp*, 8 <, *power*> );**

where the optional argument

- power* is the exponent of 2 determining the number of grid points used. *power* defaults to 8 if you do not specify this argument.
- result* returns as a column vector containing the specified evaluation of the father wavelet.

---

## WAVIFT Call

**computes inverse fast wavelet transform**

**CALL WAVIFT(result, decomp <, opt <, level>> );**

The Inverse Fast Wavelet Transform (WAVIFT) subroutine computes the inverse wavelet transform of a wavelet decomposition computed using the WAVFT subroutine. Details of this algorithm are available in many references, including Mallat (1989), Daubechies (1992), and Ogden (1997).

The inverse transform yields an exact reconstruction of the original input data, provided that no smoothing is specified. Alternatively, a smooth reconstruction of the input data can be obtained by thresholding the detail coefficients in the decomposition prior to applying the inverse transformation. Thresholding, also known as shrinkage, replaces the detail coefficient  $d_j^{(i)}$  at level  $i$  by  $\delta_{T_i}(d_j^{(i)})$ , where the  $\delta_T(x)$  is a shrinkage function and  $T_i$  is the threshold value used at level  $i$ . The SAS/IML wavelet subroutines support hard and soft shrinkage functions (Donoho and Johnstone 1994) and the non-negative garrote shrinkage function (Breiman 1995). These functions are defined as follows:

$$\begin{aligned}\delta_T^{\text{hard}}(x) &= \begin{cases} 0 & |x| \leq T \\ x & |x| > T \end{cases} \\ \delta_T^{\text{soft}}(x) &= \begin{cases} 0 & |x| \leq T \\ x - T & x > T \\ x + T & x < -T \end{cases} \\ \delta_T^{\text{garrote}}(x) &= \begin{cases} 0 & |x| \leq T \\ x - T^2/x & |x| > T \end{cases}\end{aligned}$$

You can specify several methods for choosing the threshold values. Methods in which the threshold  $T_i$  varies with the level  $i$  are called *adaptive*. Methods where the same threshold is used at all levels are called *global*.

The inputs to the WAVIFT subroutine are as follows:

*decomp* specifies a wavelet decomposition that has been computed using a call to the WAVFT subroutine.

*opt* refers to an options vector that specifies the thresholding algorithm. If this optional argument is not specified, then no thresholding is applied.

The options vector has the following components:

*opt*[1] specifies the thresholding policy.

*opt*[1]=0 specifies that no thresholding be done. If *opt*[1]=0 then all other entries in the options vector are ignored.



- $opt[1]=1$  specifies hard thresholding.
- $opt[1]=2$  specifies soft thresholding.
- $opt[1]=3$  specifies garrote thresholding.
- $opt[2]$  specifies the method for selecting the threshold.
  - $opt[2]=0$  specifies a global user-supplied threshold.
  - $opt[2]=1$  specifies a global threshold chosen using the minimax criterion of Donoho and Johnstone (1994).
  - $opt[2]=2$  specifies a global threshold defined using the universal criterion of Donoho and Johnstone (1994).
  - $opt[2]=3$  specifies an adaptive method where the thresholds at each level  $i$  are chosen to minimize an approximation of the  $L^2$  risk in estimating the true data values using the reconstruction with thresholded coefficients (Donoho and Johnstone 1995).
  - $opt[2]=4$  specifies a hybrid method of Donoho and Johnstone (1995). The universal threshold as specified by  $opt[2]=2$  is used at levels where most of the detail coefficients are essentially zero. The risk minimization method as specified by  $opt[2]=4$  is used at all other levels.
- $opt[3]$  specifies the value of the global user-supplied threshold if  $opt[2]=1$ . It is ignored if  $opt[2] \neq 1$ .
- $opt[4]$  specifies the number of levels starting at the highest detail coefficient level at which thresholding is to be applied. If this value is negative or missing, thresholding is applied at all levels in *decomp*.

Some common examples of threshold options specifications are:

- $opt=\{1\ 3\ .\ -1\}$ ; specifies hard thresholding with a minimax threshold applied at all levels in the decomposition. This threshold is named “*RiskShrink*” in Donoho and Johnstone (1994).
- $opt=\{2\ 2\ .\ -1\}$ ; specifies soft thresholding with a universal threshold applied at all levels in the decomposition. This threshold is named “*VisuShrink*” in Donoho and Johnstone (1994).
- $opt=\{2\ 4\ .\ -1\}$ ; specifies soft thresholding with level-dependent thresholds that minimize the Stein Unbiased Estimate of Risk (SURE). This threshold is named “*SureShrink*” in Donoho and Johnstone (1995).

*level* is an optional scalar argument that specifies the level at which the reconstructed data are to be returned. If this argument is not specified then the reconstructed data are returned at the top level defined in *decomp*.

The WAVIFT subroutine returns

*result* a vector obtained by inverting, after thresholding the detail coefficients, the discrete wavelet transform encoded in *decomp*. The row or column orientation of *result* is the same as that of the input data specified in the corresponding WAVFT subroutine call. If you specify the optional *level* argument, *result*

contains the reconstruction at the specified level, otherwise the reconstruction corresponds to the top level in the decomposition.

---

## WAVPRINT Call

**displays wavelet information**

**CALL WAVPRINT(*decomp*, *request* <, *options*> );**

The WAVPRINT subroutine is used to display the information that is encoded in a wavelet decomposition.

The required inputs are

*decomp* specifies a wavelet decomposition that has been computed using a call to the WAVFT subroutine.

*request* specifies a scalar indicating what information is to be displayed.

You can specify different optional arguments depending on the value of *request*:

*request*=1 displays information about the wavelet family used to perform the wavelet transform. No additional arguments need to be specified.

*request*=2 displays the detail coefficients by level. Valid syntax is

**CALL WAVPRINT(*decomp*, 2 <, *lower* <, *upper*>> );**

where the argument

*lower* is optional and specifies the lowest level to be displayed. The default value of *lower* is the lowest level in *decomp*.

*upper* is optional and specifies the upper level to be displayed. The default value of *upper* is the highest detail level in *decomp*.

*request*=3 displays the scaling coefficients by level. Valid syntax is

**CALL WAVPRINT(*decomp*, 3 <, *lower* <, *upper*>> );**

where the argument

*lower* is optional and specifies the lowest level to be displayed. The default value of *lower* is the lowest level in *decomp*.

*upper* is optional and specifies the upper level to be displayed. The default value of *upper* is the top level in *decomp*.

*request=4* displays thresholded detail coefficients by level. Valid syntax is

**CALL WAVPRINT(*decomp*, 4, *opt* <, *lower* <, *upper* > > );**

where the argument

- opt* is a required options vector that specifies the thresholding algorithm used. See the WAVIFT subroutine call for a description of this options vector.
- lower* is optional and specifies the lowest level to be displayed. The default value of *lower* is the lowest level in *decomp*.
- upper* is optional and specifies the upper level to be displayed. The default value of *upper* is the highest detail level in *decomp*.

---

## WAVTHRSH Call

**thresholds wavelet detail coefficients**

**CALL WAVTHRSH(*decomp*, *opt*);**

The Wavelet Threshold (WAVTHRSH) subroutine thresholds the detail coefficients in a wavelet decomposition.

The required inputs are

- decomp* specifies a wavelet decomposition that has been computed using a call to the WAVFT subroutine.
- opt* refers to an options vector that specifies the thresholding algorithm used. See the WAVIFT subroutine call for a description of this options vector.

On return, the detail coefficients encoded in *decomp* are replaced by their thresholded values. Note that this action is not reversible. If you want to retain the original detail coefficients, you should not use the WAVTHRSH subroutine to do thresholding. Rather, you should supply the thresholding argument where appropriate in the WAVIFT, WAVGET, and WAVPRINT subroutine calls.

---

## Details

---

### Using Symbolic Names

---

Several of the wavelet subroutines take arguments that are options vectors that specify user input. For example, the third argument in a WAVFT subroutine call is an options vector that specifies which wavelet and which boundary treatment are used in computing the wavelet transform. Typical code that defines this options vector is

```
optn      = j(1, 4, .);
optn[1] = 0;
optn[3] = 1;
optn[4] = 3;
```

A problem with such code is that it is not easily readable. By using symbolic names readability is greatly enhanced. SAS macro variables provide a convenient mechanism for creating such symbolic names. For example, the previous code could be replaced by

```
optn      = &waveSpec;
optn[&family] = &daubechies;
optn[&member] = 3;
optn[&boundary] = &zeroExtension;
```

where the symbolic macro variables (names with a preceding ampersand) resolve to the relevant quantities. Another example where symbolic names improve code readability is to use symbolic names for an integer argument that controls what action a multipurpose subroutine performs. An illustration is replacing code such as

```
call wavget(n,decomposition,1);
call wavget(fWavelet,decompostion,8);
```

by

```
call wavget(n,decomposition,&numPoints);
call wavget(fWavelet,decompostion,&fatherWavelet);
```

A set of symbolic names is defined in the autocall WAVINIT macro. The following tables list the symbolic names that are defined in this macro:

**Table 1.1.** Macro Variables for Wavelet Specification

| Position  |       | Admissible Values        |       |
|-----------|-------|--------------------------|-------|
| Name      | Value | Name                     | Value |
| &boundary | 1     | &zeroExtension           | 0     |
|           |       | &periodic                | 1     |
|           |       | &polynomial              | 2     |
|           |       | &reflection              | 3     |
|           |       | &antisymmetricReflection | 4     |
| &degree   | 2     | &constant                | 0     |
|           |       | &linear                  | 1     |
|           |       | &quadratic               | 2     |
| &family   | 3     | &daubechies              | 1     |
|           |       | &symmlet                 | 2     |
| &member   | 4     | 1 - 10                   |       |

**Table 1.2.** Macro Variables for Threshold Specification

| Position |       | Admissible Values       |       |
|----------|-------|-------------------------|-------|
| Name     | Value | Name                    | Value |
| &policy  | 1     | &none                   | 0     |
|          |       | &hard                   | 1     |
|          |       | &soft                   | 2     |
|          |       | &garrote                | 3     |
| &method  | 2     | &absolute               | 0     |
|          |       | &minimax                | 1     |
|          |       | &universal              | 2     |
|          |       | &sure                   | 3     |
|          |       | &sureHybrid             | 4     |
|          |       | &nhoodCoeffs            | 5     |
| &value   | 3     | <i>positive real</i>    |       |
| &levels  | 4     | &all                    | -1    |
|          |       | <i>positive integer</i> |       |

**Table 1.3.** Symbolic Names for the Third Argument of WAVGET

| Name                | Value |
|---------------------|-------|
| &numPoints          | 1     |
| &detailCoeffs       | 2     |
| &scalingCoeffs      | 3     |
| &thresholdingStatus | 4     |
| &specification      | 5     |
| &topLevel           | 6     |
| &startLevel         | 7     |
| &fatherWavelet      | 8     |

**Table 1.4.** Macro Variables for the Second Argument of WAVPRINT

| Name                     | Value |
|--------------------------|-------|
| &summary                 | 1     |
| &detailCoeffs            | 2     |
| &scalingCoeffs           | 3     |
| &thresholdedDetailCoeffs | 4     |

**Table 1.5.** Macro Variables for Predefined Wavelet Specifications

| Name         | &boundary   | &degree | &family     | &member |
|--------------|-------------|---------|-------------|---------|
| &waveSpec    | { .         | .       | .           | }       |
| &haar        | { &periodic | .       | &daubechies | 1 }     |
| &daubechies3 | { &periodic | .       | &daubechies | 3 }     |
| &daubechies5 | { &periodic | .       | &daubechies | 5 }     |
| &symmlet5    | { &periodic | .       | &symmlet    | 5 }     |
| &symmlet8    | { &periodic | .       | &symmlet    | 8 }     |

**Table 1.6.** Macro Variables for Predefined Threshold Specifications

| Name        | &policy | &method     | &value | &levels |
|-------------|---------|-------------|--------|---------|
| &threshSpec | { .     | .           | .      | }       |
| &RiskShrink | { &hard | &minimax    | .      | &all }  |
| &VisuShrink | { &soft | &universal  | .      | &all }  |
| &SureShrink | { &soft | &sureHybrid | .      | &all }  |

---

## Obtaining Help for the Wavelet Macros and Modules

The WAVINIT macro that you call to define symbolic macro variables and wavelet plot modules also defines a macro WAVHELP that you can call to obtain help for the wavelet macros and plot modules. The syntax for calling the WAVHELP macro is

```
%WAVHELP < ( name )>;
```

where *name* is one of wavginit, wavinit, coefficientPlot, mraApprox, mraDecomp, or scalogram. This macro displays usage and argument information for the specified macro or module. If you call the WAVHELP macro with no arguments, it lists the names of the macros and modules for which help is available. Note that you can obtain help for the built-in IML wavelet subroutines using the SAS Online Help.

---

## References

- Daubechies, I. (1992), *Ten Lectures on Wavelets*, Volume 61, CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, PA: Society for Industrial and Applied Mathematics.
- Donoho, D.L. and Johnstone, I.M. (1994), "Ideal Spatial Adaptation via Wavelet

- Shrinkage,” *Biometrika*, 81, 425–455.
- Donoho, D.L. and Johnstone, I.M. (1995), “Adapting to Unknown Smoothness via Wavelet Shrinkage,” *Journal of the American Statistical Association*, 90, 1200–1224.
- Mallat, S. (1989), “Multiresolution Approximation and Wavelets,” *Transactions of the American Mathematical Society*, 315, 69–88.
- Ogden, R.T. (1997), *Essential Wavelets for Statistical Applications and Data Analysis*, Boston: Birkhäuser.
- Sullivan, D. (2000), “FT-IR Library,” [[http://www.che.utexas.edu/~dls/ir/ir\\_dir.html](http://www.che.utexas.edu/~dls/ir/ir_dir.html)], accessed 16 October 2000.





# Chapter 2

## Fractionally Integrated Time Series Analysis

### Chapter Table of Contents

---

|                                               |    |
|-----------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                     | 37 |
| <b>GETTING STARTED</b> . . . . .              | 37 |
| Fractionally Integrated Time Series . . . . . | 37 |
| <b>REFERENCES</b> . . . . .                   | 45 |



## Chapter 2

# Fractionally Integrated Time Series Analysis

---

## Overview

This chapter describes SAS/IML subroutines related to fractionally integrated time series analysis.

The following subroutines are supported:

|                 |                                                                                |
|-----------------|--------------------------------------------------------------------------------|
| <b>FARMACOV</b> | computes the auto-covariance function for a fractionally integrated ARMA model |
| <b>FARMAFIT</b> | estimates the parameters for a fractionally integrated ARMA model              |
| <b>FARMALIK</b> | computes the log-likelihood function for a fractionally integrated ARMA model  |
| <b>FARMASIM</b> | generates a fractionally integrated ARMA process                               |
| <b>FDIF</b>     | computes a fractionally differenced process                                    |

---

## Getting Started

---

### Fractionally Integrated Time Series

The fractional differencing enables the degree of differencing  $d$  to take any real value rather than being restricted to integer values. The fractionally differenced processes are capable of modeling long-term persistence. The process

$$(1 - B)^d y_t = \epsilon_t$$

is known as a fractional Gaussian noise process or an ARFIMA(0,  $d$ , 0) process, where  $d \in (-1, 1) \setminus \{0\}$ ,  $\epsilon_t$  is a white noise process with mean zero and variance  $\sigma_\epsilon^2$ , and  $B$  is the backshift operator such that  $B^j \mathbf{y}_t = \mathbf{y}_{t-j}$ . The extension of an ARFIMA(0,  $d$ , 0) model combines fractional differencing with an ARMA( $p$ ,  $q$ ) model, known as an ARFIMA( $p$ ,  $d$ ,  $q$ ) model.

Consider an ARFIMA(0, 0.2, 0) represented as  $(1 - B)^{0.2} y_t = \epsilon_t$  where  $\epsilon_t \sim NID(0, 1)$ . With the following statements you can

- compute the auto-covariance function
- generate the simulated data

- compute the log-likelihood function
- fit a fractionally integrated time series model to the data
- obtain the fractionally differenced data

```
d = 0.2;
call farmacov(cov, d); print cov;
call farmasim(yt, d); print yt;
call farmalik(lnl, yt, d); print lnl;
call farmafit(d, ar, ma, sigma, yt); print d sigma;
call fdif(zt, yt, d); print zt;
```

---

## FARMACOV Call

computes the auto-covariance function for an ARFIMA( $p, d, q$ ) process

**CALL FARMACOV(cov, d <, phi, theta, sigma, p, q, lag>);**

The inputs to the FARMACOV subroutine are as follows:

- |              |                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>d</i>     | specifies a fractional differencing order. The value of $d$ must be in the open interval $(-0.5, 0.5)$ excluding zero. This input is required.                                                                                                                                                                                                                                      |
| <i>phi</i>   | specifies an $m_p$ -dimensional vector containing the autoregressive coefficients, where $m_p$ is the number of the elements in the subset of the AR order. The default is zero. All the roots of $\phi(B) = 0$ should be greater than one in absolute value, where $\phi(B)$ is the finite order matrix polynomial in the backshift operator $B$ , such that $B^j y_t = y_{t-j}$ . |
| <i>theta</i> | specifies an $m_q$ -dimensional vector containing the moving-average coefficients, where $m_q$ is the number of the elements in the subset of the MA order. The default is zero.                                                                                                                                                                                                    |
| <i>p</i>     | specifies the subset of the AR order. The quantity $m_p$ is defined as the number of elements of <i>phi</i> .                                                                                                                                                                                                                                                                       |

If you do not specify  $p$ , the default subset is  $p = \{1, 2, \dots, m_p\}$ .

For example, consider  $phi=0.5$ .

If you specify  $p=1$  (the default), the FARMACOV subroutine computes the theoretical auto-covariance function of an ARFIMA(1,  $d$ , 0) process as  $y_t = 0.5 y_{t-1} + \epsilon_t$ .

If you specify  $p=2$ , the FARMACOV subroutine computes the auto-covariance function of an ARFIMA(2,  $d$ , 0) process as  $y_t = 0.5 y_{t-2} + \epsilon_t$ .

- |          |                                                                                                                 |
|----------|-----------------------------------------------------------------------------------------------------------------|
| <i>q</i> | specifies the subset of the MA order. The quantity $m_q$ is defined as the number of elements of <i>theta</i> . |
|----------|-----------------------------------------------------------------------------------------------------------------|

If you do not specify  $q$ , the default subset is  $q = \{1, 2, \dots, m_q\}$ .

The usage of  $q$  is the same as that of  $p$ .

*lag* specifies the length of lags, which must be a positive number. The default is  $lag = 12$ .

The FARMACOV subroutine returns the following value:

*cov* is a  $lag + 1$  vector containing the auto-covariance function of an ARFIMA( $p, d, q$ ) process.

To compute the auto-covariance of an ARFIMA(1, 0.3, 1) process

$$(1 - 0.5B)(1 - B)^{0.3}y_t = (1 + 0.1B)\epsilon_t$$

where  $\epsilon_t \sim NID(0, 1.2)$ , you can specify

```
d      = 0.3;
phi    = 0.5;
theta= -0.1;
sigma= 1.2;
call farmacov(cov, d, phi, theta, sigma) lag=5;
print cov;
```

For  $d \in (0.5, 0.5) \setminus \{0\}$ , the series  $y_t$  represented as  $(1 - B)^d y_t = \epsilon_t$  is a stationary and invertible ARFIMA(0,  $d$ , 0) process with the auto-covariance function

$$\gamma_k = E(y_t y_{t-k}) = \frac{(-1)^k \Gamma(-2d + 1)}{\Gamma(k - d + 1) \Gamma(-k - d + 1)}$$

and the auto-correlation function

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{\Gamma(-d + 1) \Gamma(k + d)}{\Gamma(d) \Gamma(k - d + 1)} \sim \frac{\Gamma(-d + 1)}{\Gamma(d)} k^{2d-1}, \quad k \rightarrow \infty$$

Notice that  $\rho_k$  decays hyperbolically as the lag increases, rather than showing the exponential decay of the auto-correlation function of a stationary ARMA( $p, q$ ) process.

The FARMACOV subroutine computes the auto-covariance function of an ARFIMA( $p, d, q$ ) process.

For  $d \in (0.5, 0.5) \setminus \{0\}$ , the series  $y_t$  is a stationary and invertible ARFIMA( $p, d, q$ ) process represented as

$$\phi(B)(1 - B)^d y_t = \theta(B)\epsilon_t$$

where  $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \dots - \phi_p B^p$  and  $\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \dots - \theta_q B^q$  and  $\epsilon_t$  is a white noise process; all the roots of the characteristic AR and MA polynomial lie outside the unit circle.

Let  $x_t = \theta(B)^{-1} \phi(B) y_t$ , so that  $x_t$  follows an ARFIMA(0,  $d$ , 0) process; let  $z_t = (1 - B)^d y_t$ , so that  $z_t$  follows an ARMA( $p, q$ ) process; let  $\gamma_k^x$  be the auto-covariance function of  $\{x_t\}$  and  $\gamma_k^z$  be the auto-covariance function of  $\{z_t\}$ .

Then the auto-covariance function of  $\{y_t\}$  is as follows:

$$\gamma_k = \sum_{j=-\infty}^{j=\infty} \gamma_j^z \gamma_{k-j}^x$$

The explicit form of the auto-covariance function of  $\{y_t\}$  is given by Sowell (1992, p. 175).

---

## FARMAFIT Call

**estimate the parameters of an ARFIMA( $p, d, q$ ) model**

**CALL FARMAFIT( $d, phi, theta, sigma, series <, p, q, opt>$ );**

The inputs to the FARMAFIT subroutine are as follows:

- series* specifies a time series (assuming mean zero).
- p* specifies the set or subset of the AR order. If you do not specify *p*, the default is *p*=0.
- If you specify *p*=3, the FARMAFIT subroutine estimates the coefficient of the lagged variable  $y_{t-3}$ .
- If you specify *p*= $\{1, 2, 3\}$ , the FARMAFIT subroutine estimates the coefficients of lagged variables  $y_{t-1}$ ,  $y_{t-2}$ , and  $y_{t-3}$ .
- q* specifies the subset of the MA order. If you do not specify *q*, the default is *q*=0.
- If you specify *q*=2, the FARMAFIT subroutine estimates the coefficient of the lagged variable  $\epsilon_{t-2}$ .
- If you specify *q*= $\{1, 2\}$ , the FARMAFIT subroutine estimates the coefficients of lagged variables  $\epsilon_{t-1}$  and  $\epsilon_{t-2}$ .
- opt* specifies the method of computing the log-likelihood function.
- opt*=0 requests the conditional sum of squares function. This is the default.
- opt*=1 requests the exact log-likelihood function. This option requires that the time series be stationary and invertible.

The FARMAFIT subroutine returns the following values:

- d* is a scalar containing a fractional differencing order.
- phi* is a vector containing the autoregressive coefficients.
- theta* is a vector containing the moving-average coefficients.
- sigma* is a scalar containing a variance of the innovation series.

To estimate parameters of an ARFIMA(1, 0.3, 1) model

$$(1 - 0.5B)(1 - B)^{0.3}y_t = (1 + 0.1B)\epsilon_t$$

where  $\epsilon_t \sim NID(0, 1)$ , you can specify

```
d      = 0.3;
phi    = 0.5;
theta  = -0.1;
call farmasim(yt, d, phi, theta);
call farmafit(d, ar, ma, sigma, yt) p=1 q=1;
print d ar ma sigma;
```

The FARMAFIT subroutine estimates parameters  $d$ ,  $\phi(B)$ ,  $\theta(B)$ , and  $\sigma_\epsilon^2$  of an ARFIMA( $p, d, q$ ) model. The log-likelihood function needs to be solved by iterative numerical procedures such as the quasi-Newton optimization. The starting value  $d$  is obtained by the approach of Geweke and Porter-Hudak (1983); the starting value of the AR and MA parameters are obtained from the least squares estimates.

---

## FARMALIK Call

**computes the log-likelihood function of an ARFIMA( $p, d, q$ ) model**

**CALL FARMALIK(*lnl, series, d <, phi, theta, sigma, p, q, opt>*);**

The inputs to the FARMALIK subroutine are as follows:

- |               |                                                                                                                                                                                  |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>series</i> | specifies a time series (assuming mean zero).                                                                                                                                    |
| <i>d</i>      | specifies a fractional differencing order. This argument is required; the value of $d$ should be in the open interval $(-1, 1)$ excluding zero.                                  |
| <i>phi</i>    | specifies an $m_p$ -dimensional vector containing the autoregressive coefficients, where $m_p$ is the number of the elements in the subset of the AR order. The default is zero. |
| <i>theta</i>  | specifies an $m_q$ -dimensional vector containing the moving-average coefficients, where $m_q$ is the number of the elements in the subset of the MA order. The default is zero. |
| <i>sigma</i>  | specifies a variance of the innovation series. The default is one.                                                                                                               |
| <i>p</i>      | specifies the subset of the AR order. See the FARMACOV subroutine for additional details.                                                                                        |
| <i>q</i>      | specifies the subset of the MA order. See the FARMACOV subroutine for additional details.                                                                                        |
| <i>opt</i>    | specifies the method of computing the log-likelihood function.                                                                                                                   |
|               | <i>opt=0</i> requests the conditional sum of squares function. This is the default.                                                                                              |

`opt=1` requests the exact log-likelihood function. This option requires that the time series be stationary and invertible.

The FARMALIK subroutine returns the following value:

`lnl` is 3-dimensional vector. `lnl[1]` contains the log-likelihood function of the model; `lnl[2]` contains the sum of the log determinant of the innovation variance; and `lnl[3]` contains the weighted sum of squares of residuals. The log-likelihood function is computed as  $-0.5 \times (\lnl[2] + \lnl[3])$ . If the `opt=0` is specified, only the weighted sum of squares of residuals returns in `lnl[1]`.

To compute the log-likelihood function of an ARFIMA(1, 0.3, 1) model

$$(1 - 0.5B)(1 - B)^{0.3}y_t = (1 + 0.1B)\epsilon_t$$

where  $\epsilon_t \sim NID(0, 1.2)$ , you can specify

```
d      = 0.3;
phi    = 0.5;
theta= -0.1;
sigma= 1.2;
call farmasim(yt, d, phi, theta, sigma);
call farmalik(lnl, yt, d, phi, theta, sigma);
print lnl;
```

The FARMALIK subroutine computes a log-likelihood function of the ARFIMA( $p, d, q$ ) model. The exact log-likelihood function is worked by Sowell (1992); the conditional sum of squares function is worked by Chung (1996).

The exact log-likelihood function only considers a stationary and invertible ARFIMA( $p, d, q$ ) process with  $d \in (-0.5, 0.5) \setminus \{0\}$  represented as

$$\phi(B)(1 - B)^d y_t = \theta(B)\epsilon_t$$

where  $\epsilon_t \sim NID(0, \sigma^2)$ .

Let  $Y_T = [y_1, y_2, \dots, y_T]'$  and the log-likelihood function is as follows without a constant term:

$$\ell = -\frac{1}{2}(\log |\Sigma| + Y_T' \Sigma^{-1} Y_T)$$

where  $\Sigma = [\gamma_{i-j}]$  for  $i, j = 1, 2, \dots, T$ .

The conditional sum of squares function does not require the normality assumption. The initial observations  $y_0, y_{-1}, \dots$  and  $\epsilon_0, \epsilon_{-1}, \dots$  are set to zero.

Let  $y_t$  be an ARFIMA( $p, d, q$ ) process represented as

$$\phi(B)(1 - B)^d y_t = \theta(B)\epsilon_t$$



then the conditional sum of squares function is

$$\ell = -\frac{T}{2} \log \left( \frac{1}{T} \sum_{t=1}^T \epsilon_t^2 \right)$$

---

## FARMASIM Call

**generates an ARFIMA( $p, d, q$ ) process**

**CALL FARMASIM**(*series, d <, phi, theta, mu, sigma, n, p, q, initial, seed*>);

The inputs to the FARMASIM subroutine are as follows:

- |                |                                                                                                                                                                                                                                                                                          |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>d</i>       | specifies a fractional differencing order. This argument is required; the value of <i>d</i> should be in the open interval $(-1, 1)$ excluding zero.                                                                                                                                     |
| <i>phi</i>     | specifies an $m_p$ -dimensional vector containing the autoregressive coefficients, where $m_p$ is the number of the elements in the subset of the AR order. The default is zero.                                                                                                         |
| <i>theta</i>   | specifies an $m_q$ -dimensional vector containing the moving-average coefficients, where $m_q$ is the number of the elements in the subset of the MA order. The default is zero.                                                                                                         |
| <i>mu</i>      | specifies a mean value. The default is zero.                                                                                                                                                                                                                                             |
| <i>sigma</i>   | specifies a variance of the innovation series. The default is one.                                                                                                                                                                                                                       |
| <i>n</i>       | specifies the length of the series. The value of <i>n</i> should be greater than or equal to the AR order. The default is $n = 100$ is used.                                                                                                                                             |
| <i>p</i>       | specifies the subset of the AR order. See the FARMACOV subroutine for additional details.                                                                                                                                                                                                |
| <i>q</i>       | specifies the subset of the MA order. See the FARMACOV subroutine for additional details.                                                                                                                                                                                                |
| <i>initial</i> | specifies the initial values of random variables. The initial value is used for the nonstationary process. If <i>initial</i> = $a_0$ , then $y_{-p+1}, \dots, y_0$ take the same value $a_0$ . If the <i>initial</i> option is not specified, the initial values are set to zero.        |
| <i>seed</i>    | specifies the random number seed. If it is not supplied, the system clock is used to generate the seed. If it is negative, then the absolute value is used as the starting seed; otherwise, subsequent calls ignore the value of <i>seed</i> and use the last seed generated internally. |

The FARMASIM subroutine returns the following value:

- |               |                                                                             |
|---------------|-----------------------------------------------------------------------------|
| <i>series</i> | is an <i>n</i> vector containing the generated ARFIMA( $p, d, q$ ) process. |
|---------------|-----------------------------------------------------------------------------|

To generate an ARFIMA(1, 0.3, 1) process

$$(1 - 0.5B)(1 - B)^{0.3}(y_t - 10) = (1 + 0.1B)\epsilon_t$$

where  $\epsilon_t \sim NID(0, 1.2)$ , you can specify

```
d      = 0.3;
phi    = 0.5;
theta= -0.1;
mu     = 10;
sigma= 1.2;
call farmasim(yt, d, phi, theta, mu, sigma, 100);
print yt;
```

The FARMASIM subroutine generates a time series of length  $n$  from an ARFIMA( $p, d, q$ ) model. If the process is stationary and invertible, the initial values  $y_{-p+1}, \dots, y_0$  are produced using covariance matrices obtained from FARMACOV. If the process is nonstationary, the time series is recursively generated using the user-defined initial value or the zero initial value.

To generate an ARFIMA( $p, d, q$ ) process with  $d \in [0.5, 1)$ ,  $x_t$  is first generated for  $d' \in (-0.5, 0)$ , where  $d' = 1 - d$  and then  $y_t$  is generated by  $y_t = y_{t-1} + x_t$ .

To generate an ARFIMA( $p, d, q$ ) process with  $d \in (-1, -0.5]$ , a two-step approximation based on a truncation of the expansion  $(1 - B)^d$  is used; the first step is to generate an ARFIMA(0,  $d, 0$ ) process  $x_t = (1 - B)^{-d}\epsilon_t$ , with truncated moving-average weights; the second step is to generate  $y_t = \phi(B)^{-1}\theta(B)x_t$ .

---

## FDIF Call

**obtain a fractionally differenced process**

```
CALL FDIF(out, series, d);
```

The inputs to the FDIF subroutine are as follows:

*series*    specifies a time series with  $n$  length.

*d*        specifies a fractional differencing order. This argument is required; the value of  $d$  should be in the open interval  $(-1, 1)$  excluding zero.

The FDIF subroutine returns the following value:

*out*       is an  $n$  vector containing the fractionally differenced process.

Consider an ARFIMA(1, 0.3, 1) process

$$(1 - 0.5B)(1 - B)^{0.3}y_t = (1 + 0.1B)\epsilon_t$$

Let  $z_t = (1 - B)^{0.3}y_t$ , that is,  $z_t$  follows an ARMA(1,1). To get the filtered series  $z_t$ , you can specify

```

d      = 0.3;
phi    = 0.5;
theta= -0.1;
call farmasim(yt, d, phi, theta) n=100;
call fdif(zt, yt, d);
print zt;

```

---

## References

- Chung, C.F. (1996), "A Generalized Fractionally Integrated ARMA Process," *Journal of Time Series Analysis*, 2, 111–140.
- Geweke, J. and Porter-Hudak, S. (1983), "The Estimation and Application of Long Memory Time Series Models," *Journal of Time Series Analysis*, 4, 221–238.
- Granger, C.W.J. and Joyeux, R. (1980), "An Introduction to Long Memory Time Series Models and Fractional Differencing," *Journal of Time Series Analysis*, 1, 15–39.
- Hosking, J.R.M. (1981), "Fractional Differencing," *Biometrika*, 68, 165–176.
- Li, W.K. and McLeod, A.I. (1986), "Fractional Time Series Modeling," *Biometrika*, 73, 217–221.
- Sowell, F. (1992), "Maximum Likelihood Estimation of Stationary Univariate Fractionally Integrated Time Series Models," *Journal of Econometrics*, 53, 165–88.



# Subject Index

## F

FARMACOV Call

generating an ARFIMA( $p, d, q$ ) process, 38

FARMAFIT Call

estimation of an ARFIMA( $p, d, q$ ) model, 40

FARMALIK Call

generating an ARFIMA( $p, d, q$ ) model, 41

FARMASIM Call

generating an ARFIMA( $p, d, q$ ) process, 43

FDIF Call

obtaining a fractionally differenced process, 44

## W

Wavelet Analysis Calls, 22

WAVFT call

computing fast wavelet transform, 22

WAVGET call

extracting wavelet information, 24

WAVIFT call

computing inverse fast wavelet transform, 26

WAVPRINT call

printing wavelet information, 28

WAVTHRSH call

thresholding wavelet detail coefficients, 29



# Syntax Index

## **F**

FARMACOV Call, 38  
FARMAFIT Call, 40  
FARMALIK Call, 41  
FARMASIM Call, 43  
FDIF Call, 44

## **W**

WAVFT call, 22  
WAVGET call, 24  
WAVIFT call, 26  
WAVPRINT call, 28  
WAVTHRSH call, 29





# Your Turn

---

If you have comments or suggestions about *SAS/IML<sup>®</sup> Software: Changes and Enhancements, Release 8.2*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute Inc.  
SAS Publishing  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute Inc.  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [suggest@sas.com](mailto:suggest@sas.com)



*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS® Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

1. Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
2. Leave the "Subject" line blank.
3. Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.



SAS Publishing

TECHNICAL REPORT

# **SAS/QC<sup>®</sup> Software: Changes and Enhancements, Release 8.2**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/QC® Software: Changes and Enhancements, Release 8.2*, Cary, NC: SAS Institute Inc., 2001

**SAS/QC® Software: Changes and Enhancements, Release 8.2**

Copyright © 2001 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-868-9

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2001

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

|                                                                               |    |
|-------------------------------------------------------------------------------|----|
| Chapter 1. The CAPABILITY Procedure . . . . .                                 | 1  |
| Chapter 2. The RELIABILITY Procedure . . . . .                                | 17 |
| Chapter 3. The SHEWHART Procedure . . . . .                                   | 31 |
| Chapter 4. The ADX Interface for Design and Analysis of Experiments . . . . . | 47 |
| Subject Index . . . . .                                                       | 51 |
| Syntax Index . . . . .                                                        | 53 |





# Chapter 1

## The CAPABILITY Procedure

### Chapter Table of Contents

---

|                                                      |    |
|------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                            | 3  |
| <b>SYNTAX</b> . . . . .                              | 3  |
| PROC CAPABILITY Statement . . . . .                  | 3  |
| COMPHISTOGRAM Statement . . . . .                    | 4  |
| HISTOGRAM Statement . . . . .                        | 5  |
| <b>DETAILS</b> . . . . .                             | 6  |
| Specialized Capability Indices . . . . .             | 6  |
| Histogram Interval Midpoints and Endpoints . . . . . | 9  |
| Identifying Histogram Intervals . . . . .            | 10 |
| <b>EXAMPLES</b> . . . . .                            | 10 |
| Example 1.1 Clipping Histogram Bars . . . . .        | 10 |
| Example 1.2 Labeling Histogram Endpoints . . . . .   | 13 |
| <b>REFERENCES</b> . . . . .                          | 14 |



# Chapter 1

## The CAPABILITY Procedure

---

### Overview

You can specify several new options to enhance histograms and comparative histograms produced by the CAPABILITY procedure. In both the COMPHISTOGRAM and HISTOGRAM statements you can specify:

- the BARLABEL= option to display labels over histogram bars
- the CLIPSPEC= option to clip histogram bars at the specification limits
- the ENDPOINTS= option to label histogram endpoints instead of midpoints

In the COMPHISTOGRAM statement you can specify the CTEXTSIDE= and CTEXTTOP= options to control the colors of row and column labels. In the HISTOGRAM statement you can use the NMIDPOINTS= or NENDPOINTS= option to specify the number of histogram interval midpoints or endpoints.

The table produced by the SPECIALINDICES option in the PROC CAPABILITY statement now contains additional specialized process capability indices. The new indices are  $S_{j kp}$ ,  $C_{pp}$ ,  $C_{pp}''$ ,  $C_{pg}$ ,  $C_{pq}$ ,  $C_p^W$ ,  $C_{pk}^W$ ,  $C_{pm}^W$ ,  $C_{pc}$ ,  $C_p(u, v)$ , and  $C_p(v)$ .

---

### Syntax

---

#### PROC CAPABILITY Statement

You can assign parameter values for several of the specialized capability indices calculated when you specify the SPECIALINDICES option.

**SPECIALINDICES** <(parameter-list)>

requests a table of specialized process capability indices. These indices are  $k$ , Boyles' modified  $C_{pm}$  (also denoted as  $C_{pm+}$ ),  $C_{j kp}$ ,  $C_{pm}(a)$ ,  $C_p(5.15)$ ,  $C_{pk}(5.15)$ ,  $C_{pmk}$ , Wright's  $C_s$ , Boyles'  $S_{j kp}$ ,  $C_{pp}$ ,  $C_{pp}''$ ,  $C_{pg}$ ,  $C_{pq}$ ,  $C_p^W$ ,  $C_{pk}^W$ ,  $C_{pm}^W$ ,  $C_{pc}$ , and Vännmann's  $C_p(u, v)$  and  $C_p(v)$ . You can provide values for the parameters  $a$  for  $C_{pm}(a)$ ,  $u$  and  $v$  for  $C_p(u, v)$  and  $C_p(v)$ , and for the  $\gamma$  multiplier for  $C_s$  by specifying the following options in parentheses after the SPECIALINDICES option.

**CPMA=***value*

specifies the *value* of the parameter  $a$  for the capability index  $C_{pm}(a)$  described in Section 3.7 of Kotz and Johnson (1993). The *value* must be positive. The default *value* is 0.5. The existing CPMA= option in the PROC CAPABILITY statement is considered obsolete but still works.

**CPU=value**

specifies the *value* of the parameter  $u$  for Vännmann's capability index  $C_p(u, v)$ . The *value* must be greater than or equal to zero. The default *value* is zero.

**CPV=value**

specifies the *value* of the parameter  $v$  for Vännmann's capability indices  $C_p(u, v)$  and  $C_p(v)$ . The *value* must be greater than or equal to zero. The default *value* is 4.

**CSGAMMA=value**

specifies the *value* of the  $\gamma$  multiplier suggested by Chen and Kotz (1996) for Wright's capability index  $C_s$ . The *value* must be greater than zero. The default *value* is 1.

---

## COMPHISTOGRAM Statement

**BARLABEL=COUNT | PERCENT | PROPORTION**

displays labels above the histogram bars. You can specify BARLABEL=COUNT to display the number of observations associated with a given bar, BARLABEL=PERCENT to display the percentage of observations, and BARLABEL=PROPORTION to display the proportion of observations.

**CLIPSPEC=CLIP | NOFILL**

specifies that histogram bars are to be clipped at the upper and lower specification limit lines when there are no observations outside the specification limits. The bar intersecting the lower specification limit is clipped if there are no observations less than the lower limit; the bar intersecting the upper specification limit is clipped if there are no observations greater than the upper limit. If you specify CLIPSPEC=CLIP, histogram bars are truncated at the specification limits. If you specify CLIPSPEC=NOFILL, the portion of a filled histogram bar outside the specification limit is left unfilled. Specifying CLIPSPEC=NOFILL when histogram bars are not filled has no effect.

**CTEXTSIDE=color**

specifies the color of the row labels displayed along the left side of a comparative histogram. By default, the CTEXT= color is used for these labels.

**CTEXTTOP=color**

specifies the color of the column labels displayed along the top of a comparative histogram. By default, the CTEXT= color is used for these labels.

**ENDPOINTS=value-list | KEY | UNIFORM**

specifies that comparative histogram interval endpoints, rather than midpoints, are to be aligned with horizontal axis tick marks, and specifies how the endpoints are to be determined. The method you specify is used for all process variables analyzed with the COMPHISTOGRAM statement.

If you specify ENDPOINTS=value-list, the *values* must be listed in increasing order and must be evenly spaced. The number of endpoint values is one greater than the number of intervals in the resulting comparative histogram. The first value is the lower bound of the first interval and the last value is the upper bound of the last

interval. If the range of the *values* does not cover the range of the data and any specification limits (LSL and USL) that are given, the list is extended in both directions as necessary to cover these values.

If you specify `ENDPOINTS=KEY`, the procedure first determines the endpoints for the data in the key cell. The initial number of intervals is based on the number of observations in the key cell using the method of Terrell and Scott (1985). The endpoint list for the key cell is then extended in both directions as necessary until it spans the data in the remaining cells. If the key cell contains no observations, the method of determining intervals reverts to `ENDPOINTS=UNIFORM`.

If you specify `ENDPOINTS=UNIFORM`, the procedure determines the endpoints using all the observations as if there were no cells. In other words, the number of endpoints is based on the total sample size using the method of Terrell and Scott (1985).

---

## HISTOGRAM Statement

### **ENDPOINTS**

#### **ENDPOINTS=***value-list*

specifies that histogram interval endpoints, rather than midpoints, are to be aligned with horizontal axis tick marks. If you specify `ENDPOINTS=`*value-list*, the *values* must be listed in increasing order and must be evenly spaced. The number of endpoint values is one greater than the number of intervals in the resulting histogram. The first value is the lower bound of the first interval and the last value is the upper bound of the last interval. If the range of the *values* does not cover the range of the data and any specification limits (LSL and USL) that are given, the list is extended in both directions as necessary to cover these values.

If you specify `ENDPOINTS`, the procedure determines the number of intervals using the method of Terrell and Scott (1985), with interval boundaries at horizontal axis tick mark values.

#### **NENDPOINTS=***value*

specifies the number of endpoints for histogram intervals, which is equal to the number of intervals plus one. If you specify the `NENDPOINTS=` option you do not need to specify the `ENDPOINTS` option.

#### **NMIDPOINTS=***value*

specifies the number of midpoints for histogram intervals.

---

## Details

---

### Specialized Capability Indices

---

This section describes new specialized capability indices that are computed when you specify the SPECIALINDICES option on the PROC CAPABILITY statement.

**The Index**  $S_{jkp}$

Boyles (1994) proposed a smooth version of  $C_{jkp}$  defined as

$$S_{jkp} = S \left( \frac{USL - T}{\sqrt{2E_{X>T}[(X - T)^2]}}, \frac{T - LSL}{\sqrt{2E_{X<T}[(X - T)^2]}} \right)$$

The CAPABILITY procedure estimates  $S_{jkp}$  as

$$\hat{S}_{jkp} = S \left( \frac{USL - T}{\sqrt{2 \sum_{X_i > T} (X_i - T)^2 / n}}, \frac{T - LSL}{\sqrt{2 \sum_{X_i < T} (X_i - T)^2 / n}} \right)$$

where  $S(x, y) = \Phi^{-1}[\{\Phi(x) + \Phi(y)\}/2]/3$ .

**The Index**  $C_{pp}$

Chen (1998) devised a process incapability index based on the  $C_{pm}^*$  index. The first term measures *inaccuracy* and the second measures *imprecision*. The  $C_{pp}$  index is estimated as

$$\hat{C}_{pp} = \left( \frac{\bar{X} - T}{d^*/3} \right)^2 + \left( \frac{s}{d^*/3} \right)^2$$

where  $d^* = \min(USL - T, T - LSL)$ .

**The Index**  $C_{pp}''$

The index  $C_{pp}$  does not handle asymmetric tolerances well, as discussed by Kotz and Lovelace (1998). To address that shortcoming, Chen (1998) defined the index  $C_{pp}''$ , which is estimated by

$$\hat{C}_{pp}'' = \left( \frac{\hat{A}}{d^*/3} \right)^2 + \left( \frac{s}{d^*/3} \right)^2$$

where

$$\hat{A} = \max \left\{ \frac{(\bar{X} - T)d}{T - LSL}, \frac{(T - \bar{X})d}{USL - T} \right\}$$

and  $d = (USL - LSL)/2$ .

**The Index  $C_{pg}$** 

Marcucci and Beazley (1988) defined the index

$$C_{pg} = \frac{1}{C_{pm}^2}$$

which is estimated as

$$\hat{C}_{pg} = \frac{1}{\hat{C}_{pm}^2}$$

**The Index  $C_{pq}$** 

Gupta and Kotz (1997) introduced the index  $C_{pq}$ , which is estimated by

$$\hat{C}_{pq} = \hat{C}_p \left[ 1 - \frac{1}{2} \left( \frac{\bar{X} - T}{s} \right)^2 \right]$$

**The Index  $C_p^W$** 

Bai and Choi (1997) defined the index

$$C_p^W = \frac{C_p}{\sqrt{1 + |1 - 2P_x|}}$$

where  $P_x = \Pr(X \leq \mu)$ . It is estimated by

$$\hat{C}_p^W = \frac{\hat{C}_p}{\sqrt{1 + |1 - 2\hat{P}_x|}}$$

where  $\hat{P}_x$  is the fraction of observations less than or equal to  $\bar{X}$ . For more information on  $C_p^W$ , see Kotz and Lovelace (1998).

**The Index  $C_{pk}^W$** 

Bai and Choi (1997) also proposed the index

$$C_{pk}^W = \min \left\{ \frac{USL - \mu}{3\sigma\sqrt{2P_x}}, \frac{\mu - LSL}{3\sigma\sqrt{2(1 - P_x)}} \right\}$$

It is estimated by

$$\hat{C}_{pk}^W = \min \left\{ \frac{USL - \bar{X}}{3s\sqrt{2\hat{P}_x}}, \frac{\bar{X} - LSL}{3s\sqrt{2(1 - \hat{P}_x)}} \right\}$$

where  $\hat{P}_x$  is the fraction of observations less than or equal to  $\bar{X}$ . For more information on  $C_{pk}^W$ , see Kotz and Lovelace (1998).

**The Index  $C_{pm}^W$** 

The index  $C_{pm}^W$ , also introduced by Bai and Choi (1997), is defined as

$$C_{pm}^W = \frac{C_{pm}}{\sqrt{1 + |1 - 2P_T|}}$$

where  $P_T = \Pr(X \leq T)$ . It is estimated by

$$\hat{C}_{pm}^W = \frac{\hat{C}_{pm}}{\sqrt{1 + |1 - 2\hat{P}_T|}}$$

where  $\hat{P}_T$  is the fraction of observations less than or equal to  $T$ . For more information on  $C_{pm}^W$ , see Kotz and Lovelace (1998).

### **The Index $C_{pc}$**

Luceño (1996) proposed the index

$$C_{pc} = \frac{USL - LSL}{6\sqrt{\frac{\pi}{2}}E|X - M|}$$

where  $M = (USL + LSL)/2$ . It is estimated by

$$\hat{C}_{pc} = \frac{USL - LSL}{6\sqrt{\frac{\pi}{2}}c}$$

where

$$c = \frac{1}{n} \sum_{i=1}^n |X_i - M|$$

### **Vännmann's Index $C_p(u, v)$**

Vännmann (1995) introduced the generalized index  $C_p(u, v)$ , which reduces to the following capability indices given appropriate choices of  $u$  and  $v$ :

- $C_p(0, 0) = C_p$
- $C_p(0, 1) = C_{pk}$
- $C_p(1, 0) = C_{pm}$
- $C_p(1, 1) = C_{pmk}$

$C_p(u, v)$  is defined as

$$C_p(u, v) = \frac{d - u|\mu - M|}{3\sqrt{\sigma^2 + v(\mu - T)^2}}$$

and estimated by

$$\hat{C}_p(u, v) = \frac{d - u|\bar{X} - M|}{3\sqrt{(\frac{n-1}{n})s^2 + v(\bar{X} - T)^2}}$$



You can specify  $u$  with the SPECIALINDICES(CPU=) option and  $v$  with the SPECIALINDICES(CPV=) option. By default,  $u = 0$  and  $v = 4$ .

### Vännmann's Index $C_p(v)$

Vännmann (1997) also proposed the index  $C_p(v)$ , which is equivalent to  $C_p(u, v)$  with  $u = 1$ . It is estimated as

$$\hat{C}_p(v) = \frac{d - |\bar{X} - M|}{3\sqrt{(\frac{n-1}{n})s^2 + v(\bar{X} - T)^2}}$$

You can specify  $v$  with the SPECIALINDICES(CPV=) option. By default,  $v = 4$ .

### The Modified Index $C_s$

Chen and Kotz (1996) proposed a modification to Wright's  $C_s$  index which introduces a multiplier,  $\gamma > 0$ , and is estimated as

$$\hat{C}_s = \frac{(\text{USL} - \text{LSL})/2 - |\bar{X} - m|}{3\sqrt{(\frac{n-1}{n})s^2 + (\bar{X} - T)^2 + \gamma|c_4s^2b_3|}}$$

You can specify a value for  $\gamma$  with the SPECIALINDICES(CSGAMMA=) option.

---

## Histogram Interval Midpoints and Endpoints

By default, histogram intervals are centered on the major tick marks of the horizontal axis. The major tick mark values are the *midpoints* of the histogram intervals. The midpoints are evenly spaced, and the difference between consecutive midpoints is used as the width of the histogram bars. The CAPABILITY procedure determines the number of intervals using the algorithm described in Terrell and Scott (1985).

You can specify the number of intervals and their midpoint values with the MIDPOINTS=*value-list* option in a HISTOGRAM or COMPHISTOGRAM statement. In a COMPHISTOGRAM statement, you can specify MIDPOINTS=KEY or MIDPOINTS=UNIFORM. If you specify MIDPOINTS=KEY, the procedure first determines the midpoints for the data in the key cell using the method of Terrell and Scott (1985). It then extends the midpoint list for the key cell in both directions as necessary until it spans the data in the remaining cells. If you specify MIDPOINTS=UNIFORM, the procedure determines the midpoints using all the observations as if there were no cells.

In a HISTOGRAM statement you can specify the number of histogram intervals using the NMIDPOINTS= option. The CAPABILITY procedure then determines the midpoint values.

You can specify the ENDPOINTS option in a HISTOGRAM statement to produce a histogram whose interval boundaries, or *endpoints*, are aligned with major tick marks on the horizontal axis. The procedure uses the method of Terrell and Scott (1985) to determine the number of histogram intervals. The number of major tick marks on the horizontal axis of the resulting histogram is the number of intervals plus one. The least major tick mark value is the lower bound of the first interval and the greatest value is the upper bound of the last interval.

You can use the `ENDPOINTS=value-list` option in a `HISTOGRAM` or `COMPHISTOGRAM` statement to specify the number of interval endpoints and their values. The number of values is one greater than the number of intervals on the resulting histogram. You can specify `ENDPOINTS=KEY` or `ENDPOINTS=UNIFORM` in a `COMPHISTOGRAM` statement. The number of intervals is determined just as with `MIDPOINTS=KEY` or `MIDPOINTS=UNIFORM`, but the interval boundary values will not be the same, in general.

You can use the `NENDPOINTS=` option in a `HISTOGRAM` statement to align interval endpoints with major tick marks and to specify the number of endpoints. The procedure determines the endpoint values. If you specify the `ENDPOINTS=` or `NENDPOINTS=` option, you need not specify the `ENDPOINTS` option.

The `ENDPOINTS=`, `MIDPOINTS=`, `NENDPOINTS=`, and `NMIDPOINTS=` options are mutually exclusive. The `MIDPOINTS=` option takes precedence over the `NMIDPOINTS=` option, and the `ENDPOINTS=` options takes precedence over the `NENDPOINTS=` option. The `ENDPOINTS=` and `NENDPOINTS=` options override either `MIDPOINTS=` or `NMIDPOINTS=`.

---

## Identifying Histogram Intervals

When you specify the `OUTHISTOGRAM=` or `MIDPERCENTS` option in a `HISTOGRAM` or `COMPHISTOGRAM` statement, the resulting data set or table identifies histogram intervals by their midpoint values by default. The `_MIDPT_` variable identifies intervals in an `OUTHISTOGRAM=` data set. When the `ENDPOINTS`, `ENDPOINTS=`, or `NENDPOINTS=` option is specified, interval endpoints are labeled. Each interval has two endpoints, but is identified in a data set or table by one endpoint value. By default, the left endpoint is used and an `OUTHISTOGRAM=` data set contains the variable `_MINPT_`. If the `RTINCLUDE` option is specified, right endpoints are used to identify intervals and the `OUTHISTOGRAM=` data set variable is `_MAXPT_`.

---

## Examples

---

### Example 1.1. Clipping Histogram Bars

A semiconductor manufacturer produces printed circuit boards that are sampled to determine whether the thickness of their copper plating lies between a lower specification limit of 3.45 mils and an upper specification limit of 3.55 mils. The plating process is assumed to be in statistical control. The plating thicknesses of 100 boards are saved in a data set named `trans`, created by the following statements:

```
data trans;
  input thick @@;
  label thick = 'Plating Thickness (mils)';
  datalines;
3.468 3.488 3.509 3.506 3.481 3.492 3.478 3.546 3.502 3.512
3.490 3.482 3.498 3.519 3.504 3.469 3.497 3.495 3.508 3.523
3.449 3.488 3.463 3.500 3.549 3.525 3.461 3.489 3.514 3.470
```

```

3.461 3.506 3.464 3.489 3.524 3.531 3.501 3.495 3.543 3.510
3.481 3.497 3.461 3.513 3.528 3.496 3.533 3.480 3.516 3.476
3.512 3.550 3.481 3.541 3.549 3.531 3.468 3.494 3.522 3.520
3.505 3.523 3.475 3.470 3.507 3.536 3.528 3.477 3.536 3.491
3.510 3.461 3.531 3.502 3.491 3.506 3.539 3.513 3.496 3.539
3.469 3.481 3.515 3.535 3.460 3.475 3.488 3.515 3.484 3.482
3.517 3.483 3.467 3.467 3.502 3.471 3.516 3.474 3.500 3.466
;

```

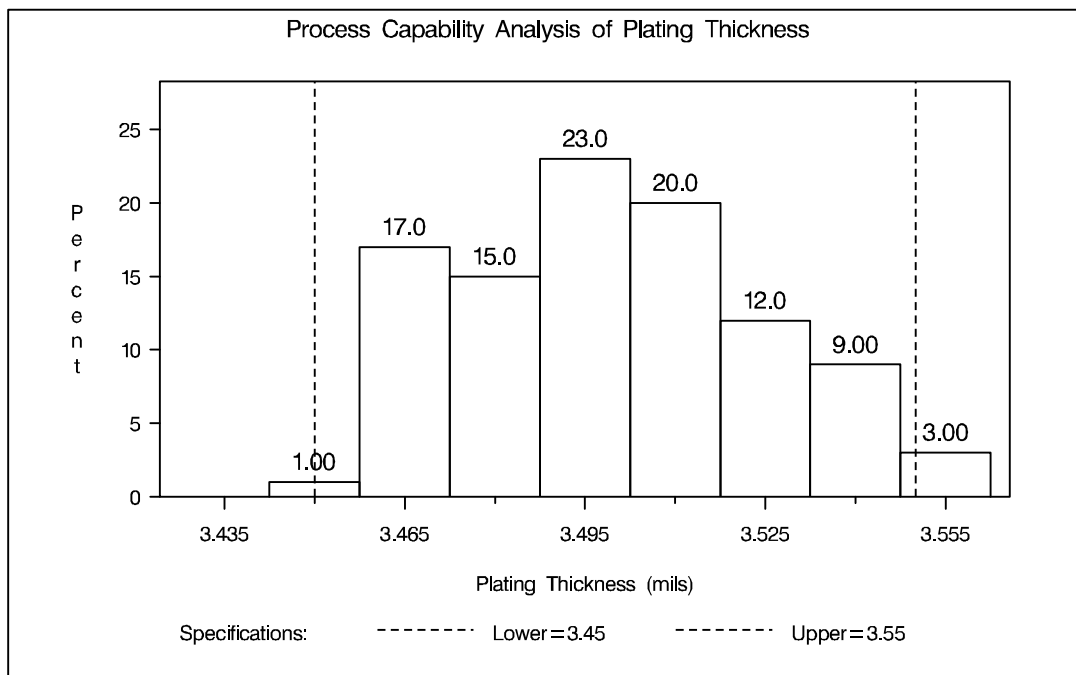
The following statements create the histogram shown in Output 1.1.1. Note that the BARLABEL=PERCENT option displays a label above each histogram bar indicating the percentage of observations represented by that bar.

```

title 'Process Capability Analysis of Plating Thickness';
proc capability data=trans noprint;
  spec lsl = 3.45 llsl = 2 clsl = black
      usl = 3.55 lusl = 2 cusl = black;
  histogram thick / cframe = ligr
                  cfill   = blue
                  barlabel = percent;
run;

```

**Output 1.1.1.** Histogram with Bar Labels



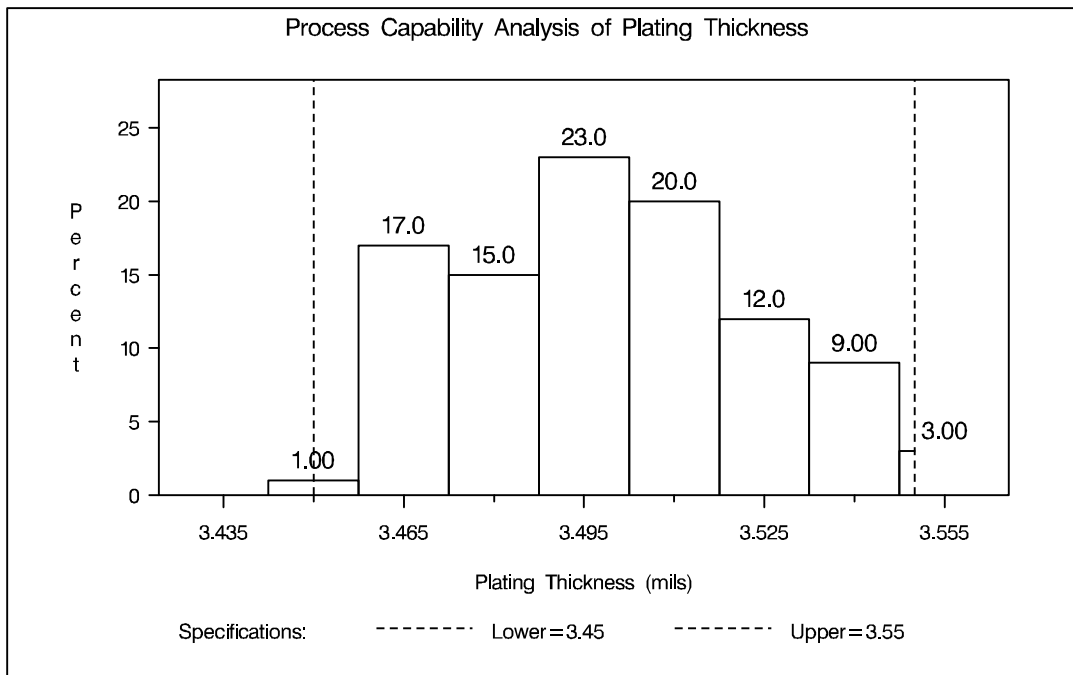
You can use the CLIPSPEC= option to clip histogram bars at the specification limits. A histogram bar intersected by a specification limit is clipped only when no observations lie outside that limit, so clipped bars indicate that all observations are within specifications. These statements create the histogram shown in Output 1.1.2:

```

title 'Process Capability Analysis of Plating Thickness';
proc capability data=trans noprint;
  spec lsl = 3.45 lls1 = 2 cls1 = black
    usl = 3.55 lus1 = 2 cus1 = black;
  histogram thick / cframe = ligr
    cfill = blue
    barlabel = percent
    clipspec = clip;
run;

```

**Output 1.1.2.** Histogram with Bar Clipping



The maximum observation in the `trans` data set is 3.55, which is equal to the upper specification limit (USL). Therefore, the rightmost histogram bar in Output 1.1.2 is clipped at the USL, because no observations lie outside that limit. The lower specification limit (LSL) intersects the leftmost histogram bar. However, the minimum observation is 3.449, which is less than the LSL of 3.45. Therefore, the leftmost histogram bar is not clipped.

## Example 1.2. Labeling Histogram Endpoints

This example uses the `trans` data set from the preceding example.

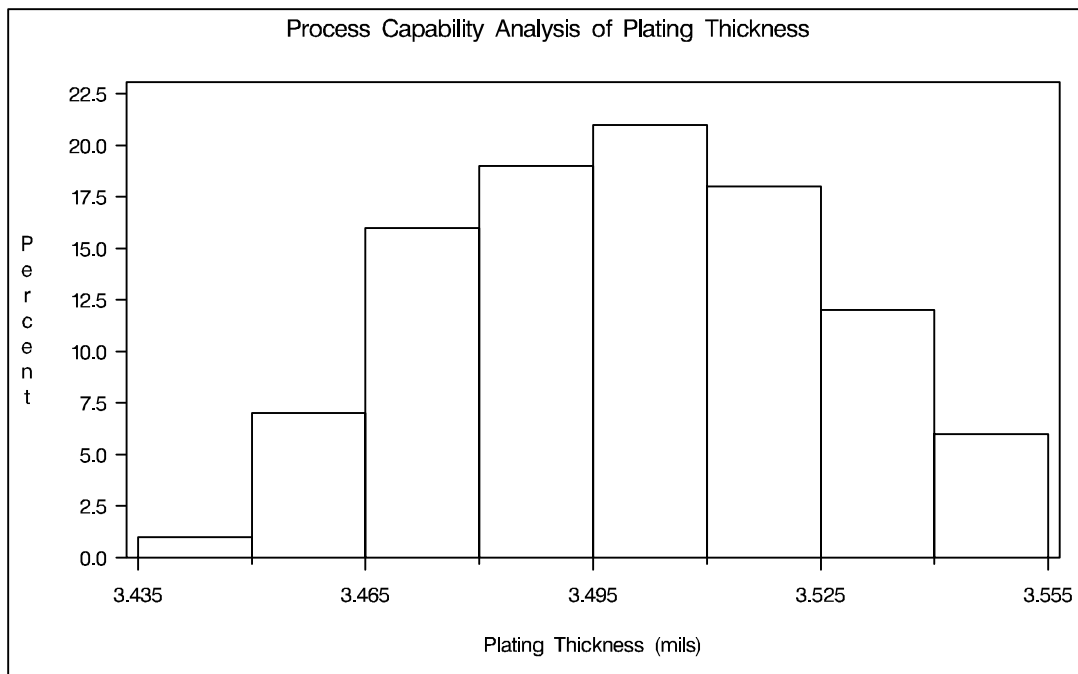
By default, major tick marks on the horizontal axis of a histogram correspond to midpoints of the histogram intervals (see Output 1.1.1). You can use the `ENDPOINTS=` and `NENDPOINTS=` options to create histograms and comparative histograms whose interval boundaries, or endpoints, are aligned with major tick marks. The following statements create the histogram shown in Output 1.2.1:

```

title 'Process Capability Analysis of Plating Thickness';
proc capability data=trans noprint;
  spec lsl = 3.45 llsl = 2 clsl = black
      usl = 3.55 lusl = 2 cusl = black;
  histogram thick / cframe = ligr
                  cfill = blue
                  endpoints;
run;

```

**Output 1.2.1.** Histogram with Endpoints Labeled



The tick mark values in Output 1.2.1 are identical to those in Output 1.1.1, but the histogram intervals lie between the tick marks. Because their intervals have different boundaries, the histograms in Output 1.1.1 and Output 1.2.1 look somewhat different. Note that a histogram produced with the `ENDPOINTS` option will not *necessarily* have the same tick mark values on its horizontal axis as a histogram of the same data produced without the `ENDPOINTS` option.

You can use the `NMIDPOINTS=` or the `NENDPOINTS=` option to specify the number of interval midpoints or endpoints in a histogram. The following statements create

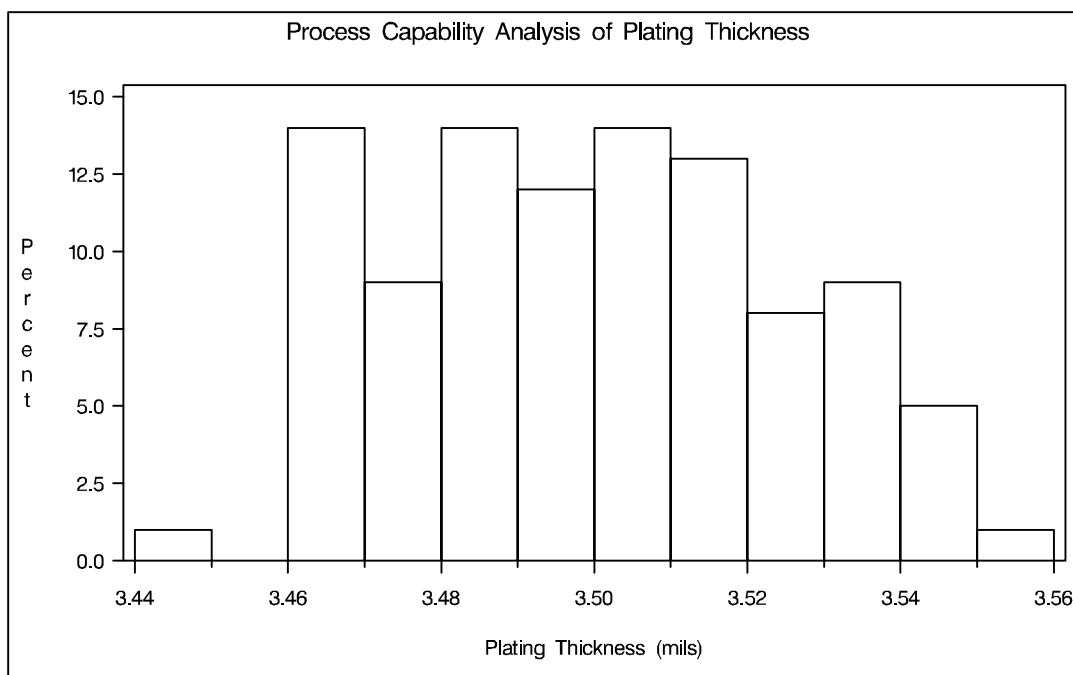
the histogram shown in Output 1.2.2:

```

title 'Process Capability Analysis of Plating Thickness';
proc capability data=trans noprint;
  spec lsl = 3.45 llsl = 2 clsl = black
      usl = 3.55 lusl = 2 cusl = black;
  histogram thick / cframe      = ligr
                    cfill      = blue
                    nendpoints = 13;
run;

```

**Output 1.2.2.** Histogram with Number of Endpoints Specified



## References

- Bai, D. S. and Choi, I. S. (1997), "Process Capability Indices for Skewed Populations," *Manuscript*, Korean Advanced Institute of Science and Technology, Taejeon, Korea.
- Boyles, R. A. (1994), "Process Capability with Asymmetric Tolerances," *Communication and Statistics, Part B—Simulation and Computation*, 23(3), 615–643.
- Chen, H. F. and Kotz, S. (1996), "An Asymptotic Distribution of Wright's Process Capability Index Sensitive to Skewness," *Journal of Statistical Computation and Simulation*, 55, 147–158.
- Chen, K. S. (1998), "Incapability Index with Asymmetric Tolerances," *Statistica Sinica*, 8, 253–262.
- Gupta, A. K. and Kotz, S. (1997), "A New Process Capability Index," *Metrika*, 45, 213–224.

- Kotz, S. and Johnson, N. L. (1993), *Process Capability Indices*, London: Chapman & Hall.
- Kotz, S. and Lovelace, C. R. (1998), *Process Capability Indices in Theory and Practice*, London: Arnold.
- Luceño, A. (1996), “A Process Capability Index with Reliable Confidence Intervals,” *Communications in Statistics – Simulation*, 25(1), 235–245.
- Marcucci, M. O. and Beazley, C. F. (1988), “Capability Indices: Process Performance Measures,” *Transactions of ASQC Congress*, 516–523.
- Terrell, G. R. and Scott, D. W. (1985), “Oversmoothed Nonparametric Density Estimates,” *Journal of the American Statistical Association*, 80, 209–214.
- Vännmann, K. (1995), “A Unified Approach to Capability Indices,” *Statistica Sinica*, 5(2), 805–820.
- Vännmann, K. (1997), “A General Class of Capability Indices in the Case of Asymmetric Tolerances,” *Communications in Statistics – Theory and Methods*, 26(8), 2049–2072.





# Chapter 2

## The RELIABILITY Procedure

### Chapter Table of Contents

---

|                                                             |    |
|-------------------------------------------------------------|----|
| <b>OVERVIEW</b>                                             | 19 |
| <b>SYNTAX</b>                                               | 19 |
| LOGSCALE Statement                                          | 19 |
| MCFPLOT Statement                                           | 20 |
| MODEL Statement                                             | 20 |
| PROBPLOT Statement                                          | 21 |
| RELATIONPLOT Statement                                      | 21 |
| <b>EXAMPLES</b>                                             | 24 |
| Example 2.1 Regression Model with Non-Constant Scale        | 24 |
| Example 2.2 Regression Model with Two Independent Variables | 27 |
| <b>REFERENCES</b>                                           | 30 |



## Chapter 2

# The RELIABILITY Procedure

---

### Overview

The experimental TRELIABILITY procedure in Release 8.1 of the SAS System has now been incorporated into the RELIABILITY procedure.

The following are the major changes and enhancements available in the RELIABILITY procedure:

- You can specify the logarithm of the distribution scale parameter as a function of explanatory variables in regression models.
- You can construct simultaneous confidence bands on probability plots.
- The default confidence intervals on probability plots are pointwise asymptotic normal parametric intervals on cumulative failure probability. The default intervals in PROC RELIABILITY are pointwise parametric intervals on distribution percentiles.
- Multiple overlaying symbols on plots are labeled with the count of points instead of being jittered as in PROC RELIABILITY.
- You can plot nonlinear relationships on relation plots.
- You can construct relation plots with two independent variables.

---

### Syntax

The following statements and statement options are now available in the RELIABILITY procedure.

---

#### LOGSCALE Statement

**LOGSCALE** *<effect-list>* *</options>* ;

LOGSCALE is a new statement in the RELIABILITY procedure. You use the LOGSCALE statement to model the logarithm of the distribution scale parameter as a function of explanatory variables. A MODEL statement must be present to specify the model for the distribution location parameter. *effect-list* is a list of variables in the input data set representing the values of the independent variables in the model for each observation, and combinations of variables representing interaction terms. It can contain any variables or combination of variables in the input data set. It can

contain the same variables as the MODEL statement, or it can contain different variables. The variables in the *effect-list* can be any combination of indicator variables named in a CLASS statement as well as continuous variables. The coefficients of the explanatory variables are estimated by maximum likelihood.

The following *options* are available for the LOGSCALE statement.

**Table 2.1.** LOGSCALE Statement Options

| Option                               | Option Description                                                                                      |
|--------------------------------------|---------------------------------------------------------------------------------------------------------|
| INITIAL= <i>number list</i>          | specifies initial values for log-scale regression parameters other than the location, or intercept term |
| INTERCEPT= <i>number</i> < INTINIT > | specifies initial or fixed value of the intercept parameter, depending on whether INTINIT is present    |

## MCFPLOT Statement

The following options are new or can take additional values.

**Table 2.2.** MCFPLOT Statement Options

| Option                                                | Option Description                                   |
|-------------------------------------------------------|------------------------------------------------------|
| PLOTSYMBOL= <i>symbol</i>  <br>( <i>symbol list</i> ) | symbols representing events in an MCF plot           |
| PLOTCOLOR= <i>color</i>  <br>( <i>color list</i> )    | colors of symbols representing events in an MCF plot |

## MODEL Statement

The following are new options in the MODEL statement.

**Table 2.3.** New MODEL Statement Options

| Option                                                                                                                                                                                      | Option Description                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RELATION=ARRHENIUS  <br>ARRHENIUS2  <br>POWER  <br>LOGISTIC<br>RELATION=( ARRHENIUS  <br>ARRHENIUS2  <br>POWER<br>LOGISTIC<br>< , ><br>ARRHENIUS  <br>ARRHENIUS2  <br>POWER  <br>LOGISTIC ) | the <i>logistic</i> function, defined as $T(x) = \log\left(\frac{x}{1-x}\right)$ , has been added as an optional transformation of the independent variable in a regression model. The logistic transformation is useful when the variable is naturally between 0 and 1. Values of 0, 1, and values outside 0 and 1 are treated as missing values. |

## PROBPLOT Statement

The following options are new or can take additional values.

**Table 2.4.** PROBPLOT Statement Options

| Option                                                  | Option Description                                                                                                                                                                                                                                                    |
|---------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CFIT= <i>color</i>   ( <i>color list</i> )              | color for fit lines and confidence curves in a probability plot                                                                                                                                                                                                       |
| LFIT= <i>linetype</i>   ( <i>linetype list</i> )        | line styles for fit lines and confidence curves in a probability plot. The <i>linetype list</i> is a list of numbers from 1 to 46 representing different linetypes, and can be separated by blanks or commas or can be a list in the form $n_1$ to $n_2$ <by $n_3$ >. |
| NOPPOS                                                  | suppresses plotting of symbols for failures in a probability plot                                                                                                                                                                                                     |
| NPINTERVALS=POINTWIASE  <br>SIMULTANEOUS                | type of nonparametric confidence interval displayed in a probability plot                                                                                                                                                                                             |
| PINTERVALS=PROBABILITY  <br>PERCENTILES  <br>LIKELIHOOD | type of parametric pointwise confidence interval displayed in a probability plot. The default type is PROBABILITY, pointwise confidence intervals on cumulative failure probability.                                                                                  |
| PPOSSYMBOL= <i>symbol</i>   ( <i>symbol list</i> )      | symbols representing failures on a probability plot                                                                                                                                                                                                                   |
| PPOSCOLOR= <i>color</i>   ( <i>color list</i> )         | colors of symbols representing failures on a probability plot                                                                                                                                                                                                         |
| SHOWMULTIPLES                                           | display the count for multiple overlaying symbols                                                                                                                                                                                                                     |

## RELATIONPLOT Statement

The following options are new or can take additional values.

**Table 2.5.** RELATIONPLOT Statement Options

| Option                                         | Option Description                                                                          |
|------------------------------------------------|---------------------------------------------------------------------------------------------|
| CFIT= <i>color</i>   ( <i>color list</i> )     | color for fit lines and confidence curves in a probability plot                             |
| CPLOTFIT= <i>color</i>   ( <i>color list</i> ) | colors for percentile lines                                                                 |
| FITTYPE=                                       | specifies method of estimating distribution parameters                                      |
| LSYX                                           | -least squares fit to the probability plot. The probability axis is the dependent variable. |
| LSXY                                           | -least squares fit to the probability plot. The lifetime axis is the dependent variable.    |
| MLE                                            | -maximum likelihood (default)                                                               |

**Table 2.5.** RELATIONPLOT Statement Options (continued)

| Option                                                        | Option Description                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MODEL                                                         | -use the fit from the preceding MODEL statement                                                                                                                                                                                                                                                     |
| NONE                                                          | -no fit is computed                                                                                                                                                                                                                                                                                 |
| REGRESSION                                                    | -use the fit from the preceding MODEL statement. Non-linear relations and percentiles from models using two independent variables can be plotted.                                                                                                                                                   |
| WEIBAYES                                                      | -Weibayes method                                                                                                                                                                                                                                                                                    |
| LFIT= <i>linetype</i>   ( <i>linetype list</i> )              | line styles for fit lines and confidence curves in a probability plot. The <i>linetype list</i> is a list of numbers from 1 to 46 representing different linetypes, and can be separated by blanks or commas or can be a list in the form $n_1$ to $n_2$ <by $n_3$ >.                               |
| LPLOTFIT= <i>linetype</i>   ( <i>linetype list</i> )          | line styles for percentile lines. <i>linetype list</i> is a list of numbers representing different line-types, and can be separated by blanks or commas or can be a list in the form $n_1$ to $n_2$ <by $n_3$ >.                                                                                    |
| NOPPOS                                                        | suppresses plotting of symbols for failures in a probability plot                                                                                                                                                                                                                                   |
| PINTERVALS=PROBABILITY  <br>PERCENTILES  <br>LIKELIHOOD       | type of parametric pointwise confidence interval displayed in a probability plot. The default type is PROBABILITY, pointwise confidence intervals on cumulative failure probability.                                                                                                                |
| RCENCOLOR= <i>color</i>   ( <i>color list</i> )               | colors for the symbols representing uncensored, right censored, and left censored observations in a relation plot                                                                                                                                                                                   |
| RCENSYMBOL= <i>symbol</i>   ( <i>symbol list</i> )            | symbols representing right censored and left censored observations in a relation plot. The <i>symbol</i> is one of the symbol names (plus, star, square, diamond, triangle, hash, paw, point, dot, circle) or a letter (A–Z).                                                                       |
| RELATION=ARRHENIUS  <br>ARRHENIUS2  <br>POWER  <br>LOGISTIC ) | the <i>logistic</i> function, defined as $T(x) = \log\left(\frac{x}{1-x}\right)$ , has been added as an optional transformation of the stress axis. The logistic transformation is useful when the variable is naturally between 0 and 1. Values of 0, 1, and values outside 0 and 1 are not valid. |
| SHOWMULTIPLES                                                 | display the count for multiple overlaying symbols                                                                                                                                                                                                                                                   |

**Table 2.5.** RELATIONPLOT Statement Options (continued)

| Option                      | Option Description                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>variable=number list</i> | allows plots of percentiles from a regression model when two independent variables are used in a MODEL statement <i>effect list</i> . The FIT=REGRESSION option must be used with this option. Percentile plots are created for each value of the independent <i>variable</i> in the <i>number list</i> . <i>number list</i> is a list of numeric values separated by blanks or commas, or in the form of a list $n_1$ to $n_2$ <by $n_3$ >. |

## Examples

The examples in this section illustrate some of the new features in the RELIABILITY procedure.

### Example 2.1. Regression Model with Non-Constant Scale

Nelson (1990, p. 272) and Meeker and Escobar (1998, p. 439) analyzed data from a strain-controlled fatigue test on 26 specimens of a type of superalloy. The following SAS statements create a SAS data set containing for each specimen the level of pseudo-stress (PSTRESS), the number of cycles (in thousands) (KCYCLES) until failure or removal from the test, and a variable to indicate whether a specimen failed (F) or was right censored (C) (STATUS):

```
data alloy;
  input pstress kcycles status$ @@;
  cen = ( status = 'C' );
  datalines;
80.3 211.629 F 99.8 43.331 F
80.6 200.027 F 100.1 12.076 F
80.8 57.923 C 100.5 13.181 F
84.3 155.000 F 113.0 18.067 F
85.2 13.949 F 114.8 21.300 F
85.6 112.968 C 116.4 15.616 F
85.8 152.680 F 118.0 13.030 F
86.4 156.725 F 118.4 8.489 F
86.7 138.114 C 118.6 12.434 F
87.2 56.723 F 120.4 9.750 F
87.3 121.075 F 142.5 11.865 F
89.7 122.372 C 144.5 6.705 F
91.3 112.002 F 145.9 5.733 F
;
run;
```

The following statements use PROC RELIABILITY to fit a Weibull regression model with the number of cycles to failure as the response variable. The data set=RESIDS contains standardized residuals created with the ODS OUTPUT statement. The MODEL statement specifies a model quadratic in the log of pseudo-stress for the extreme value location parameter. The quadratic model in pseudo-stress PSTRESS is specified in the MODEL statement, and the RELATION=POW option specifies that the log transformation be applied to PSTRESS in the MODEL statement and the LOGSCALE statement. The LOGSCALE statement specifies the log of the scale parameter as a linear function of the log of PSTRESS. The RPLOT statement specifies a plot of the data and the fitted regression model versus the variable PSTRESS. The FIT=REGRESSION option specifies plotting the regression model fitted with the preceding MODEL statement. The RELATION=POW option specifies using a log stress axis. The PLOTFIT option specifies plotting the 10th, 50th, and 90th percentiles of the regression model at each stress level. The SLOWER, SUPPER, and LUPPER options control limits on the stress and lifetime axes:



```

ods output ModObstats = Resids;
proc reliability data = alloy;
  distribution Weibull;
  model kcycles*cen(1) = pstress pstress*pstress / relation = pow;
  logscale pstress;
  rplot kcycles*cen(1) = pstress / fit=regression
                                relation = pow
                                plotfit 10 50 90
                                slower=60 supper=160
                                lupper=500;

  label pstress = "Pseudo-Stress";
  label kcycles = "Thousands of Cycles";
run;

```

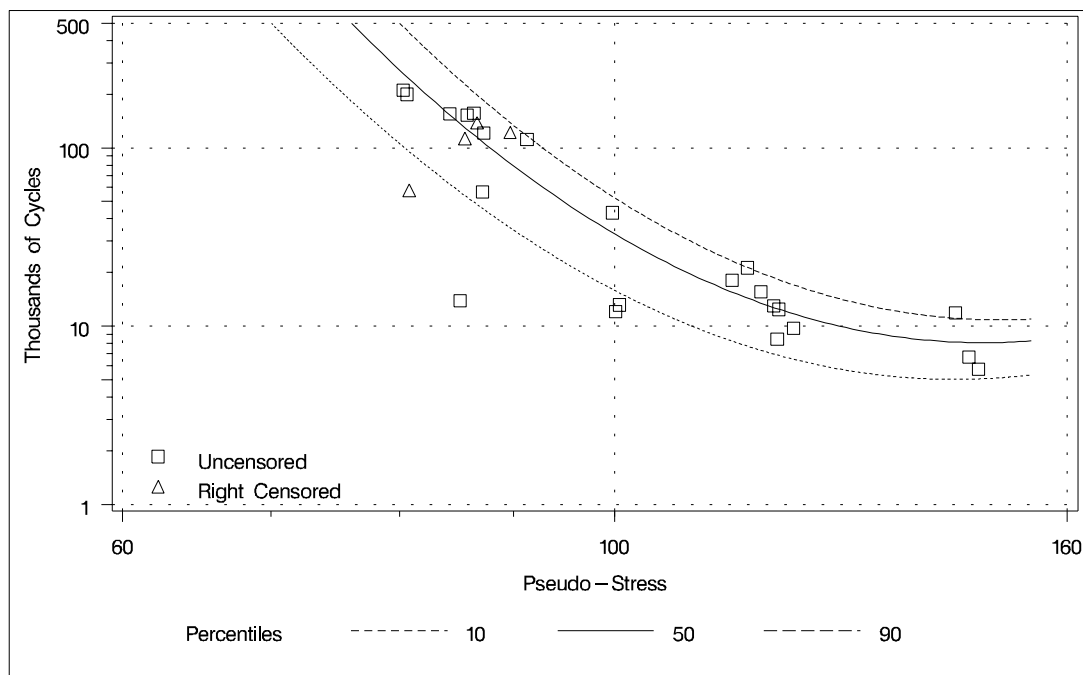
Output 2.1.1 displays the parameter estimates from the fitted regression model. Parameter estimates for both the model for the location parameter and the scale parameter models are shown. Standard errors and confidence limits for all parameter estimates are included.

**Output 2.1.1.** Parameter Estimates for Fitted Regression Model

| The RELIABILITY Procedure     |          |                |                                         |          |
|-------------------------------|----------|----------------|-----------------------------------------|----------|
| Weibull Parameter Estimates   |          |                |                                         |          |
| Parameter                     | Estimate | Standard Error | Asymptotic Normal 95% Confidence Limits |          |
|                               |          |                | Lower                                   | Upper    |
| Intercept                     | 243.1681 | 58.0666        | 129.3596                                | 356.9766 |
| pstress                       | -96.5240 | 24.7075        | -144.9498                               | -48.0983 |
| pstress*pstress               | 9.6653   | 2.6247         | 4.5210                                  | 14.8095  |
| Log-Scale Parameter Estimates |          |                |                                         |          |
| Parameter                     | Estimate | Standard Error | Asymptotic Normal 95% Confidence Limits |          |
|                               |          |                | Lower                                   | Upper    |
| Intercept                     | 4.4666   | 4.0724         | -3.5152                                 | 12.4485  |
| pstress                       | -1.1757  | 0.8731         | -2.8870                                 | 0.5355   |

Output 2.1.2 displays the plot of the data and fitted regression model.

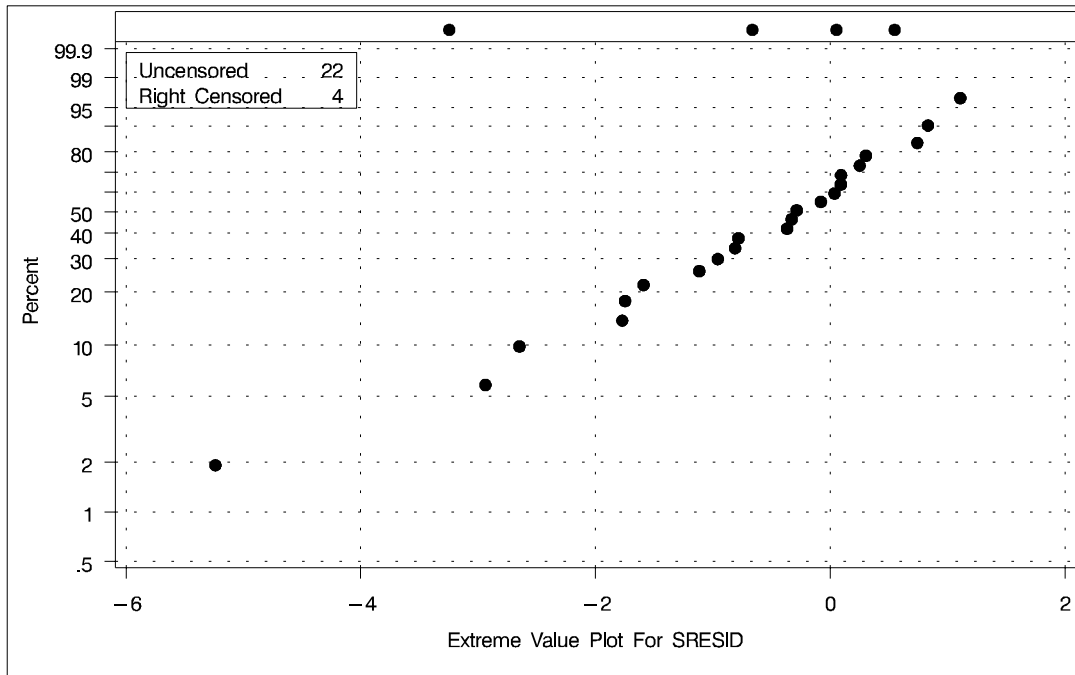
**Output 2.1.2.** Superalloy Fatigue Data with Fitted Regression Model



The following SAS statements create an extreme values probability plot of standardized residuals from the regression model shown in Output 2.1.3:

```
proc reliability data = Resids;
  distribution ev;
  pplot sresid*cen(1) / nofit;
run;
```

**Output 2.1.3.** Residuals for Superalloy Fatigue Data Regression Model



## Example 2.2. Regression Model with Two Independent Variables

Meeker and Escobar (1998, p. 447) analyze data from an accelerated life test on the lifetimes of glass capacitors as a function of operating voltage and temperature. The following SAS statements create a SAS data set containing the data. There are four lifetimes for each of eight combinations and four censored observations after the fourth failure for each combination:

```
data glass;
  input Temp Voltage @;
  do i = 1 to 4;
    Cen = 0;
    input Hours @;
    output;
  end;
  do i = 1 to 4;
    Cen = 1;
    output;
  end;
  datalines;
170 200 439 904 1092 1105
170 250 572 690 904 1090
170 300 315 315 439 628
170 350 258 258 347 588
180 200 959 1065 1065 1087
180 250 216 315 455 473
180 300 241 315 332 380
180 350 241 241 435 455
;
```

The following statements use PROC RELIABILITY to analyze the capacitor data. The MODEL statement fits a regression model with TEMP and VOLTAGE as independent variables. Parameter estimates from the fitted regression model are shown in Output 2.2.1. An interaction term between TEMP and VOLTAGE is included. The PPLOT statement creates a Weibull probability plot shown in Output 2.2.2 with all temperature-voltage combinations overlaid on the same plot. The regression model fit is also plotted. The RPLOT statement creates the plot shown in Output 2.2.3 of the data and Weibull distribution percentiles from the regression model as a function of voltage for values of temperature of 150, 170, and 180:

```
proc reliability data = glass;
  distribution Weibull;
  model Hours*Cen(1) = temp voltage temp * voltage;

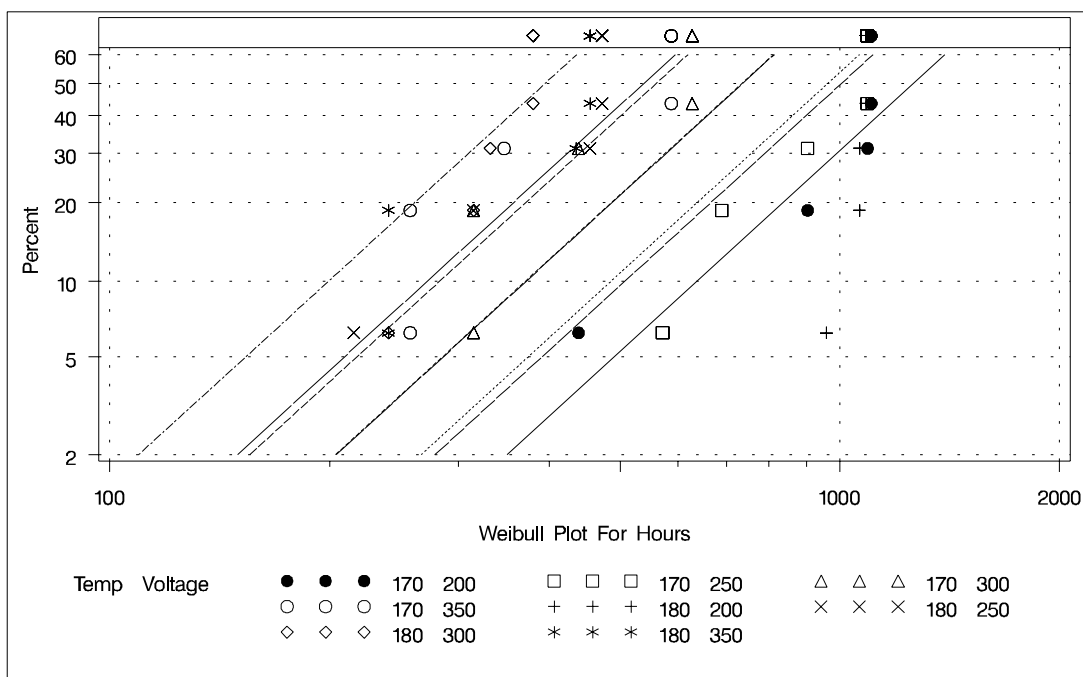
  pplot Hours*Cen(1) = ( temp voltage ) / fit = model
                                overlay
                                noconf
                                lupper = 2000
                                lfit = ( 1 to 8 );

  rplot Hours*Cen(1) = voltage / fit = regression
                                plotfit
                                temp = 150, 170, 180;
run;
```

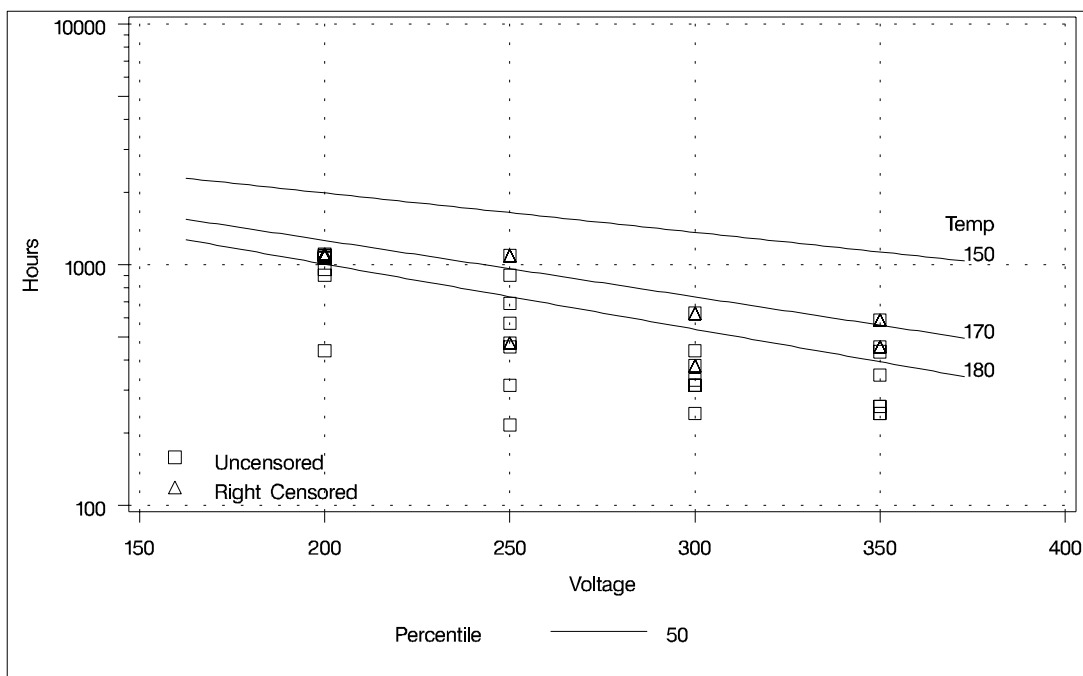
**Output 2.2.1.** Parameter Estimates for Fitted Regression Model

| The RELIABILITY Procedure   |          |                   |                                            |         |
|-----------------------------|----------|-------------------|--------------------------------------------|---------|
| Weibull Parameter Estimates |          |                   |                                            |         |
| Parameter                   | Estimate | Standard<br>Error | Asymptotic Normal<br>95% Confidence Limits |         |
|                             |          |                   | Lower                                      | Upper   |
| Intercept                   | 9.4135   | 10.5402           | -11.2449                                   | 30.0719 |
| Temp                        | -0.0062  | 0.0598            | -0.1235                                    | 0.1110  |
| Voltage                     | 0.0086   | 0.0374            | -0.0648                                    | 0.0820  |
| Temp*Voltage                | -0.0001  | 0.0002            | -0.0005                                    | 0.0003  |
| EV Scale                    | 0.3624   | 0.0553            | 0.2687                                     | 0.4887  |
| Weibull Shape               | 2.7593   | 0.4210            | 2.0461                                     | 3.7209  |

**Output 2.2.2.** Probability Plot for Glass Capacitor Regression Model



**Output 2.2.3.** Plot of Data and Fitted Weibull Percentiles for Glass Capacitor Regression Model



---

## References

- Meeker, W.Q. and Escobar, L.A. (1998), *Statistical Methods for Reliability Data*, New York: John Wiley & Sons.
- Nelson, W. (1990), *Accelerated Testing: Statistical Models, Test Plans, and Data Analyses*, New York: John Wiley & Sons.

# Chapter 3

## The SHEWHART Procedure

### Chapter Table of Contents

---

|                                                         |    |
|---------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                               | 33 |
| <b>SYNTAX</b> . . . . .                                 | 33 |
| <b>DETAILS</b> . . . . .                                | 37 |
| OVERLAY= and Related Options . . . . .                  | 37 |
| LIMITS= Data Set Extensions . . . . .                   | 37 |
| <b>EXAMPLES</b> . . . . .                               | 38 |
| Example 3.1 Using BOXSTYLE=POINTSSCHEMATIC . . . . .    | 38 |
| Example 3.2 Using Overlays . . . . .                    | 40 |
| Example 3.3 Using LIMITS= Data Set Extensions . . . . . | 43 |
| <b>REFERENCES</b> . . . . .                             | 45 |





## Chapter 3

# The SHEWHART Procedure

---

### Overview

You can now specify variables from the input data set to be plotted as overlays on control charts. You can control the appearance of the symbols used to plot the overlay variables and the lines connecting the plotted points. You can also specify uniform resource locators (URLs) to be associated with overlay points. When graphics output is directed to HTML, these URLs can provide links to other Web pages.

Using new LIMITS= data set variables, you can vary the appearance of a control chart from one phase to another:

- The `_CFRAME_` variable specifies the color of the area enclosed by the axes and frame within a phase.
- The `_CINFILL_` variable specifies the color of the area inside the upper and lower control limits within a phase.
- The `_CLIMITS_` variable specifies the color for the control limits and central line within a phase.
- The `_WLIMITS_` variable specifies the width for the control limits and central line within a phase.
- The `_LLIMITS_` variable specifies the line type for the control limits within a phase.

The new `BOXSTYLE=POINTSSCHEMATIC` option in the `BOXCHART` statement combines the `POINTS` and `SCHEMATIC` box styles. The `MRRESTART` option in the `IRCHART` statement restarts the moving range computation when a missing value is encountered. The `TESTRESET=` and `TEST2RESET=` options reset tests for special causes on the primary and secondary control charts at specified subgroups.

---

### Syntax

#### **BOXSTYLE=POINTSSCHEMATIC**

requests a schematic box-and-whisker plot representing each subgroup sample, overlaid with points plotting all the values in the subgroup. This box style combines the `POINTS` and `SCHEMATIC` box styles. By default, a square plotting symbol is used to plot the values. You can specify a symbol with the `IDSYMBOL=` option. You can specify the color of the symbols with the `IDCOLOR=` option (the default color is the color specified with the `CBOXES=` option or the second color in the device color list). This option is available only in the `BOXCHART` statement.

**CCOVERLAY=(color-list)**

specifies colors for the line segments connecting points on primary chart overlays. Colors in the CCOVERLAY= list are matched with variables in the corresponding positions in the OVERLAY= list. By default, points are connected by line segments of the same color as the plotted points. You can specify the value NONE to suppress the line segments connecting points on an overlay.

**CCOVERLAY2=(color-list)**

specifies colors for the line segments connecting points on secondary chart overlays. Colors in the CCOVERLAY2= list are matched with variables in the corresponding positions in the OVERLAY2= list. By default, points are connected by line segments of the same color as the plotted points. You can specify the value NONE to suppress the line segments connecting points on an overlay.

**COVERLAY=(color-list)**

specifies the colors used to plot overlay variables on the primary control chart. Colors in the COVERLAY= list are matched with variables in the corresponding positions in the OVERLAY= list.

**COVERLAY2=(color-list)**

specifies the colors used to plot overlay variables on a secondary control chart. Colors in the COVERLAY2= list are matched with variables in the corresponding positions in the OVERLAY2= list.

**COVERLAYCLIP=(color)**

specifies the color used to plot clipped values on overlay plots. The same color is used for clipped points on primary and secondary chart overlays.

**LOVERLAY=(linetypes)**

specifies line types for the line segments connecting points on primary chart overlays. Line types in the LOVERLAY= list are matched with variables in the corresponding positions in the OVERLAY= list.

**LOVERLAY2=(linetypes)**

specifies line types for the line segments connecting points on secondary chart overlays. Line types in the LOVERLAY2= list are matched with variables in the corresponding positions in the OVERLAY2= list.

**MRRESTART****MRRESTART=value**

causes the moving range computation on an individual measurements and moving ranges chart to be restarted when a missing value is encountered. Without the MRRESTART option, a missing value is simply skipped, and the moving range for the next non-missing subgroup is computed using the most recent previous non-missing values. MRRESTART restarts the moving range computation, so only the observations after the missing value are used in subsequent moving range computations. You can specify MRRESTART to restart the moving range computation on any missing value, and MRRESTART=value to restart only on a particular missing value. For example, MRRESTART=R will restart the computation only when the missing value ".R" is encountered. This option is valid only in the IRCHART statement.

**NOOVERLAYLEGEND**

suppresses the legend for overlay variables that is displayed by default when the OVERLAY= or OVERLAY2= option is specified.

**OVERLAY=(*variable-list*)**

specifies variables to be plotted as overlays on the primary control chart. A point is plotted for each overlay variable at each subgroup for which it has a non-missing value. The value of a particular overlay variable should be the same for each observation in the input data set with a given value of the subgroup variable.

**OVERLAY2=(*variable-list*)**

specifies variables to be plotted as overlays on a secondary control chart. A point is plotted for each overlay variable at each subgroup for which it has a non-missing value. The value of a particular overlay variable should be the same for each observation in the input data set with a given value of the subgroup variable.

**OVERLAY2HTML=(*variable-list*)**

specifies variables whose values are URLs to be associated with points on secondary chart overlays. These URLs are associated with points on an overlay plot when graphics output is directed into HTML. Variables in the OVERLAY2HTML= list are matched with variables in the corresponding positions in the OVERLAY2= list. The value of the OVERLAY2HTML= variable should be the same for each observation with a given value of the subgroup variable.

**OVERLAY2SYM=(*symbol-list*)**

specifies symbols used to plot points on secondary chart overlays. Symbols in the OVERLAY2SYM= list are matched with variables in the corresponding positions in the OVERLAY2= list.

**OVERLAY2SYMHT=(*value-list*)**

specifies the heights of symbols used to plot points on secondary chart overlays. Heights in the OVERLAY2SYMHT= list are matched with variables in the corresponding positions in the OVERLAY2= list.

**OVERLAYCLIPSYM=(*symbol*)**

specifies the symbol used to plot clipped values on overlay plots. The same symbol is used on primary and secondary chart overlays.

**OVERLAYCLIPSYMHT=(*value*)**

specifies the height of the symbol used to plot clipped values on overlay plots. The same height is used on primary and secondary chart overlays.

**OVERLAYLEGLAB=*'label'***

specifies the label displayed to the left of the overlay legend. The label can be up to 16 characters long and must be enclosed in quotes.

**OVERLAYHTML=(*variable-list*)**

specifies variables whose values are URLs to be associated with points on primary chart overlays. These URLs are associated with points on an overlay plot when graphics output is directed into HTML. Variables in the OVERLAYHTML= list are matched with variables in the corresponding positions in the OVERLAY= list. The value of the OVERLAYHTML= variable should be the same for each observation

with a given value of the subgroup variable.

**OVERLAYSYM=(symbol-list)**

specifies symbols used to plot points on primary chart overlays. Symbols in the OVERLAYSYM= list are matched with variables in the corresponding positions in the OVERLAY= list.

**OVERLAYSYMHT=(value-list)**

specifies the heights of symbols used to plot points on primary chart overlays. Heights in the OVERLAYSYMHT= list are matched with variables in the corresponding positions in the OVERLAY= list.

**TESTRESET=variable**

enables tests for special causes to be reset on the primary control chart. The specified variable must be a character variable of length 8, or length 16 if customized tests are requested. The variable values have the same format as those of the \_TESTS\_ variable in a TABLE= data set. A test that is flagged by the TESTRESET= variable value for a given subgroup is reset starting with that subgroup. That means that a positive result for the test can include the given subgroup only if it is the first subgroup in the pattern. For example, the value “12345678” for the TESTRESET= variable will reset all standard tests for special causes. This option is available in all chart statements.

**TEST2RESET=variable**

enables tests for special causes to be reset on a secondary chart. The specified variable must be a character variable of length 8, or length 16 if customized tests are requested. The variable values have the same format as those of the \_TESTS\_ variable in a TABLE= data set. A test that is flagged by the TEST2RESET= value for a given subgroup is reset starting with that subgroup. That means a positive result for the test can include the given subgroup only if it is the first subgroup in the pattern. For example, the value “12345678” for the TEST2RESET= variable will reset all standard tests for special causes. This option is available with the MRCHART, RCHART, SCHAT, XRCHART, and XSCHAT statements.

**WOVERLAY=(value-list)**

specifies the widths in pixels of the line segments connecting points on primary chart overlays. Widths in the WOVERLAY= list are matched with variables in the corresponding positions in the OVERLAY= list.

**WOVERLAY2=(value-list)**

specifies the widths in pixels of the line segments connecting points on secondary chart overlays. Widths in the WOVERLAY2= list are matched with variables in the corresponding positions in the OVERLAY2= list.

---

## Details

---

### OVERLAY= and Related Options

Values specified with the COVERLAY=, OVERLAYSYM=, and OVERLAYSYMHT= options are matched with variables in the corresponding positions in the OVERLAY= variable list. Values specified with the COVERLAY2=, OVERLAY2SYM=, and OVERLAY2SYMHT= options are matched with variables in the corresponding positions in the OVERLAY2= variable list. When one of these options is not specified, the associated values are taken from the SYMBOL definitions currently in effect. If an option is specified but lists fewer values than the number of overlay variables, the overlays for the remaining unmatched variables use values from the SYMBOL definitions. SYMBOL1 is used to plot the process variable, so the SYMBOL definitions associated by default with the overlay plots start with SYMBOL2.

The vertical axis of a control chart is scaled to accommodate data for the process variable and all the points on any overlays. If the range of overlay variable values is much greater than the range of process variable values, the control chart may be compressed and difficult to read.

---

### LIMITS= Data Set Extensions

You can specify the value NONE for the \_CLIMITS\_ variable in a LIMITS= data set to suppress display of the control limits and central line within a phase. You can specify the values NONE or EMPTY for the \_CFRAME\_ or \_CINFILL\_ variables to leave the corresponding areas uncolored within a phase.

A value specified with a chart statement option takes precedence over values of the corresponding LIMITS= data set variable. For example, if you specify CINFILL=BLUE in a chart statement, the area between the control limits will be blue for all phases on the chart, overriding any values of the \_CINFILL\_ variable from the LIMITS= data set.

## Examples

### Example 3.1. Using BOXSTYLE=POINTSSCHEMATIC

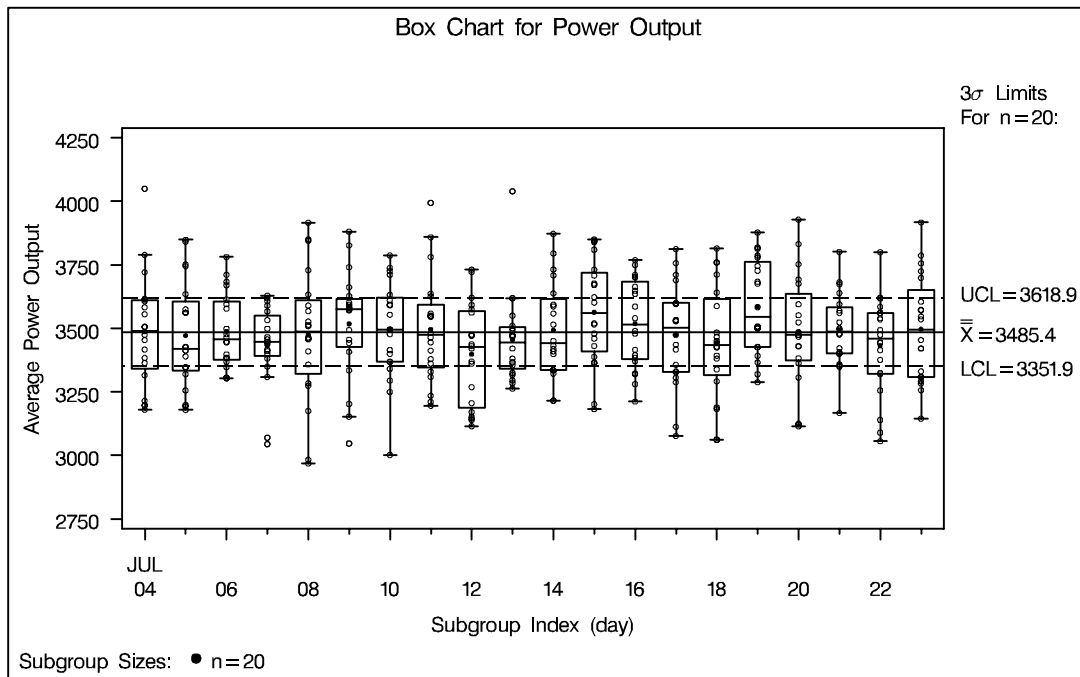
A petroleum company uses a turbine to heat water into steam that is pumped into the ground to make oil more viscous and easier to extract. This process occurs 20 times daily, and the amount of power (in kilowatts) used to heat the water to the desired temperature is recorded. The following statements create a SAS data set that contains the power output measurements for 20 days:

```
data turbine;
  informat day date7.;
  format day date5.;
  label kwatts='Average Power Output';
  input day @;
  do i=1 to 10;
    input kwatts @;
    output;
  end;
  drop i;
datalines;
04JUL94 3196 3507 4050 3215 3583 3617 3789 3180 3505 3454
04JUL94 3417 3199 3613 3384 3475 3316 3556 3607 3364 3721
05JUL94 3390 3562 3413 3193 3635 3179 3348 3199 3413 3562
05JUL94 3428 3320 3745 3426 3849 3256 3841 3575 3752 3347
06JUL94 3478 3465 3445 3383 3684 3304 3398 3578 3348 3369
06JUL94 3670 3614 3307 3595 3448 3304 3385 3499 3781 3711
.
.
.
23JUL94 3421 3787 3454 3699 3307 3917 3292 3310 3283 3536
23JUL94 3756 3145 3571 3331 3725 3605 3547 3421 3257 3574
;
```

You can use the BOXSTYLE=POINTSSCHEMATIC option to create a box chart with all observations in the turbine data set plotted. The following statements create the box chart shown in Output 3.1.1:

```
title 'Box Chart for Power Output';
symbol v=dot c=salmon h=0.4;
proc shewhart data=turbine;
  boxchart kwatts*day / cframe   = vligh
                        cboxes   = pagr
                        cboxfill = ywh
                        cinfill  = ligr
                        boxstyle = pointsschematic
                        idsymbol = circle
                        idcolor  = vibg;
run;
```

**Output 3.1.1.** Box Chart with BOXSTYLE=POINTSSCHEMATIC



## Example 3.2. Using Overlays

This example uses the SHEWHART procedure to do analysis of means (ANOM) for rate data. It is based on an example from Rodriguez (1996). That example is in ANOMP in the SAS/QC sample library.

A health care system uses ANOM to compare cesarean section rates for a set of medical groups. The following statements create a SAS data set named csection:

```
data csection;
  length id $ 2;
  input id csect94 vag94 csect95 vag95;
  total94 = csect94 + vag94;
  total95 = csect95 + vag95;
  label id = 'Medical Group Identification Number';
datalines;
1A      163      907      150      773
1K       55      314       45      253
1B       52      179       34      136
1D       19      128       18      114
3I       21       98       20       86
3M       15       81       12       93
1E       15       52       10       67
1N        6       43       19       55
1Q       12       67        7       62
3H        7       65       11       54
1R        4       43       11       38
1H        5       32        9       39
3J        7       12        7       13
1C       13       42        8       35
3B        3       33        6       37
1M        6        8        4       25
3C        6       27        5       23
10        8       26        4       23
1J        6       18        6       16
1T        1        3        3       19
3E        2       12        4       14
1G        1        7        4       11
3D        7       22        4        9
3G        2        5        1       10
1L        2        4        2        8
1I        1        3        1        7
1P       16       65        0        3
1F        0        1        0        3
1S        1        2        1        2
;

data csection;
  set csection end=eof;
  if eof then call symput('ngroups', left( put( _n_, 4. )));
  rate94 = csect94 / total94;
run;
```

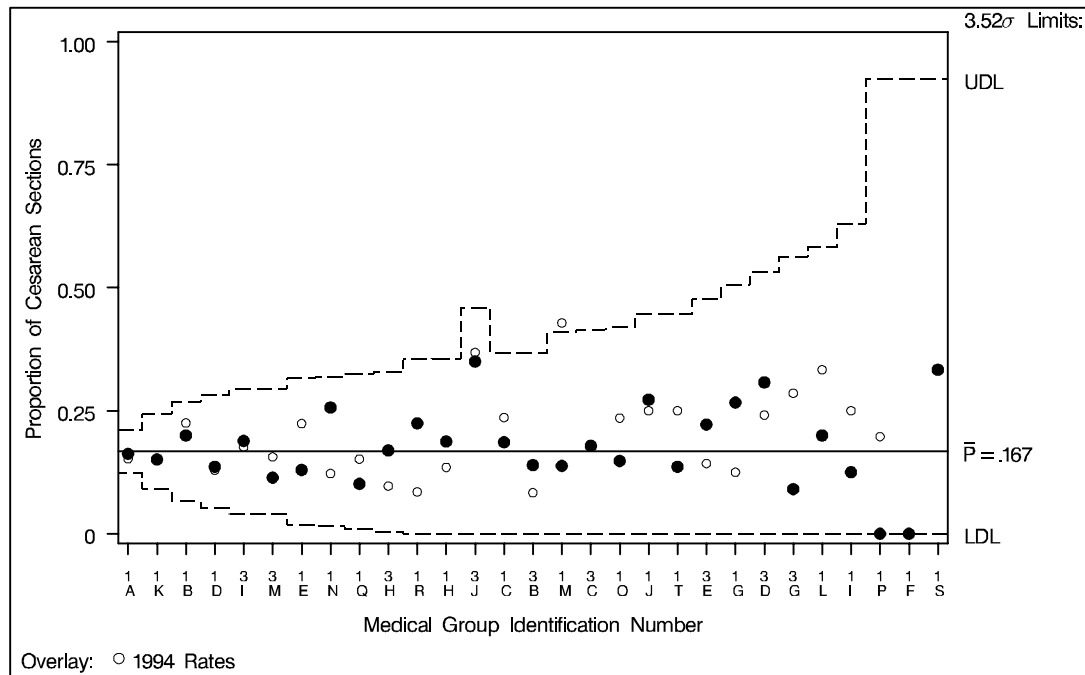


The variable `id` identifies the medical groups. The variable `csect95` provides the number of c-sections for each group during 1995, and the variable `total95` provides the total number of deliveries for each group. The variables `csect94` and `total94` provide similar counts for 1994.

The original example used an `annotate` data set to overlay rates from 1994 on the ANOM chart of 1995 rates. Now you can use `OVERLAY=` and related options to produce overlaid plots. The following statements create the ANOM chart shown in Output 3.2.1:

```
%anomsig( 0.01, &ngroups );
title 'Proportion of C-Sections:'
      ' 1994 and 1995';
symbol v=dot c=black h=2.4pct;
proc shewhart data=csection;
    pchart csect95*id / subgroupn=total95
                    noconnect
                    turnhlabels
                    nolegend
                    lclabel = 'LDL'
                    uclabel = 'UDL'
                    overlay = (rate94)
                    coverlay = (red)
                    ccoverlay = (none)
                    overlaysym = (circle)
                    overlaysymht = (2.4)
                    haxis = axis1
                    sigmas=&sigmult;
axis1 value = ( h=2.2 pct ) ;
label csect95 = 'Proportion of Cesarean Sections';
label rate94 = '1994 Rates';
run;
```

The `anomsig` macro computes the appropriate multiple of  $\sigma$  for the `SIGMAS=` option. It is defined in `ANOMSIG` in the SAS/QC sample library.

**Output 3.2.1.** ANOM Chart with Overlay

### Example 3.3. Using LIMITS= Data Set Extensions

The following statements create the HISTORY= data set flange, containing means and ranges of flange width measurements for subgroups of size five:

```
data flange;
  input day date7. sample 10-11 _phase_ $14-23
        flwidthx flwidthr flwidthn;
  format   day date7.;
datalines;
08FEB90    6  Production    0.97360    0.06247    5
09FEB90    7  Production    1.00486    0.11478    5
10FEB90    8  Production    1.00251    0.13537    5
11FEB90    9  Production    0.95509    0.08378    5
12FEB90   10  Production    1.00348    0.09993    5
15FEB90   11  Production    1.02566    0.06766    5
16FEB90   12  Production    0.97053    0.07608    5
17FEB90   13  Production    0.94713    0.10170    5
18FEB90   14  Production    1.00377    0.04875    5
19FEB90   15  Production    0.99604    0.08242    5
22FEB90   16  Change 1      0.99218    0.09787    5
23FEB90   17  Change 1      0.99526    0.02017    5
24FEB90   18  Change 1      1.02235    0.10541    5
25FEB90   19  Change 1      0.99950    0.11476    5
26FEB90   20  Change 1      0.99271    0.05395    5
01MAR90   21  Change 1      0.98695    0.03833    5
02MAR90   22  Change 1      1.00969    0.06183    5
03MAR90   23  Change 1      0.98791    0.05836    5
04MAR90   24  Change 1      1.00170    0.05243    5
05MAR90   25  Change 1      1.00412    0.04815    5
08MAR90   26  Change 2      1.00261    0.05604    5
09MAR90   27  Change 2      0.99553    0.02818    5
10MAR90   28  Change 2      1.01463    0.05558    5
11MAR90   29  Change 2      0.99812    0.03648    5
12MAR90   30  Change 2      1.00047    0.04309    5
15MAR90   31  Change 2      0.99714    0.03689    5
16MAR90   32  Change 2      0.98642    0.04809    5
17MAR90   33  Change 2      0.98891    0.07777    5
18MAR90   34  Change 2      1.00087    0.06409    5
19MAR90   35  Change 2      1.00863    0.02649    5
;
run;
```

This DATA step creates the LIMITS= data set flanim:

```
data flanim;
  input _index_ $1-10 _lclx_ _mean_ _uclx_ _r_ _uclr_ _stddev_ @;
  input _cframe_ $ _cinfll_ $ _climits_ $ _wlimits_ _llimits_;
  _var_   = 'FLWIDTH';
  _subgrp_ = 'SAMPLE';
  _type_  = 'ESTIMATE';
  _limitn_ = 5;
```

```

    _sigmas_ = 3;
    _alpha_ = .0026998;
    _lclr_ = 0;
    datalines;
    Production 0.9379 0.9883 1.0386 0.0873 0.1846 0.03753
               steel ligr red 2 4
    Change 1   0.9617 0.9992 1.0368 0.0651 0.1377 0.02800
               vibg vlib yellow 1 2
    Change 2   0.8709 0.9680 1.0652 0.1684 0.3561 0.07241
               vligb ywh vibg 3 3
    ;
    run;

```

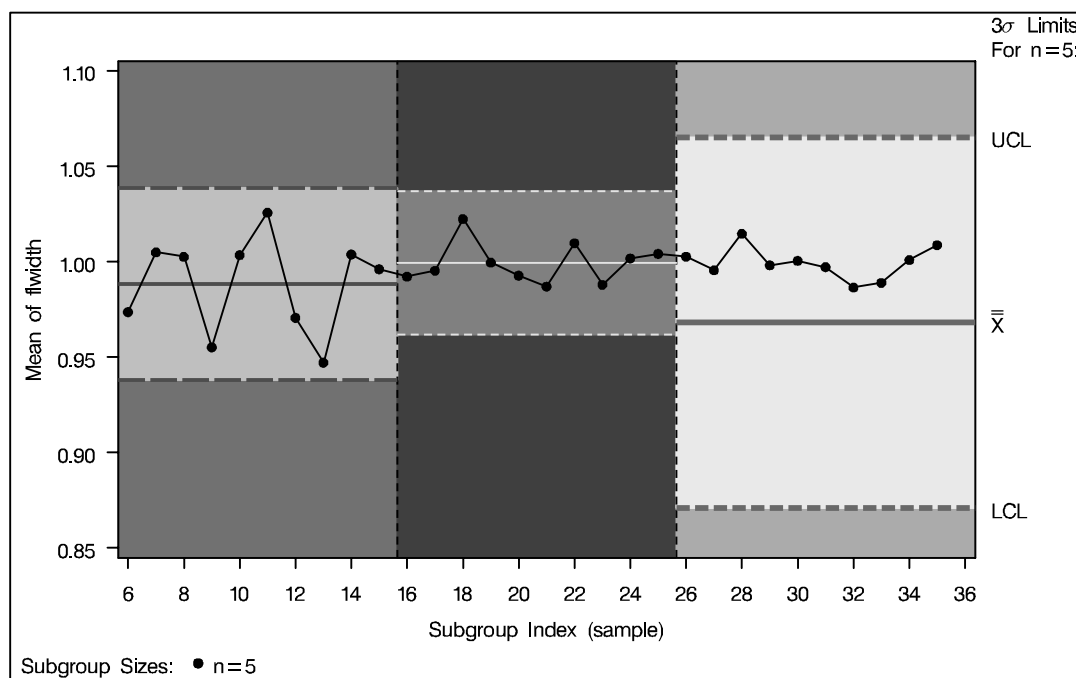
The variables `_CFRAME_`, `_CINFILL_`, `_CLIMITS_`, `_WLIMITS_`, and `_LLIMITS_` specify colors, line widths, and line types for displaying three different phases. For example, in the 'Production' phase the specified frame color is steel, the area between the control limits is light gray, the control limits are red, the control limits width is two pixels, and the control limits line style is 4. The following statements produce the chart shown in Output 3.3.1:

```

symbol1 v=dot c=black h=0.4;
proc shewhart history=flange limits=flanlim;
    xchart flwidth*sample /
        readphase = all
        readindex = all
        phaselegend;
run;

```

**Output 3.3.1.**  $\bar{X}$  Chart with Variations in Display of Phases



---

## References

- Rodriguez, R. N. (1996), “Health Care Applications of Statistical Process Control: Examples Using the SAS System,” *SAS Users Group International: Proceedings of the Twenty-First Annual Conference*, 1381–1396.



# Chapter 4

## The ADX Interface for Design and Analysis of Experiments

### Chapter Table of Contents

---

|                                       |           |
|---------------------------------------|-----------|
| <b>OVERVIEW . . . . .</b>             | <b>49</b> |
| <b>PROCESS VARIABLES . . . . .</b>    | <b>49</b> |
| <b>IMPORT DESIGN WIZARD . . . . .</b> | <b>49</b> |
| <b>DESIGN DETAILS . . . . .</b>       | <b>49</b> |





## Chapter 4

# The ADX Interface for Design and Analysis of Experiments

---

### Overview

The ADX Interface is a point-and-click interface for the design and analysis of experiments. Designed for the desktop environment of PCs and workstations, the Interface is intended primarily for engineers and researchers who require a guided interface for every step in the experimental process.

The ADX Interface has been enhanced in several ways. Mixture designs have been extended to include process variables, an import design wizard has been added, and the Design Details window now enables you to view more details of the design.

---

### Process Variables

The ADX interface now enables you to specify process variables for mixture designs. A process variable is a factor in an experiment that is not part of the mixture but whose levels, when varied, could affect the blending properties of the mixture ingredients.

---

### Import Design Wizard

An import design wizard has been added to enable external design data to be imported into ADX. The external data can be stored in a SAS data set or other file type, including Microsoft® Excel. Two-level designs, response surface designs, mixed-level designs, mixture designs, and split-plot designs can be imported.

---

### Design Details

The Design Details window now enables you to view two additional details of the basic design. Variance dispersion plots, for Resolution 5 two-level and response surface designs, provide a variance graph to evaluate a design. For mixture designs, the mixture design points can be viewed graphically.



# Subject Index

## C

capability indices

specialized, 3, 6

the index  $C_p^W$ , 7

the index  $C_{pc}$ , 8

the index  $C_{pg}$ , 7

the index  $C_{pk}^W$ , 7

the index  $C_{pm}^W$ , 7

the index  $C_{pp}$ , 6

the index  $C_{pp}''$ , 6

the index  $C_{pq}$ , 7

the index  $C_s$ , 9

the index  $S_{j_{kp}}$ , 6

Vännmann's index  $C_p(u, v)$ , 8

Vännmann's index  $C_p(v)$ , 9

## H

histograms

clipping bars, 10

labeling bars, 4

labeling endpoints, 9, 13

## R

reliability analysis

accelerated life test, 27

cumulative failure probability, 19

logarithm of distribution scale parameter, 19

probability plots, 19

probability plots, confidence intervals, 19

probability plots, extreme values, 26

regression model with non-constant scale, 24

relation plots, 19

## S

Shewhart charts

associating URLs with subgroups, 35

overlays on, 37, 40

varying display by phase, 43



# Syntax Index

## B

BARLABEL= option  
    CAPABILITY procedure, 4  
BOXSTYLE= option  
    SHEWHART procedure, 33

## C

CAPABILITY procedure, COMPHISTOGRAM statement  
    BARLABEL= option, 4  
    CLIPSPEC= option, 4  
    CTEXTSIDE= option, 4  
    CTEXTTOP= option, 4  
    ENDPOINTS= option, 4  
CAPABILITY procedure, HISTOGRAM statement  
    ENDPOINTS= option, 5  
    NENDPOINTS= option, 5  
    NMIDPOINTS= option, 5  
CAPABILITY procedure, PROC CAPABILITY statement  
    CPMA= option, 3  
    SPECIALINDICES option, 3  
CCOVERLAY= option  
    SHEWHART procedure, 34  
CCOVERLAY2= option  
    SHEWHART procedure, 34  
CLIPSPEC= option  
    CAPABILITY procedure, 4  
COVERLAY= option  
    SHEWHART procedure, 34  
COVERLAY2= option  
    SHEWHART procedure, 34  
COVERLAYCLIP= option  
    SHEWHART procedure, 34  
CPMA= option  
    CAPABILITY procedure, 3  
CTEXTSIDE= option  
    CAPABILITY procedure, 4  
CTEXTTOP= option  
    CAPABILITY procedure, 4

## E

ENDPOINTS= option  
    CAPABILITY procedure, 4–5

## L

LOVERLAY= option  
    SHEWHART procedure, 34  
LOVERLAY2= option  
    SHEWHART procedure, 34

## M

MRRESTART option  
    SHEWHART procedure, 34

## N

NENDPOINTS= option  
    CAPABILITY procedure, 5  
NMIDPOINTS= option  
    CAPABILITY procedure, 5  
NOOVERLAYLEGEND option  
    SHEWHART procedure, 35

## O

OVERLAY= option  
    SHEWHART procedure, 35  
OVERLAY2= option  
    SHEWHART procedure, 35  
OVERLAY2HTML= option  
    SHEWHART procedure, 35  
OVERLAY2SYM= option  
    SHEWHART procedure, 35  
OVERLAY2SYMHT= option  
    SHEWHART procedure, 35  
OVERLAYCLIPSYM= option  
    SHEWHART procedure, 35  
OVERLAYCLIPSYMHT= option  
    SHEWHART procedure, 35  
OVERLAYHTML= option  
    SHEWHART procedure, 35  
OVERLAYLEGLAB= option  
    SHEWHART procedure, 35  
OVERLAYSYM= option  
    SHEWHART procedure, 36  
OVERLAYSYMHT= option  
    SHEWHART procedure, 36

## R

RELIABILITY procedure, LOGSCALE statement, 19  
RELIABILITY procedure, MCFPLOT statement, 20  
RELIABILITY procedure, MODEL statement, 20  
RELIABILITY procedure, PROBPLOT statement, 21  
RELIABILITY procedure, RELATIONPLOT statement, 21

## S

SHEWHART procedure  
    BOXSTYLE= option, 33  
    CCOVERLAY= option, 34  
    CCOVERLAY2= option, 34

COVERLAY= option, 34  
COVERLAY2= option, 34  
COVERLAYCLIP= option, 34  
LOVERLAY= option, 34  
LOVERLAY2= option, 34  
MRRESTART option, 34  
NOOVERLAYLEGEND option, 35  
OVERLAY= option, 35  
OVERLAY2= option, 35  
OVERLAY2HTML= option, 35  
OVERLAY2SYM= option, 35  
OVERLAY2SYMHT= option, 35  
OVERLAYCLIPSYM= option, 35  
OVERLAYCLIPSYMHT= option, 35  
OVERLAYHTML= option, 35  
OVERLAYLEGLAB= option, 35  
OVERLAYSYM= option, 36  
OVERLAYSYMHT= option, 36  
TEST2RESET= option, 36  
TESTRESET= option, 36  
WOVERLAY= option, 36  
WOVERLAY2= option, 36  
SPECIALINDICES option  
    CAPABILITY procedure, 3

**T**

TEST2RESET= option  
    SHEWHART procedure, 36  
TESTRESET= option  
    SHEWHART procedure, 36

**W**

WOVERLAY= option  
    SHEWHART procedure, 36  
WOVERLAY2= option  
    SHEWHART procedure, 36

# Your Turn

---

If you have comments or suggestions about *SAS/QC<sup>®</sup> Software: Changes and Enhancements, Release 8.2*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute Inc.  
SAS Publishing  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute Inc.  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [suggest@sas.com](mailto:suggest@sas.com)





*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS® Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

1. Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
2. Leave the "Subject" line blank.
3. Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.



SAS Publishing

TECHNICAL REPORT

# SAS/OR<sup>®</sup> Software: Changes and Enhancements, Release 8.2



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/OR® Software: Changes and Enhancements, Release 8.2*, Cary, NC: SAS Institute Inc., 2001

**SAS/OR® Software: Changes and Enhancements, Release 8.2**

Copyright © 2001 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-851-4

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2001

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

|                                             |     |
|---------------------------------------------|-----|
| Chapter 1. The BOM Procedure . . . . .      | 1   |
| Chapter 2. The CPM Procedure . . . . .      | 37  |
| Chapter 3. The GANTT Procedure . . . . .    | 49  |
| Chapter 4. The INTPOINT Procedure . . . . . | 53  |
| Chapter 5. The LP Procedure . . . . .       | 183 |
| Chapter 6. The PM Procedure . . . . .       | 187 |
| Subject Index . . . . .                     | 191 |
| Syntax Index . . . . .                      | 197 |



# Chapter 1

## The BOM Procedure

### Chapter Table of Contents

---

|                                                                   |    |
|-------------------------------------------------------------------|----|
| <b>OVERVIEW</b> . . . . .                                         | 3  |
| <b>GETTING STARTED</b> . . . . .                                  | 3  |
| <b>SYNTAX</b> . . . . .                                           | 11 |
| Functional Summary . . . . .                                      | 11 |
| PROC BOM Statement . . . . .                                      | 12 |
| STRUCTURE Statement . . . . .                                     | 13 |
| <b>DETAILS</b> . . . . .                                          | 15 |
| Single-Level Bills of Material Data Set . . . . .                 | 15 |
| Indented Bills of Material Data Set . . . . .                     | 17 |
| Summarized Bills of Material Data Set . . . . .                   | 18 |
| Missing Values in the Input Data Set . . . . .                    | 19 |
| Macro Variable _ORBOM_ . . . . .                                  | 20 |
| Computer Resource Requirements . . . . .                          | 21 |
| <b>EXAMPLES</b> . . . . .                                         | 21 |
| Example 1.1 Bill of Material with Lead Time Information . . . . . | 21 |
| Example 1.2 Planning Bill of Material . . . . .                   | 27 |
| Example 1.3 Bills of Material Verification . . . . .              | 33 |
| <b>REFERENCES</b> . . . . .                                       | 35 |





# Chapter 1

## The BOM Procedure

---

### Overview

The new BOM procedure performs bill-of-material processing. It composes a series of single-level bills of material into a multilevel, tree-structured bill of material; determines the level of each part in the bill; and represents the multilevel bill of material structure in the form of an indented bill of material. PROC BOM can also output a summarized bill of material.

A *bill of material* (BOM) is a list of all parts, ingredients, or materials needed to make one production run of a product. The bill of material may also be called the *formula*, *recipe*, or *ingredients list* in certain process industries (Cox and Blackstone 1998). The way in which the bill of material data are organized and presented is called the *structure* of the bill of material or the structure of the product.

A variety of display formats are available for bills of material. The simplest format is the *single-level BOM*. It consists of a list of all components that are directly used in a parent item. A *multilevel bill of material* provides a display of all components that are directly or indirectly used in a parent item. When an item is a subcomponent, blend, intermediate, etc., then all of its components, including purchased parts and raw materials, are also exhibited. A multilevel structure can be illustrated by a tree with several levels. An *indented bill of material* is a form of multilevel BOM. It exhibits the final product as level 0 and all its components as level 1. The level numbers increase as you look down the tree structure. If an item is used in more than one parent within a given product structure, it appears more than once, under every subassembly in which it is used. The *summarized bill of material* is another form of a multilevel bill of material. It lists all the parts and their quantities required in a given product structure. Unlike the indented bill of material, it does not list the levels of parts and does not illustrate the part-component relationships. Moreover, the summarized bill of material lists each item only once for the total quantity required. Refer to Cox and Blackstone (1998) for further details.

The indented bill of material data generated by PROC BOM are organized in such a manner that they can be easily retrieved and manipulated to generate reports, and can also be used by other SAS/OR procedures to perform additional analysis. The summarized bill of material data are quite useful in gross requirements planning and other applications.

---

### Getting Started

The example of the ABC Lamp Company product structure example from Fogarty, Blackstone, and Hoffmann (1991) illustrates the basic features of PROC BOM. The

single-level bill of material data are depicted in Figure 1.1. These consists of a list of each part in the product structure, the components that are directly used in the part, and the quantity of each component needed to make one unit of that particular part. The data set also includes a short description and the unit of measure for each part. The SAS DATA step code to create and display the single-level BOM data set is as follows:

```

/* single-level BOM data */
data SLBOM0;
  input Part      $8.
        Desc      $24.
        Unit      $8.
        Component $8.
        QtyPer    8.0
        ;
  datalines;
LA01    Lamp LA          Each    B100          1
                                           S100          1
                                           A100          1
B100    Base assembly    Each    1100          1
                                           1200          1
                                           1300          1
                                           1400          4
S100    Black shade      Each
A100    Socket assembly  Each    1500          1
                                           1600          1
                                           1700          1
1100    Finished shaft   Each    2100         26
1200    7-Diameter steel plate Each
1300    Hub              Each
1400    1/4-20 Screw     Each
1500    Steel holder     Each    1400          2
1600    One-way socket   Each
1700    Wiring assembly  Each    2200         12
                                           2300          1
2100    3/8 Steel tubing  Inches
2200    16-Gauge lamp cord Feet
2300    Standard plug terminal Each
;

/* Display the input data set */
proc print data=SLBOM0 noobs;
  title 'ABC Lamp Company';
  title3 'Single-Level Bill of Material';
run;

```

| ABC Lamp Company              |                        |        |           |         |
|-------------------------------|------------------------|--------|-----------|---------|
| Single-Level Bill of Material |                        |        |           |         |
| Part                          | Desc                   | Unit   | Component | Qty Per |
| LA01                          | Lamp LA                | Each   | B100      | 1       |
|                               |                        |        | S100      | 1       |
|                               |                        |        | A100      | 1       |
|                               |                        |        |           |         |
| B100                          | Base assembly          | Each   | 1100      | 1       |
|                               |                        |        | 1200      | 1       |
|                               |                        |        | 1300      | 1       |
|                               |                        |        | 1400      | 4       |
|                               |                        |        |           |         |
| S100                          | Black shade            | Each   |           | .       |
| A100                          | Socket assembly        | Each   | 1500      | 1       |
|                               |                        |        | 1600      | 1       |
|                               |                        |        | 1700      | 1       |
| 1100                          | Finished shaft         | Each   | 2100      | 26      |
| 1200                          | 7-Diameter steel plate | Each   |           | .       |
| 1300                          | Hub                    | Each   |           | .       |
| 1400                          | 1/4-20 Screw           | Each   |           | .       |
| 1500                          | Steel holder           | Each   | 1400      | 2       |
| 1600                          | One-way socket         | Each   |           | .       |
| 1700                          | Wiring assembly        | Each   | 2200      | 12      |
|                               |                        |        | 2300      | 1       |
| 2100                          | 3/8 Steel tubing       | Inches |           | .       |
| 2200                          | 16-Gauge lamp cord     | Feet   |           | .       |
| 2300                          | Standard plug terminal | Each   |           | .       |

**Figure 1.1.** Single-Level Bill of Material

The following code invokes PROC BOM to produce the indented bill of material and the summarized bill of material. It also uses PROC NETDRAW to draw a *family tree* diagram for illustrating the multilevel product structure. For further details about PROC NETDRAW, refer to “The NETDRAW Procedure” chapter in the *SAS/OR User’s Guide: Project Management*.

```

/* Create the indented BOM and the summarized BOM */
proc bom data=SlBOM0 out=IndBOM0 summaryout=SumBOM0;
  structure / part=Part
              component=Component
              quantity=QtyPer
              id=(Desc Unit);
run;

/* Draw a tree diagram for illustrating the product structure */
/* Each record denotes a node in the tree */
data IndBOM0a(drop=Part_ID);
  set IndBOM0;
  Paren_ID=Part_ID;
run;

/* Extract the Parent - Part information */
data IndBOM0b;
  set IndBOM0(keep=Paren_ID Part_ID);
run;

```

```

/* Prepare the data set for running NETDRAW */
data TreBOM0;
  set IndBOM0a IndBOM0b;
run;

/* Specify graphics options */
goptions hpos=32 vpos=80 border;
pattern1 v=s c=white;

title h=5 j=c 'Multilevel Bill of Material';
footnote h=2 j=l
  'Node shows ID Number, Part Number, and Quantity Required';

/* Invoke PROC NETDRAW to display BOM tree */
proc netdraw data=TreBOM0( where=(Paren_ID NE .) );
  actnet / act=Paren_ID succ=Part_ID id=(Paren_ID _Part_ QtyPer)
  ctext=black font=swiss htext=3 cars=black
  ybetween=3 xbetween=8 centerid
  tree pcompress rotatetext rotate
  arrowhead=0 rectilinear nodefid nodelabel;
run;

```

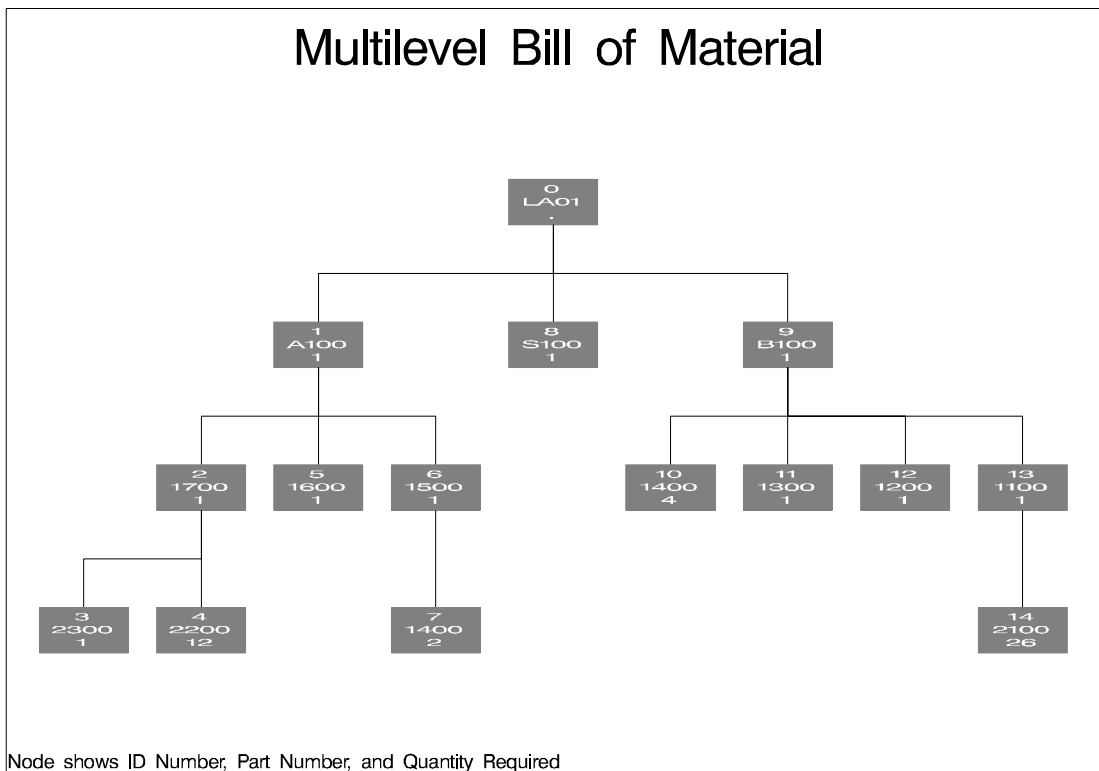


**Figure 1.2.** Multilevel Product Structure

Figure 1.2 displays the tree structure of the ABC Lamp Company. Corresponding to this tree diagram is the list of indented bill of material for the company as in Figure 1.3. The following code displays the indented BOM:

```

/* Display the indented BOM data */
proc print data=IndBOM0 noobs;
  var _Level_ _Part_ Part_ID Desc QtyPer Qty_Prod
      Unit _Parent_ Paren_ID _Prod_;
  title 'ABC Lamp Company';
  title3 'Indented Bill of Material, Part LA01';
run;

```

| ABC Lamp Company                     |      |         |                        |         |        |        |          |      |
|--------------------------------------|------|---------|------------------------|---------|--------|--------|----------|------|
| Indented Bill of Material, Part LA01 |      |         |                        |         |        |        |          |      |
| Level                                | Part | Part ID | Desc                   | Qty Per | Unit   | Parent | Paren ID | Prod |
| 0                                    | LA01 | 0       | Lamp LA                | 1       | Each   | .      | .        | LA01 |
| 1                                    | A100 | 1       | Socket assembly        | 1       | Each   | LA01   | 0        | LA01 |
| 2                                    | 1700 | 2       | Wiring assembly        | 1       | Each   | A100   | 1        | LA01 |
| 3                                    | 2300 | 3       | Standard plug terminal | 1       | Each   | 1700   | 2        | LA01 |
| 3                                    | 2200 | 4       | 16-Gauge lamp cord     | 12      | Feet   | 1700   | 2        | LA01 |
| 2                                    | 1600 | 5       | One-way socket         | 1       | Each   | A100   | 1        | LA01 |
| 2                                    | 1500 | 6       | Steel holder           | 1       | Each   | A100   | 1        | LA01 |
| 3                                    | 1400 | 7       | 1/4-20 Screw           | 2       | Each   | 1500   | 6        | LA01 |
| 1                                    | S100 | 8       | Black shade            | 1       | Each   | LA01   | 0        | LA01 |
| 1                                    | B100 | 9       | Base assembly          | 1       | Each   | LA01   | 0        | LA01 |
| 2                                    | 1400 | 10      | 1/4-20 Screw           | 4       | Each   | B100   | 9        | LA01 |
| 2                                    | 1300 | 11      | Hub                    | 1       | Each   | B100   | 9        | LA01 |
| 2                                    | 1200 | 12      | 7-Diameter steel plate | 1       | Each   | B100   | 9        | LA01 |
| 2                                    | 1100 | 13      | Finished shaft         | 1       | Each   | B100   | 9        | LA01 |
| 3                                    | 2100 | 14      | 3/8 Steel tubing       | 26      | Inches | 1100   | 13       | LA01 |

**Figure 1.3.** Indented Bill of Material

Each *record* in the indented BOM data set is associated with a *node* in the tree structure, and the node or record is uniquely identified by a *sequence number* that is assigned to it by the procedure. The `Part_ID` variable denotes this sequence number of the node or record. The `Part_ID` information is needed in case some items are required at more than one place in the BOM. For example, the item '1400' (1/4-20 Screw) is required at 'B100' (Base assembly) and '1500' (Steel holder). In this situation the part number is not sufficient to identify each node in the tree structure. Thus, the two different nodes are uniquely identified by the values '7' and '10' for the variable `Part_ID`.

The `_Level_` variable denotes the level number of each node. The final product, 'LA01', is at level 0, and the level numbers increase as you look down the tree. The `Part`, `QtyPer`, and `ID` variables (`Desc` and `Unit`) in the single-level BOM data set, `SIBOM0`, are also included in the indented BOM data set. Note that the name of the `Part` variable is changed to `_Part_` in the indented BOM data set.

There are two variables in the indented BOM data set identifying parent information: the `_Parent_` variable denotes the *part number* for the parent item (next assembly)

of the part identified by the `_Part_` variable, while the `Paren_ID` variable specifies the *sequence number* for the parent node of the current node (record) identified by the `Part_ID` variable. For example, the indented BOM data set, as shown in Figure 1.3, reveals that the part '1400' has two parent items: part '1500' and part 'B100'. Meanwhile, the parent node of node '7' is node '6', and the parent node of node '10' is node '9'.

Finally, the `_Prod_` variable denotes the part number of the final product in the production structure. The `Qty_Prod` variable contains the required quantity of the item identified by the `_Part_` variable in order to make one unit of the final product identified by the `_Prod_` variable. Note that in this particular example, the values of the `Qty_Prod` variable are identical to the values of the `QtyPer` variable. This is because the values of the `QtyPer` variable for all parent items are 1.

Figure 1.2 and Figure 1.3 show that the Indented BOM data set lists all of the parts of the tree structure in *depth-first* order (Aho, Hopcroft, and Ullman 1983). Unlike the Single-level BOM data set, the Indented BOM data set provides the “part-parent” information rather than the “part-component” relationship. The part-component information can still be easily retrieved. For example, the following code creates a list of components that are directly used in the part 'LA01':

```
/* Display the components that are directly used */
/* in item LA01 */
proc print data=IndBOM0(where=(_Parent_='LA01')
                        rename=(_Part_=Component))
    noobs;
    var Component Desc QtyPer Unit _Prod_;
    title 'ABC Lamp Company';
    title3 'Single-Level Bill of Material Retrieval, Part LA01';
run;
```

| ABC Lamp Company                                   |                 |            |      |        |
|----------------------------------------------------|-----------------|------------|------|--------|
| Single-Level Bill of Material Retrieval, Part LA01 |                 |            |      |        |
| Component                                          | Desc            | Qty<br>Per | Unit | _Prod_ |
| A100                                               | Socket assembly | 1          | Each | LA01   |
| S100                                               | Black shade     | 1          | Each | LA01   |
| B100                                               | Base assembly   | 1          | Each | LA01   |

**Figure 1.4.** Components That Are Directly Used in Part LA01

You can also create *single-level where-used* reports from the Indented BOM data set. For example, the single-level where-used data set `Used0b`, displayed in Figure 1.5, lists all parent items that directly use the part '1400'. The SAS code that creates the data set `Used0b` from the Indented BOM data set `IndBOM0` is as follows:

```

/* Create the where-used data set */
data Used0a(keep=_Parent_ Paren_ID QtyPer Unit _Prod_);
  set IndBOM0(where=(Part_='1400'));
run;

/* Get the part description from the IndBOM0 data set */
proc sql;
  create table Used0b as
    select Used0a._Parent_, IndBOM0.Desc,
           Used0a.QtyPer, Used0a.Unit, Used0a._Prod_
    from Used0a left join IndBOM0
      on Used0a.Paren_ID=IndBOM0.Part_ID;
quit;

/* Display the where-used data set */
proc print data=Used0b noobs;
  var _Parent_ Desc QtyPer Unit _Prod_;
  title 'ABC Lamp Company';
  title3 'Single-Level Where-used Report, Part 1400';
run;

```

| ABC Lamp Company                          |               |            |      |        |
|-------------------------------------------|---------------|------------|------|--------|
| Single-Level Where-used Report, Part 1400 |               |            |      |        |
| _Parent_                                  | Desc          | Qty<br>Per | Unit | _Prod_ |
| 1500                                      | Steel holder  | 2          | Each | LA01   |
| B100                                      | Base assembly | 4          | Each | LA01   |

**Figure 1.5.** Parents in Which the Part 1400 Is Directly Used

The following SAS code sorts and displays the Summarized BOM data set that is produced by the BOM procedure:

```

/* Sort and display the summarized BOM data */
proc sort data=SumBOM0;
  by _Part_;
run;

proc print data=SumBOM0 noobs;
  title 'ABC Lamp Company';
  title3 'Summarized Bill of Material, Part LA01';
run;

```

| ABC Lamp Company                       |          |          |         |         |                        |        |
|----------------------------------------|----------|----------|---------|---------|------------------------|--------|
| Summarized Bill of Material, Part LA01 |          |          |         |         |                        |        |
| _Part_                                 | Low_Code | Gros_Req | On_Hand | Net_Req | Desc                   | Unit   |
| 1100                                   | 2        | 1        | 0       | 1       | Finished shaft         | Each   |
| 1200                                   | 2        | 1        | 0       | 1       | 7-Diameter steel plate | Each   |
| 1300                                   | 2        | 1        | 0       | 1       | Hub                    | Each   |
| 1400                                   | 3        | 6        | 0       | 6       | 1/4-20 Screw           | Each   |
| 1500                                   | 2        | 1        | 0       | 1       | Steel holder           | Each   |
| 1600                                   | 2        | 1        | 0       | 1       | One-way socket         | Each   |
| 1700                                   | 2        | 1        | 0       | 1       | Wiring assembly        | Each   |
| 2100                                   | 3        | 26       | 0       | 26      | 3/8 Steel tubing       | Inches |
| 2200                                   | 3        | 12       | 0       | 12      | 16-Gauge lamp cord     | Feet   |
| 2300                                   | 3        | 1        | 0       | 1       | Standard plug terminal | Each   |
| A100                                   | 1        | 1        | 0       | 1       | Socket assembly        | Each   |
| B100                                   | 1        | 1        | 0       | 1       | Base assembly          | Each   |
| LA01                                   | 0        | 1        | 0       | 1       | Lamp LA                | Each   |
| S100                                   | 1        | 1        | 0       | 1       | Black shade            | Each   |

**Figure 1.6.** Summarized Bill of Material

The Summarized BOM data set, as displayed in Figure 1.6, lists all the parts and their quantity required in the given ABC Lamp Company product structure. Since the Single-level BOM data set, `SIBOM0`, does not contain any gross requirement information, PROC BOM assumes that the final product, 'LA01', is the only master schedule item. The *master schedule items* (MSI) are items selected to be planned by the master scheduler. These items are deemed critical in their impact on lower-level components or resources. Therefore, the master scheduler, not the computer, maintains the plan for these items. The BOM procedure also assumes that the gross requirement of 'LA01' is 1 unit and that there are no items currently on hand. The net requirement of the part 'LA01' is the gross requirement (1) minus the quantity on hand (0). The gross requirements of the other items are then calculated sequentially, level by level, as follows:

**'B100' Base assembly** (1 per lamp)

|                   |                  |
|-------------------|------------------|
| Gross requirement | $1 \times 1 = 1$ |
| Quantity on hand  | -0               |
| Net requirement   | <hr/> 1          |

**'1100' Finished shaft** (1 per base assembly)

|                   |                  |
|-------------------|------------------|
| Gross requirement | $1 \times 1 = 1$ |
| Quantity on hand  | -0               |
| Net requirement   | <hr/> 1          |

**'2100' 3/8 Steel tubing** (26 inches per shaft)

|                   |                    |
|-------------------|--------------------|
| Gross requirement | $1 \times 26 = 26$ |
| Quantity on hand  | -0                 |
| Net requirement   | <hr/> 26           |

If a part (like item '1400') is required at more than one assembly, the gross requirement is the total needed in all assemblies where it is required.

The `Net_Req` variable denotes the net requirement, and the `Low_Code` variable denotes the *low-level code* of the part. The low-level code is a number that indicates



the lowest level in any bill of material at which the item appears. The low-level codes are necessary to make sure that the net requirement for a given item is not calculated until all the gross requirements have been calculated down to that level.

---

## Syntax

The following statements are available in PROC BOM.

```
PROC BOM options ;
      STRUCTURE / options ;
```

---

## Functional Summary

The following table outlines the options available for the BOM procedure.

**Table 1.1.** PROC BOM Options

| Task                                                                                                                                                 | Statement                                        | Option                                       |
|------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|----------------------------------------------|
| <b>Data Set Specification</b><br>Single-level BOM data set<br>Indented BOM data set<br>Summarized BOM data set                                       | BOM<br>BOM<br>BOM                                | DATA=<br>OUT=<br>SUMMARYOUT=                 |
| <b>Output Control Options</b><br>ID variables                                                                                                        | STRUCTURE                                        | ID=                                          |
| <b>Part Information Specification</b><br>lead time variable<br>quantity on hand variable<br>gross requirement variable                               | STRUCTURE<br>STRUCTURE<br>STRUCTURE              | LEADTIME=<br>QTYONHAND=<br>REQUIREMENT=      |
| <b>Problem Size Options</b><br>number of distinct part names<br>do not use utility data set<br>number of items                                       | BOM<br>BOM<br>BOM                                | N NAMES=<br>NOUTIL<br>NPARTS=                |
| <b>Product Structure Specification</b><br>component variables<br>end items<br>part variable<br>quantity of component needed to make one part<br>item | STRUCTURE<br>STRUCTURE<br>STRUCTURE<br>STRUCTURE | COMPONENT=<br>ENDITEM=<br>PART=<br>QUANTITY= |

---

## PROC BOM Statement

### PROC BOM *options* ;

The PROC BOM statement invokes the procedure. You can specify the following options in the PROC BOM statement.

#### **DATA=SAS-data-set**

names the input SAS data set. This data set (also referred to as the Single-level BOM data set) contains all of the single-level bill of material information. See the “Single-Level Bills of Material Data Set” section on page 15 for information about the variables that can be included in this data set. If you do not specify the DATA= option, PROC BOM uses the most recently created SAS data set as the input data set.

#### **NNAMES=tsize**

specifies the size of the symbolic table used to look up the part names/numbers for which memory is initially allocated in core by the procedure. This can help PROC BOM to make better use of available memory. See the “Computer Resource Requirements” section on page 21 for details. If the number of part names/numbers exceeds *tsize*, the procedure uses a utility data set for storing the tree used for the table lookup. The default value for *tsize* is set to  $nobs \times (ncomp + 1)$ , where *nobs* is the number of observations in the Single-level BOM data set and *ncomp* is the number of Component variables.

#### **NOUTIL**

specifies that if the number of items in the problem exceeds the number of items for which memory is allocated in core, the procedure will try to expend the amount of allocated memory so that the whole problem can fit in. If you do not specify this option, the procedure resorts to the use of utility data sets and swaps between core memory and utility data sets as necessary if the number of items in the problem is larger than the number of items for which memory is allocated in core. On the other hand, if you specified this option and the problem is too large to fit in core memory, PROC BOM will stop with an error message.

#### **NPARTS=nparts**

specifies the number of items for which memory is initially allocated in core by the procedure. This can help PROC BOM to make better use of available memory. See the “Computer Resource Requirements” section on page 21 for details. If the number of items exceeds *nparts*, the procedure uses a utility data set for storing the item information array. The default value for *nparts* is set to  $nobs \times (ncomp + 1)$ , where *nobs* is the number of observations in the Single-level BOM data set and *ncomp* is the number of Component variables.

**OUT=SAS-data-set**

specifies a name for the output SAS data set that contains all of the indented bill of material information. This data set is also referred to as the Indented BOM data set. See the “Indented Bills of Material Data Set” section on page 17 for information about the variables that are included in this data set. If the OUT= option is omitted, the SAS System creates a data set and names it according to the DATA $n$  naming convention.

**SUMMARYOUT=SAS-data-set**

specifies a name for the output SAS data set that contains all of the summarized bill of material information. This data set is also referred to as the Summarized BOM data set. See the “Summarized Bills of Material Data Set” section on page 18 for information about the variables that are included in this data set. If the SUMMARYOUT= option is omitted, PROC BOM does not produce the summarized bill of material.

---

## STRUCTURE Statement

**STRUCTURE / options ;**

The STRUCTURE statement lists the variables in the input data set (the Single-level BOM data set). You should specify the **Part** variable and at least one **Component** variable. All other variables are optional. In this statement, you can also explicitly specify the parts that are to be used as the level-0 items.

**COMPONENT=(variables)****COMP=(variables)**

specifies variables in the input data set that name all of the components directly used in the item identified by the **Part** variable. These variables are also referred to as **Component** variables. The **Component** variables must be of the same type, format, and length as the **Part** variable. At least one **Component** variable must be specified.

**NOTE:** This variable is case-sensitive when it is in character format.

**ENDITEM=(enditems)**

specifies the parts that are to be used as the highest level or level-0 items in the indented bills of material. In other words, this option can be used to construct indented bills of material for subassemblies. The values for *enditems* must be either numbers (if the **Part** variable is numeric) or quoted strings (if the **Part** variable is character). If you do not specify this option, the procedure uses all final products as end items and builds an indented BOM for each of these items.

**ID=(variables)**

identifies all variables not specified in the COMPONENT=, LEADTIME=, PART=, QTYONHAND=, QUANTITY=, or REQUIREMENT= options that are to be included in the Indented BOM and the Summarized BOM output data sets. These variables are also referred to as **ID** variables. The ID= option is useful for carrying any relevant information about each part (identified by the **Part** variable) from the Single-level BOM data set to the output data sets. The ID variables cannot be named as Gros\_Req, \_Level\_, Low\_Code, Net\_Req, On\_Hand, Qty\_Per, Qty\_Prod, \_Parent\_, Paren\_ID, \_Part\_, Part\_ID, \_Prod\_, or Tot\_Lead.

**LEADTIME=***variable***DUR=***variable*

identifies the variable in the input data set that contains the lead time information for the part identified by the **Part** variable. This variable is also referred to as the **LeadTime** variable. The *lead time* of a part is the length of time between recognition of the need for an order of this part and the receipt of the parts. The variable specified must be numeric. If you do not specify this option, PROC BOM assumes the lead time for each part to be 0. The information in the **LeadTime** variable is carried to the Indented BOM output data set. Moreover, the procedure creates a new variable, **Tot\_Lead**, in the Indented BOM data set if this option is specified. See the “Indented Bills of Material Data Set” section on page 17 for more information about the **Tot\_Lead** variable.

**PART=***variable***ITEM=***variable*

specifies the variable in the input data set that names the parent item. This variable is also referred to as the **Part** variable. The **Part** variable can be either character or numeric. You must specify the **PART=** option in the **STRUCTURE** statement. All the information specified by the **ID** variables corresponds to the part identified by the **Part** variable. All the information in the **Part** variable, except the name of the variable, is carried to the Indented BOM and the Summarized BOM output data sets. The name of this variable is changed to **\_Part\_** in both output data sets.

**NOTE:** This variable is case-sensitive when it is in character format.

**QTYONHAND=***variable***INVENTORY=***variable*

identifies the variable in the input data set that contains the quantity currently on hand for the part identified by the **Part** variable. This variable is also referred to as the **QtyOnHand** variable. The **QtyOnHand** variable must be numeric. If you do not specify the **QtyOnHand** variable, the procedure assumes that you do not have any items on hand. In addition, the default name, **On\_Hand**, is used for this variable. The information in the **QtyOnHand** variable is carried to the Summarized BOM output data set if the **SUMMARYOUT=** option is specified.

**QUANTITY=**(*variables*)**QTYPER=**(*variables*)

identifies variables in the input data set that contain the quantity of items identified by the **Component** variables required to make 1 unit of the item identified by the **Part** variable. These variables are also referred to as the **Quantity** or **QtyPer** variables. The variables specified must be numeric and the number of **Quantity** variables must be equal to the number of **Component** variables. If you do not specify these variables, the procedure assumes that you need 1 unit of each listed component identified by a **Component** variable to make 1 unit of the parent item identified by the **Part** variable. If there is exactly one **Quantity** variable, the information in this variable is carried to the Indented BOM output data set. Otherwise, the output data set uses the default name **Qty\_Per**.

**REQUIREMENT=***variable*

**GROSSREQ=***variable*

identifies the variable in the input data set that contains the gross requirement of the part identified by the **Part** variable. This variable is also referred to as the **Requirement** variable and must be numeric. If you do not specify this option, the procedure assumes that the gross requirement for all parts are missing except for the final products. All final products are assumed to have gross requirements of 1 unit. In addition, the default name, **Gros\_Req**, is used for this variable. The information in this variable is carried to the Summarized BOM output data set if the **SUMMARYOUT=** option is specified.

## Details

### Single-Level Bills of Material Data Set

The BOM procedure uses the Single-level BOM data set as input data with key variable names being used to identify the appropriate information. Table 1.2 lists all of the variables in this input data set, with their type and their interpretation by the BOM procedure. It also lists the options in the **STRUCTURE** statement that are used to identify those variables.

**Table 1.2.** Single-Level BOM Data Set and Associated Variables

| Variable    | Type                 | Option       | Interpretation                                      |
|-------------|----------------------|--------------|-----------------------------------------------------|
| Component   | same as Part         | COMPONENT=   | Component name or number                            |
| ID          | character or numeric | ID=          | Additional information about the part               |
| LeadTime    | numeric              | LEADTIME=    | Lead time of the part                               |
| Part        | character or numeric | PART=        | Part name or number                                 |
| QtyOnHand   | numeric              | QTYONHAND=   | Quantity of the part that is currently on hand      |
| Quantity    | numeric              | QUANTITY=    | Quantity of the component required per unit of part |
| Requirement | numeric              | REQUIREMENT= | Gross requirement of the part                       |

The value of the **Requirement** variable should be missing for most parts, except master schedule items. A *master schedule item* (MSI) is a part that is selected to be planned by the master scheduler. In general, final products are master schedule items. Some companies like to select a few items that are deemed critical in their impact on lower level components or resources as master schedule items. The requirements of the master schedule items are also called *independent demands*. In the Single-level BOM data set, if the value of the **Requirement** variable is greater than or equal to

0, then the procedure treats the item identified by the **Part** variable as an MSI and assumes that its gross requirement is planned by the master scheduler. The gross requirements of other parts (also known as the *dependent demands*) are determined by the procedure as the value of the net requirements of their parent items times the quantity required to make one unit of the parent items. This process is called the *dependent demand process* (Clement, Coldrick, and Sari 1992).

In general, the information in the **Part** and the **LeadTime** variables and all ID variables are stored in the company's *part master data file*. The part master data file gives various details of fairly constant data associated with the individual parts. It might contain the following kinds of data (Clement, Coldrick, and Sari 1992):

- part number or name
- part description
- engineering drawing number
- revision level
- commodity code
- cost
- product code
- lead time
- order quantity
- item type or status
- latest engineering change dates
- lot size
- planner/buyer codes
- unit of measure
- make or buy

The information for the **Part**, **Component**, and **Quantity** variables can usually be found in the *product structure data*. The product structure data describe the material content of a product at each stocking level in the manufacturing process. The quantity of each part that is currently on hand is contained in the company's inventory data, and the gross requirements of master schedule items come from the *master production schedule* (MPS). The Single-level BOM data set contains information from all these typical data sets. To obtain the Single-level BOM data set, you can combine the part master data file, inventory data file, and master production schedule, based on the common variable **Part**, to create a new data set. Then you can either append the product structure data (part-component relationship information) to the end of the new data set (as in Example 1.1 on page 21) or merge them together (as in Example 1.2 on page 27).

## Indented Bills of Material Data Set

The Indented BOM data set produced by PROC BOM contains all the information in the Single-level BOM data set, plus a few more variables, to describe the structure of the products. The `Part`, `LeadTime`, `Quantity`, and all ID variables in the Single-level BOM data set are carried to this output data set. The name of the `Part` variable is changed to `_Part_`. Moreover, if there is more than one `Quantity` variable in the Single-level BOM data set, the procedure uses the name `Qty_Per` as the `Quantity` variable in the Indented BOM data set. Like the Single-level BOM data set, the information contained in those variables is associated with the item identified by the `_Part_` variable. A few new variables are also added to the Indented BOM data set.

The `_Level_` variable contains the indenture level of the item identified by the `_Part_` variable. The top-most parts (the final products) have level 0, and all components that are directly used by these parts have level 1. All subsequent components of those items have the level number increased by 1. This process continues until there are no subsequent components. The `_Parent_` variable contains the part name or number of the parent item. The `_Prod_` variable contains the part name or number of the final product in the product structure. If the input data set contains the lead time information for each part, PROC BOM measures the total lead time accumulated from the final product (identified by the `_Prod_` variable) to the part (identified by the `_Part_` variable) and puts it in the `Tot_Lead` variable. The `Qty_Prod` variable denotes the quantity of the part (identified by the `_Part_` variable) required to make one unit of the product identified by the `_Prod_` variable.

Two more variables are added to the output data set: variable `Part_ID` and variable `Paren_ID`. Note that if an item is used in more than one parent item, it appears more than once, under every subassembly in which it is used. In order to distinguish those parts that have multiple appearances, PROC BOM assigns a unique number to each part in the multilevel tree structure. The `Part_ID` variable contains this identification number of the part (identified by the `_Part_` variable), and the `Paren_ID` variable contains the identification number of the parent item. These two variables are useful when you use the Indented BOM data set as the input data set for other SAS/OR procedures, such as PROC NETDRAW and PROC CPM, which require unique identification of each node.

Table 1.3 lists all of the variables in the Indented BOM data set. It also lists the type and a brief description of these variables.

**Table 1.3.** Indented BOM Data Set and Associated Variables

| Variable | Type                        | Interpretation                        |
|----------|-----------------------------|---------------------------------------|
| ID       | character or numeric        | Additional information about the part |
| LeadTime | numeric                     | Lead time of the part                 |
| _Level_  | numeric                     | Level number of the part              |
| _Parent_ | same as <code>_Part_</code> | Parent item of the part               |
| Paren_ID | numeric                     | Identification number of the parent   |

**Table 1.3.** (continued)

| Variable | Type                 | Interpretation                                         |
|----------|----------------------|--------------------------------------------------------|
| _Part_   | character or numeric | Part name or number                                    |
| Part_ID  | numeric              | Identification number of the part                      |
| _Prod_   | same as _Part_       | Name or number of the final product                    |
| Qty_Prod | numeric              | Quantity required to produce one unit of final product |
| Quantity | numeric              | Quantity per assembly                                  |
| Tot_Lead | numeric              | Total lead time                                        |

The records in the Indented BOM data set are organized so that the final product item is always listed first. In addition, each item is listed directly after its parent and before any right siblings of the parent. For example, from the Indented BOM data as shown in Figure 1.3 on page 7, you can easily see that the final product 'LA01' is the first record of the Indented BOM data set. Moreover, the part '1700' is listed directly after its parent, 'A100', and before the parts 'S100' and 'B100' (the right siblings of the part 'A100').

The single-level (part-component) and multilevel relationships of the product structure are clearly shown in the Indented BOM data. For example, the three components that go into the parent 'LA01' are easily determined by the level 1 identifiers. It is also easy to see that the part 'B100' (Base assembly) requires four components at level 2.

---

## Summarized Bills of Material Data Set

If the SUMMARYOUT= option in the PROC BOM statement is specified, the BOM procedure creates a Summarized BOM data set. The summarized bill of material data set lists all the parts and their quantities required and currently on hand. Unlike the indented bill of material data set, it does not list the part-parent relationships and the levels of manufacture, and it lists a part only once for the total quantity used.

The Part, QtyOnHand, and all ID variables in the Single-level BOM data set are carried to this output data set. The name of the Part variable is changed to \_Part\_. If the item identified by the \_Part\_ variable is a master schedule item, the value of the Requirement variable (the independent demand) is also carried to this data set. The value of the Requirement variable (dependent demand) for those items that are not master schedule items is determined by the net requirement of the parent item times the quantity needed per parent item.



The new variable, **Net\_Req**, denotes the net requirement of the item identified by the **\_Part\_** variable. The value of the **Net\_Req** variable is determined as

$$\text{Net\_Req} = \text{Requirement} - \text{QtyOnHand}$$

The **Low\_Code** variable denotes the low-level code of the item identified by the **\_Part\_** variable, that is, the lowest level in any bill of material at which the item appears. The low-level codes are necessary to make sure that the net requirement of any item is not calculated until all the gross requirements have been calculated down to that level.

Table 1.4 lists all of the variables in the Summarized BOM data set. It also lists the type and a brief description of these variables.

**Table 1.4.** Summarized BOM Data Set and Associated Variables

| Variable    | Type                 | Interpretation                                 |
|-------------|----------------------|------------------------------------------------|
| ID          | character or numeric | Additional information about the part          |
| Low_Code    | numeric              | Low-level code of the part                     |
| Net_Req     | numeric              | Net requirement of the part                    |
| _Part_      | character or numeric | Part name or number                            |
| QtyOnHand   | numeric              | Quantity of the part that is currently on hand |
| Requirement | numeric              | Gross requirement of the part                  |

## Missing Values in the Input Data Set

The following table summarizes the treatment of missing values for variables in the Single-level BOM input data set.

**Table 1.5.** Treatment of Missing Values in the BOM Procedure

| Variable  | Value Used / Assumption Made / Action Taken                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Component | value ignored                                                                                                                                          |
| ID        | missing, if Part is not missing;<br>otherwise ignored                                                                                                  |
| LeadTime  | 0, if Part is not missing;<br>otherwise ignored                                                                                                        |
| Part      | input error: procedure stops with error message, if this is the first record;<br>otherwise, treat this record as a continuation of the previous record |

**Table 1.5.** (continued)

| Variable    | Value Used / Assumption Made / Action Taken                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| QtyOnHand   | 0, if Part is not missing;<br>otherwise ignored                                                                                                                                     |
| Quantity    | 1, if corresponding Component variable is not missing;<br>otherwise ignored                                                                                                         |
| Requirement | ignored if Part is missing;<br>1, if Part is not missing and the part identified by the Part<br>variable is a final product;<br>otherwise, the value is determined by the procedure |

Note that a missing value is allowed for the Part variable only when the information for an item is specified in more than one record (see Figure 1.1 on page 5 as an example). If the Part variable is missing, the values for the ID, LeadTime, QtyOnHand, and Requirement variables are ignored by the procedure. If the Part variable is not missing and the item identified by the Part variable is not a master schedule item (that is, it is not a final product and the value of the corresponding Requirement variable is missing), the gross requirement of this part is determined based on the dependent demand process. See the “Summarized Bills of Material Data Set” section on page 18 for details about the dependent demand process.

---

## Macro Variable `_ORBOM_`

The BOM procedure defines a macro variable named `_ORBOM_`. This variable contains a character string that indicates the status of the procedure. It is set at procedure termination. The form of the `_ORBOM_` character string is `STATUS= REASON=`, where `STATUS=` is either `SUCCESSFUL` or `ERROR_EXIT`, and `REASON=` (if PROC BOM terminated unsuccessfully) can be one of the following:

```

CYCLE
BADDATA_ERROR
MEMORY_ERROR
IO_ERROR
SEMANTIC_ERROR
SYNTAX_ERROR
BOM_BUG
UNKNOWN_ERROR

```

This information can be used when PROC BOM is one step in a larger program that needs to determine whether the procedure terminated successfully or not. Because `_ORBOM_` is a standard SAS macro variable, it can be used in the ways that all macro variables can be used.

---

## Computer Resource Requirements

There is no inherent limit on the size of the product structure that can be handled with the BOM procedure. The number of parts and components are constrained only by the amount of memory available. Naturally, there needs to be a sufficient amount of core memory available in order to invoke and initialize the SAS system. As far as possible, the procedure attempts to store all the data in core memory.

However, if the problem is too large to fit in core memory, the procedure resorts to the use of utility data sets and swaps between core memory and utility data sets as necessary, unless the NOUTIL option is specified. The procedure uses the NPARTS= and NNAMEs= options to determine approximate problem size. If these options are not specified, the procedure estimates default values on the basis of the number of observations in the Single-level BOM data set. See the “Syntax” section on page 11 for default specifications.

The storage requirement for the data area and the time required by the procedure are proportional to the number of items in the product structure.

---

## Examples

This section contains examples that illustrate the features of the BOM procedure. Most of the examples use the data from the ABC Lamp Company product structure described in the “Getting Started” section beginning on page 3.

---

### Example 1.1. Bill of Material with Lead Time Information

As explained in the section “Single-Level Bills of Material Data Set” beginning on page 15, most companies store the relatively constant information for parts in a part master file. In this example, you start with two different data files, Pmaster1 and ParComp1. The Pmaster1 data set, as shown in Output 1.1.1, is a simplified version of the ABC Lamp Company’s part master file. Note that the data set contains the description, the lead time, and the unit of measure information for each part. The data set ParComp1 (as shown in Output 1.1.2) lists the part-component relationship of each part along with the quantity of the component item needed to make one unit of that particular part.

The SAS DATA step code to accomplish this is as follows:

```

/* Part Master data */
data PMaster1;
  input Part      $8.
         Desc     $24.
         Unit     $8.
         LeadTime 8.0
        ;

  datalines;
1100   Finished shaft           Each           2
1200   6-Diameter steel plate  Each           3

```

|      |                        |        |   |
|------|------------------------|--------|---|
| 1300 | Hub                    | Each   | 2 |
| 1400 | 1/4-20 Screw           | Each   | 1 |
| 1500 | Steel holder           | Each   | 2 |
| 1600 | One-way socket         | Each   | 2 |
| 1700 | Wiring assembly        | Each   | 1 |
| 2100 | 3/8 Steel tubing       | Inches | 3 |
| 2200 | 16-Gauge lamp cord     | Feet   | 2 |
| 2300 | Standard plug terminal | Each   | 1 |
| A100 | Socket assembly        | Each   | 1 |
| B100 | Base assembly          | Each   | 1 |
| LA01 | Lamp LA                | Each   | 2 |
| S100 | Black shade            | Each   | 2 |

;

```
/* Part-Component relationship data */
```

```
data ParComp1;
```

```
input Part $8.
```

```
Component $8.
```

```
QtyPer 8.0 ;
```

```
datalines;
```

```
LA01 B100 1
```

```
S100 1
```

```
A100 1
```

```
B100 1100 1
```

```
1200 1
```

```
1300 1
```

```
1400 4
```

```
A100 1500 1
```

```
1600 1
```

```
1700 1
```

```
1100 2100 26
```

```
1500 1400 2
```

```
1700 2200 12
```

```
2300 1
```

```
;
```

```
/* Display the part master data */
```

```
proc print data=PMaster1 noobs;
```

```
title 'ABC Lamp Company';
```

```
title3 'Part Master Data';
```

```
run;
```

```
/* Display the part-component relationship data */
```

```
proc print data=ParComp1 noobs;
```

```
title 'ABC Lamp Company';
```

```
title3 'Part-Component Relationship Data';
```

```
run;
```

**Output 1.1.1.** Part Master Data Set

| ABC Lamp Company |                        |        |           |
|------------------|------------------------|--------|-----------|
| Part Master Data |                        |        |           |
| Part             | Desc                   | Unit   | Lead Time |
| 1100             | Finished shaft         | Each   | 2         |
| 1200             | 6-Diameter steel plate | Each   | 3         |
| 1300             | Hub                    | Each   | 2         |
| 1400             | 1/4-20 Screw           | Each   | 1         |
| 1500             | Steel holder           | Each   | 2         |
| 1600             | One-way socket         | Each   | 2         |
| 1700             | Wiring assembly        | Each   | 1         |
| 2100             | 3/8 Steel tubing       | Inches | 3         |
| 2200             | 16-Gauge lamp cord     | Feet   | 2         |
| 2300             | Standard plug terminal | Each   | 1         |
| A100             | Socket assembly        | Each   | 1         |
| B100             | Base assembly          | Each   | 1         |
| LA01             | Lamp LA                | Each   | 2         |
| S100             | Black shade            | Each   | 2         |

**Output 1.1.2.** Part-Component Relationship Data Set

| ABC Lamp Company                 |           |         |
|----------------------------------|-----------|---------|
| Part-Component Relationship Data |           |         |
| Part                             | Component | Qty Per |
| LA01                             | B100      | 1       |
|                                  | S100      | 1       |
|                                  | A100      | 1       |
|                                  |           |         |
| B100                             | 1100      | 1       |
|                                  | 1200      | 1       |
|                                  | 1300      | 1       |
|                                  | 1400      | 4       |
|                                  |           |         |
| A100                             | 1500      | 1       |
|                                  | 1600      | 1       |
|                                  | 1700      | 1       |
|                                  |           |         |
| 1100                             | 2100      | 26      |
| 1500                             | 1400      | 2       |
| 1700                             | 2200      | 12      |
|                                  | 2300      | 1       |

The SIBOM1 data set is created by appending the part-component data set, ParComp1, to the end of the data set PMaster1 to form an input data set for the procedure:

```

/* Append the Part-Component information to */
/* the part master data                      */
data SIBOM1;
  set PMaster1 ParComp1;
run;
```

The following code produces the Indented BOM data set and displays it in Output 1.1.3. A new variable, `Tot_Lead`, has been added to the output data set. This variable denotes the total lead time accumulated from the product 'LA01' to the item identified by the `_Part_` variable:

```

/* Create the indented BOM with lead time */
proc bom data=SlBOM1 out=IndBOM1;
  structure / part=Part
             leadtime=LeadTime
             component=Component
             quantity=QtyPer
             id=(Desc Unit);
run;

/* Display the indented BOM data */
proc print data=IndBOM1 noobs;
  var _Level_ _Part_ Desc QtyPer Qty_Prod
      Unit LeadTime Tot_Lead _Parent_ _Prod_;
  title 'ABC Lamp Company';
  title3 'Indented Bill of Material, Part LA01';
run;

```

**Output 1.1.3.** Indented Bill of Material with Lead Time

| ABC Lamp Company                     |      |                        |          |        |           |                 |        |         |  |
|--------------------------------------|------|------------------------|----------|--------|-----------|-----------------|--------|---------|--|
| Indented Bill of Material, Part LA01 |      |                        |          |        |           |                 |        |         |  |
| Level                                | Part | Description            | Quantity | Unit   | Lead Time | Total Lead Time | Parent | Product |  |
| 0                                    | LA01 | Lamp LA                | 1        | Each   | 2         | 2               |        | LA01    |  |
| 1                                    | A100 | Socket assembly        | 1        | Each   | 1         | 3               | LA01   | LA01    |  |
| 2                                    | 1700 | Wiring assembly        | 1        | Each   | 1         | 4               | A100   | LA01    |  |
| 3                                    | 2300 | Standard plug terminal | 1        | Each   | 1         | 5               | 1700   | LA01    |  |
| 3                                    | 2200 | 16-Gauge lamp cord     | 12       | Feet   | 2         | 6               | 1700   | LA01    |  |
| 2                                    | 1600 | One-way socket         | 1        | Each   | 2         | 5               | A100   | LA01    |  |
| 2                                    | 1500 | Steel holder           | 1        | Each   | 2         | 5               | A100   | LA01    |  |
| 3                                    | 1400 | 1/4-20 Screw           | 2        | Each   | 1         | 6               | 1500   | LA01    |  |
| 1                                    | S100 | Black shade            | 1        | Each   | 2         | 4               | LA01   | LA01    |  |
| 1                                    | B100 | Base assembly          | 1        | Each   | 1         | 3               | LA01   | LA01    |  |
| 2                                    | 1400 | 1/4-20 Screw           | 4        | Each   | 1         | 4               | B100   | LA01    |  |
| 2                                    | 1300 | Hub                    | 1        | Each   | 2         | 5               | B100   | LA01    |  |
| 2                                    | 1200 | 6-Diameter steel plate | 1        | Each   | 3         | 6               | B100   | LA01    |  |
| 2                                    | 1100 | Finished shaft         | 1        | Each   | 2         | 5               | B100   | LA01    |  |
| 3                                    | 2100 | 3/8 Steel tubing       | 26       | Inches | 3         | 8               | 1100   | LA01    |  |

The following code uses the Indented BOM data set produced in the previous invocation of PROC BOM to produce a data set that can be used by the NETDRAW procedure:

```

/* Prepare the data set for running NETDRAW */
data IndBOM1a(drop=Part_ID);
  set IndBOM1;
  Paren_ID=Part_ID;
run;

data IndBOM1b;
  set IndBOM1(keep=Paren_ID Part_ID);
run;

data TreBOM1;
  set IndBOM1a IndBOM1b;
  LTnQP = put(LeadTime, f3.)||" "||put(QtyPer, f3.);
run;

```

PROC NETDRAW is invoked with the NODISPLAY option to create a data set (Layout1) that contains all the layout information for the tree structure. The OUT= option specifies the name of the layout data set:

```

/* Specify graphics options */
title h=1 j=c 'Multilevel Bill of Material with Lead-time Offset';
footnote h=0.5 j=l
  'Node shows Part Number, Lead-time, and Quantity Required';
pattern1 v=e c=gray;

/* get the layout information for the BOM tree */
proc netdraw data=TreBOM1( where=(Paren_ID NE .) )
  out=Layout1 nodisplay;
  actnet / act=Paren_ID succ=Part_ID id=(Part_ LTnQP Tot_Lead)
  ybetween=3 xbetween=15 tree
  rectilinear nodefid nodelabel;
run;

```

In the next invocation, PROC NETDRAW uses a modified layout of the nodes to produce a diagram where the nodes are aligned according to the total lead time:

```

/* Lead time offset the X coordinate of each node */
data TreBOM1a;
  set Layout1;
  if _seq_ = 0;
  drop _seq_;
  _x_ = Tot_Lead;
run;

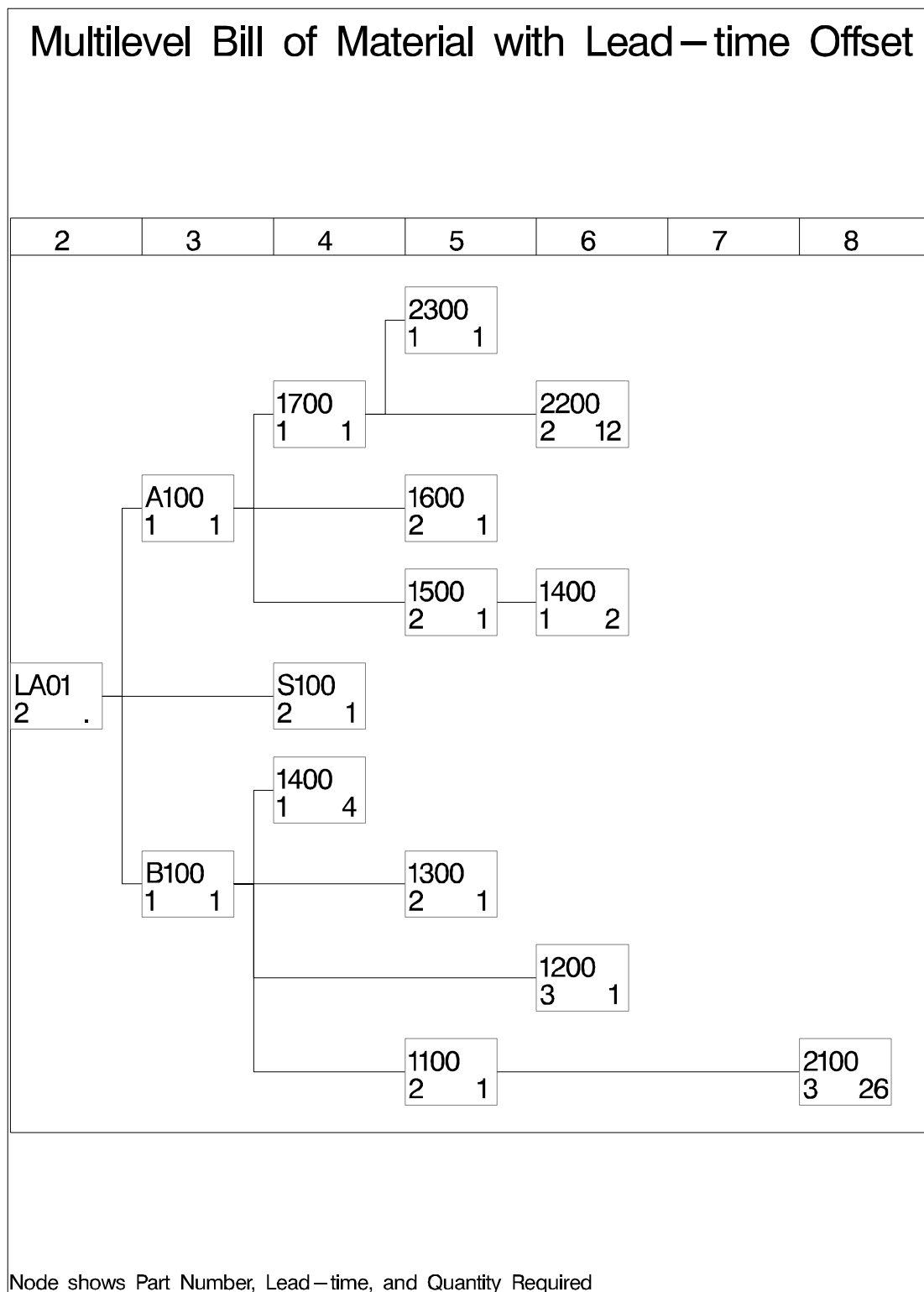
/* display the BOM tree with lead-time offset */
proc netdraw data=TreBOM1a;
  actnet / id=(Part_ LTnQP)
  font=swiss ctext=black htext=3 carcs=black
  align=Tot_Lead frame pcompress
  xbetween=15 ybetween=3
  arrowhead=0 rectilinear nodefid nodelabel;
run;

```

Refer to “The NETDRAW Procedure” chapter in the *SAS/OR User’s Guide: Project Management* for details.

The resulting tree diagram is shown in Output 1.1.4.

#### Output 1.1.4. BOM Tree Structure with Lead-time Offset





## Example 1.2. Planning Bill of Material

The previous examples dealt with the bill of material from the perspective of the manufacturing process. In this example, let's turn to another type of BOM that is often useful in planning and handling engineering charges. It is referred to as a *planning BOM*, a *pseudo BOM*, *super BOM*, or a *phantom BOM*.

Let's assume that the ABC Lamp Company sells lamps with 8 different shades (either 14 inches or 15 inches, in the color black, white, cream, or yellow), 3 alternate base plates (6 inches, 7 inches, or 8 inches), and 2 types of sockets ('One-way' and 'Three-way'). Working with all different options, the company now has 48 ( $= 8 \times 3 \times 2$ ) different final products. In other words, the ABC Lamp Company has 48 different end items to forecast and plan. This might not sound too complicated; however, this example is only a simplified case. In the real world, some manufacturing companies making automobiles, computers, or machine tools can easily make over 5,000 different final products. So many final products are obviously impossible to forecast accurately and plan with any hope of validity. However, a reasonable forecast for total lamp sales could be made, for example, 10,000 per week. Some parts like the '1100 Finished shaft' are common to all configurations. Although these parts are produced independently of one another, they can be grouped as common parts on the BOM for administrative purposes. The new part that is created to group all common parts is never stocked and hence it is called a *phantom part*. Phantom items are generally assigned 0 lead times and lot-for-lot order quantity. Besides the common parts phantom item, other phantom items are created for planning different options. These items are also referred to as *model items*. This process is known as the *modularized bill of material construction*.

The following SAS code creates three data sets: The Phantom2 data set contains the generic part 'LAXX' and all phantom items. The data set Option2 lists all options of each model item that are not already in the part master data set PMaster1 (described in Example 1.1 on page 21). Finally, the ParComp2 data set contains the re-grouped part-component relationship information:

```

/* Generic and phantom part data */
data Phantom2;
  input Part      $8.
         Desc     $24.
         Req      8.0
         Unit     $8.
         LeadTime 4.0
        ;
  datalines;
LAXX    Lamp LA                      10000 Each      3
4000    Common parts                  . Each      0
A10X    Socket assembly options      . Each      0
B10X    Base assembly options        . Each      0
S10X    Shade options                 . Each      0
;

```

```

/* Additional options and alternative parts */
data Option2;
  input Part      $8.
         Desc     $24.
         Req      8.0
         Unit     $8.
         LeadTime 4.0
  ;
datalines;
A101  Three-way socket assem.      . Each      1
B101  7in Base assembly           . Each      1
B102  8in Base assembly           . Each      1
S101  14in White shade            . Each      2
S102  14in Cream shade            . Each      2
S103  14in Yellow shade           . Each      2
S104  15in Black shade            . Each      2
S105  15in White shade            . Each      2
S106  15in Cream shade            . Each      2
S107  15in Yellow shade           . Each      2
1201  7-Diameter steel plate      . Each      3
1202  8-Diameter steel plate      . Each      3
1601  Three-way socket            . Each      2
;

/* part-component relationship data */
data ParComp2;
  input Part      $8.
         Component $8.
         QtyPer   8.2
  ;
datalines;
LAXX  4000      1.00
LAXX  B10X      1.00
LAXX  S10X      1.00
LAXX  A10X      1.00
4000  1100      1.00
4000  1300      1.00
4000  1400      4.00
4000  1500      1.00
4000  1700      1.00
B10X  B100      0.32
B10X  B101      0.41
B10X  B102      0.33
S10X  S100      0.07
S10X  S101      0.18
S10X  S102      0.24
S10X  S103      0.10
S10X  S104      0.06
S10X  S105      0.14
S10X  S106      0.22
S10X  S107      0.10
A10X  A100      0.11
A10X  A101      0.92
B100  1200      1.00

```

|      |      |       |
|------|------|-------|
| B101 | 1201 | 1.00  |
| B102 | 1202 | 1.00  |
| A100 | 1600 | 1.00  |
| A101 | 1601 | 1.00  |
| 1100 | 2100 | 26.00 |
| 1500 | 1400 | 2.00  |
| 1700 | 2200 | 12.00 |
| 1700 | 2300 | 1.00  |

;

The part identified as '4000' is the phantom item for common parts '1100', '1300', '1400', '1500', and '1700'. Each available option is structured as a model item with a quantity per parent (identified as the QtyPer variable) that represents its forecast popularity or option percentage. Note that the total percentage of each option in this example is more than 100 percent (for example, the total percentage of the 'A10X: Socket assembly options' is  $0.11 + 0.92 = 1.03$ ). This extra percentage is used to cover the uncertainty of the exact percentage split. Using this procedure to cover possible high side demand for each option is called *option overplanning* (Fogarty, Blackstone, and Hoffmann 1991).

The new part master data set PMaster2 combines the generic and phantom part data, the additional options data, and the old part master file shown in Output 1.1.1 on page 23. The modularized single-level BOM data set, SIBOM2, combines the data set PMaster2 with the part-component data set ParComp2 by using PROC SQL. The SAS code to accomplish this is as follows:

```

/* Append the old part master data to the new */
/* phantom item and additional option data sets */
data PMaster2;
  set Phantom2
      Option2
      PMaster1(where=(Part NE 'LA01'));
run;

proc sort data=PMaster2;
  by Part;
run;

proc sort data=ParComp2;
  by Part;
run;

/* Merge the Part-Component information to the new */
/* part master data to create an input data set */
proc sql;
  create table SIBOM2 as
  select PMaster2.*,
         ParComp2.Component, ParComp2.QtyPer
  from PMaster2 left join ParComp2
    on PMaster2.Part=ParComp2.Part;
quit;

```

Output 1.2.1. Modularized Single-Level Bills of Material

| ABC Lamp Company                           |                         |       |        |           |           |         |
|--------------------------------------------|-------------------------|-------|--------|-----------|-----------|---------|
| Modularized Single-Level Bills of Material |                         |       |        |           |           |         |
| Part                                       | Desc                    | Req   | Unit   | Lead Time | Component | Qty Per |
| 1100                                       | Finished shaft          | .     | Each   | 2         | 2100      | 26.00   |
| 1200                                       | 6-Diameter steel plate  | .     | Each   | 3         |           | .       |
| 1201                                       | 7-Diameter steel plate  | .     | Each   | 3         |           | .       |
| 1202                                       | 8-Diameter steel plate  | .     | Each   | 3         |           | .       |
| 1300                                       | Hub                     | .     | Each   | 2         |           | .       |
| 1400                                       | 1/4-20 Screw            | .     | Each   | 1         |           | .       |
| 1500                                       | Steel holder            | .     | Each   | 2         | 1400      | 2.00    |
| 1600                                       | One-way socket          | .     | Each   | 2         |           | .       |
| 1601                                       | Three-way socket        | .     | Each   | 2         |           | .       |
| 1700                                       | Wiring assembly         | .     | Each   | 1         | 2200      | 12.00   |
| 1700                                       | Wiring assembly         | .     | Each   | 1         | 2300      | 1.00    |
| 2100                                       | 3/8 Steel tubing        | .     | Inches | 3         |           | .       |
| 2200                                       | 16-Gauge lamp cord      | .     | Feet   | 2         |           | .       |
| 2300                                       | Standard plug terminal  | .     | Each   | 1         |           | .       |
| 4000                                       | Common parts            | .     | Each   | 0         | 1100      | 1.00    |
| 4000                                       | Common parts            | .     | Each   | 0         | 1300      | 1.00    |
| 4000                                       | Common parts            | .     | Each   | 0         | 1400      | 4.00    |
| 4000                                       | Common parts            | .     | Each   | 0         | 1500      | 1.00    |
| 4000                                       | Common parts            | .     | Each   | 0         | 1700      | 1.00    |
| A100                                       | Socket assembly         | .     | Each   | 1         | 1600      | 1.00    |
| A101                                       | Three-way socket assem. | .     | Each   | 1         | 1601      | 1.00    |
| A10X                                       | Socket assembly options | .     | Each   | 0         | A100      | 0.11    |
| A10X                                       | Socket assembly options | .     | Each   | 0         | A101      | 0.92    |
| B100                                       | Base assembly           | .     | Each   | 1         | 1200      | 1.00    |
| B101                                       | 7in Base assembly       | .     | Each   | 1         | 1201      | 1.00    |
| B102                                       | 8in Base assembly       | .     | Each   | 1         | 1202      | 1.00    |
| B10X                                       | Base assembly options   | .     | Each   | 0         | B100      | 0.32    |
| B10X                                       | Base assembly options   | .     | Each   | 0         | B101      | 0.41    |
| B10X                                       | Base assembly options   | .     | Each   | 0         | B102      | 0.33    |
| LAXX                                       | Lamp LA                 | 10000 | Each   | 3         | 4000      | 1.00    |
| LAXX                                       | Lamp LA                 | 10000 | Each   | 3         | B10X      | 1.00    |
| LAXX                                       | Lamp LA                 | 10000 | Each   | 3         | S10X      | 1.00    |
| LAXX                                       | Lamp LA                 | 10000 | Each   | 3         | A10X      | 1.00    |
| S100                                       | Black shade             | .     | Each   | 2         |           | .       |
| S101                                       | 14in White shade        | .     | Each   | 2         |           | .       |
| S102                                       | 14in Cream shade        | .     | Each   | 2         |           | .       |
| S103                                       | 14in Yellow shade       | .     | Each   | 2         |           | .       |
| S104                                       | 15in Black shade        | .     | Each   | 2         |           | .       |
| S105                                       | 15in White shade        | .     | Each   | 2         |           | .       |
| S106                                       | 15in Cream shade        | .     | Each   | 2         |           | .       |
| S107                                       | 15in Yellow shade       | .     | Each   | 2         |           | .       |
| S10X                                       | Shade options           | .     | Each   | 0         | S100      | 0.07    |
| S10X                                       | Shade options           | .     | Each   | 0         | S101      | 0.18    |
| S10X                                       | Shade options           | .     | Each   | 0         | S102      | 0.24    |
| S10X                                       | Shade options           | .     | Each   | 0         | S103      | 0.10    |
| S10X                                       | Shade options           | .     | Each   | 0         | S104      | 0.06    |
| S10X                                       | Shade options           | .     | Each   | 0         | S105      | 0.14    |
| S10X                                       | Shade options           | .     | Each   | 0         | S106      | 0.22    |
| S10X                                       | Shade options           | .     | Each   | 0         | S107      | 0.10    |

The following SAS code sorts and displays the modularized single-level BOM data set as shown in Output 1.2.1:

```
proc sort data=S1BOM2;
  by Part;
run;
```

```

/* Display the single-level BOM data */
proc print data=SlBOM2 noobs;
  title 'ABC Lamp Company';
  title3 'Modularized Single-Level Bills of Material';
run;

```

The following code invokes PROC BOM to generate the planning BOM and the summarized BOM from the modularized single-level BOM:

```

/* Generate the indented BOM and summarized BOM data sets */
proc bom data=SlBOM2 out=IndBOM2 summaryout=SumBOM2;
  structure / part=Part
             requirement=Req
             leadtime=LeadTime
             component=Component
             quantity=QtyPer
             id=(Desc Unit);
run;

/* Display the indented BOM data */
proc print data=IndBOM2 noobs;
  var _Level_ _Part_ Desc QtyPer Qty_Prod
      Unit LeadTime Tot_Lead _Parent_ _Prod_;
  title 'ABC Lamp Company';
  title3 'Indented Bill of Material, Part LAXX';
run;

proc sort data=SumBOM2;
  by _Part_;
run;

/* Display the summarized BOM data */
proc print data=SumBOM2 noobs;
  title 'ABC Lamp Company';
  title3 'Summarized Bill of Material, Part LAXX';
run;

```

The indented bill of material is shown in Output 1.2.2. Output 1.2.3 lists the quantity of each part that is needed to build 10,000 lamps.

Output 1.2.2. Planning Bill of Material with Option Overplanning

| ABC Lamp Company                     |      |                         |       |       |        |     |      |        |         |
|--------------------------------------|------|-------------------------|-------|-------|--------|-----|------|--------|---------|
| Indented Bill of Material, Part LAXX |      |                         |       |       |        |     |      |        |         |
| Level                                | Part | Description             | Qty   | Unit  | Parent | Qty | Unit | Parent | Product |
| —                                    | —    | —                       | —     | —     | —      | —   | —    | —      | —       |
| 0                                    | LAXX | Lamp LA                 | .     | 1.00  | Each   | 3   | 3    |        | LAXX    |
| 1                                    | A10X | Socket assembly options | 1.00  | 1.00  | Each   | 0   | 3    | LAXX   | LAXX    |
| 2                                    | A101 | Three-way socket assem. | 0.92  | 0.92  | Each   | 1   | 4    | A10X   | LAXX    |
| 3                                    | 1601 | Three-way socket        | 1.00  | 0.92  | Each   | 2   | 6    | A101   | LAXX    |
| 2                                    | A100 | Socket assembly         | 0.11  | 0.11  | Each   | 1   | 4    | A10X   | LAXX    |
| 3                                    | 1600 | One-way socket          | 1.00  | 0.11  | Each   | 2   | 6    | A100   | LAXX    |
| 1                                    | S10X | Shade options           | 1.00  | 1.00  | Each   | 0   | 3    | LAXX   | LAXX    |
| 2                                    | S107 | 15in Yellow shade       | 0.10  | 0.10  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S106 | 15in Cream shade        | 0.22  | 0.22  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S105 | 15in White shade        | 0.14  | 0.14  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S104 | 15in Black shade        | 0.06  | 0.06  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S103 | 14in Yellow shade       | 0.10  | 0.10  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S102 | 14in Cream shade        | 0.24  | 0.24  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S101 | 14in White shade        | 0.18  | 0.18  | Each   | 2   | 5    | S10X   | LAXX    |
| 2                                    | S100 | Black shade             | 0.07  | 0.07  | Each   | 2   | 5    | S10X   | LAXX    |
| 1                                    | B10X | Base assembly options   | 1.00  | 1.00  | Each   | 0   | 3    | LAXX   | LAXX    |
| 2                                    | B102 | 8in Base assembly       | 0.33  | 0.33  | Each   | 1   | 4    | B10X   | LAXX    |
| 3                                    | 1202 | 8-Diameter steel plate  | 1.00  | 0.33  | Each   | 3   | 7    | B102   | LAXX    |
| 2                                    | B101 | 7in Base assembly       | 0.41  | 0.41  | Each   | 1   | 4    | B10X   | LAXX    |
| 3                                    | 1201 | 7-Diameter steel plate  | 1.00  | 0.41  | Each   | 3   | 7    | B101   | LAXX    |
| 2                                    | B100 | Base assembly           | 0.32  | 0.32  | Each   | 1   | 4    | B10X   | LAXX    |
| 3                                    | 1200 | 6-Diameter steel plate  | 1.00  | 0.32  | Each   | 3   | 7    | B100   | LAXX    |
| 1                                    | 4000 | Common parts            | 1.00  | 1.00  | Each   | 0   | 3    | LAXX   | LAXX    |
| 2                                    | 1700 | Wiring assembly         | 1.00  | 1.00  | Each   | 1   | 4    | 4000   | LAXX    |
| 3                                    | 2300 | Standard plug terminal  | 1.00  | 1.00  | Each   | 1   | 5    | 1700   | LAXX    |
| 3                                    | 2200 | 16-Gauge lamp cord      | 12.00 | 12.00 | Feet   | 2   | 6    | 1700   | LAXX    |
| 2                                    | 1500 | Steel holder            | 1.00  | 1.00  | Each   | 2   | 5    | 4000   | LAXX    |
| 3                                    | 1400 | 1/4-20 Screw            | 2.00  | 2.00  | Each   | 1   | 6    | 1500   | LAXX    |
| 2                                    | 1400 | 1/4-20 Screw            | 4.00  | 4.00  | Each   | 1   | 4    | 4000   | LAXX    |
| 2                                    | 1300 | Hub                     | 1.00  | 1.00  | Each   | 2   | 5    | 4000   | LAXX    |
| 2                                    | 1100 | Finished shaft          | 1.00  | 1.00  | Each   | 2   | 5    | 4000   | LAXX    |
| 3                                    | 2100 | 3/8 Steel tubing        | 26.00 | 26.00 | Inches | 3   | 8    | 1100   | LAXX    |

**Output 1.2.3. Summarized Part Requirement**

| ABC Lamp Company                       |          |        |         |         |                         |        |
|----------------------------------------|----------|--------|---------|---------|-------------------------|--------|
| Summarized Bill of Material, Part LAXX |          |        |         |         |                         |        |
| _Part_                                 | Low_Code | Req    | On_Hand | Net_Req | Desc                    | Unit   |
| 1100                                   | 2        | 10000  | 0       | 10000   | Finished shaft          | Each   |
| 1200                                   | 3        | 3200   | 0       | 3200    | 6-Diameter steel plate  | Each   |
| 1201                                   | 3        | 4100   | 0       | 4100    | 7-Diameter steel plate  | Each   |
| 1202                                   | 3        | 3300   | 0       | 3300    | 8-Diameter steel plate  | Each   |
| 1300                                   | 2        | 10000  | 0       | 10000   | Hub                     | Each   |
| 1400                                   | 3        | 60000  | 0       | 60000   | 1/4-20 Screw            | Each   |
| 1500                                   | 2        | 10000  | 0       | 10000   | Steel holder            | Each   |
| 1600                                   | 3        | 1100   | 0       | 1100    | One-way socket          | Each   |
| 1601                                   | 3        | 9200   | 0       | 9200    | Three-way socket        | Each   |
| 1700                                   | 2        | 10000  | 0       | 10000   | Wiring assembly         | Each   |
| 2100                                   | 3        | 260000 | 0       | 260000  | 3/8 Steel tubing        | Inches |
| 2200                                   | 3        | 120000 | 0       | 120000  | 16-Gauge lamp cord      | Feet   |
| 2300                                   | 3        | 10000  | 0       | 10000   | Standard plug terminal  | Each   |
| 4000                                   | 1        | 10000  | 0       | 10000   | Common parts            | Each   |
| A100                                   | 2        | 1100   | 0       | 1100    | Socket assembly         | Each   |
| A101                                   | 2        | 9200   | 0       | 9200    | Three-way socket assem. | Each   |
| A10X                                   | 1        | 10000  | 0       | 10000   | Socket assembly options | Each   |
| B100                                   | 2        | 3200   | 0       | 3200    | Base assembly           | Each   |
| B101                                   | 2        | 4100   | 0       | 4100    | 7in Base assembly       | Each   |
| B102                                   | 2        | 3300   | 0       | 3300    | 8in Base assembly       | Each   |
| B10X                                   | 1        | 10000  | 0       | 10000   | Base assembly options   | Each   |
| LAXX                                   | 0        | 10000  | 0       | 10000   | Lamp LA                 | Each   |
| S100                                   | 2        | 700    | 0       | 700     | Black shade             | Each   |
| S101                                   | 2        | 1800   | 0       | 1800    | 14in White shade        | Each   |
| S102                                   | 2        | 2400   | 0       | 2400    | 14in Cream shade        | Each   |
| S103                                   | 2        | 1000   | 0       | 1000    | 14in Yellow shade       | Each   |
| S104                                   | 2        | 600    | 0       | 600     | 15in Black shade        | Each   |
| S105                                   | 2        | 1400   | 0       | 1400    | 15in White shade        | Each   |
| S106                                   | 2        | 2200   | 0       | 2200    | 15in Cream shade        | Each   |
| S107                                   | 2        | 1000   | 0       | 1000    | 15in Yellow shade       | Each   |
| S10X                                   | 1        | 10000  | 0       | 10000   | Shade options           | Each   |

**Example 1.3. Bills of Material Verification**

The previous example illustrated how some companies may have more than one type of bill of material. It is important that while the different types of BOM may differ in the way parts are grouped together, they must have the same component items (except those phantom parts) in the same quantities. This example demonstrates a simple method that compares the different types of BOM and detects any differences.

The transaction data set Trans3 contains the new values of the gross requirements of the lamp 'LAXX' and the options. Note that the options 'B100', 'S100', and 'A100' that are associated with the lamp 'LA01' described in the "Getting Started" section on page 3, as well as the lamp 'LAXX', have gross requirements of 1. All other options have 0 gross requirements. Therefore, the item 'LAXX' and all of the selected options are master schedule items. See the "Single-Level Bills of Material Data Set" section beginning on page 15 for a detailed description about master schedule items.

The following SAS DATA step code updates the value of the **Req** variable in the **SIBOM2** data set (shown in Output 1.2.1 on page 30) with the value of the variable in the transaction data set **Trans3**:

```

/* Gross requirement transaction data set */
data Trans3;
    input Part      $8.
           Req      8.0
    ;
datalines;
LAXX          1
B100          1
B101          0
B102          0
S100          1
S101          0
S102          0
S103          0
S104          0
S105          0
S106          0
S107          0
A100          1
A101          0
;

proc sort data=Trans3;
    by Part;
run;

/* Update the gross requirement values of the */
/* single-level BOM data set */
data SLBOM3(drop=OldReq);
    merge SLBOM2(rename=(Req=OldReq)) Trans3(in=in2);
    by Part;

    if not in2 then Req=OldReq;
run;

```

The following code invokes PROC BOM with the new input data set. The summarized bill of material is shown in Output 1.3.1:

```

/* Generate the indented BOM and Summarized BOM */
proc bom data=SLBOM3 out=IndBOM3 summaryout=SumBOM3;
    structure / part=Part
               requirement=Req
               leadtime=LeadTime
               component=Component
               quantity=QtyPer
               id=(Desc Unit);
run;

```



```

/* Sort and display the summarized BOM data */
proc sort data=SumBOM3;
  by _Part_;
run;

proc print data=SumBOM3(where=(Net_Req NE 0)) noobs;
  title 'ABC Lamp Company';
  title3 'Summarized Bill of Material, Part LA01';
run;

```

**Output 1.3.1.** Summarized Bills of Material

| ABC Lamp Company                       |          |     |         |         |                         |        |
|----------------------------------------|----------|-----|---------|---------|-------------------------|--------|
| Summarized Bill of Material, Part LA01 |          |     |         |         |                         |        |
| _Part_                                 | Low_Code | Req | On_Hand | Net_Req | Desc                    | Unit   |
| 1100                                   | 2        | 1   | 0       | 1       | Finished shaft          | Each   |
| 1200                                   | 3        | 1   | 0       | 1       | 6-Diameter steel plate  | Each   |
| 1300                                   | 2        | 1   | 0       | 1       | Hub                     | Each   |
| 1400                                   | 3        | 6   | 0       | 6       | 1/4-20 Screw            | Each   |
| 1500                                   | 2        | 1   | 0       | 1       | Steel holder            | Each   |
| 1600                                   | 3        | 1   | 0       | 1       | One-way socket          | Each   |
| 1700                                   | 2        | 1   | 0       | 1       | Wiring assembly         | Each   |
| 2100                                   | 3        | 26  | 0       | 26      | 3/8 Steel tubing        | Inches |
| 2200                                   | 3        | 12  | 0       | 12      | 16-Gauge lamp cord      | Feet   |
| 2300                                   | 3        | 1   | 0       | 1       | Standard plug terminal  | Each   |
| 4000                                   | 1        | 1   | 0       | 1       | Common parts            | Each   |
| A100                                   | 2        | 1   | 0       | 1       | Socket assembly         | Each   |
| A10X                                   | 1        | 1   | 0       | 1       | Socket assembly options | Each   |
| B100                                   | 2        | 1   | 0       | 1       | Base assembly           | Each   |
| B10X                                   | 1        | 1   | 0       | 1       | Base assembly options   | Each   |
| LAXX                                   | 0        | 1   | 0       | 1       | Lamp LA                 | Each   |
| S100                                   | 2        | 1   | 0       | 1       | Black shade             | Each   |
| S10X                                   | 1        | 1   | 0       | 1       | Shade options           | Each   |

By comparing the summarized BOM in Output 1.3.1 with the one displayed in Figure 1.6 on page 10, you can verify that the two bills of material are in agreement except for the low-level code of each part and the phantom items, such as '4000 Common parts', 'A10X Socket assembly options', 'B10X Base assembly options', and 'S10X Shade options'.

## References

- Aho, A.V., Hopcroft, J.E., and Ullman, J.D. (1983), *Data Structures and Algorithms*, Reading, MA: Addison-Wesley.
- Clement, J., Coldrick, A., and Sari, J. (1992), *Manufacturing Data Structures: Building Foundations for Excellence with Bills of Materials and Process Information*, Essex Junction, VT: Oliver Wright Limited Publications, Inc.
- Cox, J.F., III, and Blackstone, J.H., Jr., eds. (1998), *APICS Dictionary*, Ninth Edition, Alexandria, VA: APICS.

Fogarty, D.W., Blackstone, J.H., and Hoffmann, T.R. (1991), *Production and Inventory Management*, Second Edition, Cincinnati, OH: South-Western Publishing Co.

# Chapter 2

## The CPM Procedure

### Chapter Table of Contents

---

|                                                  |    |
|--------------------------------------------------|----|
| <b>OVERVIEW</b>                                  | 39 |
| <b>SYNTAX</b>                                    | 39 |
| PROC CPM Statement                               | 39 |
| ACTUAL Statement                                 | 40 |
| <b>DETAILS</b>                                   | 40 |
| Finish Milestones                                | 40 |
| <b>EXAMPLES</b>                                  | 41 |
| Example 2.1 Use of the SETFINISHMILESTONE Option | 41 |



## Chapter 2

# The CPM Procedure

---

### Overview

Enhancements have been added to the CPM procedure as follows:

- The SETFINISHMILESTONE option has been added to the CPM procedure statement to enable you to specify that milestones (zero duration activities) should have the same start and finish times as the finish time of their predecessor. In other words, this option enables milestones that mark the *end* of the preceding activity to coincide with its finish time.
- The FIXASTART option has been added to the ACTUAL statement to enable you to request that the actual start and finish times of an activity should be left unchanged even if they coincide with a non-working time. Thus, if the actual start time is specified to be sometime on Sunday, it is left unchanged even if Sunday is a non-working day in the activity's calendar.

---

### Syntax

---

#### PROC CPM Statement

The following new option is available in the CPM procedure statement:

##### **SETFINISHMILESTONE**

specifies that milestones (zero duration activities) should have the same start and finish times as the finish time of their predecessor. In other words, this option enables milestones that mark the *end* of the preceding activity to coincide with its finish time. By default, if a milestone M is a successor to an activity that finishes at the end of the day (say 15Mar2000), the start and finish times for the milestone are specified as the beginning of the next day (16Mar2000). This corresponds to the definition of start times in the CPM procedure: *all* start times indicate the *beginning* of the date specified. For zero duration activities, the finish time is defined to be the same as the start time. The SETFINISHMILESTONE option specifies that the start and finish times for the milestone M should be specified as 15Mar2000, with the interpretation that the milestone's schedule corresponds to the *end* of the day. There may be exceptions to this definition if there are special alignment constraints on the milestone. For details, see the "Finish Milestones" section on page 40.

---

## ACTUAL Statement

The following new option is available in the ACTUAL statement:

### FIXASTART

specifies that the actual start time of an activity should not be overwritten if it is specified to be on a non-work day. By default, none of the start or finish times of an activity can occur during a non-work period corresponding to the activity's calendar. If the actual start time is specified on a non-work day, it is moved to the nearest work day. The FIXASTART option specifies that the actual start and finish times be left unchanged even if they coincide with a non-working time. Thus, if the actual start time is specified to be sometime on Sunday, it is left unchanged even if Sunday is a non-working day in the activity's calendar.

---

## Details

---

### Finish Milestones

By default, the start and finish times for the different schedules computed by PROC CPM denote the first and last *day* of work, respectively, when the values are formatted as SAS *date* values. All start times are assumed to denote the beginning of the day and all finish times are assumed to correspond to the end of the day. If the times are SAS *time* or *datetime* values, they denote the first and last *second* of work, respectively. However, for zero duration activities, *both* the start and the finish times correspond to the beginning of the date (or second) specified.

Thus, according to the preceding definitions, the CPM procedure assumes that all milestones are scheduled at the *beginning* of the day indicated by their start times. In other words, the milestones can be regarded as *start* milestones since they correspond to the *beginning* of the time period indicated by their scheduled times. However, in some situations, you may want to treat the milestones as *finish* milestones.

Consider the following example:

Activity 'A' has a 2-day duration and is followed by a milestone (zero duration) activity, 'B'. Suppose that activity 'A' starts on March 15, 2000. The default calculations by the CPM procedure will produce the following schedule for the two activities:

| OBS | Activity | Duration | E_START   | E_FINISH  |
|-----|----------|----------|-----------|-----------|
| 1   | A        | 2        | 15MAR2000 | 16MAR2000 |
| 2   | B        | 0        | 17Mar2000 | 17MAR2000 |

The start and finish times of the milestone activity, 'B', are interpreted as the beginning of March 17, 2000. In some situations, you may want the milestones to start and finish on the same day as their predecessors. For instance, in this example, you may want the start and finish time of activity 'B' to be set to March 16, 2000, with the interpretation that the time corresponds to the *end* of the day. Such milestones will be referred to as *finish milestones*.

The SETFINISHMILESTONE option in the PROC CPM statement indicates that a milestone that is linked to its predecessor by a *Finish-to-Start* or a *Finish-to-Finish* precedence constraint should be treated as a *finish milestone*. In other words, such a milestone should have the start and finish time set to the *end* of the day that the predecessor activity finishes. There are some exceptions to this rule:

- There is an alignment constraint on activity ‘B’ that requires the milestone to start on a later day than the date dictated by the precedence constraint.
- Activity ‘B’ has an actual start or finish time specified that is inconsistent with the predecessor’s finish date.

Note that the alignment constraint that affects the early schedule of the project may not have any impact on the late schedule. Thus, a milestone may be treated as a finish milestone for the late schedule even if it is not a finish milestone according to the early schedule. See Example 2.1 for an illustration of this situation. In addition, while computing the resource-constrained schedule, a start milestone (according to the early schedule) may in fact turn out to be a finish milestone according to the resource-constrained schedule.

Since the same milestone could be treated as either a start or a finish milestone depending on the presence or absence of an alignment constraint, or depending on the type of the schedule (early, late, resource-constrained, or actual), the CPM procedure adds extra variables to the Schedule data set corresponding to each type of schedule. These variables, EFINMILE, LFINMILE, SFINMILE, and AFINMILE, indicate for each milestone activity in the project whether the corresponding schedule times (early, late, resource-constrained, or actual) are to be interpreted as finish milestone times. These variables have a value of ‘1’ if the milestone is treated as a finish milestone for the corresponding schedule; otherwise, the value is missing. In addition to providing an unambiguous interpretation for the schedule times of the milestones, these variables are useful in plotting the schedules correctly using the Gantt procedure. (See Example 2.1.)

---

## Examples

---

### Example 2.1. Use of the SETFINISHMILESTONE Option

A simple activity network is used to illustrate the use of the SETFINISHMILESTONE option in a couple of different scenarios.

The following DATA step reads the project network in AON format into a SAS data set named `tasks`. The data set (printed in Output 2.1.1) contains an Activity variable (`act`), a Successor variable (`succ`), a Lag variable (`lag`), and a Duration variable (`dur`). Note that there are several milestones linked to other activities through different types of precedence constraints. The data set also contains some alignment constraints as specified by the variables `target` and `trgttype`. Note that the treatment of the milestones will vary depending on the presence or absence of the alignment constraints. The data set also contains two variables that indicate the expected early

schedule dates for the milestones corresponding to two different invocations of PROC CPM: the variable `notrgtmd` corresponds to the non-aligned schedule and the variable `miledate` corresponds to an invocation with the `ALIGNDATE` statement.

```
data tasks;
  input act $ 1-7 succ $ 9-15 lag $ dur target date9.
        trgttype $ miledate date9. notrgtmd date9.;
  format target date7. miledate date7. notrgtmd date7.;
datalines;
Task 0 Mile 1 ss_0 1 24Jan00 SGE . .
Mile 1 Task 2 . 0 . . 24Jan00 24Jan00
Task 2 . . 1 . . . .
Task 3 Mile 4 . 1 . . . .
Mile 4 . . 0 . . 24Jan00 24Jan00
Task 5 Mile 6 . 1 . . . .
Mile 6 Mile 7 FS_1 0 . . 24Jan00 24Jan00
Mile 7 . . 0 . . 25Jan00 25Jan00
Task 8 Mile 9 SS_3 1 . . . .
Mile 9 Mile 10 . 0 . . 27Jan00 27Jan00
Mile 10 . . 0 . . 27Jan00 27Jan00
Task 11 Mile 12 . 2 . . . .
Mile 12 Mile 13 FS_1 0 26Jan00 SGE 26Jan00 25Jan00
Mile 13 . . 0 . . 27Jan00 26Jan00
;
```

**Output 2.1.1.** Input Data Set

| Input Data Set |         |         |      |     |         |          |          |          |
|----------------|---------|---------|------|-----|---------|----------|----------|----------|
| Obs            | act     | succ    | lag  | dur | target  | trgttype | miledate | notrgtmd |
| 1              | Task 0  | Mile 1  | ss_0 | 1   | 24JAN00 | SGE      | .        | .        |
| 2              | Mile 1  | Task 2  |      | 0   | .       |          | 24JAN00  | 24JAN00  |
| 3              | Task 2  |         |      | 1   | .       |          | .        | .        |
| 4              | Task 3  | Mile 4  |      | 1   | .       |          | .        | .        |
| 5              | Mile 4  |         |      | 0   | .       |          | 24JAN00  | 24JAN00  |
| 6              | Task 5  | Mile 6  |      | 1   | .       |          | .        | .        |
| 7              | Mile 6  | Mile 7  | FS_1 | 0   | .       |          | 24JAN00  | 24JAN00  |
| 8              | Mile 7  |         |      | 0   | .       |          | 25JAN00  | 25JAN00  |
| 9              | Task 8  | Mile 9  | SS_3 | 1   | .       |          | .        | .        |
| 10             | Mile 9  | Mile 10 |      | 0   | .       |          | 27JAN00  | 27JAN00  |
| 11             | Mile 10 |         |      | 0   | .       |          | 27JAN00  | 27JAN00  |
| 12             | Task 11 | Mile 12 |      | 2   | .       |          | .        | .        |
| 13             | Mile 12 | Mile 13 | FS_1 | 0   | 26JAN00 | SGE      | 26JAN00  | 25JAN00  |
| 14             | Mile 13 |         |      | 0   | .       |          | 27JAN00  | 26JAN00  |



Output 2.1.2. Default Schedule

| Default Schedule           |             |                  |             |             |                                      |                                 |                                      |                                 |                                      |                       |                       |
|----------------------------|-------------|------------------|-------------|-------------|--------------------------------------|---------------------------------|--------------------------------------|---------------------------------|--------------------------------------|-----------------------|-----------------------|
| O<br>b<br>j<br>e<br>c<br>t | a<br>c<br>t | s<br>u<br>c<br>c | d<br>u<br>r | l<br>a<br>g | n<br>o<br>t<br>r<br>g<br>t<br>m<br>d | E<br>_<br>S<br>T<br>A<br>R<br>T | E<br>_<br>F<br>I<br>N<br>I<br>S<br>H | L<br>_<br>S<br>T<br>A<br>R<br>T | L<br>_<br>F<br>I<br>N<br>I<br>S<br>H | T<br>_<br>F<br>_<br>F | T<br>_<br>F<br>_<br>F |
|                            |             |                  |             |             |                                      |                                 |                                      |                                 |                                      |                       |                       |
| 1                          | Task 0      | Mile 1           | 1           | ss_0        | .                                    | 24JAN00                         | 24JAN00                              | 26JAN00                         | 26JAN00                              | 2                     | 0                     |
| 2                          | Mile 1      | Task 2           | 0           |             | 24JAN00                              | 24JAN00                         | 24JAN00                              | 26JAN00                         | 26JAN00                              | 2                     | 0                     |
| 3                          | Task 2      |                  | 1           |             | .                                    | 24JAN00                         | 24JAN00                              | 26JAN00                         | 26JAN00                              | 2                     | 2                     |
| 4                          | Task 3      | Mile 4           | 1           |             | .                                    | 24JAN00                         | 24JAN00                              | 26JAN00                         | 26JAN00                              | 2                     | 0                     |
| 5                          | Mile 4      |                  | 0           |             | 24JAN00                              | 25JAN00                         | 25JAN00                              | 27JAN00                         | 27JAN00                              | 2                     | 2                     |
| 6                          | Task 5      | Mile 6           | 1           |             | .                                    | 24JAN00                         | 24JAN00                              | 25JAN00                         | 25JAN00                              | 1                     | 0                     |
| 7                          | Mile 6      | Mile 7           | 0           | FS_1        | 24JAN00                              | 25JAN00                         | 25JAN00                              | 26JAN00                         | 26JAN00                              | 1                     | 0                     |
| 8                          | Mile 7      |                  | 0           |             | 25JAN00                              | 26JAN00                         | 26JAN00                              | 27JAN00                         | 27JAN00                              | 1                     | 1                     |
| 9                          | Task 8      | Mile 9           | 1           | SS_3        | .                                    | 24JAN00                         | 24JAN00                              | 24JAN00                         | 24JAN00                              | 0                     | 0                     |
| 10                         | Mile 9      | Mile 10          | 0           |             | 27JAN00                              | 27JAN00                         | 27JAN00                              | 27JAN00                         | 27JAN00                              | 0                     | 0                     |
| 11                         | Mile 10     |                  | 0           |             | 27JAN00                              | 27JAN00                         | 27JAN00                              | 27JAN00                         | 27JAN00                              | 0                     | 0                     |
| 12                         | Task 11     | Mile 12          | 2           |             | .                                    | 24JAN00                         | 25JAN00                              | 24JAN00                         | 25JAN00                              | 0                     | 0                     |
| 13                         | Mile 12     | Mile 13          | 0           | FS_1        | 25JAN00                              | 26JAN00                         | 26JAN00                              | 26JAN00                         | 26JAN00                              | 0                     | 0                     |
| 14                         | Mile 13     |                  | 0           |             | 26JAN00                              | 27JAN00                         | 27JAN00                              | 27JAN00                         | 27JAN00                              | 0                     | 0                     |

First, the CPM procedure is invoked with the default treatment of milestones. The resulting schedule is printed in Output 2.1.2. Note the dates for the milestones. Compare these dates with the values of the variable `notrgtmd` that contains the desired milestone schedule dates corresponding to the finish times of their predecessor.

```

/* Schedule the project */
proc cpm data=tasks out=out0
    collapse interval=day
    date='24jan00'd;
    activity act;
    successor succ /lag=(lag);
    duration dur;
    id lag notrgtmd;
run;

title 'Default Schedule';
proc print; run;

```

Next, the CPM procedure is invoked with the option `SETFINISHMILESTONE` and the resulting schedule is printed in Output 2.1.3. The variables `EFINMILE` and `LFINMILE` indicate if the milestone is a finish milestone or not. For example, the milestone 'Mile 12' has `E_FINISH = 25JAN00` and the value of `EFINMILE` is '1', indicating that the activity finishes at the end of the day on January 25, 2000. The milestone 'Mile 13' (with a finish-to-start lag of 1 day) finishes at the end of the day on January 26, 2000. In fact, as the late finish schedule indicates, the value of `L_FINISH` for 'Mile 13' (and the project finish time) is the end of the day on 26JAN00. Note that both the variables `EFINMILE` and `LFINMILE` have the same values for all the activities in this example.

```

proc cpm data=tasks out=out1
    collapse interval=day
    date='24jan00'd
    setfinishmilestone;
    activity act;
    successor succ /lag=(lag);
    duration dur;
    id lag notrgtmd;
run;

title 'Schedule with option SETFINISHMILESTONE';
title2 'No Target Dates';
proc print;
    id act;
    var succ lag dur notrgtmd e_start e_finish
        l_start l_finish efinmile lfinmile;
run;

```

Output 2.1.3. Schedule with SETFINISHMILESTONE Option

| Schedule with option SETFINISHMILESTONE<br>No Target Dates |         |      |     |          |         |          |         |          |          |          |
|------------------------------------------------------------|---------|------|-----|----------|---------|----------|---------|----------|----------|----------|
| act                                                        | task    | lag  | dur | notrgtmd | e_start | e_finish | l_start | l_finish | efinmile | lfinmile |
| Task 0                                                     | Mile 1  | ss_0 | 1   | .        | 24JAN00 | 24JAN00  | 26JAN00 | 26JAN00  | .        | .        |
| Mile 1                                                     | Task 2  |      | 0   | 24JAN00  | 24JAN00 | 24JAN00  | 26JAN00 | 26JAN00  | .        | .        |
| Task 2                                                     |         |      | 1   | .        | 24JAN00 | 24JAN00  | 26JAN00 | 26JAN00  | .        | .        |
| Task 3                                                     | Mile 4  |      | 1   | .        | 24JAN00 | 24JAN00  | 26JAN00 | 26JAN00  | .        | .        |
| Mile 4                                                     |         |      | 0   | 24JAN00  | 24JAN00 | 24JAN00  | 26JAN00 | 26JAN00  | 1        | 1        |
| Task 5                                                     | Mile 6  |      | 1   | .        | 24JAN00 | 24JAN00  | 25JAN00 | 25JAN00  | .        | .        |
| Mile 6                                                     | Mile 7  | FS_1 | 0   | 24JAN00  | 24JAN00 | 24JAN00  | 25JAN00 | 25JAN00  | 1        | 1        |
| Mile 7                                                     |         |      | 0   | 25JAN00  | 25JAN00 | 25JAN00  | 26JAN00 | 26JAN00  | 1        | 1        |
| Task 8                                                     | Mile 9  | SS_3 | 1   | .        | 24JAN00 | 24JAN00  | 24JAN00 | 24JAN00  | .        | .        |
| Mile 9                                                     | Mile 10 |      | 0   | 27JAN00  | 27JAN00 | 27JAN00  | 27JAN00 | 27JAN00  | .        | .        |
| Mile 10                                                    |         |      | 0   | 27JAN00  | 27JAN00 | 27JAN00  | 27JAN00 | 27JAN00  | .        | .        |
| Task 11                                                    | Mile 12 |      | 2   | .        | 24JAN00 | 25JAN00  | 24JAN00 | 25JAN00  | .        | .        |
| Mile 12                                                    | Mile 13 | FS_1 | 0   | 25JAN00  | 25JAN00 | 25JAN00  | 25JAN00 | 25JAN00  | 1        | 1        |
| Mile 13                                                    |         |      | 0   | 26JAN00  | 26JAN00 | 26JAN00  | 26JAN00 | 26JAN00  | 1        | 1        |

The next invocation of CPM illustrates the effect of alignment constraints on the milestones. As explained in the “Finish Milestones” section on page 40, imposing an alignment constraint of type SGE on a milestone may change it from a finish milestone to a start milestone (default behavior) as far as the early schedule of the project is concerned. In the following program, the CPM procedure is invoked with the SETFINISHMILESTONE option and the ALIGNDATE and ALIGNTYPE statements. The resulting schedule is printed in Output 2.1.4. Note that the early schedule of the milestones now corresponds to the values in the variable miledat. Note also that the activities ‘Mile 12’ and ‘Mile 13’ are no longer finish milestones, as indicated by missing values for the variable EFINMILE.

```
proc cpm data=tasks out=out2
    collapse
    interval=day
    date='24jan00'd
    setfinishmilestone;
    activity act;
    successor succ /lag=(lag);
    duration dur;
    aligndate target;
    aligntype trgttype;
    id target trgttype lag miledat;
run;

title 'Schedule with option SETFINISHMILESTONE';
title2 'Target Dates change Early Schedule for some Milestones';
proc print;
    id act;
    var succ lag target trgttype miledat e_start e_finish
        l_start l_finish efinmile lfinmile;
run;
```

Output 2.1.4. Effect of Alignment Constraints

| Schedule with option SETFINISHMILESTONE                |         |      |         |     |         |         |         |         |         |   |   |
|--------------------------------------------------------|---------|------|---------|-----|---------|---------|---------|---------|---------|---|---|
| Target Dates change Early Schedule for some Milestones |         |      |         |     |         |         |         |         |         |   |   |
|                                                        |         |      | t       | m   | E       | L       | L       | E       | L       |   |   |
|                                                        |         |      | t       | i   | —       | —       | —       | —       | —       |   |   |
|                                                        |         |      | a       | e   | S       | I       | S       | I       | N       |   |   |
|                                                        |         |      | r       | d   | T       | N       | T       | N       | M       |   |   |
|                                                        |         |      | g       | a   | A       | I       | A       | I       | I       |   |   |
|                                                        |         |      | e       | t   | R       | S       | R       | S       | L       |   |   |
|                                                        |         |      | t       | e   | T       | H       | T       | H       | E       |   |   |
| a                                                      | s       |      |         |     |         |         |         |         |         |   |   |
| c                                                      | u       | a    |         |     |         |         |         |         |         |   |   |
| t                                                      | c       | g    |         |     |         |         |         |         |         |   |   |
| Task 0                                                 | Mile 1  | ss_0 | 24JAN00 | SGE | .       | 24JAN00 | 24JAN00 | 26JAN00 | 26JAN00 | . | . |
| Mile 1                                                 | Task 2  | .    | 24JAN00 | .   | 24JAN00 | 24JAN00 | 24JAN00 | 26JAN00 | 26JAN00 | . | . |
| Task 2                                                 | .       | .    | 24JAN00 | .   | 24JAN00 | 24JAN00 | 24JAN00 | 26JAN00 | 26JAN00 | . | . |
| Task 3                                                 | Mile 4  | .    | 24JAN00 | .   | 24JAN00 | 24JAN00 | 24JAN00 | 26JAN00 | 26JAN00 | . | . |
| Mile 4                                                 | .       | .    | 24JAN00 | .   | 24JAN00 | 24JAN00 | 24JAN00 | 26JAN00 | 26JAN00 | 1 | 1 |
| Task 5                                                 | Mile 6  | .    | 24JAN00 | .   | 24JAN00 | 24JAN00 | 24JAN00 | 25JAN00 | 25JAN00 | . | . |
| Mile 6                                                 | Mile 7  | FS_1 | 24JAN00 | .   | 24JAN00 | 24JAN00 | 24JAN00 | 25JAN00 | 25JAN00 | 1 | 1 |
| Mile 7                                                 | .       | .    | 25JAN00 | .   | 25JAN00 | 25JAN00 | 25JAN00 | 26JAN00 | 26JAN00 | 1 | 1 |
| Task 8                                                 | Mile 9  | SS_3 | 24JAN00 | .   | 24JAN00 | 24JAN00 | 24JAN00 | 24JAN00 | 24JAN00 | . | . |
| Mile 9                                                 | Mile 10 | .    | 27JAN00 | .   | 27JAN00 | 27JAN00 | 27JAN00 | 27JAN00 | 27JAN00 | . | . |
| Mile 10                                                | .       | .    | 27JAN00 | .   | 27JAN00 | 27JAN00 | 27JAN00 | 27JAN00 | 27JAN00 | . | . |
| Task 11                                                | Mile 12 | .    | 24JAN00 | .   | 24JAN00 | 25JAN00 | 24JAN00 | 24JAN00 | 25JAN00 | . | . |
| Mile 12                                                | Mile 13 | FS_1 | 26JAN00 | SGE | 26JAN00 | 26JAN00 | 26JAN00 | 25JAN00 | 25JAN00 | . | 1 |
| Mile 13                                                | .       | .    | 27JAN00 | .   | 27JAN00 | 27JAN00 | 27JAN00 | 26JAN00 | 26JAN00 | . | 1 |

The interpretation of the start and finish times for a milestone depends on whether it is a start milestone or a finish milestone. By default, all milestones are start milestones and are assumed to be scheduled at the beginning of the date specified in the start or finish time variable. As such, PROC GANTT displays these milestones at the start of the corresponding days on the Gantt chart. However, if a milestone is a finish milestone then it may not be displayed correctly on the Gantt chart, depending on the scale of the display.

In this example, PROC GANTT is used to display the schedule produced in Output 2.1.4. Recall that the schedule is saved in the data set `out2`. First, PROC GANTT is invoked without any modifications to the schedule data set. The resulting Gantt chart is displayed in Output 2.1.5. Note that the finish milestones (with values of `EFINMILE = '1'`) are not plotted correctly. For example, 'Mile 6' is plotted at the *beginning* instead of the *end* of the schedule bar for the predecessor activity, 'Act 5'. To correct this problem, you can adjust the schedule variables for the finish milestones and plot the new values, as illustrated by the second invocation of PROC GANTT. The corrected Gantt chart is displayed in Output 2.1.6.

```

title h=1.5
      'Schedule with option SETFINISHMILESTONE and ALIGNDATE';
title2 'Gantt Chart of Early Schedule without adjustment';
proc gantt data=out2(drop=l_);
    chart / compress act=act succ=succ lag=lag
            font=swiss scale=7
            cprec=cyan cmile=magenta
            caxis=black cframe=ligr;
            dur=dur nojobnum nolegend;
    id act succ lag e_start efinmile;
run;

/* Save adjusted E_START and E_FINISH times for finish
   milestones */
data temp;
    set out2;
    format estart efinish date7.;
    estart = e_start;
    efinish = e_finish;
    if efinmile then do;
        estart=estart+1;
        efinish=efinish+1;
    end;
run;

/* Plot the adjusted start and finish times for the
   early schedule */
title h=1.5
      'Schedule with option SETFINISHMILESTONE and ALIGNDATE';
title2 'Gantt Chart of Early Schedule after adjustment';
proc gantt data=temp(drop=l_);
    chart / compress act=act succ=succ lag=lag
            font=swiss scale=7
            es=estart ef=efinish

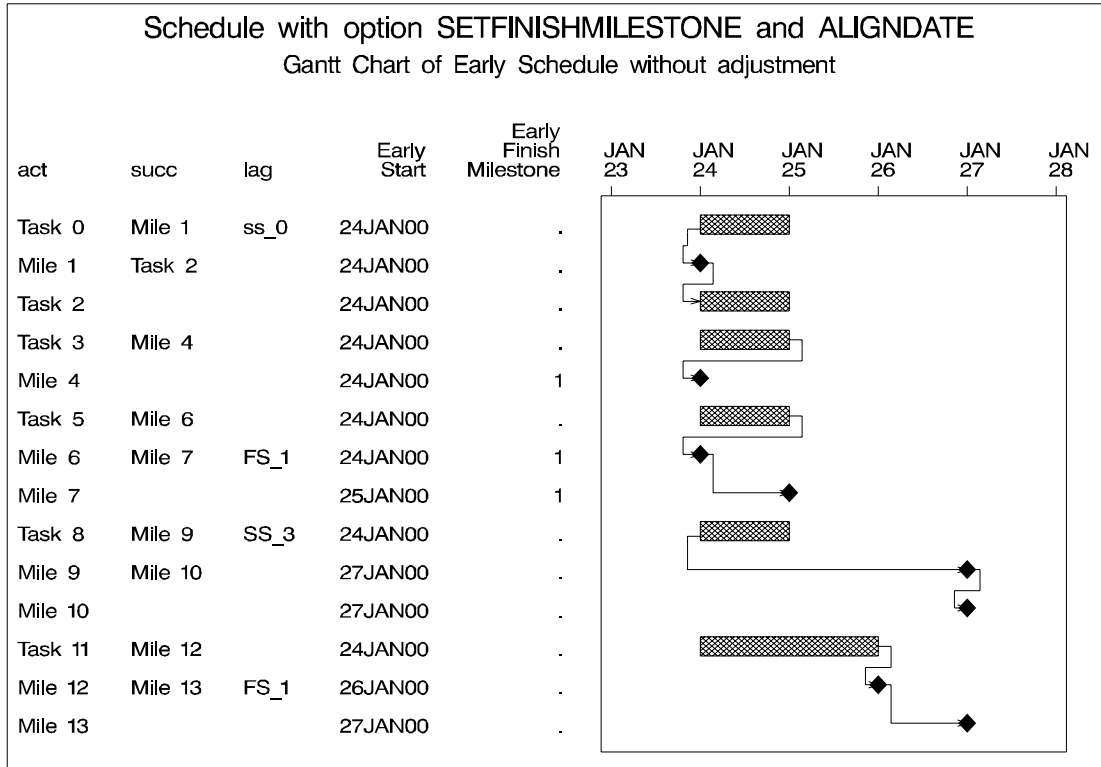
```

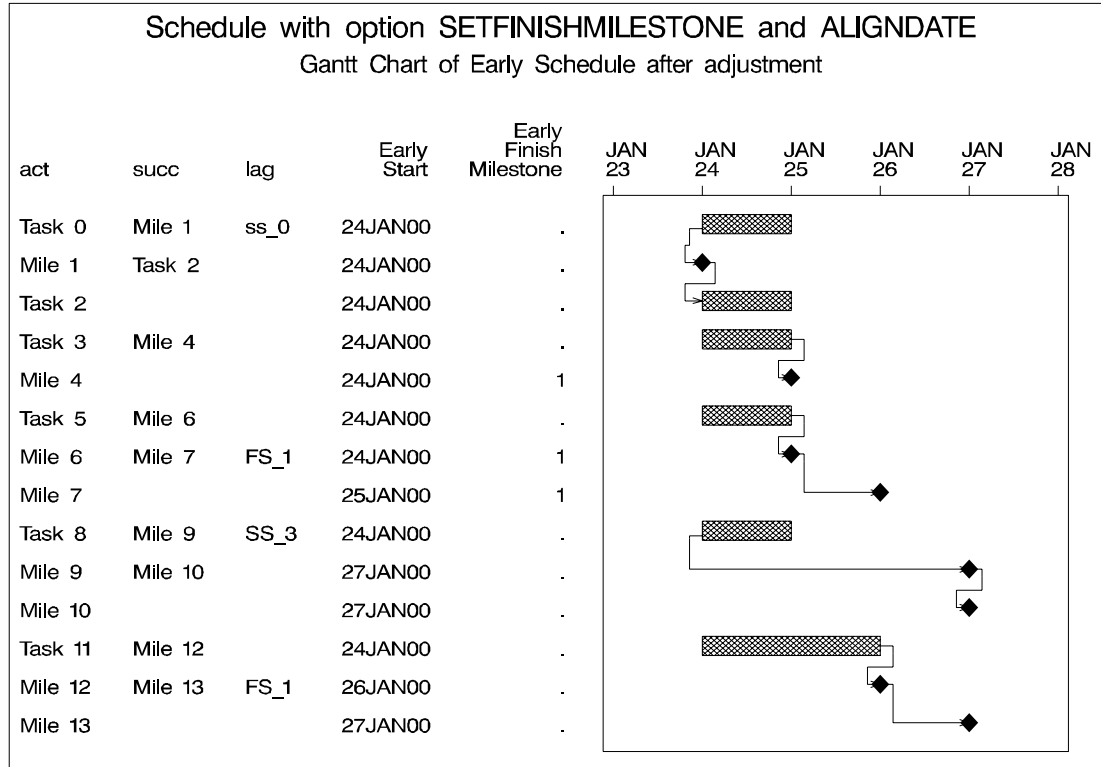
```

cprec=cyan cmile=magenta
caxis=black cframe=ligr;
dur=dur nojobnum nolegend;
id act succ lag e_start efinmile;
run;

```

**Output 2.1.5.** Gantt Chart of Unadjusted Schedule



**Output 2.1.6.** Gantt Chart of Adjusted Schedule

# Chapter 3

## The GANTT Procedure

### Chapter Table of Contents

---

|                                    |    |
|------------------------------------|----|
| <b>OVERVIEW</b> . . . . .          | 51 |
| <b>SYNTAX</b> . . . . .            | 51 |
| CHART Statement . . . . .          | 51 |
| <b>DETAILS</b> . . . . .           | 52 |
| Web-Enabled Gantt Charts . . . . . | 52 |





## Chapter 3

# The GANTT Procedure

---

### Overview

The following enhancements have been made to the Gantt procedure:

- The milestone color specifications have been made more flexible. The milestone colors are now determined using the same rules as for schedule bars, and consequently, they can also be set individually.
- The `WEB=` variable enables you to define an HTML reference for each activity. In the past the URL was only associated with the schedule bars for the activity. This association has now been extended to include milestones and ID variables.

---

### Syntax

The following options have been enhanced in PROC GANTT:

---

### CHART Statement

#### **Graphics Options**

**CMILE=***color*

specifies the color to use for drawing the milestone symbol on the chart. If the `CMILE=` option is not specified, the default color of the milestones follows the rules for coloring the bars of the relevant schedule. For example, the milestone depicting a critical activity is drawn with the color of the fill pattern used for critical activities. For an activity with slack, the early start and late start milestone are drawn with the color of the fill pattern used for the duration and the slack time of a noncritical activity, respectively. You can also control the color at the activity level by using a `PATTERN` variable.

**WEB=***variable*

**HTML=***variable*

specifies the character variable in the schedule data set that identifies an HTML page for each activity. The procedure generates an HTML image map using this information for all the schedule bars, milestones, and ID variables corresponding to an activity.

---

## Details

---

### Web-Enabled Gantt Charts

---

The WEB= variable enables you to define an HTML reference for each activity. This HTML reference is currently associated with all the schedule bars, milestones, and ID variables that correspond to the activity. The WEB= variable is a character variable, and the values need to be of the form “HREF=htmlpage.”

In addition, you can also store the coordinate and link information defined via the WEB= option in a SAS data set by specifying the IMAGEMAP= option in the PROC GANTT statement. By processing this SAS data set using a DATA step, you can generate customized HTML pages for your Gantt chart.

# Chapter 4

## The INTPOINT Procedure

### Chapter Table of Contents

---

|                                  |     |
|----------------------------------|-----|
| <b>OVERVIEW</b>                  | 55  |
| Mathematical Description of NPSC | 56  |
| Mathematical Description of LP   | 58  |
| The Interior Point Algorithm     | 58  |
| Network Models                   | 66  |
| <b>INTRODUCTION</b>              | 74  |
| Getting Started: NPSC Problems   | 74  |
| Getting Started: LP Problems     | 81  |
| Typical PROC INTPOINT Run        | 89  |
| <b>SYNTAX</b>                    | 91  |
| Functional Summary               | 93  |
| PROC INTPOINT Statement          | 96  |
| CAPACITY Statement               | 115 |
| COEF Statement                   | 116 |
| COLUMN Statement                 | 116 |
| COST Statement                   | 117 |
| DEMAND Statement                 | 117 |
| HEADNODE Statement               | 117 |
| ID Statement                     | 118 |
| LO Statement                     | 118 |
| NAME Statement                   | 119 |
| NODE Statement                   | 119 |
| QUIT Statement                   | 119 |
| RHS Statement                    | 119 |
| ROW Statement                    | 120 |
| RUN Statement                    | 120 |
| SUPDEM Statement                 | 120 |
| SUPPLY Statement                 | 121 |
| TAILNODE Statement               | 121 |
| TYPE Statement                   | 122 |
| VAR Statement                    | 124 |
| <b>DETAILS</b>                   | 125 |
| Input Data Sets                  | 125 |

|                                                                         |            |
|-------------------------------------------------------------------------|------------|
| Output Data Set . . . . .                                               | 135        |
| Case Sensitivity . . . . .                                              | 136        |
| Loop Arcs . . . . .                                                     | 137        |
| Multiple Arcs . . . . .                                                 | 137        |
| Flow and Value Bounds . . . . .                                         | 137        |
| Tightening Bounds and Side Constraints . . . . .                        | 138        |
| Reasons for Infeasibility . . . . .                                     | 138        |
| Missing S Supply and Missing D Demand Values . . . . .                  | 139        |
| Balancing Total Supply and Total Demand . . . . .                       | 144        |
| How to Make the Data Read of PROC INTPOINT More Efficient . . . . .     | 146        |
| Stopping Criteria . . . . .                                             | 151        |
| <b>EXAMPLES . . . . .</b>                                               | <b>155</b> |
| Example 4.1 Production, Inventory, Distribution Problem . . . . .       | 156        |
| Example 4.2 Altering Arc Data . . . . .                                 | 160        |
| Example 4.3 Adding Side Constraints . . . . .                           | 165        |
| Example 4.4 Using Constraints and More Alteration to Arc Data . . . . . | 170        |
| Example 4.5 Nonarc Variables in the Side Constraints . . . . .          | 174        |
| Example 4.6 Solving an LP Problem with Data in MPS Format . . . . .     | 179        |
| <b>REFERENCES . . . . .</b>                                             | <b>181</b> |

# Chapter 4

## The INTPOINT Procedure

---

### Overview

The new INTPOINT procedure solves the Network Program with Side Constraints (NPSC) problem (defined in the “Mathematical Description of NPSC” section on page 56) and the more general Linear Programming (LP) problem (defined in the “Mathematical Description of LP” section on page 58).

NPSC and LP models can be used to describe a wide variety of real-world applications ranging from production, inventory, and distribution problems to financial applications.

Whether your problem is NPSC or LP, PROC INTPOINT uses the same optimization algorithm, the Interior Point algorithm. This algorithm is outlined in the section “The Interior Point Algorithm” on page 58.

While many of your problems may best be formulated as LP problems, there may be other instances when your problems are better formulated as NPSC problems. The “Network Models” section on page 66 describes typical models that have a network component and suggests reasons why NPSC may be preferable to LP. The “Getting Started: NPSC Problems” section on page 74 outlines how you supply data of any NPSC problem to the PROC INTPOINT and call the procedure. After it reads the NPSC data, PROC INTPOINT converts the problem into an equivalent LP problem, performs Interior Point optimization, then converts the solution it finds back into a form you can use as the optimum to the original NPSC model.

If your model is an LP problem, the way you supply the data to PROC INTPOINT and run the procedure is described in the “Getting Started: LP Problems” section on page 81.

The remainder of this chapter is organized as follows:

- The “Typical PROC INTPOINT Run” section on page 89 describes how to use this procedure.
- The “Functional Summary” section on page 93 lists the statements and options that can be used to control PROC INTPOINT.
- The “Syntax” section on page 91 describes all the statements and options of PROC INTPOINT.
- The “Details” section on page 125 contains fuller explanations, descriptions, and advice on the use and behavior of the procedure.
- PROC INTPOINT is demonstrated by solving several examples in the “Examples” section on page 155.
- The “References” section on page 181 concludes the chapter.

---

## Mathematical Description of NPSC

A network consists of a collection of nodes joined by a collection of arcs. The arcs connect nodes and convey flow of one or more commodities that are supplied at supply nodes and demanded at demand nodes in the network. Each arc has a cost per unit of flow, a flow capacity, and a lower flow bound associated with it. An important concept in network modeling is *conservation of flow*. Conservation of flow means that the total flow in arcs directed toward a node, plus the supply at the node, minus the demand at the node, equals the total flow in arcs directed away from the node.

Often all the details of a problem cannot be specified in a network model alone. In many of these cases, these details can be represented by the addition of side constraints to the model. Side constraints are linear functions of arc variables (variables containing flow through an arc) and nonarc variables (variables that are not part of the network). The data for a side constraint consist of coefficients of arcs and coefficients of nonarc variables, a constraint type (that is,  $\leq$ ,  $=$ , or  $\geq$ ) and a right-hand-side value (rhs). A nonarc variable has a name, an objective function coefficient analogous to an arc cost, an upper bound analogous to an arc capacity, and a lower bound analogous to an arc lower flow bound.

If a network component of NPSC is removed by merging arcs and nonarc variables into a single set of variables, and if the flow conservation constraints and side constraints are merged into a single set of constraints, the result is an LP problem. PROC INTPOINT will automatically transform an NPSC problem into an equivalent LP problem, perform the optimization, then transform the problem back into its original form. By doing this, PROC INTPOINT finds the flow through the network and the values of any nonarc variables that minimize the total cost of the solution. Flow conservation is met, flow through each arc is on or between the arc's lower flow bound and capacity, the value of each nonarc variable is on or between the nonarc's lower and upper bounds, and the side constraints are satisfied.

Note that, since many LPs have large embedded networks, PROC INTPOINT is an attractive alternative to the LP procedure in many cases. Rather than formulating all problems as LPs, network models remain conceptually easy since they are based on network diagrams that represent the problem pictorially. PROC INTPOINT accepts the network specification in a format that is particularly suited to networks. This not only simplifies problem description but also aids in the interpretation of the solution. The conversion to and from the equivalent LP is done "behind the scenes" by the procedure.

If a network programming problem with side constraints has  $n$  nodes,  $a$  arcs,  $g$  nonarc variables, and  $k$  side constraints, then the formal statement of the problem solved by PROC INTPOINT is

$$\min\{c^T x + d^T z\}$$

$$\begin{aligned}
\text{subject to} \quad & Fx = b \\
& Hx + Qz \geq, =, \leq r \\
& l \leq x \leq u \\
& m \leq z \leq v
\end{aligned}$$

where

$c$  is the  $a \times 1$  objective function coefficient of the arc variables vector (the cost vector)

$x$  is the  $a \times 1$  arc variable value vector (the flow vector)

$d$  is the  $g \times 1$  objective function coefficient of the nonarc variables vector

$z$  is the  $g \times 1$  nonarc variable value vector

$F$  is the  $n \times a$  node-arc incidence matrix of the network, where

$$\begin{aligned}
F_{i,j} &= -1 && \text{if arc } j \text{ is directed from node } i \\
F_{i,j} &= 1 && \text{if arc } j \text{ is directed toward node } i \\
F_{i,j} &= 0 && \text{otherwise}
\end{aligned}$$

$b$  is the  $n \times 1$  node supply/demand vector, where

$$\begin{aligned}
b_i &= s && \text{if node } i \text{ has supply capability of } s \text{ units of flow} \\
b_i &= -d && \text{if node } i \text{ has demand } d \text{ of units of flow} \\
b_i &= 0 && \text{if node } i \text{ is a transshipment node}
\end{aligned}$$

$H$  is the  $k \times a$  side constraint coefficient matrix for arc variables, where  $H_{i,j}$  is the coefficient of arc  $j$  in the  $i$ th side constraint

$Q$  is the  $k \times g$  side constraint coefficient matrix for nonarc variables, where  $Q_{i,j}$  is the coefficient of nonarc  $j$  in the  $i$ th side constraint

$r$  is the  $k \times 1$  side constraint right-hand-side vector

$l$  is the  $a \times 1$  arc lower flow bound vector

$u$  is the  $a \times 1$  arc capacity vector

$m$  is the  $g \times 1$  nonarc variable value lower bound vector

$v$  is the  $g \times 1$  nonarc variable value upper bound vector

The INTPOINT procedure can also be used to solve an unconstrained network problem, that is, one in which  $H$ ,  $Q$ ,  $d$ ,  $r$ , and  $z$  do not exist.

The INTPOINT procedure can also be used to solve a network problem with side constraints but no nonarc variables, in which case  $Q$ ,  $d$ , and  $z$  do not exist.

---

## Mathematical Description of LP

PROC INTPOINT solves LP problems. These have a linear objective function and a collection of linear constraints. PROC INTPOINT finds the values of variables that minimize the total cost of the solution. The value of each variable is on or between the variable's lower and upper bounds, and the constraints are satisfied.

If an LP has  $g$  variables and  $k$  constraints, then the formal statement of the problem solved by PROC INTPOINT is

$$\begin{array}{ll} & \min\{d^T z\} \\ \text{subject to} & Qz \geq, =, \leq r \\ & m \leq z \leq v \end{array}$$

where

$d$  is the  $g \times 1$  objective function coefficient of the variables vector

$z$  is the  $g \times 1$  variable value vector

$Q$  is the  $k \times g$  constraint coefficient matrix for the variables, where  $Q_{i,j}$  is the coefficient of variable  $j$  in the  $i$ th constraint

$r$  is the  $k \times 1$  side constraint right-hand-side vector

$m$  is the  $g \times 1$  variable value lower bound vector

$v$  is the  $g \times 1$  variable value upper bound vector

---

## The Interior Point Algorithm

The Simplex algorithm, developed shortly after World War II, was for many years the main method used to solve Linear Programming problems. Over the last fifteen years, however, the Interior Point algorithm has been developed. This algorithm also solves Linear Programming problems. From the start it showed great theoretical promise, and considerable research in the area resulted in practical implementations that performed competitively with the Simplex algorithm. More recently, Interior Point algorithms have evolved to become superior to the Simplex algorithm, in general, especially when the problems are large.

There are many variations of Interior Point algorithms. PROC INTPOINT uses the Primal-Dual with Predictor-Corrector algorithm. More information on this particular algorithm and related theory can be found in the texts by Roos, Terlaky, and Vial (1997), Wright (1996), and Ye (1996).

### Interior Point Algorithmic Details

After preprocessing, the Linear Program to be solved is

$$\min\{c^T x\}$$



$$\begin{aligned} \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

This is the *primal* problem. The matrices of  $d$ ,  $z$ , and  $Q$  of NPSC have been renamed  $c$ ,  $x$ , and  $A$ , respectively, as these symbols are by convention used more, the problem to be solved is different from the original because of preprocessing, and there has been a change of primal variable to transform the LP into one whose variables have zero lower bounds. To simplify the algebra here, assume that variables have infinite upper bounds, and constraints are equalities. (Interior Point algorithms do efficiently handle finite upper bounds, and it is easy to introduce primal slack variables to change inequalities into equalities.) The problem has  $n$  variables.  $i$  is a variable number.  $k$  is an iteration number, and if used as a subscript or superscript it denotes “of iteration  $k$ ”.

There exists an equivalent problem, the *dual* problem, stated as

$$\begin{aligned} & \max \{b^T y\} \\ \text{subject to} \quad & A^T y + s = c \\ & s \geq 0 \\ \text{where} \quad & y \text{ are dual variables, and } s \text{ are dual constraint slacks} \end{aligned}$$

What the Interior Point has to do is to solve the system of equations to satisfy the Karush-Kuhn-Tucker (KKT) conditions for optimality:

$$\begin{aligned} & Ax = b \\ & A^T y + s = c \\ & XSe = 0 \\ & x \geq 0 \\ & s \geq 0 \\ \text{where} \quad & S = \text{diag}(s), \text{ (that is, } S_{i,j} = s_i \text{ if } i = j, S_{i,j} = 0 \text{ otherwise)} \\ & X = \text{diag}(x), \text{ and} \\ & e_i = 1 \forall i \end{aligned}$$

These are the conditions for feasibility, with the *complementarity* 112

condition  $XSe = 0$  added.  $c^T x = b^T y$  must occur at the optimum. Complementarity forces the optimal objectives of the primal and dual to be equal,  $c^T x_{opt} = b^T y_{opt}$ , as

$$\begin{aligned} 0 &= x_{opt}^T s_{opt} = s_{opt}^T x_{opt} = (c - A^T y_{opt})^T x_{opt} = \\ &= c^T x_{opt} - y_{opt}^T (Ax_{opt}) = c^T x_{opt} - b^T y_{opt} \end{aligned}$$

therefore  $0 = c^T x_{opt} - b^T y_{opt}$

Before the optimum is reached, a solution  $(x, y, s)$  may not satisfy the KKT conditions:

- Primal constraints can be broken,  $infeas_c = b - Ax \neq 0$ .
- Dual constraints can be broken,  $infeas_d = c - A^T y - s \neq 0$ .
- Complementarity is unsatisfied,  $x^T s = c^T x - b^T y \neq 0$ . This is called the *duality gap*.

The Interior Point algorithm works by using Newton's method to find a direction to move  $(\Delta x^k, \Delta y^k, \Delta s^k)$  from the current solution  $(x^k, y^k, s^k)$  toward a better solution:

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

$\alpha$  is the *step length* and is assigned a value as large as possible and not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  is “too close” to zero. The direction in which to move is found using

$$\begin{aligned} A\Delta x^k &= infeas_c \\ A^T \Delta y^k + \Delta s^k &= infeas_d \\ S^k \Delta x^k + X^k \Delta s^k &= -X^k S^k e \end{aligned}$$

To greatly improve performance, the third equation is changed to

$$S^k \Delta x^k + X^k \Delta s^k = -X^k S^k e + \sigma_k \mu_k e$$

where

$$\mu_k = (x^k)^T s^k / n, \text{ the average complementarity, and}$$

$$0 \leq \sigma_k \leq 1$$

The effect now is to find a direction in which to move to reduce infeasibilities and to reduce the complementarity toward zero, but if any  $x_i^k s_i^k$  is too close to zero, it is “nudged out” to  $\mu$ , and any  $x_i^k s_i^k$  that is larger than  $\mu$  is “nudged into”  $\mu$ . A  $\sigma_k$  close to or equal to 0.0 biases a direction toward the optimum, and a value for  $\sigma_k$  close to or equal to 1.0 “centers” the direction toward a point where all pairwise products  $x_i^k s_i^k = \mu$ . Such points make up the *Central Path* in the interior. Although centering directions make little, if any, progress in reducing  $\mu$  and moving the solution closer

to the optimum, substantial progress toward the optimum can usually be made in the next iteration.

The Central Path is crucial to why the Interior Point algorithm is so efficient. As  $\mu$  is decreased, this path “guides” the algorithm to the optimum through the interior of feasible space. Without centering, the algorithm would find a series of solutions near each other close to the boundary of feasible space. Step lengths along the direction would be small and many more iterations would probably be required to reach the optimum.

That in a nutshell is the Primal-Dual Interior Point algorithm. Varieties of the algorithm differ in the way  $\alpha$  and  $\sigma_k$  are chosen and the direction adjusted during each iteration. A wealth of information can be found in the texts by Roos, Terlaky, and Vial (1997), Wright (1996), and Ye (1996).

The calculation of the direction is the most time-consuming step of the Interior Point algorithm. Assume the  $k$ th iteration is being performed, so the subscript and superscript  $k$  can be dropped from the algebra:

$$\begin{aligned} A\Delta x &= \text{infeas}_c \\ A^T \Delta y + \Delta s &= \text{infeas}_d \\ S\Delta x + X\Delta s &= -XSe + \sigma\mu e \end{aligned}$$

Rearranging the second equation

$$\Delta s = \text{infeas}_d - A^T \Delta y$$

Rearranging the third equation

$$\begin{aligned} \Delta s &= X^{-1}(-S\Delta x - XSe + \sigma\mu e) \\ \Delta s &= -\Theta\Delta x - Se + X^{-1}\sigma\mu e \end{aligned}$$

where

$$\Theta = SX^{-1}$$

Equating these two expressions for  $\Delta s$  and rearranging

$$\begin{aligned} -\Theta\Delta x - Se + X^{-1}\sigma\mu e &= \text{infeas}_d - A^T \Delta y \\ -\Theta\Delta x &= Se - X^{-1}\sigma\mu e + \text{infeas}_d - A^T \Delta y \\ \Delta x &= \Theta^{-1}(-Se + X^{-1}\sigma\mu e - \text{infeas}_d + A^T \Delta y) \\ \Delta x &= \rho + \Theta^{-1}A^T \Delta y \end{aligned}$$

where

$$\rho = \Theta^{-1}(-Se + X^{-1}\sigma\mu e - \text{infeas}_d)$$

Substituting into the first direction equation

$$\begin{aligned} A\Delta x &= \text{infeas}_c \\ A(\rho + \Theta^{-1}A^T \Delta y) &= \text{infeas}_c \end{aligned}$$

$$A\Theta^{-1}A^T\Delta y = \text{infeas}_c - A\rho$$

$$\Delta y = (A\Theta^{-1}A^T)^{-1}(\text{infeas}_c - A\rho)$$

$\Theta$ ,  $\rho$ ,  $\Delta y$ ,  $\Delta x$ , and  $\Delta s$  are calculated in that order. The hardest term is the factorization of the  $(A\Theta^{-1}A^T)$  matrix to determine  $\Delta y$ . Fortunately, although the *values* of  $(A\Theta^{-1}A^T)$  are different for each iteration, the *locations* of the nonzeros in this matrix remain fixed; the nonzero locations are the same as those in the matrix  $(AA^T)$ . This is because  $\Theta^{-1} = XS^{-1}$  is a diagonal matrix that has the effect of merely scaling the columns of  $(AA^T)$ .

The fact that the nonzeros in  $A\Theta^{-1}A^T$  have a constant pattern is exploited by all Interior Point algorithms and is a major reason for their excellent performance. Before iterations begin,  $AA^T$  is examined and its rows and columns are symmetrically permuted so that during Cholesky factorization, the number of *fillins* created is smaller. A list of arithmetic operations to perform the factorization is saved in concise computer data structures (working with memory locations rather than actual numerical values). This is called *symbolic factorization*. During iterations, when memory has been initialized with numerical values, the operations list is performed sequentially. Determining how the factorization should be performed again and again is unnecessary.

### The Primal-Dual Predictor-Corrector Interior Point Algorithm

The variant of the Interior Point algorithm implemented in PROC INTPOINT is a Primal-Dual Predictor-Corrector Interior Point algorithm. At first, Newton's method is used to find a direction  $(\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k)$  to move, but calculated as if  $\mu$  is zero, that is, as a step with no centering, known as an *affine* step:

$$A\Delta x_{aff}^k = \text{infeas}_c$$

$$A^T\Delta y_{aff}^k + \Delta s_{aff}^k = \text{infeas}_d$$

$$S^k\Delta x_{aff}^k + X^k\Delta s_{aff}^k = -X^kS^ke$$

$$(x_{aff}^k, y_{aff}^k, s_{aff}^k) = (x^k, y^k, s^k) + \alpha(\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k)$$

$\alpha$  is the *step length* as before.

Complementarity  $x^Ts$  is calculated at  $(x_{aff}^k, y_{aff}^k, s_{aff}^k)$  and compared with the complementarity at the starting point  $(x^k, y^k, s^k)$ , and the success of the affine step is gauged. If the affine step was successful in reducing the complementarity by a substantial amount, the need for centering is not great, and  $\sigma_k$  in the following linear system is assigned a value close to zero. If, however, the affine step was unsuccessful, centering would be beneficial, and  $\sigma_k$  in the following linear system is assigned a value closer to 1.0. The value of  $\sigma_k$  is therefore adaptively altered depending on the progress made toward the optimum.

A second linear system is solved to determine a centering vector  $(\Delta x_c^k, \Delta y_c^k, \Delta s_c^k)$  from  $(x_{aff}^k, y_{aff}^k, s_{aff}^k)$ :

$$A\Delta x_c^k = 0$$

$$\begin{aligned}
A^T \Delta y_c^k + \Delta s_c^k &= 0 \\
S^k \Delta x_c^k + X^k \Delta s_c^k &= -X_{aff}^k S_{aff}^k e + \sigma_k \mu_k e
\end{aligned}$$

then

$$\begin{aligned}
(\Delta x^k, \Delta y^k, \Delta s^k) &= (\Delta x_{aff}^k, \Delta y_{aff}^k, \Delta s_{aff}^k) + (\Delta x_c^k, \Delta y_c^k, \Delta s_c^k) \\
(x^{k+1}, y^{k+1}, s^{k+1}) &= (x^k, y^k, s^k) + \alpha (\Delta x^k, \Delta y^k, \Delta s^k)
\end{aligned}$$

where, as before,  $\alpha$  is the *step length* assigned a value as large as possible but not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  is “too close” to zero.

Although the Predictor-Corrector variant entails solving two linear systems instead of one, fewer iterations are usually required to reach the optimum. The additional overhead of calculating the second linear system is small, as the factorization of the  $(A\Theta^{-1}A^T)$  matrix has already been performed to solve the first linear system.

### Interior Point: Upper Bounds

If the LP had upper bounds ( $0 \leq x \leq u$  where  $u$  is the upper bound vector), then the primal and dual problems, the duality gap, and the KKT conditions would have to be expanded.

The primal Linear Program to be solved is

$$\begin{aligned}
&\min \{c^T x\} \\
\text{subject to} \quad &Ax = b \\
&0 \leq x \leq u
\end{aligned}$$

$0 \leq x \leq u$  is split into  $x \geq 0$  and  $x \leq u$ . Let  $z$  be primal slack so that  $x + z = u$ , and associate dual variables  $w$  with these constraints. The Interior Point solves the system of equations to satisfy the Karush-Kuhn-Tucker (KKT) conditions for optimality:

$$\begin{aligned}
Ax &= b \\
x + z &= u \\
A^T y + s - w &= c \\
XSe &= 0 \\
ZWe &= 0 \\
x, s, z, w &\geq 0
\end{aligned}$$

These are the conditions for feasibility, with the *complementarity* conditions  $XSe = 0$  and  $ZWe = 0$  added.  $c^T x = b^T y - u^T w$  must occur at the optimum. Complementarity forces the optimal objectives of the primal and dual to be equal,  $c^T x_{opt} = b^T y_{opt} - u^T w_{opt}$ , as

$$0 = z_{opt}^T w_{opt} = (u - x_{opt})^T w_{opt} = u^T w_{opt} - x_{opt}^T w_{opt}$$

$$0 = x_{opt}^T s_{opt} = s_{opt}^T x_{opt} = (c - A^T y_{opt} + w_{opt})^T x_{opt} =$$

$$c^T x_{opt} - y_{opt}^T (Ax_{opt}) + w_{opt}^T x_{opt} = c^T x_{opt} - b^T y_{opt} + u^T w_{opt}$$

therefore  $0 = c^T x_{opt} - b^T y_{opt} + u^T w_{opt}$

Before the optimum is reached, a solution  $(x, y, s, z, w)$  might not satisfy the KKT conditions:

- Primal bound constraints can be broken,  $infeas_b = u - x - z \neq 0$ .
- Primal constraints can be broken,  $infeas_c = b - Ax \neq 0$ .
- Dual constraints can be broken,  $infeas_d = c - A^T y - s + w \neq 0$ .
- Complementarity conditions are unsatisfied,  $x^T s \neq 0$  and  $z^T w \neq 0$ .

The calculations of the Interior point algorithm can easily be derived in a fashion similar to calculations for when an LP has no upper bounds. See the paper by Lustig, Marsten, and Shanno (1992).

In some iteration  $k$ , the *affine* step system that must be solved is:

$$\begin{aligned}\Delta x_{aff} + \Delta z_{aff} &= infeas_b \\ A\Delta x_{aff} &= infeas_c \\ A^T \Delta y_{aff} + \Delta s_{aff} - \Delta w_{aff} &= infeas_d \\ S\Delta x_{aff} + X\Delta s_{aff} &= -XSe \\ Z\Delta w_{aff} + W\Delta z_{aff} &= -ZWe\end{aligned}$$

Therefore, the computations involved in solving the affine step are:

$$\begin{aligned}\Theta &= SX^{-1} + WZ^{-1} \\ \rho &= \Theta^{-1}(infeas_d + (S - W)e - Z^{-1}Winfeas_b) \\ \Delta y_{aff} &= (A\Theta^{-1}A^T)^{-1}(infeas_c + A\rho) \\ \Delta x_{aff} &= \Theta^{-1}A^T \Delta y_{aff} - \rho \\ \Delta z_{aff} &= infeas_b - \Delta x_{aff} \\ \Delta w_{aff} &= -We - Z^{-1}W\Delta z_{aff} \\ \Delta s_{aff} &= -Se - X^{-1}S\Delta x_{aff} \\ (x_{aff}, y_{aff}, s_{aff}, z_{aff}, w_{aff}) &= (x, y, s, z, w) + \\ &\alpha(\Delta x_{aff}, \Delta y_{aff}, \Delta s_{aff}, \Delta z_{aff}, \Delta w_{aff})\end{aligned}$$

$\alpha$  is the *step length* as before.

A second linear system is solved to determine a centering vector:  $(\Delta x_c, \Delta y_c, \Delta s_c, \Delta z_c, \Delta w_c)$  from  $(x_{aff}, y_{aff}, s_{aff}, z_{aff}, w_{aff})$ :

$$\begin{aligned}\Delta x_c + \Delta z_c &= 0 \\ A\Delta x_c &= 0 \\ A^T \Delta y_c + \Delta s_c - \Delta w_c &= 0 \\ S\Delta x_c + X\Delta s_c &= -X_{aff}S_{aff}e + \sigma\mu e \\ Z\Delta w_c + W\Delta z_c &= -Z_{aff}W_{aff}e + \sigma\mu e\end{aligned}$$

where

$$\begin{aligned}\zeta_{start} &= x^T s + z^T w, \text{ complementarity at the start of the iteration} \\ \zeta_{aff} &= x_{aff}^T s_{aff} + z_{aff}^T w_{aff}, \text{ the affine complementarity} \\ \mu &= \zeta_{aff}/2n, \text{ the average complementarity} \\ \sigma &= (\zeta_{aff}/\zeta_{start})^3\end{aligned}$$

Therefore, the computations involved in solving the centering step are:

$$\begin{aligned}\rho &= \Theta^{-1}(\sigma\mu(X^{-1} - Z^{-1})e - X^{-1}X_{aff}S_{aff}e + Z^{-1}Z_{aff}W_{aff}e) \\ \Delta y_c &= (A\Theta^{-1}A^T)^{-1}A\rho \\ \Delta x_c &= \Theta^{-1}A^T\Delta y_c - \rho \\ \Delta z_c &= -\Delta x_c \\ \Delta w_c &= \sigma\mu Z^{-1}e - Z^{-1}Z_{aff}W_{aff}e - Z^{-1}W_{aff}\Delta z_c \\ \Delta s_c &= \sigma\mu X^{-1}e - X^{-1}X_{aff}S_{aff}e - X^{-1}S_{aff}\Delta x_c\end{aligned}$$

Then

$$\begin{aligned}(\Delta x, \Delta y, \Delta s, \Delta z, \Delta w) &= \\ (\Delta x_{aff}, \Delta y_{aff}, \Delta s_{aff}, \Delta z_{aff}, \Delta w_{aff}) &+ \\ (\Delta x_c, \Delta y_c, \Delta s_c, \Delta z_c, \Delta w_c) & \\ (x^{k+1}, y^{k+1}, s^{k+1}, z^{k+1}, w^{k+1}) &= \\ (x^k, y^k, s^k, z^k, w^k) &+ \\ \alpha(\Delta x, \Delta y, \Delta s, \Delta z, \Delta w) &\end{aligned}$$

where, as before,  $\alpha$  is the *step length* assigned a value as large as possible but not so large that a  $x_i^{k+1}$ ,  $s_i^{k+1}$ ,  $z_i^{k+1}$ , or  $w_i^{k+1}$  is “too close” to zero.

The algebra in this section has been simplified by assuming that *all* variables have finite upper bounds. If the number of variables with finite upper bounds  $n_u < n$ , you need to change the algebra to reflect that the “z” and “w” arrays has dimension  $n_u \times 1$  or  $n_u \times n_u$ . Other computations need slight modification. For example, the average complementarity

$$\mu = x_{aff}^T s_{aff} / n + z_{aff}^T w_{aff} / n_u$$

An important point is that any upper bounds can be handled by specializing the algorithm and *not* by generating the constraints  $x \leq u$  and adding these to the main primal constraints  $Ax = b$ .

## Network Models

The following are descriptions of some typical NPSC models.

### **Production, Inventory, and Distribution (Supply Chain) Problems**

One common class of network model is the production-inventory-distribution or supply-chain problem. The diagram in Figure 4.1 illustrates this problem. The subscript on the Production, Inventory, and Sales nodes indicates the time period. By replicating sections of the model, the notion of time can be included.

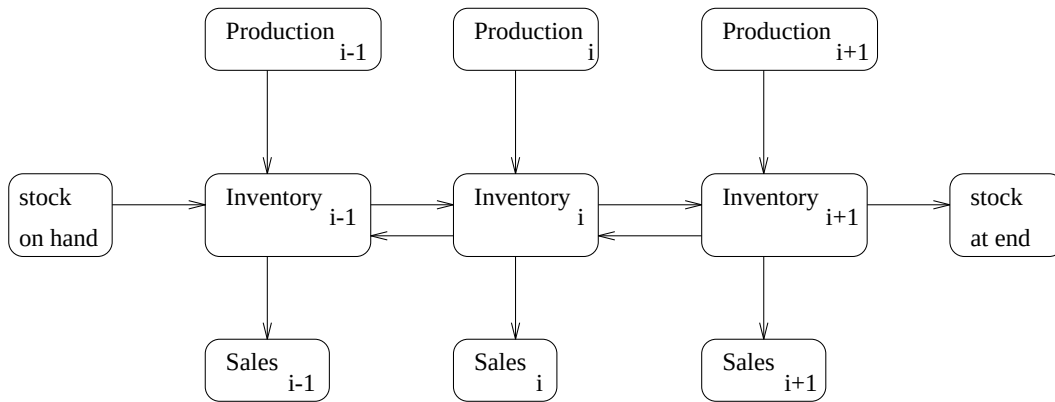


**Figure 4.1.** Production-Inventory-Distribution Problem

In this type of model, the nodes can represent a wide variety of facilities. Several examples are suppliers, spot markets, importers, farmers, manufacturers, factories, parts of a plant, production lines, waste disposal facilities, workstations, warehouses, coolstores, depots, wholesalers, export markets, ports, rail junctions, airports, road intersections, cities, regions, shops, customers, and consumers. The diversity of this selection demonstrates how rich the potential applications of this model are.

Depending upon the interpretation of the nodes, the objectives of the modeling exercise can vary widely. Some common types of objectives are

- to reduce collection or purchase costs of raw materials
- to reduce inventory holding or backorder costs. Warehouses and other storage facilities sometimes have capacities, and there can be limits on the amount of goods that can be placed on backorder.
- to decide where facilities should be located and what the capacity of these should be. Network models have been used to help decide where factories, hospitals, ambulance and fire stations, oil and water wells, and schools should be sited.



- to determine the assignment of resources (machines, production capability, workforce) to tasks, schedules, classes, or files
- to determine the optimal distribution of goods or services. This usually means minimizing transportation costs and reducing transit time or distances covered.
- to find the shortest path from one location to another
- to ensure that demands (for example, production requirements, market demands, contractual obligations) are met
- to maximize profits from the sale of products or the charge for services
- to maximize production by identifying bottlenecks

Some specific applications are

- car distribution models. These help determine which models and numbers of cars should be manufactured in which factories and where to distribute cars from these factories to zones in the United States in order to meet customer demand at least cost.
- models in the timber industry. These help determine when to plant and mill forests, schedule production of pulp, paper, and wood products, and distribute products for sale or export.
- military applications. The nodes can be theaters, bases, ammunition dumps, logistical suppliers, or radar installations. Some models are used to find the best ways to mobilize personnel and supplies and to evacuate the wounded in the least amount of time.
- communications applications. The nodes can be telephone exchanges, transmission lines, satellite links, and consumers. In a model of an electrical grid, the nodes can be transformers, powerstations, watersheds, reservoirs, dams, and consumers. The effect of high loads or outages might be of concern.

### ***Proportional Constraints***

In many models, you have the characteristic that a flow through an arc must be proportional to the flow through another arc. Side constraints are often necessary to model that situation. Such constraints are called *proportional constraints* and are useful in models where production is subject to refining or modification into different materials. The amount of each output, or any waste, evaporation, or reduction can be specified as a proportion of input.

Typically, the arcs near the supply nodes carry raw materials and the arcs near the demand nodes carry refined products. For example, in a model of the milling industry, the flow through some arcs may represent quantities of wheat. After the wheat is processed, the flow through other arcs might be flour. For others it might be bran. The side constraints model the relationship between the amount of flour or bran produced as a proportion of the amount of wheat milled. Some of the wheat can end up as neither flour, bran, nor any useful product, so this waste is drained away via arcs to a waste node.

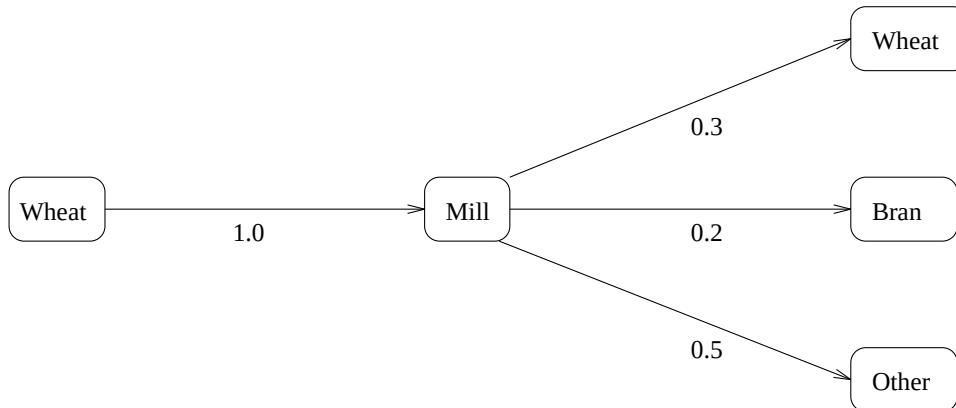


**Figure 4.2.** Proportional Constraints

In order for arcs to be specified in side constraints, they must be named. By default, PROC INTPOINT names arcs using the names of the nodes at the head and tail of the arc. An arc is named with its tail node name followed by an “\_” followed by the name of its head node name. For example, an arc from node *from* to node *to* is called *from\_to*.

Consider the network fragment in Figure 4.2. The arc *Wheat\_Mill* conveys the wheat milled. The cost of flow on this arc is the milling cost. The capacity of this arc is the capacity of the mill. The lower flow bound on this arc is the minimum quantity that must be milled for the mill to operate economically. The constraints

$$\begin{aligned} 0.3 \text{ Wheat\_Mill} - \text{Mill\_Flour} &= 0.0 \\ 0.2 \text{ Wheat\_Mill} - \text{Mill\_Bran} &= 0.0 \end{aligned}$$

force every unit of wheat that is milled to produce 0.3 units of flour and 0.2 units of bran. Note that it is not necessary to specify the constraint

$$0.5 \text{ Wheat\_Mill} - \text{Mill\_Other} = 0.0$$

since flow conservation implies that any flow that does not traverse through *Mill\_Flour* or *Mill\_Bran* must be conveyed through *Mill\_Other*. And, computationally, it is better if this constraint is not specified, since there is one less side constraint and fewer problems with numerical precision. Notice that the sum of the proportions must equal 1.0 exactly; otherwise, flow conservation is violated.

### Blending Constraints

Blending or quality constraints can also influence the recipes or proportions of ingredients that are mixed. For example, different raw materials can have different properties. In an application of the oil industry, the amount of products that are obtained could be different for each type of crude oil. Furthermore, fuel might have a minimum octane requirement or limited sulphur or lead content, so that a blending of crudes is needed to produce the product.

The network fragment in Figure 4.3 shows an example of this.

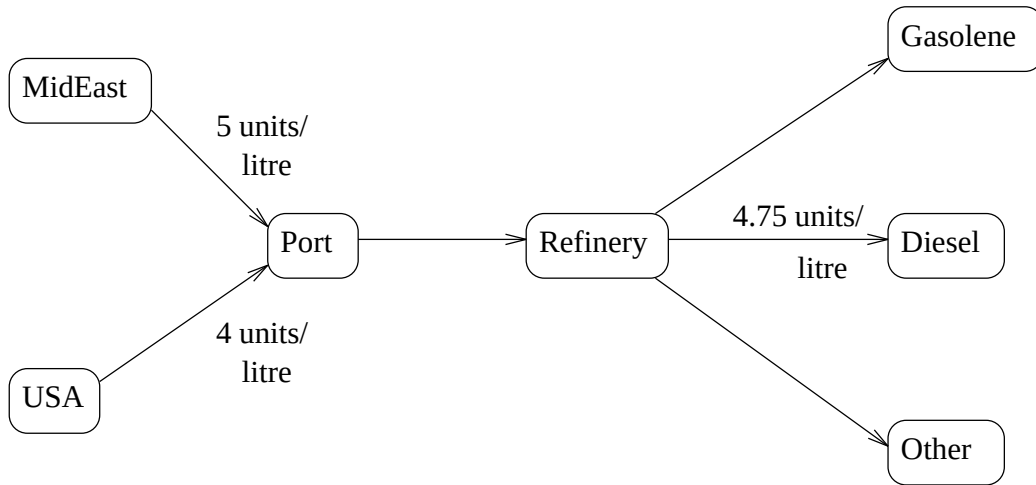


**Figure 4.3.** Blending Constraints

The arcs `MidEast_Port` and `USA_Port` convey crude oil from the two sources. The arc `Port_Refinery` represents refining while the arcs `Refinery_Gasoline` and `Refinery_Diesel` carry the gas and diesel produced. The proportional constraints

$$\begin{aligned} 0.4 \text{ Port\_Refinery} - \text{Refinery\_Gasoline} &= 0.0 \\ 0.2 \text{ Port\_Refinery} - \text{Refinery\_Diesel} &= 0.0 \end{aligned}$$

capture the restrictions for producing gasoline and diesel from crude. Suppose that only crude from the Middle East is used, then the resulting diesel would contain 5 units of sulphur per litre. If only crude from the U.S.A. is used, the resulting diesel would contain 4 units of sulphur per litre. Diesel can have at most 4.75 units of sulphur per litre. Some crude from the U.S.A. must be used if Middle East crude is used in order to meet the 4.75 sulphur per litre limit. The side constraint to model this requirement is

$$5 \text{ MidEast\_Port} + 4 \text{ USA\_Port} - 4.75 \text{ Port\_Refinery} \leq 0.0$$

Since  $\text{Port\_Refinery} = \text{MidEast\_Port} + \text{USA\_Port}$ , flow conservation allows this constraint to be simplified to

$$1 \text{ MidEast\_Port} - 3 \text{ USA\_Port} \leq 0.0$$

If, for example, 120 units of crude from the Middle East is used, then at least 40 units of crude from the U.S.A. must be used. The preceding constraint is simplified

because you assume that the sulphur concentration of diesel is proportional to the sulphur concentration of the crude mix. If this is not the case, the relation

$$0.2 \text{ Port\_Refinery} = \text{Refinery\_Diesel}$$

is used to obtain

$$5 \text{ MidEast\_Port} + 4 \text{ USA\_Port} - 4.75 (1.0/0.2 \text{ Refinery\_Diesel}) \leq 0.0$$

which equals

$$5 \text{ MidEast\_Port} + 4 \text{ USA\_Port} - 23.75 \text{ Refinery\_Diesel} \leq 0.0$$

An example similar to this oil industry problem is solved in the “Introductory NPSC Example” section on page 75.

### Multicommodity Problems

Side constraints are also used in models in which there are capacities on transportation or some other shared resource, or there are limits on overall production or demand in multicommodity, multidivisional, or multiperiod problems. Each commodity, division, or period can have a separate network coupled to one main system by the side constraints. Side constraints are used to combine the outputs of subdivisions of a problem (either commodities, outputs in distinct time periods, or different process streams) to meet overall demands or to limit overall production or expenditures. This method is more desirable than doing separate *local* optimizations for individual commodity, process, or time networks and then trying to establish relationships between each when determining an overall policy if the *global* constraint is not satisfied. Of course, to make models more realistic, side constraints may be necessary in the local problems.

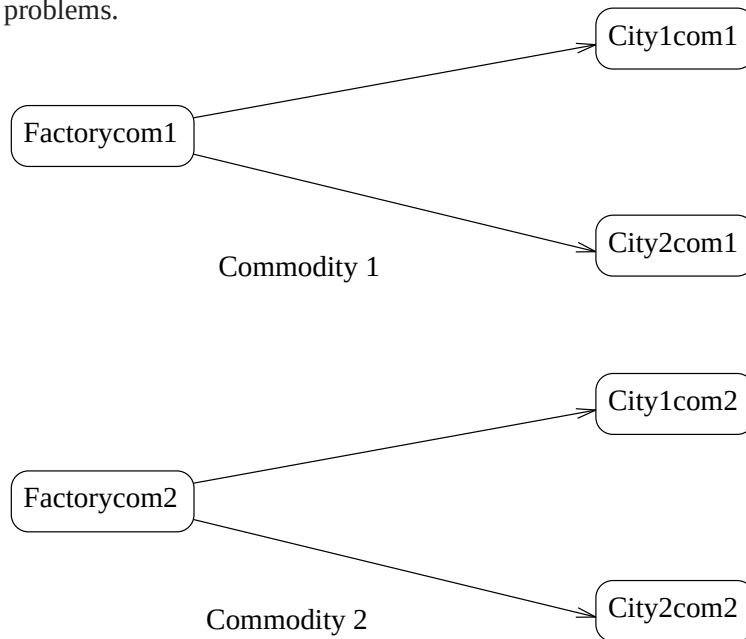


**Figure 4.4.** Multicommodity Problem

Figure 4.4 shows two network fragments. They represent identical production and distribution sites but of two different commodities. Suffix *com1* represents commodity 1 and suffix *com2* represents commodity 2. The nodes *Factorycom1* and *Factorycom2* model the same factory, and nodes *City1com1* and *City1com2* model the same location, city1. Similarly, *City2com1* and *City2com2* are the same location, city2. Suppose that commodity 1 occupies 2 cubic meters, commodity 2 occupies 3 cubic meters, the truck dispatched to city1 has a capacity of 200 cubic meters, and the truck dispatched to city2 has a capacity of 250 cubic meters. How much of each commodity can be loaded onto each truck? The side constraints for this case are

$$\begin{aligned} 2 \text{ Factorycom1\_City1com1} + 3 \text{ Factorycom2\_City1com2} &\leq 200 \\ 2 \text{ Factorycom1\_City2com1} + 3 \text{ Factorycom2\_City2com2} &\leq 250 \end{aligned}$$

### Large Modeling Strategy

In many cases, the flow through an arc might actually represent the flow or movement of a commodity from place to place or from time period to time period. However, sometimes an arc is included in the network as a method of capturing some aspect of the problem that you would not normally think of as part of a network model. There is no commodity movement associated with that arc. For example, in a multiprocess, multiproduct model (Figure 4.5), there might be subnetworks for each process and each product. The subnetworks can be joined together by a set of arcs that have flows that represent the amount of product *j* produced by process *i*. To model an upper-limit constraint on the total amount of product *j* that can be produced, direct all arcs carrying product *j* to a single node and from there through a single arc. The capacity of this arc is the upper limit of product *j* production. It is preferable to model this structure in the network rather than to include it in the side constraints because the efficiency of the optimizer may be less affected by a reasonable increase in the size of the network rather than increasing the number or complicating side constraints.

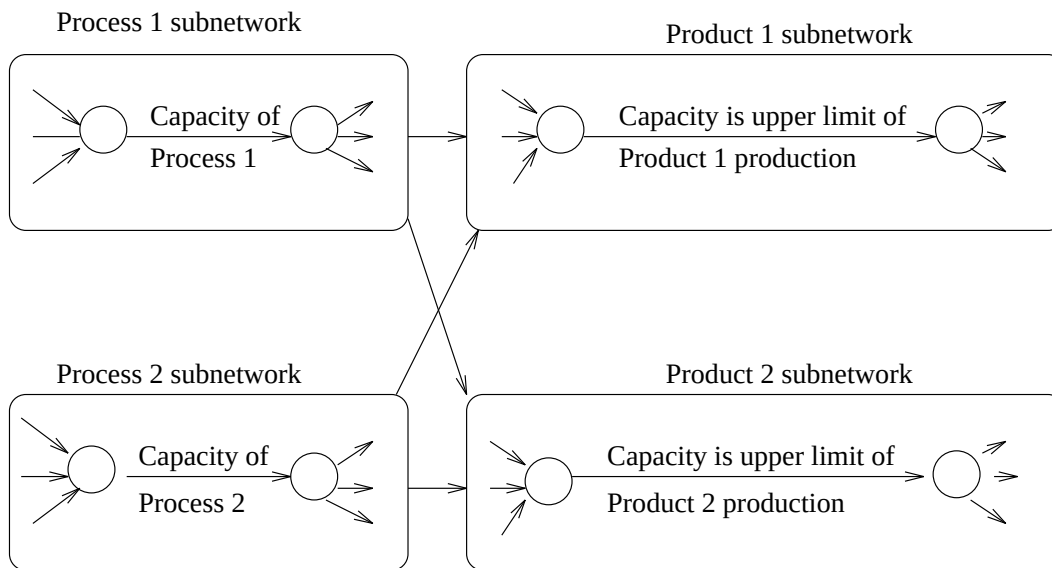


**Figure 4.5.** Multiprocess, Multiproduct Example

When starting a project, it is often a good strategy to use a small network formulation and then to use that model as a framework upon which to add detail. For example, in the multiprocess multiproduct model, you might start with the network depicted in Figure 4.5. Then, for example, the process subnetwork can be enhanced to include the distribution of products. Other phases of the operation could be included by adding more subnetworks. Initially, these subnetworks can be single nodes, but in subsequent studies they can be expanded to include greater detail.

### **Advantages of Network Models over LP Models**

Many linear programming problems have large embedded network structures. Such problems often result when modeling manufacturing processes, transportation or distribution networks, or resource allocation, or when deciding where to locate facilities. Often, some commodity is to be moved from place to place, so the more natural formulation in many applications is that of a constrained network rather than a linear program.

Using a network diagram to visualize a problem makes it possible to capture the important relationships in an easily understood picture form. The network diagram aids the communication between model builder and model user, making it easier to comprehend how the model is structured, how it can be changed, and how results can be interpreted.

If a network structure is embedded in a linear program, the problem is an NPSC (see the “Mathematical Description of NPSC” section on page 56). When the network part of the problem is large compared to the non-network part, especially if the number of side constraints is small, it is worthwhile to exploit this structure to describe the model. Rather than generating the data for the flow conservation constraints, generate instead the data for the nodes and arcs of the network.

### **Flow Conservation Constraints**

The constraints  $Fx = b$  in NPSC (see the “Mathematical Description of NPSC” section on page 56) are referred to as the nodal flow conservation constraints. These constraints algebraically state that the sum of the flow through arcs directed toward a node plus that node’s supply, if any, equals the sum of the flow through arcs directed away from that node plus that node’s demand, if any. The flow conservation constraints are implicit in the network model and should not be specified explicitly in side constraint data when using PROC INTPOINT to solve NPSC problems.

### **Nonarc Variables**

Nonarc variables can be used to simplify side constraints. For example, if a sum of flows appears in many constraints, it may be worthwhile to equate this expression with a nonarc variable and use this in the other constraints. This keeps the constraint coefficient matrix sparse. By assigning a nonarc variable a nonzero objective function, it is then possible to incur a cost for using resources above some lowest feasible limit. Similarly, a profit (a negative objective function coefficient value) can be made if all available resources are not used.

In some models, nonarc variables are used in constraints to absorb excess resources or supply needed resources. Then, either the excess resource can be used or the needed resource can be supplied to another component of the model.

For example, consider a multicommodity problem of making television sets that have either 19- or 25-inch screens. In their manufacture, three and four chips, respectively, are used. Production occurs at two factories during March and April. The supplier of chips can supply only 2,600 chips to factory 1 and 3,750 chips to factory 2 each month. The names of arcs are in the form  $\text{Prod}n\_s\_m$ , where  $n$  is the factory number,  $s$  is the screen size, and  $m$  is the month. For example,  $\text{Prod1\_25\_Apr}$  is the arc that conveys the number of 25-inch TVs produced in factory 1 during April. You might have to determine similar systematic naming schemes for your application.

As described, the constraints are

$$\begin{aligned} 3 \text{ Prod1\_19\_Mar} + 4 \text{ Prod1\_25\_Mar} &\leq 2600 \\ 3 \text{ Prod2\_19\_Mar} + 4 \text{ Prod2\_25\_Mar} &\leq 3750 \\ 3 \text{ Prod1\_19\_Apr} + 4 \text{ Prod1\_25\_Apr} &\leq 2600 \\ 3 \text{ Prod2\_19\_Apr} + 4 \text{ Prod2\_25\_Apr} &\leq 3750 \end{aligned}$$

If there are chips that could be obtained for use in March but not used for production in March, why not keep these unused chips until April? Furthermore, if the March excess chips at factory 1 could be used either at factory 1 or factory 2 in April, the model becomes

$$\begin{aligned} 3 \text{ Prod1\_19\_Mar} + 4 \text{ Prod1\_25\_Mar} + \text{F1\_Unused\_Mar} &= 2600 \\ 3 \text{ Prod2\_19\_Mar} + 4 \text{ Prod2\_25\_Mar} + \text{F2\_Unused\_Mar} &= 3750 \\ 3 \text{ Prod1\_19\_Apr} + 4 \text{ Prod1\_25\_Apr} - \text{F1\_Kept\_Since\_Mar} &= 2600 \\ 3 \text{ Prod2\_19\_Apr} + 4 \text{ Prod2\_25\_Apr} - \text{F2\_Kept\_Since\_Mar} &= 3750 \\ \text{F1\_Unused\_Mar} + \text{F2\_Unused\_Mar} &\text{ (continued)} \\ - \text{F1\_Kept\_Since\_Mar} - \text{F2\_Kept\_Since\_Mar} &\geq 0.0 \end{aligned}$$

where  $\text{F1\_Kept\_Since\_Mar}$  is the number of chips used during April at factory 1 that were obtained in March at either factory 1 or factory 2, and  $\text{F2\_Kept\_Since\_Mar}$  is the number of chips used during April at factory 2 that were obtained in March. The last constraint ensures that the number of chips used during April that were obtained in March does not exceed the number of chips not used in March. There may be a cost to hold chips in inventory. This can be modeled having a positive objective function coefficient for the nonarc variables  $\text{F1\_Kept\_Since\_Mar}$  and  $\text{F2\_Kept\_Since\_Mar}$ . Moreover, nonarc variable upper bounds represent an upper limit on the number of chips that can be held in inventory between March and April.

See Example 4.1 through Example 4.5, which use this TV problem. The use of nonarc variables as described previously is illustrated.

---

## Introduction

---

### Getting Started: NPSC Problems

---

To solve NPSC problems using PROC INTPOINT, you save a representation of the network and the side constraints in three SAS data sets. These data sets are then passed to PROC INTPOINT for solution. There are various forms that a problem's data can take. You can use any one or a combination of several of these forms.

The NODEDATA= data set contains the names of the supply and demand nodes and the supply or demand associated with each. These are the elements in the column vector  $b$  in the NPSC problem (see the “Mathematical Description of NPSC” section on page 56).

The ARCDATA= data set contains information about the variables of the problem. Usually these are arcs, but there can be data related to nonarc variables in the ARCDATA= data set as well.

An arc is identified by the names of its tail node (where it originates) and head node (where it is directed). Each observation can be used to identify an arc in the network and, optionally, the cost per flow unit across the arc, the arc's capacity, lower flow bound, and name. These data are associated with the matrix  $F$  and the vectors  $c$ ,  $l$ , and  $u$  in the NPSC problem (see the “Mathematical Description of NPSC” section on page 56).

**Note:** although  $F$  is a node-arc incidence matrix, it is specified in the ARCDATA= data set by arc definitions. Do not explicitly specify these flow conservation constraints as constraints of the problem.

In addition, the ARCDATA= data set can be used to specify information about nonarc variables, including objective function coefficients, lower and upper value bounds, and names. These data are the elements of the vectors  $d$ ,  $m$ , and  $v$  in the NPSC problem (see the “Mathematical Description of NPSC” section on page 56). Data for an arc or nonarc variable can be given in more than one observation.

Supply and demand data also can be specified in the ARCDATA= data set. In such a case, the NODEDATA= data set may not be needed.

The CONDATA= data set describes the side constraints and their right-hand-sides. These data are elements of the matrices  $H$  and  $Q$  and the vector  $r$ . Constraint types are also specified in the CONDATA= data set. You can include in this data set upper bound values or capacities, lower flow or value bounds, and costs or objective function coefficients. It is possible to give all information about some or all nonarc variables in the CONDATA= data set.

An arc is identified in this data set by its name. If you specify an arc's name in the ARCDATA= data set, then this name is used to associate data in the CONDATA= data set with that arc. Each arc also has a default name that is the name of the tail and head node of the arc concatenated together and separated by an underscore character; tail\_head, for example.



If you use the dense side constraint input format (described in the “CONDATA= Data Set” section on page 126), and want to use these default arc names, these arc names are names of SAS variables in the VAR list of the CONDATA= data set.

If you use the sparse side constraint input format (see the “CONDATA= Data Set” section on page 126) and want to use these default arc names, these arc names are values of the COLUMN list variable of the CONDATA= data set.

PROC INTPOINT reads the data from the NODEDATA= data set, the ARCDATA= data set, and the CONDATA= data set. Error checking is performed, and the model is converted into an equivalent LP. This LP is preprocessed. Preprocessing is optional but highly recommended. Preprocessing analyzes the model and tries to determine before optimization whether variables can be “fixed” to their optimal values. Knowing that, the model can be modified and these variables dropped out. It can be determined that some constraints are redundant. Sometimes, preprocessing succeeds in reducing the size of the problem, thereby making the subsequent optimization easier and faster.

The optimal solution to the equivalent LP is then found. This LP is converted back to the original NPSC problem, and the optimum for this is derived from the optimum of the equivalent LP. If the problem was preprocessed, the model is now post-processed, where fixed variables are reintroduced. The solution can be saved in the CONOUT= data set.

### ***Introductory NPSC Example***

Consider the following transshipment problem for an oil company. Crude oil is shipped to refineries where it is processed into gasoline and diesel fuel. The gasoline and diesel fuel are then distributed to service stations. At each stage, there are shipping, processing, and distribution costs. Also, there are lower flow bounds and capacities.

In addition, there are two sets of side constraints. The first set is that two times the crude from the Middle East cannot exceed the throughput of a refinery plus 15 units. (The phrase “plus 15 units” that finishes the last sentence is used to enable some side constraints in this example to have a nonzero rhs.) The second set of constraints are necessary to model the situation that one unit of crude mix processed at a refinery yields three-fourths of a unit of gasoline and one-fourth of a unit of diesel fuel.

Because there are two products that are not independent in the way in which they flow through the network, an NPSC is an appropriate model for this example (see Figure 4.6). The side constraints are used to model the limitations on the amount of Middle Eastern crude that can be processed by each refinery and the conversion proportions of crude to gasoline and diesel fuel.

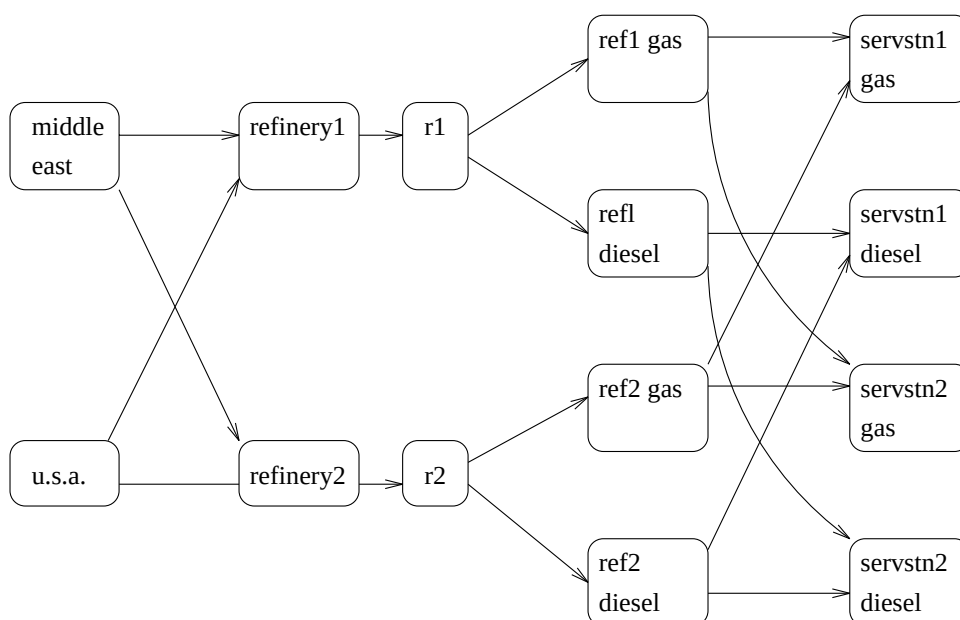


**Figure 4.6.** Oil Industry Example

To solve this problem with PROC INTPOINT, save a representation of the model in three SAS data sets. In the NODEDATA= data set, you name the supply and demand nodes and give the associated supplies and demands. To distinguish demand nodes from supply nodes, specify demands as negative quantities. For the oil example, the NODEDATA= data set can be saved as follows:

```

title 'Oil Industry Example';
title3 'Setting Up Nodedata = Noded For PROC INTPOINT';
data noded;
  input _node_&$15. _sd_;
  datalines;
middle east      100
u.s.a.           80
servstn1 gas     -95
servstn1 diesel -30
servstn2 gas     -40
servstn2 diesel -15
;

```

The ARCDATA= data set contains the rest of the information about the network. Each observation in the data set identifies an arc in the network and gives the cost per flow unit across the arc, the capacities of the arc, the lower bound on flow across the arc, and the name of the arc.

```

title3 'Setting Up Arcdata = Arcd1 For PROC INTPOINT';
data arcd1;
  input _from_&$11. _to_&$15. _cost_ _capac_ _lo_ _name_ $;
  datalines;
middle east      refinery 1      63      95      20      m_e_ref1
middle east      refinery 2      81      80      10      m_e_ref2
u.s.a.           refinery 1      55      .        .        .
u.s.a.           refinery 2      49      .        .        .
refinery 1       r1             200     175     50      thrupt1
refinery 2       r2             220     100     35      thrupt2
r1               ref1 gas        .       140     .        r1_gas
r1               ref1 diesel    .       75     .        .
r2               ref2 gas        .       100     .        r2_gas
r2               ref2 diesel    .       75     .        .
ref1 gas         servstn1 gas    15      70     .        .
ref1 gas         servstn2 gas    22      60     .        .
ref1 diesel      servstn1 diesel 18      .        .        .
ref1 diesel      servstn2 diesel 17      .        .        .
ref2 gas         servstn1 gas    17      35     5        .
ref2 gas         servstn2 gas    31      .        .        .
ref2 diesel      servstn1 diesel 36      .        .        .
ref2 diesel      servstn2 diesel 23      .        .        .
;

```

Finally, the CONDATA= data set contains the side constraints for the model:

```

title3 'Setting Up Condata = Cond1 For PROC INTPOINT';
data cond1;
  input m_e_ref1 m_e_ref2 thrupt1 r1_gas thrupt2 r2_gas
        _type_ $ _rhs_;
  datalines;
-2   .   1   .   .   .   >=   -15
.   -2   .   .   1   .   GE   -15
.   .   -3  4   .   .   EQ    0
.   .   .   .   -3  4   =    0
;

```

Note that the SAS variable names in the CONDATA= data set are the names of arcs given in the ARCDATA= data set. These are the arcs that have nonzero constraint coefficients in side constraints. For example, the proportionality constraint that specifies that one unit of crude at each refinery yields three-fourths of a unit of gasoline and one-fourth of a unit of diesel fuel is given for refinery 1 in the third observation and for refinery 2 in the last observation. The third observation requires that each unit of flow on the arc thrupt1 equals three-fourths of a unit of flow on the arc r1\_gas. Because all crude processed at refinery 1 flows through thrupt1 and all gasoline produced at refinery 1 flows through r1\_gas, the constraint models the situation. It proceeds similarly for refinery 2 in the last observation.

To find the minimum cost flow through the network that satisfies the supplies, demands, and side constraints, invoke PROC INTPOINT as follows:

```
proc intpoint
  bytes=1000000
  nodedata=noded          /* the supply and demand data */
  arcdata=arcd1           /* the arc descriptions      */
  condata=cond1           /* the side constraints     */
  conout=solution;        /* the solution data set    */
run;
```

The following messages, that appear on the SAS log, summarize the model as read by PROC INTPOINT and note the progress toward a solution:

```
NOTE: Number of nodes= 14 .
NOTE: Number of supply nodes= 2 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 180 , total demand= 180 .
NOTE: Number of arcs= 18 .
NOTE: Number of <= side constraints= 0 .
NOTE: Number of == side constraints= 2 .
NOTE: Number of >= side constraints= 2 .
NOTE: Number of side constraint coefficients= 8 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 16 .
NOTE: Number of >= constraints= 2 .
NOTE: Number of constraint coefficients= 44 .
NOTE: Number of variables= 18 .
NOTE: After preprocessing, number of <= constraints= 0.
NOTE: After preprocessing, number of == constraints= 6.
NOTE: After preprocessing, number of >= constraints= 2.
NOTE: The preprocessor eliminated 10 constraints from the
      problem.
NOTE: The preprocessor eliminated 22 constraint
      coefficients from the problem.
NOTE: After preprocessing, number of variables= 8.
NOTE: The preprocessor eliminated 10 variables from the
      problem.
NOTE: 2 columns, 0 rows and 2 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 18 nonzero elements in A * A transpose.
NOTE: Of the 8 rows and columns, 2 are sparse.
NOTE: There are 8 nonzero superdiagonal elements in the
      sparse rows of the factored A * A transpose. This
      includes fill-in.
NOTE: There are 3 operations of the form
      u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q] to factorize the
      sparse rows of A * A transpose.
NOTE: Bound feasibility attained by iteration 1.
```

NOTE: Dual feasibility attained by iteration 1.  
 NOTE: Constraint feasibility attained by iteration 2.  
 NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 12 iterations.  
 NOTE: The rate of complementarity reduction is slow. The optimum has probably been reached even though standard stopping conditions have not been met.  
 NOTE: Objective = 50875.006369.  
 NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.  
 NOTE: There were 18 observations read from the data set WORK.ARC1.  
 NOTE: There were 6 observations read from the data set WORK.NODE1.  
 NOTE: There were 4 observations read from the data set WORK.COND1.  
 NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.

The first set of messages shows the size of the problem. The next set of messages provides statistics on the size of the equivalent LP problem. The number of variables may not equal the number of arcs if the problem has nonarc variables. This example has none. To convert a network to the equivalent LP problem, a flow conservation constraint must be created for each node (including an excess or bypass node, if required). This explains why the number of equality constraints and the number of constraint coefficients differ from the number of equality side constraints and the number of coefficients in all side constraints.

If the preprocessor was successful in decreasing the problem size, some messages will report how well it did. In this example, the model size was cut approximately in half!

The next set of messages describes aspects of the Interior Point algorithm. Of particular interest are those concerned with the Cholesky factorization of  $AA^T$  where  $A$  is the coefficient matrix of the final LP. It is crucial to preorder the rows and columns of this matrix to prevent *fill-in* and reduce the number of row operations to undertake the factorization. See the “Interior Point Algorithmic Details” section on page 58 for a more extensive explanation.

Unlike PROC LP, which displays the solution and other information as output, PROC INTPOINT saves the optimum in the output SAS data set that you specify. For this example, the solution is saved in the SOLUTION data set. It can be displayed with the PRINT procedure as:

```
title3 'Optimum';
proc print data=solution;
  var _from_ _to_ _cost_ _capac_ _lo_ _name_
      _supply_ _demand_ _flow_ _fcost_;
  sum _fcost_;
run;
```

| Oil Industry Example           |             |                 |       |          |          |          |       |          |       |
|--------------------------------|-------------|-----------------|-------|----------|----------|----------|-------|----------|-------|
| Optimum                        |             |                 |       |          |          |          |       |          |       |
| Flow                           | Cost        | Flow            | Cost  | Flow     | Cost     | Flow     | Cost  | Flow     | Cost  |
| Variable                       | Value       | Variable        | Value | Variable | Value    | Variable | Value | Variable | Value |
| 1 refinery 1 r1                | 200         | 175 50 thruput1 | . .   | 145.000  | 28999.99 |          |       |          |       |
| 2 refinery 2 r2                | 220         | 100 35 thruput2 | . .   | 35.000   | 7700.01  |          |       |          |       |
| 3 r1 ref1 diesel               | 0           | 75 0            | . .   | 36.250   | 0.00     |          |       |          |       |
| 4 r1 ref1 gas                  | 0           | 140 0 r1_gas    | . .   | 108.750  | 0.00     |          |       |          |       |
| 5 r2 ref2 diesel               | 0           | 75 0            | . .   | 8.750    | 0.00     |          |       |          |       |
| 6 r2 ref2 gas                  | 0           | 100 0 r2_gas    | . .   | 26.250   | 0.00     |          |       |          |       |
| 7 middle east refinery 1       | 63          | 95 20 m_e_ref1  | 100 . | 80.000   | 5039.99  |          |       |          |       |
| 8 u.s.a. refinery 1            | 55 99999999 | 0               | 80 .  | 65.000   | 3575.00  |          |       |          |       |
| 9 middle east refinery 2       | 81          | 80 10 m_e_ref2  | 100 . | 20.000   | 1620.01  |          |       |          |       |
| 10 u.s.a. refinery 2           | 49 99999999 | 0               | 80 .  | 15.000   | 735.00   |          |       |          |       |
| 11 ref1 diesel servstn1 diesel | 18 99999999 | 0               | . 30  | 30.000   | 540.00   |          |       |          |       |
| 12 ref2 diesel servstn1 diesel | 36 99999999 | 0               | . 30  | 0.000    | 0.00     |          |       |          |       |
| 13 ref1 gas servstn1 gas       | 15          | 70 0            | . 95  | 68.750   | 1031.25  |          |       |          |       |
| 14 ref2 gas servstn1 gas       | 17          | 35 5            | . 95  | 26.250   | 446.25   |          |       |          |       |
| 15 ref1 diesel servstn2 diesel | 17 99999999 | 0               | . 15  | 6.250    | 106.25   |          |       |          |       |
| 16 ref2 diesel servstn2 diesel | 23 99999999 | 0               | . 15  | 8.750    | 201.25   |          |       |          |       |
| 17 ref1 gas servstn2 gas       | 22          | 60 0            | . 40  | 40.000   | 879.99   |          |       |          |       |
| 18 ref2 gas servstn2 gas       | 31 99999999 | 0               | . 40  | 0.000    | 0.01     |          |       |          |       |
| =====                          |             |                 |       |          |          |          |       |          |       |
| 50875.01                       |             |                 |       |          |          |          |       |          |       |

Figure 4.7. CONOUT=SOLUTION

Notice that, in CONOUT=SOLUTION (Figure 4.7), the optimal flow through each arc in the network is given in the variable named `_FLOW_`, and the cost of flow through each arc is given in the variable `_FCOST_`.

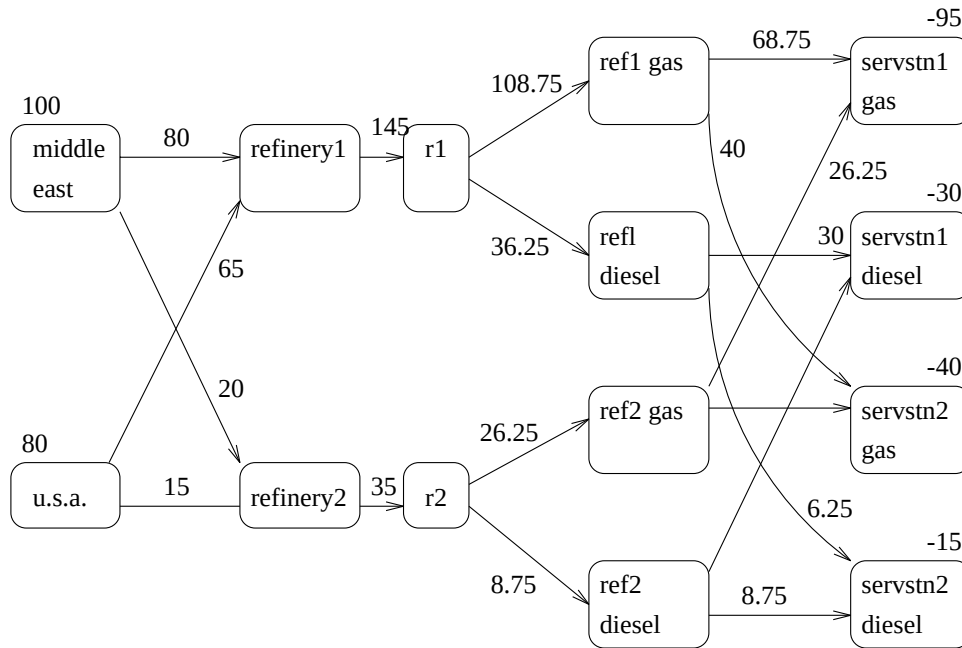


Figure 4.8. Oil Industry Solution

## Getting Started: LP Problems

Data for an LP problem resembles the data for side constraints and nonarc variables supplied to PROC INTPOINT when solving an NPSC problem. It is also very similar to the data required by the LP procedure.

To solve LP problems using PROC INTPOINT, you save a representation of the LP variables and the constraints in one or two SAS data sets. These data sets are then passed to PROC INTPOINT for solution. There are various forms that a problem's data can take. You can use any one or a combination of several of these forms.

The ARCDATA= data set contains information about the LP variables of the problem. Although this data set is called ARCDATA, it contains data for no arcs. Instead, all data in this data set are related to LP variables. This data set has no SAS variables containing values that are node names.

The ARCDATA= data set can be used to specify information about LP variables, including objective function coefficients, lower and upper value bounds, and names. These data are the elements of the vectors  $d$ ,  $m$ , and  $v$  in problem (LP). Data for an LP variable can be given in more than one observation.

The CONDATA= data set describes the constraints and their right-hand-sides. These data are elements of the matrix  $Q$  and the vector  $r$ .

Constraint types are also specified in the CONDATA= data set. You can include in this data set LP variable data such as upper bound values, lower value bounds, and

objective function coefficients. It is possible to give all information about some or all LP variables in the CONDATA= data set.

Because PROC INTPOINT evolved from PROC NETFLOW, another procedure in SAS/OR software that was originally designed to solve models with networks, the ARCDATA= data set is always expected. If the ARCDATA= data set is not specified, by default the last data set created before PROC INTPOINT is invoked is assumed to be the ARCDATA= data set. However, these characteristics of PROC INTPOINT are not helpful when an LP problem is being solved and all data is provided in a single data set specified by the CONDATA= data set, and that data set is not the last data set created before PROC INTPOINT starts. In this case, you must specify that the ARCDATA= data set and the CONDATA= data set are both equal to the input data set. PROC INTPOINT then knows that an LP problem is to be solved and that the data reside in one data set.

An LP variable is identified in this data set by its name. If you specify an LP variable's name in the ARCDATA= data set, then this name is used to associate data in the CONDATA= data set with that LP variable.

If you use the dense constraint input format (described in the “CONDATA= Data Set” section on page 126), these LP variable names are names of SAS variables in the VAR list of the CONDATA= data set.

If you use the sparse constraint input format (described in the “CONDATA= Data Set” section on page 126), these LP variable names are values of the SAS variables in the COLUMN list of the CONDATA= data set.

PROC INTPOINT reads the data from the ARCDATA= data set (if there is one) and the CONDATA= data set. Error checking is performed, and the LP is preprocessed. Preprocessing is optional but highly recommended. The preprocessor analyzes the model and tries to determine before optimization whether LP variables can be “fixed” to their optimal values. Knowing that, the model can be modified and these LP variables dropped out. Some constraints may be found to be redundant. Sometimes, preprocessing succeeds in reducing the size of the problem, thereby making the subsequent optimization easier and faster.

The optimal solution is then found for the resulting LP. If the problem was preprocessed, the model is now post-processed, where fixed LP variables are reintroduced. The solution can be saved in the CONOUT= data set.



**Introductory LP Example**

Consider the Linear Programming problem in the chapter on the LP procedure. The SAS data set in that section is created the same way here:

```

title 'Linear Programming Example';
title3 'Setting Up Condata = dcon1 For PROC INTPOINT';
data dcon1;
  input _id_ $14.
        a_light a_heavy brega naphthal naphthai
        heatingo jet_1 jet_2
        _type_ $ _rhs_;
  datalines;
profit      -175 -165 -205  0  0  0 300 300 max      .
naphtha_l_conv .035 .030 .045 -1  0  0  0  0 eq      0
naphtha_i_conv .100 .075 .135  0 -1  0  0  0 eq      0
heating_o_conv .390 .300 .430  0  0 -1  0  0 eq      0
recipe_1       0    0    0  0 .3 .7 -1  0 eq      0
recipe_2       0    0    0 .2  0 .8  0 -1 eq      0
available      110  165  80  .  .  .  .  . upperbd .
;

```

To solve this problem, use:

```

proc intpoint
  bytes=1000000
  condata=dcon1
  conout=solutn1;
run;

```

Note how it is possible to use an input SAS data set of PROC LP and, without requiring any changes to be made to the data set, to use that as an input data set for PROC INTPOINT.

The following messages that appear on the SAS log summarize the model as read by PROC INTPOINT and note the progress toward a solution:

```

NOTE: Number of variables= 8 .
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 5 .
NOTE: Number of >= constraints= 0 .
NOTE: Number of constraint coefficients= 18 .
NOTE: After preprocessing, number of <= constraints= 0.
NOTE: After preprocessing, number of == constraints= 0.
NOTE: After preprocessing, number of >= constraints= 0.
NOTE: The preprocessor eliminated 5 constraints from the
      problem.
NOTE: The preprocessor eliminated 18 constraint
      coefficients from the problem.
NOTE: After preprocessing, number of variables= 0.
NOTE: The preprocessor eliminated 8 variables from the
      problem.
WARNING: Optimization is unnecessary as the problem has 0
         variables and 0 rows.

```

NOTE: Preprocessing could have caused that.  
 NOTE: Objective = 1544.  
 NOTE: The data set WORK.SOLUTN1 has 8 observations and 6 variables.  
 NOTE: There were 7 observations read from the data set WORK.DCON1.  
 NOTE: The data set WORK.SOLUTN1 has 8 observations and 6 variables.

Notice that the preprocessor succeeded in fixing *all* LP variables to their optimal values, eliminating the need to do any actual optimization.

Unlike PROC LP, which displays the solution and other information as output, PROC INTPOINT saves the optimum in the output SAS data set you specify. For this example, the solution is saved in the SOLUTION data set. It can be displayed with PROC PRINT as:

```
title3 'LP Optimum';
proc print data=solutn1;
  var _name_ _cost_ _capac_ _lo_ _flow_ _fcost_;
  sum _fcost_;
run;
```

Notice that in the CONOUT=SOLUTION (Figure 4.9) the optimal value through each variable in the LP is given in the variable named `_FLOW_`, and that the cost of value for each variable is given in the variable `_FCOST_`.

| Linear Programming Example |          |        |          |      |        |         |
|----------------------------|----------|--------|----------|------|--------|---------|
| LP Optimum                 |          |        |          |      |        |         |
| Obs                        | _NAME_   | _COST_ | _CAPAC_  | _LO_ | _FLOW_ | _FCOST_ |
| 1                          | a_heavy  | -165   | 165      | 0    | 0.00   | 0       |
| 2                          | a_light  | -175   | 110      | 0    | 110.00 | -19250  |
| 3                          | brega    | -205   | 80       | 0    | 80.00  | -16400  |
| 4                          | heatingo | 0      | 99999999 | 0    | 77.30  | 0       |
| 5                          | jet_1    | 300    | 99999999 | 0    | 60.65  | 18195   |
| 6                          | jet_2    | 300    | 99999999 | 0    | 63.33  | 18999   |
| 7                          | naphthai | 0      | 99999999 | 0    | 21.80  | 0       |
| 8                          | naphthal | 0      | 99999999 | 0    | 7.45   | 0       |
|                            |          |        |          |      |        | =====   |
|                            |          |        |          |      |        | 1544    |

Figure 4.9. CONOUT=SOLUTN1

The same model can be specified in the sparse format as in the following scon2 dataset. This format enables you to omit the zero coefficients:

```

title3 'Setting Up Condata = scon2 For PROC INTPOINT';
data scon2;
  input _type_ $ @10 _col_ $13. @24 _row_ $16. _coef_;
  datalines;
max      .                profit                .
eq       .                napha_l_conv           .
eq       .                napha_i_conv           .
eq       .                heating_oil_conv        .
eq       .                recipe_1                .
eq       .                recipe_2                .
upperbd  .                available              .
.         a_light         profit                -175
.         a_light         napha_l_conv           .035
.         a_light         napha_i_conv           .100
.         a_light         heating_oil_conv        .390
.         a_light         available              110
.         a_heavy         profit                -165
.         a_heavy         napha_l_conv           .030
.         a_heavy         napha_i_conv           .075
.         a_heavy         heating_oil_conv        .300
.         a_heavy         available              165
.         brega           profit                -205
.         brega           napha_l_conv           .045
.         brega           napha_i_conv           .135
.         brega           heating_oil_conv        .430
.         brega           available              80
.         naphthal        napha_l_conv          -1
.         naphthal        recipe_2               .2
.         naphthai        napha_i_conv          -1
.         naphthai        recipe_1               .3
.         heatingo        heating_oil_conv       -1
.         heatingo        recipe_1               .7
.         heatingo        recipe_2               .8
.         jet_1           profit                 300
.         jet_1           recipe_1               -1
.         jet_2           profit                 300
.         jet_2           recipe_2               -1
;

```

To find the minimum cost solution, invoke PROC INTPOINT (note the SPARSECONDATA option which must be specified) as follows:

```

proc intpoint
  bytes=1000000
  sparsecondata
  condata=scon2
  conout=solutn2;
run;

```

A data set that can be used as the ARCDATA= data set can be initialized as follows:

```
data vars3;
  input _name_ $ profit available;
  datalines;
a_heavy -165 165
a_light -175 110
brega -205 80
heatingo 0 .
jet_1 300 .
jet_2 300 .
naphthai 0 .
naphthal 0 .
;
```

The following CONDATA= data set is the original dense format CONDATA= dcon1 data set after the LP variable's nonconstraint information has been removed. (You could have left some or all of that information in CONDATA as PROC INTPOINT "merges" data, but doing that and checking for consistency takes time.)

```
data dcon3;
  input _id_ $14.
  a_light a_heavy brega naphthal naphthai
  heatingo jet_1 jet_2
  _type_ $ _rhs_;
  datalines;
naphtha_l_conv .035 .030 .045 -1 0 0 0 0 eq 0
naphtha_i_conv .100 .075 .135 0 -1 0 0 0 eq 0
heating_o_conv .390 .300 .430 0 0 -1 0 0 eq 0
recipe_1 0 0 0 0 .3 .7 -1 0 eq 0
recipe_2 0 0 0 .2 0 .8 0 -1 eq 0
;
```

**Note:** You must now specify the MAXIMIZE option; otherwise, PROC INTPOINT will optimize to the minimum (which, incidentally, has a total objective = -3539.25). You must indicate that the SAS variable profit in the ARCDATA= vars3 data set has values that are objective function coefficients, by specifying the OBJFN statement. The UPPERBD must be specified as the SAS variable available that has as values upper bounds:

```
proc intpoint
  maximize /* ***** necessary ***** */
  bytes=1000000
  arcdata=vars3
  condata=dcon3
  conout=solutn3;
objfn profit;
upperbd available;
run;
```

The ARCDATA=vars3 data set can become more concise by noting that the model variables heatingo, naphthai, and naphthal have zero objective function coeffi-

cients (the default) and default upper bounds, so those observations need not be present:

```
data vars4;
  input _name_ $ profit available;
  datalines;
a_heavy -165 165
a_light -175 110
brega -205 80
jet_1 300 .
jet_2 300 .
;
```

The CONDATA=dcon3 data set can become more concise by noting that all the constraints have the same type (eq) and zero (the default) rhs values. This model is a good candidate for using the DEFCONTYPE= options.

The DEFCONTYPE= option can be useful not only when *all* constraints have the same type as is the case here, but also when *most* constraints have the same type and you want to change the default type from  $\leq$  to  $=$  or  $\geq$ . The essential constraint type data in the CONDATA= data set is that which overrides the DEFCONTYPE= type you specified:

```
data dcon4;
  input _id_ $14.
  a_light a_heavy brega naphthal naphthai
  heatingo jet_1 jet_2;
  datalines;
naphtha_l_conv .035 .030 .045 -1 0 0 0 0
naphtha_i_conv .100 .075 .135 0 -1 0 0 0
heating_o_conv .390 .300 .430 0 0 -1 0 0
recipe_1 0 0 0 0 .3 .7 -1 0
recipe_2 0 0 0 .2 0 .8 0 -1
;

proc intpoint
  maximize defcontype=eq
  arcdata=vars3
  condata=dcon3
  conout=solutn3;
objfn profit;
upperbd available;
run;
```

Here are several different ways of using the ARCDATA= data set and a sparse format CONDATA= data set for this LP. The following CONDATA= data set is the result of removing the profit and available data from the original sparse format CONDATA=scon2 data set.

```

data scon5;
  input _type_ $ @10 _col_ $13. @24 _row_ $16. _coef_;
  datalines;
eq      .      napha_l_conv      .
eq      .      napha_i_conv      .
eq      .      heating_oil_conv  .
eq      .      recipe_1          .
eq      .      recipe_2          .
.      a_light  napha_l_conv      .035
.      a_light  napha_i_conv      .100
.      a_light  heating_oil_conv  .390
.      a_heavy  napha_l_conv      .030
.      a_heavy  napha_i_conv      .075
.      a_heavy  heating_oil_conv  .300
.      brega    napha_l_conv      .045
.      brega    napha_i_conv      .135
.      brega    heating_oil_conv  .430
.      naphthal napha_l_conv      -1
.      naphthal recipe_2          .2
.      naphthai napha_i_conv      -1
.      naphthai recipe_1          .3
.      heatingo heating_oil_conv  -1
.      heatingo recipe_1          .7
.      heatingo recipe_2          .8
.      jet_1    recipe_1          -1
.      jet_2    recipe_2          -1
;

proc intpoint
  maximize
  sparsesecondata
  arcdata=vars3      /* or arcdata=vars4 */
  condata=scon5
  conout=solutn5;
objfn profit;
upperbd available;
run;

```

The CONDATA=scon5 data set can become more concise by noting that all the constraints have the same type (eq) and zero (the default) rhs values. Use the DEFCON-  
TYPE= option again. Once the first five observations of the CONDATA=scon5 data  
set are removed, the \_type\_ SAS variable has values that are missing in all of the  
remaining observations. Therefore, this SAS variable can be removed.

```

data scon6;
  input _col_ $ _row_&$16. _coef_;
  datalines;
a_light  napha_l_conv          .035
a_light  napha_i_conv          .100
a_light  heating_oil_conv      .390
a_heavy  napha_l_conv          .030
a_heavy  napha_i_conv          .075
a_heavy  heating_oil_conv      .300
brega    napha_l_conv          .045
brega    napha_i_conv          .135
brega    heating_oil_conv      .430
naphthal napha_l_conv          -1
naphthal recipe_2             .2
naphthai napha_i_conv          -1
naphthai recipe_1             .3
heatingo heating_oil_conv      -1
heatingo recipe_1             .7
heatingo recipe_2             .8
jet_1    recipe_1             -1
jet_2    recipe_2             -1
;

proc intpoint
  maximize
  defcontype=eq
  sparsesecondata
  arcdata=vars4
  condata=scon6
  conout=solutn6;
objfn profit;
upperbd available;
run;

```

---

## Typical PROC INTPOINT Run

You start PROC INTPOINT by giving the PROC INTPOINT statement. You can specify many options that control the procedure in the PROC INTPOINT statement, or you can rely on default settings and specify no options at all. However, there are some options you must specify, or will probably have to specify:

- You must specify the BYTES= parameter indicating the size of the working memory that the procedure is allowed to use. This option has no default.
- In many instances (and certainly when solving NPSC problems), you need to specify the ARCDATA= data set. This option has a default (which is the SAS data set that was created last before PROC INTPOINT began running), but that may need to be overridden.
- The CONDATA= data set must also be specified if the problem is NPSC and has side constraints, or if it is an LP problem.
- When solving a network problem, you have to specify the NODEDATA= data set, if some model data is given in such a data set.

Some options, while optional, are frequently required. To have the optimal solution output to a SAS data set, you have to specify the CONOUT= data set. You may want to indicate reasons why optimization should stop (for example, you can indicate the maximum number of iterations that can be performed), or you might want to alter stopping criteria so that optimization does not stop prematurely. Some options enable you to control other aspects of the Interior Point algorithm. Specifying certain values for these options can reduce the time it takes to solve a problem.

The SAS variable lists should be given next. If you have SAS variables in the input data sets that have special names (for example, a SAS variable in the ARCDATA= data set named \_TAIL\_ that has tail nodes of arcs as values), it may not be necessary to have many or any variable lists. If you do not specify a TAIL variable list, PROC INTPOINT will search the ARCDATA= data set for a SAS variable named \_TAIL\_.

What usually follows is a RUN statement, which indicates that all information that you, the user, need to supply to PROC INTPOINT has been given, and the procedure is to start running. This also happens if you specify a statement in your SAS program that PROC INTPOINT does not recognize as one of its own, the next DATA step or procedure.

The QUIT statement indicates that PROC INTPOINT must immediately finish.

For example, a PROC INTPOINT run might look something like this:

```
proc intpoint
    bytes=    /* working memory size */
    arcdata= /* data set */
    condata= /* data set */
    /* other options */
;
variable list specifications; /* if necessary */
run;      /* start running, read data, */
          /* and do the optimization. */
```



---

## Syntax

Below are statements used in PROC INTPOINT, listed in alphabetical order as they appear in the text that follows.

```
PROC INTPOINT options ;  
  CAPACITY variable ;  
  COEF variables ;  
  COLUMN variable ;  
  COST variable ;  
  DEMAND variable ;  
  HEADNODE variable ;  
  ID variables ;  
  LO variable ;  
  NAME variable ;  
  NODE variable ;  
  QUIT ;  
  RHS variable ;  
  ROW variables ;  
  RUN ;  
  SUPDEM variable ;  
  SUPPLY variable ;  
  TAILNODE variable ;  
  TYPE variable ;  
  VAR variables ;
```

Most of the statements of the INTPOINT procedure are variable lists. The variables specified in the variable lists must be present in one of three input SAS data sets:

|                                                                                                                                                                                                                                            |                         |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| <i>PROC INTPOINT options;</i>                                                                                                                                                                                                              | required statement      |
| <i>CAPACITY variable;</i><br><i>COST variable;</i><br><i>DEMAND variable;</i><br><i>HEADNODE variable;</i><br><i>ID variables;</i><br><i>LO variable;</i><br><i>NAME variable;</i><br><i>SUPPLY variable;</i><br><i>TAILNODE variable;</i> | optional ARCDATA lists  |
| <i>NODE variable;</i><br><i>SUPDEM variable;</i>                                                                                                                                                                                           | optional NODEDATA lists |
| <i>COEF variables;</i><br><i>COLUMN variable;</i><br><i>RHS variable;</i><br><i>ROW variables;</i><br><i>TYPE variable;</i><br><i>VAR variables;</i>                                                                                       | optional CONDATA lists  |
| <b>RUN;</b><br><b>QUIT;</b>                                                                                                                                                                                                                | optional statements     |

## Functional Summary

The following tables outline the options that can be specified in the PROC INTPOINT statement. This statement invokes the procedure.

**Table 4.1.** Input Data Set Options

| Description               | Statement | Option    |
|---------------------------|-----------|-----------|
| arcs input data set       | INTPOINT  | ARCDATA=  |
| nodes input data set      | INTPOINT  | NODEDATA= |
| constraint input data set | INTPOINT  | CONDATA=  |

**Table 4.2.** Options for Networks

| Description                                         | Statement | Option       |
|-----------------------------------------------------|-----------|--------------|
| default arc cost and                                | INTPOINT  | DEFCOST=     |
| default variable objective function coefficient     |           |              |
| default arc capacity                                | INTPOINT  | DEFCAPACITY= |
| default variable upper bound                        |           |              |
| default arc lower flow bound                        | INTPOINT  | DEFMINFLOW=  |
| default variable lower bound                        |           |              |
| network's only supply node                          | INTPOINT  | SOURCE=      |
| SOURCE's supply capability                          | INTPOINT  | SUPPLY=      |
| network's only demand node                          | INTPOINT  | SINK=        |
| SINK's demand                                       | INTPOINT  | DEMAND=      |
| excess supply or demand is conveyed through network | INTPOINT  | THRUNET      |
| find maximal flow between SOURCE and SINK           | INTPOINT  | MAXFLOW      |
| cost of bypass arc when solving MAXFLOW problem     | INTPOINT  | BYPASSDIV=   |
| find shortest path from SOURCE to SINK              | INTPOINT  | SHORTPATH    |

**Table 4.3.** Miscellaneous Options

| Description                                                           | Statement | Option    |
|-----------------------------------------------------------------------|-----------|-----------|
| infinity value                                                        | INTPOINT  | INFINITY= |
| scale constraint row, or nonarc variable column coefficients, or both | INTPOINT  | SCALE=    |
| maximization instead of minimization                                  | INTPOINT  | MAXIMIZE  |

**Table 4.4.** Data Set Read Options

| Description                                                    | Statement | Option            |
|----------------------------------------------------------------|-----------|-------------------|
| CONDATA has sparse data format                                 | INTPOINT  | SPARSECONDATA     |
| default constraint type                                        | INTPOINT  | DEFCONTYPE=       |
| special COLUMN variable value                                  | INTPOINT  | TYPEOBS=          |
| special COLUMN variable value                                  | INTPOINT  | RHSOBS=           |
| used to interpret arc and nonarc variable names in the CONDATA | INTPOINT  | NAMECTRL=         |
| no nonarc data in the ARCDATA                                  | INTPOINT  | ARCS_ONLY_ARCDATA |
| data for an arc found in only one obs of ARCDATA               | INTPOINT  | ARC_SINGLE_OBS    |
| data for an constraint found in only one obs of the CONDATA    | INTPOINT  | CON_SINGLE_OBS    |
| data for a coefficient found once in the CONDATA               | INTPOINT  | NON_REPLIC=       |
| data is grouped, exploited during data read                    | INTPOINT  | GROUPED=          |

**Table 4.5.** Problem Size (approx.) Options

| Description                             | Statement | Option  |
|-----------------------------------------|-----------|---------|
| number of nodes                         | INTPOINT  | NNODES= |
| number of arcs                          | INTPOINT  | NARCS=  |
| number of nonarc variables or variables | INTPOINT  | NNAS=   |
| number of coefficients                  | INTPOINT  | NCOEFS= |
| number of constraints                   | INTPOINT  | NCONS=  |

**Table 4.6.** Memory Control Options

| Description                            | Statement | Option |
|----------------------------------------|-----------|--------|
| issue memory usage messages to SASLOG  | INTPOINT  | MEMREP |
| number of bytes to use for main memory | INTPOINT  | BYTES= |

**Table 4.7.** Output Data Set Option

| Description                   | Statement | Option  |
|-------------------------------|-----------|---------|
| constrained solution data set | INTPOINT  | CONOUT= |

**Table 4.8.** Miscellaneous Options

| Description                                                        | Statement | Option   |
|--------------------------------------------------------------------|-----------|----------|
| zero tolerance - optimization                                      | INTPOINT  | ZERO2=   |
| zero tolerance - real number comparisons                           | INTPOINT  | ZEROTOL= |
| display this number of similar SAS log messages, suppress the rest | INTPOINT  | VERBOSE= |

**Table 4.9.** Interior Point Algorithm Options

| Description                                  | Statement | Option       |
|----------------------------------------------|-----------|--------------|
| allowed amount of dual infeasibility         | INTPOINT  | TOLDINF=     |
| allowed amount of primal infeasibility       | INTPOINT  | TOLPINF=     |
| allowed total amount of dual infeasibility   | INTPOINT  | TOLTOTDINF=  |
| allowed total amount of primal infeasibility | INTPOINT  | TOLTOTPINF=  |
| cut-off tolerance for Cholesky factorization | INTPOINT  | CHOLTINYTOL= |
| density threshold for Cholesky processing    | INTPOINT  | DENSETHR=    |
| step-length multiplier                       | INTPOINT  | PDSTEPMULT=  |
| preprocessing type                           | INTPOINT  | PRSLTYPE=    |
| print optimization progress on SAS log       | INTPOINT  | PRINTLEVEL2= |
| ratio test zero tolerance                    | INTPOINT  | RTTOL=       |

**Table 4.10.** Interior Point Algorithm Options: Stopping Criteria

| Description                                           | Statement | Option             |
|-------------------------------------------------------|-----------|--------------------|
| maximum number of Interior Point algorithm iterations | INTPOINT  | MAXITERB=          |
| Primal-Dual (Duality) gap tolerance                   | INTPOINT  | PDGAPTOL=          |
| stop because of complementarity                       | INTPOINT  | STOP_C=            |
| stop because of duality gap                           | INTPOINT  | STOP_DG=           |
| stop because of <i>infeas<sub>b</sub></i>             | INTPOINT  | STOP_IB=           |
| stop because of <i>infeas<sub>c</sub></i>             | INTPOINT  | STOP_IC=           |
| stop because of <i>infeas<sub>d</sub></i>             | INTPOINT  | STOP_ID=           |
| stop because of complementarity                       | INTPOINT  | AND_STOP_C=        |
| stop because of duality gap                           | INTPOINT  | AND_STOP_DG=       |
| stop because of <i>infeas<sub>b</sub></i>             | INTPOINT  | AND_STOP_IB=       |
| stop because of <i>infeas<sub>c</sub></i>             | INTPOINT  | AND_STOP_IC=       |
| stop because of <i>infeas<sub>d</sub></i>             | INTPOINT  | AND_STOP_ID=       |
| stop because of complementarity                       | INTPOINT  | KEEPGOING_C=       |
| stop because of duality gap                           | INTPOINT  | KEEPGOING_DG=      |
| stop because of <i>infeas<sub>b</sub></i>             | INTPOINT  | KEEPGOING_IB=      |
| stop because of <i>infeas<sub>c</sub></i>             | INTPOINT  | KEEPGOING_IC=      |
| stop because of <i>infeas<sub>d</sub></i>             | INTPOINT  | KEEPGOING_ID=      |
| stop because of complementarity                       | INTPOINT  | AND_KEEPPGOING_C=  |
| stop because of duality gap                           | INTPOINT  | AND_KEEPPGOING_DG= |
| stop because of <i>infeas<sub>b</sub></i>             | INTPOINT  | AND_KEEPPGOING_IB= |
| stop because of <i>infeas<sub>c</sub></i>             | INTPOINT  | AND_KEEPPGOING_IC= |
| stop because of <i>infeas<sub>d</sub></i>             | INTPOINT  | AND_KEEPPGOING_ID= |

---

## PROC INTPOINT Statement

**PROC INTPOINT** *options* ;

This statement invokes the procedure. The following options can be specified in the PROC INTPOINT statement.

### **Data Set Options**

This section briefly describes all the input and output data sets used by PROC INTPOINT. The ARCDATA= data set, the NODEDATA= data set, and the CONDATA= data set can contain SAS variables that have special names, for instance `_CAPAC_`, `_COST_`, and `_HEAD_`. PROC INTPOINT looks for such variables if you do not give explicit variable list specifications. If a SAS variable with a special name is found and that SAS variable is not in another variable list specification, PROC INTPOINT determines that values of the SAS variable are to be interpreted in a special way. By using SAS variables that have special names, you may not need to have any variable list specifications.

#### **ARCDATA=SAS-data-set**

names the data set that contains arc and, optionally, nonarc variable information and nodal supply/demand data. The ARCDATA= data set must be specified in all PROC INTPOINT statements when solving NPSC problems.

If your problem is an LP, the ARCDATA= data set is optional. You can specify LP variable information such as objective function coefficients, and lower and upper bounds.

#### **CONDATA=SAS-data-set**

names the data set that contains the side constraint data. The data set can also contain other data such as arc costs, capacities, lower flow bounds, nonarc variable upper and lower bounds, and objective function coefficients. PROC INTPOINT needs a CONDATA= data set to solve a constrained problem. See the “CONDATA= Data Set” section on page 126 for more information.

If your problem is an LP, this data set contains the constraint data, and can also contain other data such as objective function coefficients, and lower and upper bounds. PROC INTPOINT needs a CONDATA= data set to solve an LP.

#### **CONOUT=SAS-data-set**

#### **COUT=SAS-data-set**

names the output data set that receives an optimal solution. See the “CONOUT= Data Set” section on page 135 for more information.

If PROC INTPOINT is outputting observations to the output data set and you want this to stop, press the keys used to stop SAS procedures.

#### **NODEDATA=SAS-data-set**

names the data set that contains the node supply and demand specifications. You do not need observations in the NODEDATA= data set for transshipment nodes. (Transshipment nodes neither supply nor demand flow.) All nodes are assumed to be trans-

shipment nodes until supply or demand data indicate otherwise. It is acceptable for some arcs to be directed toward supply nodes or away from demand nodes.

This data set is used only when you are solving network problems (not when solving LP problems), in which case the use of the NODEDATA= data set is optional provided that, if the NODEDATA= data set is not used, supply and demand details are specified by other means. Other means include using the MAXFLOW or SHORT-PATH option, SUPPLY or DEMAND variable list (or both) in the ARCDATA= data set, and the SOURCE=, SUPPLY=, SINK=, or DEMAND= option in the PROC INTPOINT statement.

### **General Options**

The following is a list of options you can use with PROC INTPOINT. The options are listed in alphabetical order.

#### **ARCS\_ONLY\_ARCDATA**

indicates that data for arcs only are in the ARCDATA= data set. When PROC INTPOINT reads the data in the ARCDATA= data set, memory would not be wasted to receive data for nonarc variables. The read might then be performed faster. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 146.

#### **ARC\_SINGLE\_OBS**

indicates that for all arcs and nonarc variables, data for each arc or nonarc variable is found in only one observation of the ARCDATA= data set. When reading the data in the ARCDATA= data set, PROC INTPOINT knows that the data in an observation is for an arc or a nonarc variable that has not had data previously read and that needs to be checked for consistency. The read might then be performed faster.

When solving an LP, specifying the ARC\_SINGLE\_OBS option indicates that for all LP variables, data for each LP variable is found in only one observation of the ARCDATA= data set. When reading the data in the ARCDATA= data set, PROC INTPOINT knows that the data in an observation is for an LP variable that has not had data previously read and that needs to be checked for consistency. The read might then be performed faster.

If you specify ARC\_SINGLE\_OBS, PROC INTPOINT automatically works as if GROUPED=ARCDATA is also specified.

See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 146.

#### **BYPASSDIVIDE=*b***

#### **BYPASSDIV=*b***

#### **BPD=*b***

should be used only when the MAXFLOW option has been specified; that is, PROC INTPOINT is solving a maximal flow problem. PROC INTPOINT prepares to solve maximal flow problems by setting up a bypass arc. This arc is directed from the SOURCE= to the SINK= and will eventually convey flow equal to INFINITY minus the maximal flow through the network. The cost of the bypass arc must be great enough to drive flow through the network, rather than through the bypass arc. Also,

the cost of the bypass arc must be greater than the eventual total cost of the maximal flow, which can be nonzero if some network arcs have nonzero costs. The cost of the bypass is set to the value of the INFINITY= option. Valid values for the BYPASS-DIV= option must be greater than or equal to 1.1.

If there are no nonzero costs of arcs in the MAXFLOW problem, the cost of the bypass arc is set to 1.0 (-1.0 if maximizing) if you do not specify the BYPASSDIV= option. The default value for the BYPASSDIV= option (in the presence of nonzero arc costs) is 100.0.

#### **BYTES=*b***

indicates the size of the main working memory (in bytes) that PROC INTPOINT will allocate. Specifying this option is mandatory. The working memory is used to store all the arrays and buffers used by PROC INTPOINT. If this memory has a size smaller than what is required to store all arrays and buffers, PROC INTPOINT uses various schemes that page information between auxiliary memory (often your machine's disk) and RAM.

For small problems, specify BYTES=100000. For large problems (those with hundreds of thousands or millions of variables), BYTES=1000000 might do. For solving problems of that size, if you are running on a machine with an inadequate amount of RAM, PROC INTPOINT's performance will suffer since it will be forced to page or to rely on virtual memory.

If you specify the MEMREP option, PROC INTPOINT will issue messages on the SAS log informing you of its memory usage; that is, how much memory is required to prevent paging, and details about the amount of paging that must be performed, if applicable.

#### **CON\_SINGLE\_OBS**

improves how the CONDATA= data set is read. How it works depends on whether the CONDATA has a dense or sparse format.

If the CONDATA= data set has the dense format, specifying CON\_SINGLE\_OBS indicates that, for each constraint, data for each can be found in only one observation of the CONDATA= data set.

If the CONDATA= data set has a sparse format, and data for each arc, nonarc variable, or LP variable can be found in only one observation of the CONDATA, then specify the CON\_SINGLE\_OBS option. If there are  $n$  SAS variables in the ROW and COEF list, then each arc or nonarc can have at most  $n$  constraint coefficients in the model. See the "How to Make the Data Read of PROC INTPOINT More Efficient" section on page 146.

#### **DEFCAPACITY=*c***

##### **DC=*c***

requests that the default arc capacity and the default nonarc variable value upper bound (or for LP problems, the default LP variable value upper bound) be  $c$ . If this option is not specified, then DEFCAPACITY= INFINITY.



**DEFCTYPE=*c*****DEFTYPE=*c*****DCT=*c***

specifies the default constraint type. This default constraint type is either *less than or equal to* or is the type indicated by DEFCTYPE=*c*. Valid values for this option are

LE, le, or <=     for *less than or equal to*

EQ, eq, or =       for *equal to*

GE, ge, or >=     for *greater than or equal to*

The values do not need to be enclosed in quotes.

**DEFCOST=*c***

requests that the default arc cost and the default nonarc variable objective function coefficient (or for an LP, the default LP variable objective function coefficient) be *c*. If this option is not specified, then DEFCOST=0.0.

**DEFMINFLOW=*m*****DMF=*m***

requests that the default lower flow bound through arcs and the default lower value bound of nonarc variables (or for an LP, the default lower value bound of LP variables) be *m*. If a value is not specified, then DEFMINFLOW=0.0.

**DEMAND=*d***

specifies the demand at the SINK node specified by the SINK= option. The DEMAND= option should be used only if the SINK= option is given in the PROC INTPOINT statement and neither the SHORTPATH option nor the MAXFLOW option is specified. If you are solving a minimum cost network problem and the SINK= option is used to identify the sink node, and the DEMAND= option is not specified, then the demand at the sink node is made equal to the network's total supply.

**GROUPED=*c***

PROC INTPOINT can take a much shorter time to read data if the data have been grouped prior to the PROC INTPOINT call. This enables PROC INTPOINT to conclude that, for instance, a new NAME list variable value seen in the ARCDATA= data set grouped by the values of the NAME list variable before PROC INTPOINT was called is new. PROC INTPOINT does not need to check that the NAME has been read in a previous observation. See the "How to Make the Data Read of PROC INTPOINT More Efficient" section on page 146.

- GROUPED=ARCDATA indicates that the ARCDATA= data set has been grouped by values of the NAME list variable. If `_NAME_` is the name of the NAME list variable, you could use:

```
proc sort data=arcdata;by _name_;
```

prior to calling PROC INTPOINT. Technically, you do not have to sort the data, only to ensure that all similar values of the NAME list variable are grouped

together. If you specify the ARCS\_ONLY\_ARCDATA option, PROC INTPOINT automatically works as if GROUPED=ARCDATA is also specified.

- GROUPED=CONDATA indicates that the CONDATA= data set has been grouped.

If the CONDATA= data set has a dense format, GROUPED=CONDATA indicates that the CONDATA= data set has been grouped by values of the ROW list variable. If `_ROW_` is the name of the ROW list variable, you could use:

```
proc sort data=condata;by _row_;
```

prior to calling PROC INTPOINT. Technically, you do not have to sort the data, only to ensure that all similar values of the ROW list variable are grouped together. If you specify the CON\_SINGLE\_OBS option, or if there is no ROW list variable, PROC INTPOINT automatically works as if GROUPED=CONDATA has been specified.

If the CONDATA has the sparse format, GROUPED=CONDATA indicates that the CONDATA has been grouped by values of the COLUMN list variable. If `_COL_` is the name of the COLUMN list variable, you could use:

```
proc sort data=condata;by _col_;
```

prior to calling PROC INTPOINT. Technically, you do not have to sort the data, only to ensure that all similar values of the COLUMN list variable are grouped together.

- GROUPED=BOTH indicates that both GROUPED=ARCDATA and GROUPED=CONDATA are TRUE.
- GROUPED=NONE indicates that the data sets have not been grouped, that is, neither GROUPED=ARCDATA nor GROUPED=CONDATA is TRUE. This is the default, but it is much better if GROUPED=ARCDATA, or GROUPED=CONDATA, or GROUPED=BOTH.

A data set like

```
... _XXXXX_ ....
    bbb
    bbb
    aaa
    ccc
    ccc
```

is a candidate for the GROUPED= option. Similar values are grouped together. When PROC INTPOINT is reading the  $i$ th observation, either the value of the `_XXXXX_` variable is the same as the  $(i - 1)$ th (that is, the previous observation's) `_XXXXX_` value, or it is a new `_XXXXX_` value not seen in any previous observation. This also means that if the  $i$ th `_XXXXX_` value is different from the  $(i - 1)$ th `_XXXXX_` value, the value of the  $(i - 1)$ th `_XXXXX_` variable will not be seen in any observations  $i$ ,  $i + 1$ , ... .

**INFINITY=*i*****INF=*i***

is the largest number used by PROC INTPOINT in computations. A number too small can adversely affect the solution process. You should avoid specifying an enormous value for the INFINITY= option because numerical roundoff errors can result. If a value is not specified, then INFINITY=999999. The INFINITY= option cannot be assigned a value less than 9999.

**MAXFLOW****MF**

specifies that PROC INTPOINT solve a maximum flow problem. In this case, the PROC INTPOINT procedure finds the maximum flow from the node specified by the SOURCE= option to the node specified by the SINK= option. PROC INTPOINT automatically assigns an INFINITY= option supply to the SOURCE= option node and the SINK= option is assigned the INFINITY= option demand. In this way, the MAXFLOW option sets up a maximum flow problem as an equivalent minimum cost problem.

You can use the MAXFLOW option when solving any flow problem (not necessarily a maximum flow problem) when the network has one supply node (with infinite supply) and one demand node (with infinite demand). The MAXFLOW option can be used in conjunction with all other options (except SHORTPATH, SUPPLY=, and DEMAND=) and capabilities of PROC INTPOINT.

**MAXIMIZE****MAX**

specifies that PROC INTPOINT find the maximum cost flow through the network. If both the MAXIMIZE and the SHORTPATH options are specified, the solution obtained is the longest path between the SOURCE= and SINK= nodes. Similarly, MAXIMIZE and MAXFLOW together cause PROC INTPOINT to find the minimum flow between these two nodes; this is zero if there are no nonzero lower flow bounds. If solving an LP, specifying the MAXIMIZE option is necessary if you want the maximal optimal solution found instead of the minimal optimum.

**MEMREP**

indicates that information on the memory usage and paging schemes (if necessary) is reported by PROC INTPOINT on the SAS log.

**NAMECTRL=*i***

is used to interpret arc and nonarc variable names in the CONDATA= data set. In the ARCDATA= data set, an arc is identified by its tail and head node. In the CONDATA= data set, arcs are identified by names. You can give a name to an arc by having a NAME list specification that indicates a SAS variable in the ARCDATA= data set that has names of arcs as values.

PROC INTPOINT requires that arcs that have information about them in the CONDATA= data set have names, but arcs that do not have information about them in the CONDATA= data set can also have names. Unlike a nonarc variable whose name uniquely identifies it, an arc can have several different names. An arc has a default name in the form *tail\_head*, that is, the name of the arc's tail node followed by an underscore and the name of the arc's head node.

In the CONDATA= data set, if the dense data format is used (described in the “CONDATA= Data Set” section on page 126), a name of an arc or a nonarc variable is the *name* of a SAS variable listed in the VAR list specification. If the sparse data format of the CONDATA= data set is used, a name of an arc or a nonarc variable is a *value* of the SAS variable listed in the COLUMN list specification.

The NAMECTRL= option is used when a name of an arc or a nonarc variable in the CONDATA= data set (either a VAR list variable name or a value of the COLUMN list variable) is in the form *tail\_head* and there exists an arc with these end nodes. If *tail\_head* has not already been tagged as belonging to an arc or nonarc variable in the ARCDATA= data set, PROC INTPOINT needs to know whether *tail\_head* is the name of the arc or the name of a nonarc variable.

If you specify NAMECTRL=1, a name that is not defined in the ARCDATA= data set is assumed to be the name of a nonarc variable. NAMECTRL=2 treats *tail\_head* as the name of the arc with these endnodes, provided no other name is used to associate data in the CONDATA= data set with this arc. If the arc does have other names that appear in the CONDATA= data set, *tail\_head* is assumed to be the name of a nonarc variable. If you specify NAMECTRL=3, *tail\_head* is assumed to be a name of the arc with these end nodes, whether the arc has other names or not. The default value of NAMECTRL is 3.

If the dense format is used for the CONDATA= data set, there are two circumstances that affect how this data set is read:

1. if you are running SAS Version 6, or a previous version to that, or if you are running SAS Version 7 onward and you specify

**options validvarname=v6;**

in your SAS session. Let's refer to this as *case 1*.

2. if you are running SAS Version 7 onward and you do not specify

**options validvarname=v6;**

in your SAS session. Let's refer to this as *case 2*.

If *case 1* (and the text in this and the five paragraphs that follow are relevant only for *case 1*), the SAS System converts SAS variable names in a SAS program to uppercase. The VAR list variable names are uppercased. Because of this, PROC INTPOINT automatically uppercases names of arcs and nonarc variables or LP variables (the values of the NAME list variable) in the ARCDATA= data set. The names of arcs and nonarc variables or LP variables (the values of the NAME list variable) appear uppercased in the CONOUT= data set.

Also, if the dense format is used for the CONDATA= data set, be careful with default arc names (names in the form *tailnode\_headnode*). Node names (values in the TAILNODE and HEADNODE list variables) in the ARCDATA= data set are not automatically uppercased by PROC INTPOINT. Consider the following code:

```

data arcdata;
    input _from_ $ _to_ $ _name $ ;
    datalines;
from to1 .
from to2 arc2
TAIL TO3 .
;
data densecon;
    input from_to1 from_to2 arc2 tail_to3;
    datalines;
2 3 5
;
proc intpoint
    arcdata=arcdata condata=densecon;
run;

```

The SAS System does not uppercase character string values within SAS data sets. PROC INTPOINT never uppercases node names, so the arcs in observations 1, 2, and 3 in the preceding ARCDATA= data set have the default names `from_to1`, `from_to2`, and `TAIL_TO3`, respectively. When the dense format of the CONDATA= data set is used, PROC INTPOINT does uppercase values of the NAME list variable, so the name of the arc in the second observation of the ARCDATA= data set is `ARC2`. Thus, the second arc has two names: its default `from_to2` and the other that was specified `ARC2`.

As the SAS System uppercases program code, you must think of the input statement

```
input from_to1 from_to2 arc2 tail_to3;
```

as really being

```
INPUT FROM_TO1 FROM_TO2 ARC2 TAIL_TO3;
```

The SAS variables named `FROM_TO1` and `FROM_TO2` are *not* associated with any of the arcs in the preceding ARCDATA= data set. The values `FROM_TO1` and `FROM_TO2` are different from all of the arc names `from_to1`, `from_to2`, `TAIL_TO3`, and `ARC2`. `FROM_TO1` and `FROM_TO2` could end up being the names of two nonarc variables.

The SAS variable named `ARC2` is the name of the second arc in the ARCDATA= data set, even though the name specified in the ARCDATA= data set looks like `arc2`. The SAS variable named `TAIL_TO3` is the default name of the third arc in the ARCDATA= data set.

If case 2, the SAS System does not convert SAS variable names in a SAS program to uppercase. The VAR list variable names are not uppercased. PROC INTPOINT does not automatically uppercase names of arcs and nonarc variables or LP variables (the values of the NAME list variable) in the ARCDATA= data set. PROC INTPOINT does not uppercase any SAS variable names, data set values, or indeed anything. Therefore, PROC INTPOINT respects case, and characters in the data if compared must have the right case if you mean them to be the same. Note how the input

statement in the data step that initialized the data set densecon below is specified in the following code:

```
data arcdata;
  input _from_ $ _to_ $ _name $ ;
  datalines;
from to1 .
from to2 arc2
TAIL T03 .
;
data densecon;
  input from_to1 from_to2 arc2 TAIL_T03;
  datalines;
2 3 5
;
proc intpoint
  arcdata=arcdata condata=densecon;
run;
```

**NARCS=*n***

specifies the approximate number of arcs. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 146.

**NCOEFS=*n***

specifies the approximate number of constraint coefficients. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 146.

**NCONS=*n***

specifies the approximate number of constraints. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 146.

**NNAS=*n***

specifies the approximate number of nonarc variables. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 146.

**NNODES=*n***

specifies the approximate number of nodes. See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 146.

**NON\_REPLIC=*c***

prevents PROC INTPOINT from doing unnecessary checks of data previously read.

- NON\_REPLIC=COEFS indicates that each constraint coefficient is specified *once* in the CONDATA= data set.
- NON\_REPLIC=NONE indicates that constraint coefficients can be specified more than once in the CONDATA= data set. NON\_REPLIC=NONE is the default.

See the “How to Make the Data Read of PROC INTPOINT More Efficient” section on page 146.

**RHSOBS=***charstr*

specifies the keyword that identifies a right-hand-side observation when using the sparse format for data in the CONDATA= data set. The keyword is expected as a value of the SAS variable in the CONDATA= data set named in the COLUMN list specification. The default value of the RHSOBS= option is `_RHS_` or `_rhs_`. If *charstr* is not a valid SAS variable name, enclose it in quotes.

**SCALE=***s*

indicates that the NPSC side constraints or the LP constraints are to be scaled. Scaling is useful when some coefficients are either much larger or much smaller than other coefficients. Scaling might make all coefficients have values that have a smaller range, and this can make computations more stable numerically. Try the SCALE= option if PROC INTPOINT is unable to solve a problem because of numerical instability. Specify

- SCALE=ROW, SCALE=CON, or SCALE=CONSTRAINT if you want the largest absolute value of coefficients in each constraint to be about 1.0
- SCALE=COL, SCALE=COLUMN, or SCALE=NONARC if you want NPSC nonarc variable columns or LP variable columns to be scaled so that the absolute value of the largest constraint coefficient of that variable is near to 1
- SCALE=BOTH if you want the largest absolute value of coefficients in each constraint, and the absolute value of the largest constraint coefficient of an NPSC nonarc variable or LP variable to be near to 1. This is the default.
- SCALE=NONE if no scaling should be done

**SHORTPATH****SP**

specifies that PROC INTPOINT solve a shortest path problem. The INTPOINT procedure finds the shortest path between the nodes specified in the SOURCE= option and the SINK= option. The costs of arcs are their *lengths*. PROC INTPOINT automatically assigns a supply of one flow unit to the SOURCE= node, and the SINK= node is assigned to have a one flow unit demand. In this way, the SHORTPATH option sets up a shortest path problem as an equivalent minimum cost problem.

If a network has one supply node (with supply of one unit) and one demand node (with demand of one unit), you could specify the SHORTPATH option, with the SOURCE= and SINK= nodes, even if the problem is not a shortest path problem. You then should not provide any supply or demand data in the NODEDATA= data set or the ARCDATA= data set.

**SINK=***sinkname***SINKNODE=***sinkname*

identifies the demand node. The SINK= option is useful when you specify the MAXFLOW option or the SHORTPATH option and you need to specify toward which node the shortest path or maximum flow is directed. The SINK= option also can be used when a minimum cost problem has only one demand node. Rather than having this information in the ARCDATA= data set or the NODEDATA= data set, use the SINK= option with an accompanying DEMAND= specification for this node.

The SINK= option must be the name of a head node of at least one arc; thus, it must have a character value. If the value of the SINK= option is not a valid SAS character variable name (if, for example, it contains embedded blanks), it must be enclosed in quotes.

**SOURCE=***sourcename*

**SOURCENODE=***sourcename*

identifies a supply node. The SOURCE= option is useful when you specify the MAXFLOW or the SHORTPATH option and need to specify from which node the shortest path or maximum flow originates. The SOURCE= option also can be used when a minimum cost problem has only one supply node. Rather than having this information in the ARCDATA= data set or the NODEDATA= data set, use the SOURCE= option with an accompanying SUPPLY= amount of supply at this node. The SOURCE= option must be the name of a tail node of at least one arc; thus, it must have a character value. If the value of the SOURCE= option is not a valid SAS character variable name (if, for example, it contains embedded blanks), it must be enclosed in quotes.

**SPARSECONDATA**

**SCDATA**

indicates that the CONDATA= data set has data in the sparse data format. Otherwise, it is assumed that the data are in the dense format.

**Note:** If the SPARSECONDATA option is not specified, and you are running SAS software Version 6 or you have specified

**options validvarname=v6;**

all NAME list variable values in the ARCDATA= data set are uppercased. See the “Case Sensitivity” section on page 136.

**SUPPLY=s**

specifies the supply at the source node specified by the SOURCE= option. The SUPPLY= option should be used only if the SOURCE= option is given in the PROC INTPOINT statement and neither the SHORTPATH option nor the MAXFLOW option is specified. If you are solving a minimum cost network problem and the SOURCE= option is used to identify the source node and the SUPPLY= option is not specified, then by default the supply at the source node is made equal to the network’s total demand.

**THRUNET**

tells PROC INTPOINT to force through the network any excess supply (the amount by which total supply exceeds total demand) or any excess demand (the amount by which total demand exceeds total supply) as is required. If a network problem has unequal total supply and total demand and the THRUNET option is not specified, PROC INTPOINT drains away the excess supply or excess demand in an optimal manner. The consequences of specifying or not specifying THRUNET are discussed in the “Balancing Total Supply and Total Demand” section on page 144.



**TYPEOBS=charstr**

specifies the keyword that identifies a type observation when using the sparse format for data in the CONDATA= data set. The keyword is expected as a value of the SAS variable in the CONDATA= data set named in the COLUMN list specification. The default value of the TYPEOBS= option is `_TYPE_` or `_type_`. If *charstr* is not a valid SAS variable name, enclose it in quotes.

**Options to Halt Optimization****VERBOSE=v**

limits the number of similar messages that are displayed on the SAS log.

For example, when reading the ARCDATA= data set, PROC INTPOINT might have cause to issue the following message many times:

**ERROR: The HEAD list variable value in obs i in the ARCDATA is missing, - the TAIL list variable value of this obs is nonmissing. This is an incomplete arc specification.**

If there are lots of observations that have this fault, messages that are similar are issued for only the first VERBOSE= such observations. After the ARCDATA= data set has been read, PROC INTPOINT will issue the message

**NOTE: More messages similar to the ones immediately above could have been issued but were suppressed as VERBOSE= v.**

If observations in the ARCDATA= data set have this error, PROC INTPOINT stops and you have to fix the data. Imagine that this error is only a warning and PROC INTPOINT proceeded to other operations such as reading the CONDATA= data set. If PROC INTPOINT finds there are numerous errors when reading that data set, the number of messages issued to the SAS log are also limited by the VERBOSE= option.

When PROC INTPOINT finishes and messages have been suppressed, the message

**NOTE: To see all messages, specify VERBOSE=v.**

is issued. The value of *v* is the smallest value that should be specified for the VERBOSE= option so that *all* messages are displayed if PROC INTPOINT is run again with the same data and everything else.

The default value for the VERBOSE= option is 12.

**ZERO2=z****Z2=z**

specifies the zero tolerance level used when determining whether the final solution has been reached. ZERO2= is also used when outputting the solution to the CONOUT= data set. Values within *z* of zero are set to 0.0, where *z* is the value of the ZERO2= option. Flows close to the lower flow bound or capacity of arcs are reassigned those exact values. If there are nonarc variables, values close to the lower or upper value bound of nonarc variables are reassigned those exact values. When solving an LP problem, values close to the lower or upper value bound of LP variables are reassigned those exact values.

The ZERO2= option works when determining whether optimality has been reached or whether an element in the vector  $(\Delta x^k, \Delta y^k, \Delta s^k)$  is less than or greater than zero. It is crucial to know that when determining the maximal value for the step length  $\alpha$  in the formula

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

See the description of the PDSTEPMULT= option for more details on this computation.

Two values are deemed to be close if one is within  $z$  of the other. The default value for the ZERO2= option is 0.000001. Any value specified for the ZERO2= option that is  $< 0.0$  or  $> 0.0001$  is not valid.

#### **ZEROTOL=z**

specifies the zero tolerance used when PROC INTPOINT must compare any real number with another real number, or zero. For example, if  $x$  and  $y$  are real numbers, then for  $x$  to be considered greater than  $y$ ,  $x$  must be at least  $y + \text{ZEROTOL}$ . The ZEROTOL= option is used throughout any PROC INTPOINT run.

ZEROTOL=z controls the way PROC INTPOINT performs all double precision comparisons; that is, whether a double precision number is equal to, not equal to, greater than (or equal to), or less than (or equal to) zero or some other double precision number. A double precision number is deemed to be the same as another such value if the absolute differences between them is less than or equal to the value of the ZEROTOL= option.

The default value for the ZEROTOL= option is 1.0E-14. You can specify the ZEROTOL= option in the INTPOINT statement. Valid values for the ZEROTOL= option must be  $> 0.0$  and  $< 0.0001$ . Do not specify a value too close to zero as this defeats the purpose of the ZEROTOL= option. Neither should the value be too large, as comparisons might be incorrectly performed.

#### **TOLDINF=t**

##### **RTOLDINF=t**

specifies the allowed amount of dual infeasibility. In the “Interior Point Algorithmic Details” section on page 58, the vector  $infeas_d$  is defined. If all elements of this vector are  $\leq t$ , the solution is considered dual feasible.  $infeas_d$  is replaced by a zero vector, which makes computations faster. This option is the dual equivalent to the TOLPINF= option. Increasing the value of the TOLDINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are greater than  $1.0E - 12$ . The default is  $1.0E - 7$ .

#### **TOLPINF=t**

##### **RTOLPINF=t**

specifies the allowed amount of primal infeasibility. This option is the dual equivalent to the TOLDINF= option. In the “Interior Point: Upper Bounds” section on page 63, the vector  $infeas_b$  is defined. In the “Interior Point Algorithmic Details” section on page 58, the vector  $infeas_c$  is defined. If all elements in these vectors are  $\leq t$ , the solution is considered primal feasible.  $infeas_b$  and  $infeas_c$  are replaced by zero

vectors, which makes computations faster. Increasing the value of the TOLPINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are greater than  $1.0E - 12$ . The default is  $1.0E - 7$ .

**TOLTOTDINF= $t$**

**RTOLTOTDINF= $t$**

specifies the allowed total amount of dual infeasibility. In the “Interior Point Algorithmic Details” section on page 58, the vector  $infeas_d$  is defined. If  $\sum_{i=1}^n infeas_{di} \leq t$ , the solution is considered dual feasible.  $infeas_d$  is replaced by a zero vector, which makes computations faster. This option is the dual equivalent to the TOLTOTPINF= option. Increasing the value of the TOLTOTDINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are greater than  $1.0E - 12$ . The default is  $1.0E - 7$ .

**TOLTOTPINF= $t$**

**RTOLTOTPINF= $t$**

specifies the allowed total amount of primal infeasibility. This option is the dual equivalent to the TOLTOTDINF= option. In the “Interior Point: Upper Bounds” section on page 63, the vector  $infeas_b$  is defined. In the “Interior Point Algorithmic Details” section on page 58, the vector  $infeas_c$  is defined. If  $\sum_{i=1}^n infeas_{bi} \leq t$  and  $\sum_{i=1}^m infeas_{ci} \leq t$ , the solution is considered primal feasible.  $infeas_b$  and  $infeas_c$  are replaced by zero vectors, which makes computations faster. Increasing the value of the TOLTOTPINF= option too much can lead to instability, but a modest increase can give the algorithm added flexibility and decrease the iteration count. Valid values for  $t$  are greater than  $1.0E - 12$ . The default is  $1.0E - 7$ .

**CHOLTINYTOL= $c$**

**RCHOLTINYTOL= $c$**

specifies the cut-off tolerance for Cholesky factorization of the  $A\Theta A^{-1}$ . If a diagonal value drops below  $c$ , the row is essentially treated as dependent and is ignored in the factorization. Valid values for  $c$  are between  $1.0E - 30$  and  $1.0E - 6$ . The default value is  $1.0E - 8$ .

**DENSETHR= $d$**

**RDENSETHR= $d$**

specifies the density threshold for Cholesky factorization. When the symbolic factorization encounters a column of  $L$  (where  $L$  is the remaining unfactorized submatrix) that has DENSETHR= proportion of nonzeros and the remaining part of  $L$  is at least  $12 \times 12$ , the remainder of  $L$  is treated as dense. In practice, the lower right part of the Cholesky triangular factor  $L$  is quite dense and it can be computationally more efficient to treat it as 100% dense. The default value for  $d$  is 0.7. A specification of  $d \leq 0.0$  causes all dense processing;  $d \geq 1.0$  causes all sparse processing.

**PDSTEPMULT= $p$** **RPDSTEPMULT= $p$** 

specifies the step-length multiplier. The maximum feasible step-length chosen by the Interior point algorithm is multiplied by the value of the PDSTEPMULT= option. This number must be less than 1 to avoid moving beyond the barrier. An actual step-length greater than 1 indicates numerical difficulties. Valid values for  $p$  are between 0.01 and 0.999999. The default value is 0.99995.

In the “Interior Point Algorithmic Details” section on page 58, the solution of the next iteration is obtained by moving along a direction from the current iteration’s solution.

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

Let  $\alpha = \text{Min}_i \{\alpha : x_i^k + \alpha \Delta x = 0 \mid s_i^k + \alpha \Delta s = 0\}$ . If  $\alpha \leq 1$ , then  $\alpha = p\alpha$ .  $\alpha$  is a value as large as possible but  $\leq 1.0$  and not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  of some variable  $i$  is not “too close” to zero.

**PRSLTYPE= $p$** **IPRSLTYPE= $p$** 

Preprocessing the Linear Programming problem often succeeds in allowing some variables and constraints to be temporarily eliminated from the resulting LP that must be solved. This reduces the solution time and possibly also the chance that the optimizer will run into numerical difficulties. The task of preprocessing is inexpensive to do.

You control how much preprocessing to do by specifying the PRSLTYPE=  $p$ , where  $p$  can be -1, 0, 1, 2, or 3:

- |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -1 | Do not perform preprocessing. For most problems, specifying PRSLTYPE=-1 is <i>not</i> recommended.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 0  | Given upper and lower bounds on each variable, the greatest and least contribution to the row activity of each variable is computed. If these are within the limits set by the upper and lower bounds on the row activity, then the row is redundant and can be discarded. Otherwise, whenever possible, the bounds on any of the variables are tightened. For example, if all coefficients in a constraint are positive and all variables have zero lower bounds, then the row’s smallest contribution is zero. If the rhs value of this constraint is zero, then if the constraint type is = or $\leq$ , all the variables in that constraint are fixed to zero. These variables and the constraint are removed. If the constraint type is $\geq$ , the constraint is redundant. If the rhs is negative and the constraint is $\leq$ , the problem is infeasible. If just one variable in a row is not fixed, the row is used to impose an implicit upper or lower bound on the variable and then this row is eliminated. The preprocessor also tries to tighten the bounds on constraint right hand sides. |
| 1  | When there are exactly two unfixed variables with coefficients in an equality constraint, one variable is solved in terms of the other.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

The problem will have one less variable. The new matrix will have at least two fewer coefficients and one less constraint. In other constraints where both variables appear, two coefficients are combined into one. PRSLTYPE=0 reductions are also done.

- 2 It may be possible to determine that an equality constraint is not constraining a variable. That is, if all variables are non-negative, then  $x - \sum_i y_i = 0$  does not constrain  $x$ , since it must be non-negative if all the  $y_i$ 's are non-negative. In this case,  $x$  is eliminated by subtracting this equation from all others containing  $x$ . This is useful when the only other entry for  $x$  is in the objective function. This reduction is performed if there is at most one other nonobjective coefficient. PRSLTYPE=0 reductions are also done.
- 3 All possible reductions are performed. PRSLTYPE=3 is the default.

Preprocessing is iterative. As variables are fixed and eliminated, and constraints are found to be redundant and they too are eliminated, and as variable bounds and constraint right hand sides are tightened, the LP to be optimized is modified to reflect these changes. Another iteration of preprocessing of the modified LP may reveal more variables and constraints that are eliminated, or tightened.

#### **PRINTLEVEL2=*p***

is used when you want to see PROC INTPOINT's progress to the optimum. PROC INTPOINT will produce a table on the SAS log. A row of the table is generated during each iteration and may consist of values of

- the affine step complementarity
- the complementarity of the solution for the next iteration
- the total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the *infeas<sub>b</sub>* array in the "Interior Point: Upper Bounds" section on page 63)
- the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the *infeas<sub>c</sub>* array in the "Interior Point Algorithmic Details" section on page 58)
- the total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the *infeas<sub>d</sub>* array in the "Interior Point Algorithmic Details" section on page 58)

As optimization progresses, the values in all columns should converge to zero. If you specify PRINTLEVEL2=2, all columns will appear in the table. If PRINTLEVEL2=1 is specified, only the affine step complementarity, the complementarity of the solution for the next iteration, will appear. Some time is saved by not calculating the infeasibility values.

PRINTLEVEL2=2 is specified in all PROC INTPOINT runs in the "Examples" section on page 155.

#### **RTTOL=*r***

specifies the zero tolerance used during the ratio test of the Interior Point algorithm. The ratio test determines  $\alpha$ , the maximum feasible step-length.

Valid values for  $r$  are greater than  $1.0\text{e-}14$ . The default value is  $1.0\text{e-}10$ .

In the “Interior Point Algorithmic Details” section on page 58, the solution of the next iteration is obtained by moving along a direction from the current iteration’s solution.

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha(\Delta x^k, \Delta y^k, \Delta s^k)$$

Let  $\alpha = \text{Min}_i\{\alpha : x_i^k + \alpha\Delta x_i = 0 | s_i^k + \alpha\Delta s_i = 0\}$ . If  $\alpha \leq 1$ , then  $\alpha$  is reduced slightly by multiplying it by PDSTEPMULT=.  $\alpha$  is a value as large as possible but  $\leq 1.0$  and not so large that a  $x_i^{k+1}$  or  $s_i^{k+1}$  of some variable  $i$  is negative. When determining  $\alpha$ , only negative elements of  $\Delta x$  and  $\Delta s$  are important.

RTTOL= $r$  indicates a number close to zero so that another number  $n$  is considered truly negative if  $n \leq -r$ . Even though  $n < 0$ , if  $n > -r$ ,  $n$  may be too close to zero and may have the wrong sign due to rounding error.

### Interior Point Algorithm Options: Stopping Criteria

**MAXITERB= $m$**

**IMAXITERB= $m$**

specifies the maximum number of iterations that the Interior point algorithm can perform. The default value for  $m$  is 100. One of the most remarkable aspects of the Interior Point algorithm is that for most problems, it usually needs to do a small number of iterations, *no matter the size of the problem*.

**PDGAPTOL= $p$**

**RPDGAPTOL= $p$**

specifies the Primal-Dual gap or Duality gap tolerance. Duality gap is defined in the “Interior Point Algorithmic Details” section on page 58. If the relative gap ( $\text{dualitygap}/(c^T x)$ ) between the primal and dual objectives is smaller than the value of the PDGAPTOL= option and both the primal and dual problems are feasible, then PROC INTPOINT stops optimization with a solution that is deemed optimal. Valid values for  $p$  are between  $1.0E - 12$  and  $1.0E - 1$ . the default is  $1.0E - 7$ .

**STOP\_C= $s$**

is used to determine whether optimization should stop. At the beginning of each iteration, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.

**STOP\_DG= $s$**

is used to determine whether optimization should stop. At the beginning of each iteration, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.

**STOP\_IB= $s$**

is used to determine whether optimization should stop. At the beginning of each iteration, if total bound infeasibility  $\sum_{i=1}^n \text{infeas}_{bi}$  (see the  $\text{infeas}_b$  array in the “Interior Point: Upper Bounds” section on page 63; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the

“Stopping Criteria” section on page 151.

#### **STOP\_IC=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 58; this value appears in the Tot\_infeas<sub>c</sub> column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.

#### **STOP\_ID=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 58; this value appears in the Tot\_infeas<sub>d</sub> column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.

#### **AND\_STOP\_C=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , *and* the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.

#### **AND\_STOP\_DG=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , *and* the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.

#### **AND\_STOP\_IB=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 63; this value appears in the Tot\_infeas<sub>b</sub> column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $\leq s$ , *and* the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.

#### **AND\_STOP\_IC=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 58; this value appears in the Tot\_infeas<sub>c</sub> column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , *and* the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.



**AND\_STOP\_ID=s**

is used to determine whether optimization should stop. At the beginning of each iteration, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 58; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $\leq s$ , and the other conditions related to other AND\_STOP parameters are also satisfied, optimization will stop. This option is discussed in the “Stopping Criteria” section on page 151.

**KEEPGOING\_C=s**

is used to determine whether optimization should stop. If a stopping condition is met, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

**KEEPGOING\_DG=s**

is used to determine whether optimization should stop. If a stopping condition is met, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

**KEEPGOING\_IB=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 63; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

**KEEPGOING\_IC=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 58; this value appears in the Tot\_infeasc column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

**KEEPGOING\_ID=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 58; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

**AND\_KEEPGOING\_C=s**

is used to determine whether optimization should stop. If a stopping condition is met, if complementarity (the value of the Complem-ity column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.



**AND\_KEEPPGOING\_DG=s**

is used to determine whether optimization should stop. If a stopping condition is met, if the duality gap (the value of the Duality\_gap column in the table produced when you specify PRINTLEVEL2=1 or PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

**AND\_KEEPPGOING\_IB=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 63; this value appears in the Tot\_infeasb column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

**AND\_KEEPPGOING\_IC=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 58; this value appears in the Tot\_infeasc column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

**AND\_KEEPPGOING\_ID=s**

is used to determine whether optimization should stop. If a stopping condition is met, if total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 58; this value appears in the Tot\_infeasd column in the table produced when you specify PRINTLEVEL2=2) is  $> s$ , and the other conditions related to other AND\_KEEPPGOING parameters are also satisfied, optimization will continue. This option is discussed in the “Stopping Criteria” section on page 151.

---

**CAPACITY Statement**

**CAPACITY** *variable* ;

**CAPAC** *variable* ;

**UPPERBD** *variable* ;

The CAPACITY statement identifies the SAS variable in the ARCDATA= data set that contains the maximum feasible flow or capacity of the network arcs. If an observation contains nonarc variable information, the CAPACITY list variable is the upper value bound for the nonarc variable named in the NAME list variable in that observation.

When solving an LP, the CAPACITY statement identifies the SAS variable in the

ARCDATA= data set that contains the maximum feasible value of the LP variables.

The CAPACITY list variable must have numeric values. It is not necessary to have a CAPACITY statement if the name of the SAS variable is `_CAPAC_`, `_UPPER_`, `_UPPERBD`, or `_HI_`.

---

## COEF Statement

**COEF** *variables* ;

The COEF list is used with the sparse input format of the CONDATA= data set. The COEF list can contain more than one SAS variable, each of which must have numeric values. If the COEF statement is not specified, the CONDATA= data set is searched and SAS variables with names beginning with `_COE` are used. The number of SAS variables in the COEF list must be no greater than the number of SAS variables in the ROW list.

The values of the COEF list variables in an observation can be interpreted differently than these variables' values in other observations. The values can be coefficients in the side constraints, costs and objective function coefficients, bound data, constraint type data, or rhs data. If the COLUMN list variable has a value that is a name of an arc or a nonarc variable, the *i*th COEF list variable is associated with the constraint or special row name named in the *i*th ROW list variable. Otherwise, the COEF list variables indicate type values, rhs values, or missing values.

When solving an LP, the values of the COEF list variables in an observation can be interpreted differently than these variables' values in other observations. The values can be coefficients in the constraints, objective function coefficients, bound data, constraint type data, or rhs data. If the COLUMN list variable has a value that is a name of an LP variable, the *i*th COEF list variable is associated with the constraint or special row name named in the *i*th ROW list variable. Otherwise, the COEF list variables indicate type values, rhs values, or missing values.

---

## COLUMN Statement

**COLUMN** *variable* ;

The COLUMN list is used with the sparse input format of the CONDATA= data set.

This list consists of one SAS variable in the CONDATA= data set that has as values the names of arc variables, nonarc variables, or missing values. When solving an LP, this list consists of one SAS variable in the CONDATA= data set that has as values the names of LP variables, or missing values. Some, if not all, of these values also can be values of the NAME list variables of the ARCDATA= data set. The COLUMN list variable can have other special values (Refer to the TYPEOBS= and RHSOBS= options). If the COLUMN list is not specified after the PROC INTPOINT statement, the CONDATA= data set is searched and a SAS variable named `_COLUMN_` is

used. The COLUMN list variable must have character values.

---

## COST Statement

**COST** *variable* ;  
**OBJFN** *variable* ;

The COST statement identifies the SAS variable in the ARCDATA= data set that contains the per unit flow cost through an arc. If an observation contains nonarc variable information, the value of the COST list variable is the objective function coefficient of the nonarc variable named in the NAME list variable in that observation.

If solving an LP, the COST statement identifies the SAS variable in the ARCDATA= data set that contains the per unit objective function coefficient of an LP variable named in the NAME list variable in that observation.

The COST list variable must have numeric values. It is not necessary to specify a COST statement if the name of the SAS variable is `_COST_` or `_LENGTH_`.

---

## DEMAND Statement

**DEMAND** *variable* ;

The DEMAND statement identifies the SAS variable in the ARCDATA= data set that contains the demand at the node named in the corresponding HEADNODE list variable. The DEMAND list variable must have numeric values. It is not necessary to have a DEMAND statement if the name of this SAS variable is `_DEMAND_`. See the “Missing S Supply and Missing D Demand Values” section on page 139 for cases when the SUPDEM list variable values can have other values. There should be no DEMAND statement if you are solving an LP.

---

## HEADNODE Statement

**HEADNODE** *variable* ;  
**HEAD** *variable* ;  
**TONODE** *variable* ;  
**TO** *variable* ;

The HEADNODE statement specifies the SAS variable that must be present in the ARCDATA= data set that contains the names of nodes toward which arcs are directed. It is not necessary to have a HEADNODE statement if the name of the SAS variable

is `_HEAD_` or `_TO_`. The `HEADNODE` variable must have character values.

There should be no `HEAD` statement if you are solving an LP.

---

## ID Statement

**ID** *variables* ;

The ID statement specifies SAS variables containing values for pre- and post-optimal processing and analysis. These variables are not processed by PROC INTPOINT but are read by the procedure and written in the `CONOUT=` data set. For example, imagine a network used to model a distribution system. The SAS variables listed on the ID statement can contain information on the type of vehicle, the transportation mode, the condition of the road, the time to complete the journey, the name of the driver, or other ancillary information useful for report writing or describing facets of the operation that do not have bearing on the optimization. The ID variables can be character, numeric, or both.

If no ID list is specified, the procedure forms an ID list of all SAS variables not included in any other implicit or explicit list specification. If the ID list is specified, any SAS variables in the `ARCDATA=` data set not in any list are dropped and do not appear in the `CONOUT=` data set.

---

## LO Statement

**LO** *variable* ;  
**LOWERBD** *variable* ;  
**MINFLOW** *variable* ;

The LO statement identifies the SAS variable in the `ARCDATA=` data set that contains the minimum feasible flow or lower flow bound for arcs in the network. If an observation contains nonarc variable information, the LO list variable has the value of the lower bound for the nonarc variable named in the `NAME` list variable. If solving an LP, the LO statement identifies the SAS variable in the `ARCDATA=` data set that contains the lower value bound for LP variables. The LO list variables must have numeric values. It is not necessary to have a LO statement if the name of this SAS variable is `_LOWER_`, `_LO_`, `_LOWERBD`, or `_MINFLOW`.

---

## NAME Statement

**NAME** *variable* ;  
**ARCNAME** *variable* ;  
**VARNAME** *variable* ;

Each arc and nonarc variable in an NPSC, or each variable in an LP, that has data in the CONDATA= data set must have a unique name. This name is a value of the NAME list variable. The NAME list variable must have character values (see the NAMECTRL= option in the PROC INTPOINT statement for more information). It is not necessary to have a NAME statement if the name of this SAS variable is \_NAME\_.

---

## NODE Statement

**NODE** *variable* ;

The NODE list variable, which must be present in the NODEDATA= data set, has names of nodes as values. These values must also be values of the TAILNODE list variable, the HEADNODE list variable, or both. If this list is not explicitly specified, the NODEDATA= data set is searched for a SAS variable with the name \_NODE\_. The NODE list variable must have character values.

---

## QUIT Statement

**QUIT**;

The QUIT statement indicates that PROC INTPOINT is to stop immediately. The solution is not saved in the CONOUT= data set. The QUIT statement has no options.

---

## RHS Statement

**RHS** *variable* ;

The RHS variable list is used when the dense format of the CONDATA= data set is used. The values of the SAS variable specified in the RHS list are constraint right-hand-side values. If the RHS list is not specified, the CONDATA= data set is searched and a SAS variable with the name \_RHS\_ is used. The RHS list variable must have numeric values. If there is no RHS list and no SAS variable named \_RHS\_, all constraints are assumed to have zero right-hand-side values.

---

## ROW Statement

**ROW** *variables* ;

The ROW list is used when either the sparse or dense format of the CONDATA= data set is being used. SAS variables in the ROW list have values that are constraint or special row names. The SAS variables in the ROW list must have character values.

If the dense data format is used, there must be only one SAS variable in this list. In this case, if a ROW list is not specified, the CONDATA= data set is searched and the SAS variable with the name `_ROW_` or `_CON_` is used. If that search fail to find a suitable SAS variable, data for each constraint must reside in only one observation.

If the sparse data format is used and the ROW statement is not specified, the CONDATA= data set is searched and SAS variables with names beginning with `_ROW` or `_CON` are used. The number of SAS variables in the ROW list must not be less than the number of SAS variables in the COEF list. The  $i$ th ROW list variable is paired with the  $i$ th COEF list variable. If the number of ROW list variables is greater than the number of COEF list variables, the last ROW list variables have no COEF partner. These ROW list variables that have no corresponding COEF list variable are used in observations that have a TYPE list variable value. All ROW list variable values are tagged as having the type indicated. If there is no TYPE list variable, all ROW list variable values are constraint names.

---

## RUN Statement

**RUN**;

The RUN statement causes optimization to be started. The RUN statement has no options. If PROC INTPOINT is called and is not terminated because of an error or a QUIT statement, and you have not used a RUN statement, a RUN statement is assumed implicitly as the last statement of PROC INTPOINT. Therefore, PROC INTPOINT reads that data, performs optimization, and saves the optimal solution in the CONOUT= data set.

---

## SUPDEM Statement

**SUPDEM** *variable* ;

The SAS variable in this list, which must be present in the NODEDATA= data set, contains supply and demand information for the nodes in the NODE list. A positive SUPDEM list variable value  $s$  ( $s > 0$ ) denotes that the node named in the NODE list variable can supply  $s$  units of flow. A negative SUPDEM list variable value  $-d$  ( $d > 0$ ) means that this node demands  $d$  units of flow. If a SAS variable is not explicitly specified, a SAS variable with the name `_SUPDEM_` or `_SD_` in the NODEDATA=

data set is used as the SUPDEM variable. If a node is a transshipment node (neither a supply nor a demand node), an observation associated with this node need not be present in the NODEDATA= data set. If present, the SUPDEM list variable value must be zero or a missing value. See the “Missing S Supply and Missing D Demand Values” section on page 139 for cases when the SUPDEM list variable values can have other values.

---

## SUPPLY Statement

**SUPPLY** *variable* ;

The SUPPLY statement identifies the SAS variable in the ARCDATA= data set that contains the supply at the node named in that observation’s TAILNODE list variable. If a tail node does not supply flow, use zero or a missing value for the observation’s SUPPLY list variable value. If a tail node has supply capability, a missing value indicates that the supply quantity is given in another observation. It is not necessary to have a SUPPLY statement if the name of this SAS variable is `_SUPPLY_`. See the “Missing S Supply and Missing D Demand Values” section on page 139 for cases when the SUPDEM list variable values can have other values. There should be no SUPPLY statement if you are solving an LP.

---

## TAILNODE Statement

**TAILNODE** *variable* ;

**TAIL** *variable* ;

**FROMNODE** *variable* ;

**FROM** *variable* ;

The TAILNODE statement specifies the SAS variable that must (when solving an NPSC problem) be present in the ARCDATA= data set that has as values the names of tail nodes of arcs. The TAILNODE variable must have character values. It is not necessary to have a TAILNODE statement if the name of the SAS variable is `_TAIL_` or `_FROM_`. If the TAILNODE list variable value is missing, it is assumed that the observation of the ARCDATA= data set contains information concerning a nonarc variable. There should be no TAILNODE statement if you are solving an LP.

---

## TYPE Statement

```
TYPE variable ;  
CONTYPE variable ;
```

The TYPE list, which is optional, names the SAS variable that has as values keywords that indicate either the constraint type for each constraint or the type of special rows in the CONDATA= data set. The values of the TYPE list variable also indicate, in each observation of the CONDATA= data set, how values of the VAR or COEF list variables are to be interpreted and how the type of each constraint or special row name is determined. If the TYPE list is not specified, the CONDATA= data set is searched and a SAS variable with the name `_TYPE_` is used. Valid keywords for the TYPE variable are given below. If there is no TYPE statement and no other method is used to furnish type information (see the DEFCONTYPE= option), all constraints are assumed to be of the type “less than or equal to” and no special rows are used. The TYPE list variable must have character values and can be used when the data in the CONDATA= data set is in either the sparse or dense format. If the TYPE list variable value has a \* as its first character, the observation is ignored because it is a comment observation.



**TYPE List Variable Values**

The following are valid TYPE list variable values. The letters in boldface denote the characters that PROC INTPOINT uses to determine what type the value suggests. You need to have at least these characters. Below, the minimal TYPE list variable values have additional characters to aid you in remembering these values.

The valid TYPE list variable values are

|                         |                                                                                |
|-------------------------|--------------------------------------------------------------------------------|
| <                       | less than or equal to ( $\leq$ )                                               |
| =                       | equal to (=)                                                                   |
| >                       | greater than or equal to ( $\geq$ )                                            |
| <b>CAPAC</b>            | capacity                                                                       |
| <b>COST</b>             | cost                                                                           |
| <b>EQ</b>               | equal to                                                                       |
| <b>FREE</b>             | free row (used only for Linear Programs solved by Interior Point)              |
| <b>GE</b>               | greater than or equal to                                                       |
| <b>LE</b>               | less than or equal to                                                          |
| <b>LOWERBD</b>          | lower flow or value bound                                                      |
| <b>LOW</b> <i>blank</i> | lower flow or value bound                                                      |
| <b>MAXIMIZE</b>         | maximize (opposite of cost)                                                    |
| <b>MINIMIZE</b>         | minimize (same as cost)                                                        |
| <b>OBJECTIVE</b>        | objective function (same as cost)                                              |
| <b>RHS</b>              | rhs of constraint                                                              |
| <b>TYPE</b>             | type of constraint                                                             |
| <b>UPPCOST</b>          | reserved for future use                                                        |
| <b>UNREST</b>           | unrestricted variable (used only for Linear Programs solved by Interior Point) |
| <b>UPPER</b>            | upper value bound or capacity; second letter must not be N                     |

The valid TYPE list variable values in function order are

- **LE** less than or equal to ( $\leq$ )
- **EQ** Equal to (=)
- **GE** Greater than or equal to ( $\geq$ )
- **COST**  
**MINIMIZE**  
**MAXIMIZE**  
**OBJECTIVE**  
cost or objective function coefficient
- **CAPAC**  
**UPPER** capacity or upper value bound

- **LOWERBD**  
**LOW***blank* Lower flow or value bound
- **RHS** rhs of constraint
- **TYPE** type of constraint

A TYPE list variable value that has the first character \* causes the observation to be treated as a comment. If the first character is a negative sign, then  $\leq$  is the type. If the first character is a zero, then  $=$  is the type. If the first character is a positive number, then  $\geq$  is the type.

---

## VAR Statement

**VAR** *variables* ;

The VAR variable list is used when the dense data format is used for the CONDATA= data set. The names of these SAS variables are also names of the arc and nonarc variables that have data in the CONDATA= data set. If solving an LP, the names of these SAS variables are also names of the LP variables. If no explicit VAR list is specified, all numeric SAS variables in the CONDATA= data set that are not in other SAS variable lists are put onto the VAR list. The VAR list variables must have numeric values. The values of the VAR list variables in some observations can be interpreted differently than in other observations. The values can be coefficients in the side constraints, costs and objective function coefficients, or bound data. When solving an LP, the values of the SAS variables in the VAR list can be constraint coefficients, objective function coefficients, or bound data. How these numeric values are interpreted depends on the value of each observation's TYPE or ROW list variable value. If there are no TYPE list variables, the VAR list variable values are all assumed to be side constraint coefficients.

## Details

### Input Data Sets

PROC INTPOINT is designed so that there are as few rules as possible that you must obey when inputting a problem's data. Raw data are acceptable. This should cut the amount of processing required to groom the data before it is input to PROC INTPOINT. Data formats are so flexible that, due to space restrictions, all possible forms for a problem's data are not shown here. Try any reasonable form for your problem's data; it should be acceptable. PROC INTPOINT will outline its objections.

You can supply the same piece of data several ways. You do not have to restrict yourself to using any particular one. If you use several ways, PROC INTPOINT checks that the data are consistent each time that the data are encountered. After all input data sets have been read, data are merged so that the problem is described completely. The observations can be in any order.

#### **ARCDATA= Data Set**

See the "Getting Started: NPSC Problems" section on page 74 and the "Introductory NPSC Example" section on page 75 for a description of this input data set.

**Note:** Information for an arc or nonarc variable can be specified in more than one observation. For example, consider an arc directed from node A toward node B that has a cost of 50, capacity of 100, and lower flow bound of 10 flow units. Some possible observations in the ARCDATA= data set are

| <b>_tail_</b> | <b>_head_</b> | <b>_cost_</b> | <b>_capac_</b> | <b>_lo_</b> |
|---------------|---------------|---------------|----------------|-------------|
| A             | B             | 50            | .              | .           |
| A             | B             | .             | 100            | .           |
| A             | B             | .             | .              | 10          |
| A             | B             | 50            | 100            | .           |
| A             | B             | .             | 100            | 10          |
| A             | B             | 50            | .              | 10          |
| A             | B             | 50            | 100            | 10          |

Similarly, for a nonarc variable that has an upper bound of 100, a lower bound of 10, and an objective function coefficient of 50, the **\_TAIL\_** and **\_HEAD\_** values are missing.

When solving an LP that has an LP variable named **my\_var** with an upper bound of 100, a lower bound of 10, and an objective function coefficient of 50, some possible observations in the ARCDATA= data set are

| <b>_name_</b> | <b>_cost_</b> | <b>_capac_</b> | <b>_lo_</b> |
|---------------|---------------|----------------|-------------|
| my_var        | 50            | .              | .           |
| my_var        | .             | 100            | .           |
| my_var        | .             | .              | 10          |
| my_var        | 50            | 100            | .           |
| my_var        | .             | 100            | 10          |
| my_var        | 50            | .              | 10          |
| my_var        | 50            | 100            | 10          |

**CONDATA= Data Set**

Regardless of whether the data in the CONDATA= data set is in the sparse or dense format, you will receive a warning if PROC INTPOINT finds a constraint row that has no coefficients. You will also be warned if any nonarc or LP variable has no constraint coefficients.

**Dense Input Format**

If the dense format is used, most SAS variables in the CONDATA= data set belong to the VAR list. The names of the SAS variables belonging to this list have names of arc and nonarc variables or, if solving an LP, names of the LP variables. These names can be values of the SAS variables in the ARCDATA= data set that belong to the NAME list, or names of nonarc variables, or names in the form *tail\_head*, or any combination of these three forms. Names in the form *tail\_head* are default arc names, and if you use them, you must specify node names in the ARCDATA= data set (values of the TAILNODE and HEADNODE list variables).

The CONDATA= data set can have three other SAS variables belonging, respectively, to the ROW, the TYPE, and the RHS lists. The CONDATA= data set of the oil industry example in the “Introductory NPSC Example” section on page 75 uses the dense data format.

Consider the SAS code that creates a dense format CONDATA= data set that has data for three constraints. This data set was used in the “Introductory NPSC Example” section on page 75:

```
data cond1;
  input m_e_ref1 m_e_ref2 thruput1 r1_gas thruput2 r2_gas
        _type_ $ _rhs_;
  datalines;
-2   .   1   .   .   .   >=   -15
.   -2   .   .   1   .   GE   -15
.   .   -3  4   .   .   EQ    0
.   .   .   .   -3  4   =     0
;
```

You can use nonconstraint type values to furnish data on costs, capacities, lower flow bounds (and, if there are nonarc or LP variables, objective function coefficients and upper and lower bounds). You need not have such (or as much) data in the ARCDATA= data set. The first three observations in the following data set are examples of observations that provide cost, capacity, and lower bound data:

```

data cond1b;
  input m_e_ref1 m_e_ref2 thrupt1 r1_gas thrupt2 r2_gas
        _type_ $ _rhs_;
  datalines;
63 81 200 . 220 . cost .
95 80 175 140 100 100 capac .
20 10 50 . 35 . lo .
-2 . 1 . . . >= -15
. -2 . . 1 . GE -15
. . -3 4 . . EQ 0
. . . . -3 4 = 0
;

```

If a ROW list variable is used, the data for a constraint can be spread over more than one observation. To illustrate, the data for the first constraint (which is called con1) and the cost and capacity data (in special rows called costrow and caprow, respectively) will be spread over more than one observation in the following data set:

```

data cond1b;
  input _row_ $
        m_e_ref1 m_e_ref2 thrupt1 r1_gas thrupt2 r2_gas
        _type_ $ _rhs_;
  datalines;
costrow 63 . . . . .
costrow . 81 200 . . . cost .
. . . . 220 . cost .
caprow . . . . . capac .
caprow 95 . 175 . 100 100 .
caprow . 80 175 140 . . .
lorow 20 10 50 . 35 . lo .
con1 -2 . 1 . . . .
con1 . . . . . >= -15
con2 . -2 . . 1 . GE -15
con3 . . -3 4 . . EQ 0
con4 . . . . -3 4 = 0
;

```

Using both ROW and TYPE lists, you can use special row names. Examples of these are costrow and caprow in the last data set. It should be restated that in any of the input data sets of PROC INTPOINT, the order of the observations does not matter. However, the CONDATA= data set can be read more quickly if PROC INTPOINT knows what type of constraint or special row a ROW list variable value is. For example, when the first observation is read, PROC INTPOINT does not know whether COSTROW is a constraint or special row and how to interpret the value 63 for the arc with the name m\_e\_ref1. When PROC INTPOINT reads the second observation, it learns that COSTROW has cost type and that the values 81 and 200 are costs. When the entire CONDATA= data set has been read, PROC INTPOINT knows the type of all special rows and constraints. Data that PROC INTPOINT had to set aside (such as the first observation 63 value and the COSTROW ROW list variable value, which at the time had unknown type, but is subsequently known to be a cost special row) is reprocessed. During this second pass, if a ROW list variable value has unassigned

constraint or special row type, it is treated as a constraint with DEFCONTYPE= (or DEFCONTYPE= default) type. Associated VAR list variable values are coefficients of that constraint.

### Sparse Input Format

The side constraints usually become sparse as the problem size increases. When the sparse data format of the CONDATA= data set is used, only nonzero constraint coefficients must be specified. Remember to specify the SPARSECONDATA option in the PROC INTPOINT statement. With the sparse method of specifying constraint information, the names of arc and nonarc variables or, if solving an LP, the names of LP variables do not have to be valid SAS variable names.

A sparse format CONDATA= data set for the oil industry example in the “Introductory NPSC Example” section on page 75 is displayed below.

```

title 'Setting Up Condata = Cond2 for PROC INTPOINT';
data cond2;
  input _column_ $ _row1 $ _coef1 _row2 $ _coef2 ;
  datalines;
m_e_ref1  con1  -2      .      .
m_e_ref2  con2  -2      .      .
thruput1  con1   1  con3  -3
r1_gas    .      .  con3   4
thruput2  con2   1  con4  -3
r2_gas    .      .  con4   4
_type_    con1   1  con2   1
_type_    con3   0  con4   0
_rhs_     con1 -15  con2 -15
;

```

Recall that the COLUMN list variable values `_type_` and `_rhs_` are the default values of the TYPEOBS= and RHSOBS= options. Also, the default rhs value of constraints (con3 and con4) is zero. The third to last observation has the value `_type_` for the COLUMN list variable. The `_ROW1` variable value is con1, and the `_COEF1_` variable has the value 1. This indicates that the constraint con1 is *greater than* or equal to type (because the value 1 is *greater than* zero). Similarly, the data in the second to last observation's `_ROW2` and `_COEF2` variables indicate that con2 is an *equality* constraint (0 *equals* zero).

An alternative, using a TYPE list variable, is

```

title 'Setting Up Condata = Cond3 for PROC INTPOINT';
data cond3;
    input _column_ $ _row1 $ _coef1 _row2 $ _coef2 _type_ $ ;
    datalines;
m_e_ref1  con1  -2      .      .  >=
m_e_ref2  con2  -2      .      .  .
thruput1  con1   1  con3  -3  .
r1_gas    .      .  con3   4  .
thruput2  con2   1  con4  -3  .
r2_gas    .      .  con4   4  .
.          con3   .  con4   .  eq
.          con1 -15  con2 -15  ge
;

```

If the COLUMN list variable is missing in a particular observation (the last 2 observations in the data set `cond3`, for instance), the constraints named in the ROW list variables all have the constraint type indicated by the value in the TYPE list variable. It is for this type of observation that you are allowed more ROW list variables than COEF list variables. If corresponding COEF list variables are not missing (for example, the last observation in the data set `cond3`), these values are the rhs values of those constraints. Therefore, you can specify both constraint type and rhs in the same observation.

As in the previous `CONDATA=` data set, if the COLUMN list variable is an arc or nonarc variable, the COEF list variable values are coefficient values for that arc or nonarc variable in the constraints indicated in the corresponding ROW list variables. If in this same observation the TYPE list variable contains a constraint type, all constraints named in the ROW list variables in that observation have this constraint type (for example, the first observation in the data set `cond3`). Therefore, you can specify both constraint type and coefficient information in the same observation.

Also note that `DEFCONTTYPE=EQ` could have been specified, saving you from having to include in the data that `con3` and `con4` are of this type.

In the oil industry example, arc costs, capacities, and lower flow bounds are presented in the `ARCDATA=` data set. Alternatively, you could have used the following input data sets. The `arcd2` data set has only two SAS variables. For each arc, there is an observation in which the arc's tail and head node is specified:

```

title3 'Setting Up Arcdata = Arcd2 for PROC INTPOINT';
data arcd2;
    input _from_&$11. _to_&$15. ;
    datalines;
middle east refinery 1
middle east refinery 2
u.s.a. refinery 1
u.s.a. refinery 2
refinery 1 r1
refinery 2 r2
r1 ref1 gas
r1 ref1 diesel
r2 ref2 gas
r2 ref2 diesel
ref1 gas servstn1 gas
ref1 gas servstn2 gas
ref1 diesel servstn1 diesel
ref1 diesel servstn2 diesel
ref2 gas servstn1 gas
ref2 gas servstn2 gas
ref2 diesel servstn1 diesel
ref2 diesel servstn2 diesel
;

title 'Setting Up Condata = Cond4 for PROC INTPOINT';
data cond4;
    input _column_&$27. _row1 $ _coef1 _row2 $ _coef2 _type_ $ ;
    datalines;
. con1 -15 con2 -15 ge
. costrow . . cost
. . caprow . capac
middle east_refinery 1 con1 -2 . .
middle east_refinery 2 con2 -2 . .
refinery 1_r1 con1 1 con3 -3
r1_ref1 gas . . con3 4 =
refinery 2_r2 con2 1 con4 -3
r2_ref2 gas . . con4 4 eq
middle east_refinery 1 costrow 63 caprow 95
middle east_refinery 2 costrow 81 caprow 80
u.s.a._refinery 1 costrow 55 . .
u.s.a._refinery 2 costrow 49 . .
refinery 1_r1 costrow 200 caprow 175
refinery 2_r2 costrow 220 caprow 100
r1_ref1 gas . . caprow 140
r1_ref1 diesel . . caprow 75
r2_ref2 gas . . caprow 100
r2_ref2 diesel . . caprow 75
ref1 gas_servstn1 gas costrow 15 caprow 70
ref1 gas_servstn2 gas costrow 22 caprow 60
ref1 diesel_servstn1 diesel costrow 18 . .
ref1 diesel_servstn2 diesel costrow 17 . .
ref2 gas_servstn1 gas costrow 17 caprow 35
ref2 gas_servstn2 gas costrow 31 . .
ref2 diesel_servstn1 diesel costrow 36 . .

```



```

ref2 diesel_servstn2 diesel costrow 23      .      .      .
middle east_refinery 1      . 20      .      .      lo
middle east_refinery 2      . 10      .      .      lo
refinery 1_r1                . 50      .      .      lo
refinery 2_r2                . 35      .      .      lo
ref2 gas_servstn1 gas       . 5       .      .      lo
;

```

The first observation in the cond4 data set defines con1 and con2 as *greater than or equal to* ( $\geq$ ) constraints that both (by coincidence) have rhs values of -15. The second observation defines the special row costrow as a cost row. When costrow is a ROW list variable value, the associated COEF list variable value is interpreted as a cost or objective function coefficient. PROC INTPOINT has to do less work if constraint names and special rows are defined in observations near the top of a data set, but this is not a strict requirement. The fourth to ninth observations contain constraint coefficient data. Observations seven and nine have TYPE list variable values that indicate that constraints con3 and con4 are equality constraints. The last five observations contain lower flow bound data. Observations that have an arc or nonarc variable name in the COLUMN list variable, a nonconstraint type TYPE list variable value, and a value in (one of) the COEF list variables are valid.

The following data set is equivalent to the cond4 data set:

```

title 'Setting Up Condata = Cond5 for PROC INTPOINT';
data cond5;
  input _column_&$27. _row1 $ _coef1 _row2 $ _coef2 _type_ $ ;
  datalines;
middle east_refinery 1      con1 -2 costrow 63      .
middle east_refinery 2      con2 -2 lorow 10      .
refinery 1_r1                .      .      con3 -3      =
r1_ref1 gas                  caprow 140      con3 4      .
refinery 2_r2                con2 1      con4 -3      .
r2_ref2 gas                  .      .      con4 4      eq
.                             CON1 -15      CON2 -15      GE
ref2 diesel_servstn1 diesel . 36 costrow . cost
.                             .      .      caprow . capac
.                             lorow .      .      .      lo
middle east_refinery 1      caprow 95      lorow 20      .
middle east_refinery 2      caprow 80 costrow 81      .
u.s.a._refinery 1          .      .      .      55      cost
u.s.a._refinery 2          costrow 49      .      .      .
refinery 1_r1                con1 1      caprow 175      .
refinery 1_r1                lorow 50 costrow 200      .
refinery 2_r2                costrow 220 caprow 100      .
refinery 2_r2                . 35      .      .      lo
r1_ref1 diesel              caprow2 75      .      .      capac
r2_ref2 gas                  .      .      caprow 100      .
r2_ref2 diesel              caprow2 75      .      .      .
ref1 gas_servstn1 gas       costrow 15      caprow 70      .
ref1 gas_servstn2 gas       caprow2 60 costrow 22      .
ref1 diesel_servstn1 diesel .      .      costrow 18      .
ref1 diesel_servstn2 diesel costrow 17      .      .      .

```

```

ref2 gas_servstn1 gas          costrow 17   lowrow 5   .
ref2 gas_servstn1 gas          .   . caprow2 35   .
ref2 gas_servstn2 gas          . 31   .   . cost
ref2 diesel_servstn2 diesel    .   . costrow 23   .
;

```

### Converting from an NPSC to an LP Problem

If you have data for a linear programming program that has an embedded network, the steps required to change that data into a form that is acceptable by PROC INTPOINT are

1. Identify the nodal flow conservation constraints. The coefficient matrix of these constraints (a submatrix of the LP's constraint coefficient matrix) has only two nonzero elements in each column, -1 and 1.
2. Assign a node to each nodal flow conservation constraint.
3. The rhs values of conservation constraints are the corresponding node's supplies and demands. Use this information to create the NODEDATA= data set.
4. Assign an arc to each column of the flow conservation constraint coefficient matrix. The arc is directed from the node associated with the row that has the 1 element in it and directed toward to the node associated with the row that has the -1 element in it. Set up the ARCDATA= data set that has two SAS variables. This data set could resemble ARCDATA=arcd2. These will eventually be the TAILNODE and HEADNODE list variables when PROC INTPOINT is used. Each observation consists of the tail and head node of each arc.
5. Remove from the data of the linear program all data concerning the nodal flow conservation constraints.
6. Put the remaining data into a CONDATA= data set. This data set will probably resemble CONDATA=cond4 or CONDATA=cond5.

### The Sparse Format Summary

The following list illustrates possible CONDATA= data set observation sparse formats. a1, b1, b2, b3 and c1 have as a `_COLUMN_` variable value either the name of an arc (possibly in the form *tail\_head*) or the name of a nonarc variable (if you are solving an NPSC), or the name of the LP variable (if you are solving an LP). These are collectively referred to as *variable* in the tables that follow.

- If there is no TYPE list variable in the CONDATA= data set, the problem must be constrained and there is no nonconstraint data in the CONDATA= data set:

|    | <code>_COLUMN_</code>                                       | <code>_ROWx_</code> | <code>_COEFx_</code> | <code>_ROWy_</code><br>(no <code>_COEFy_</code> )<br>(may not be<br>in CONDATA) |
|----|-------------------------------------------------------------|---------------------|----------------------|---------------------------------------------------------------------------------|
| a1 | variable                                                    | constraint          | lhs coef             | +-----+                                                                         |
| a2 | <code>_TYPE_</code> or<br><code>TYPEOBS=</code>             | constraint          | -1 0 1               |                                                                                 |
| a3 | <code>_RHS_</code> or<br><code>RHSOBS=</code> or<br>missing | constraint          | rhs value            | constraint<br>or<br>missing                                                     |
| a4 | <code>_TYPE_</code> or<br><code>TYPEOBS=</code>             | constraint          | missing              |                                                                                 |
| a5 | <code>_RHS_</code> or<br><code>RHSOBS=</code> or<br>missing | constraint          | missing              |                                                                                 |
|    |                                                             |                     |                      | +-----+                                                                         |

Observations of the form a4 and a5 serve no useful purpose but are still allowed to make problem generation easier.

- If there are no ROW list variables in the data set, the problem has no constraints and the information is nonconstraint data. There must be a TYPE list variable and only one COEF list variable in this case. The COLUMN list variable has as values the names of arcs or nonarc variables and must not have missing values or special row names as values:

|    | <code>_COLUMN_</code> | <code>_TYPE_</code> | <code>_COEFx_</code> |
|----|-----------------------|---------------------|----------------------|
| b1 | variable              | UPPERBD             | capacity             |
| b2 | variable              | LOWERBD             | lower flow           |
| b3 | variable              | COST                | cost                 |

- Using a TYPE list variable for constraint data implies the following:

|    | <b>_COLUMN_</b>                          | <b>_TYPE_</b> | <b>_ROWx_</b>       | <b>_COEFx_</b> | <b>_ROWy_</b><br>(no <b>_COEFy_</b> )<br>(may not be<br>in CONDATA) |
|----|------------------------------------------|---------------|---------------------|----------------|---------------------------------------------------------------------|
| c1 | variable                                 | missing       | +-----+             | lhs coef       | +-----+                                                             |
| c2 | <b>_TYPE_</b> or<br>TYPEOBS=             | missing       | c                   | -1 0 1         |                                                                     |
| c3 | <b>_RHS_</b> or<br>missing<br>or RHSOBS= | missing       | n  <br>  s  <br>  t | rhs value      | constraint<br>or<br>missing                                         |
| c4 | variable                                 | con type      | r                   | lhs coef       |                                                                     |
| c5 | <b>_RHS_</b> or<br>missing<br>or RHSOBS= | con type      | a  <br>  i  <br>  n | rhs value      |                                                                     |
| c6 | missing                                  | TYPE          | t                   | -1 0 1         |                                                                     |
| c7 | missing                                  | RHS           | +-----+             | rhs value      | +-----+                                                             |

If the observation is in form c4 or c5, and the **\_COEFx\_** values are missing, the constraint is assigned the type data specified in the **\_TYPE\_** variable.

- Using a TYPE list variable for arc and nonarc variable data implies the following:

|    | <b>_COLUMN_</b> | <b>_TYPE_</b> | <b>_ROWx_</b>                       | <b>_COEFx_</b>                                                | <b>_ROWy_</b><br>(no <b>_COEFy_</b> )<br>(may not be<br>in CONDATA) |
|----|-----------------|---------------|-------------------------------------|---------------------------------------------------------------|---------------------------------------------------------------------|
| d1 | variable        | UPPERBD       | missing                             | capacity                                                      | missing                                                             |
| d2 | variable        | LOWERBD       | or                                  | lowerflow                                                     | or                                                                  |
| d3 | variable        | COST          | special<br>row<br>name              | cost                                                          | special<br>row<br>name                                              |
| d4 | missing         |               | +-----+<br>  special<br>row<br>name |                                                               | +-----+                                                             |
| d5 | variable        | missing       | +-----+                             | value that<br>is interpreted<br>according to<br><b>_ROWx_</b> | +-----+<br>  missing                                                |

The observation with form d1 to d5 can have ROW list variable values. Observation d4 must have ROW list variable values. The ROW value is put into the ROW name tree so that when dealing with observation d4 or d5, the COEF list variable value is interpreted according to the type of ROW list variable value. For example, the following three observations define the **\_ROWx\_** variable values up\_row, lo\_row and co\_row as being an upper value bound row, lower value bound row, and cost row, respectively:

|                 |               |               |                |
|-----------------|---------------|---------------|----------------|
| <b>_COLUMN_</b> | <b>_TYPE_</b> | <b>_ROWx_</b> | <b>_COEFx_</b> |
| .               | UPPERBD       | up_row        | .              |
| variable_a      | LOWERBD       | lo_row        | lower flow     |
| variable_b      | COST          | co_row        | cost           |

PROC INTPOINT is now able to correctly interpret the following observation:

|                 |               |               |                |               |                |               |                |
|-----------------|---------------|---------------|----------------|---------------|----------------|---------------|----------------|
| <b>_COLUMN_</b> | <b>_TYPE_</b> | <b>_ROW1_</b> | <b>_COEF1_</b> | <b>_ROW2_</b> | <b>_COEF2_</b> | <b>_ROW3_</b> | <b>_COEF3_</b> |
| var_c           | .             | up_row        | upval          | lo_row        | loval          | co_row        | cost           |

If the TYPE list variable value is a constraint type and the value of the COLUMN list variable equals the value of the TYPEOBS= option or the default value \_TYPE\_, the TYPE list variable value is ignored.

### **NODEDATA= Data Set**

See the “Getting Started: NPSC Problems” section on page 74 and the “Introductory NPSC Example” section on page 75 for a description of this input data set.

---

## **Output Data Set**

For NPSC problems, the procedure determines the flow that should pass through each arc as well as the value that should be assigned to each nonarc variable. The goal is that the minimum flow bounds, capacities, lower and upper value bounds, and side constraints are not violated. This goal is reached when total cost incurred by such a flow pattern and value assignment is feasible and optimal. The solution found must also conserve flow at each node.

For LP problems, the procedure determines the value that should be assigned to each variable. The goal is that the lower and upper value bounds and the constraints are not violated. This goal is reached when the total cost incurred by such a value assignment is feasible and optimal.

The CONOUT= data set can be produced and contains a solution obtained after performing optimization.

### **CONOUT= Data Set**

The CONOUT= data set contains the following variables, and their possible values in an observation are

|                |                                                                                                           |
|----------------|-----------------------------------------------------------------------------------------------------------|
| <b>_FROM_</b>  | a tail node of an arc. This is a missing value if an observation has information about a nonarc variable. |
| <b>_TO_</b>    | a head node of an arc. This is a missing value if an observation has information about a nonarc variable. |
| <b>_COST_</b>  | the cost of an arc or the objective function coefficient of a nonarc variable.                            |
| <b>_CAPAC_</b> | the capacity of an arc or upper value bound of a nonarc variable                                          |
| <b>_LO_</b>    | the lower flow bound of an arc or lower value bound of a nonarc variable                                  |
| <b>_NAME_</b>  | a name of an arc or nonarc variable                                                                       |

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_SUPPLY_</code> | the supply of the tail node of the arc in the observation. This is a missing value if an observation has information about a nonarc variable. |
| <code>_DEMAND_</code> | the demand of the head node of the arc in the observation. This is a missing value if an observation has information about a nonarc variable. |
| <code>_FLOW_</code>   | the flow through the arc or value of the nonarc variable                                                                                      |
| <code>_FCOST_</code>  | flow cost, the product of <code>_COST_</code> and <code>_FLOW_</code>                                                                         |
| <code>_RCOST_</code>  | the reduced cost of the arc or nonarc variable                                                                                                |
| <code>_STATUS_</code> | the status of the arc or nonarc variable                                                                                                      |

The variables present in the `ARCDATA=` data set are present in a `CONOUT=` data set. For example, if there is a variable called `tail` in the `ARCDATA=` data set and you specified the SAS variable list:

```
from tail;
```

then `tail` is a variable in the `CONOUT=` data sets instead of `_FROM_`. Any ID list variables also appear in the `CONOUT=` data sets.

---

## Case Sensitivity

Whenever the INTPOINT procedure has to compare character strings, whether they are node names, arc names, nonarc names, LP variable names, or constraint names, if the two strings have different lengths, or on a character by character basis the character is different *or has different cases*, PROC INTPOINT judges the character strings to be different.

Not only is this rule enforced when one or both character strings are obtained as values of SAS variables in PROC INTPOINT's input data sets, it also should be obeyed if one or both character strings were originally SAS variable names, or were obtained as the values of options or statements parsed to PROC INTPOINT. For example, if the network has only one node that has supply capability, or if you are solving a MAXFLOW or SHORTPATH problem, you can indicate that node using the SOURCE option. If you specify:

```
proc intpoint source=NotableNode
```

then PROC INTPOINT looks for a value of the TAILNODE list variable that is `NotableNode`.

Version 6 of the SAS System converts text that makes up statements into uppercase. The name of the node searched for would be `NOTABLENODE`, even if this was your SAS code:

```
proc intpoint source=NotableNode
```

If you want PROC INTPOINT to behave as it did in Version 6, specify:

```
options validvarname=v6;
```

If the SPARSECONDATA option is not specified, and you are running SAS software Version 6, or you are running SAS software Version 7 onward and have specified:

```
options validvarname=v6;
```

all values of the SAS variables that belong to the NAME list are uppercased. This is because the SAS System has uppercased all SAS variable names, particularly those in the VAR list of the CONDATA= data set.

Entities that contain blanks must be enclosed in quotes.

---

## Loop Arcs

Loop arcs (which are arcs directed toward nodes from which they originate) are prohibited. Rather, introduce a dummy intermediate node in loop arcs. For example, replace arc (A,A) with (A,B) and (B,A). B is the name of a new node, and it must be distinct for each loop arc.

---

## Multiple Arcs

Multiple arcs with the same tail and head nodes are prohibited. PROC INTPOINT checks to ensure there are no such arcs before proceeding with the optimization. Introduce a new dummy intermediate node in multiple arcs. This node must be distinct for each multiple arc. For example, if some network has three arcs directed from node A toward node B, then replace one of these three with arcs (A,C) and (C,B) and replace another one with (A,D) and (D,B). C and D are new nodes added to the network.

---

## Flow and Value Bounds

The capacity and lower flow bound of an arc can be equal. Negative arc capacities and lower flow bounds are permitted. If both arc capacities and lower flow bounds are negative, the lower flow bound must be at least as negative as the capacity. An arc  $(A, B)$  that has a negative flow of  $-f$  units can be interpreted as an arc that conveys  $f$  units of flow from node  $B$  to node  $A$ .

The upper and lower value bound of a nonarc variable can be equal. Negative upper and lower bounds are permitted. If both are negative, the lower bound must be at least as negative as the upper bound.

When solving an LP, the upper and lower value bound of an LP variable can be equal. Negative upper and lower bounds are permitted. If both are negative, the lower bound must be at least as negative as the upper bound.

In short, for any problem to be feasible, a lower bound must be  $\leq$  the associated upper bound.

---

## Tightening Bounds and Side Constraints

If any piece of data is furnished to PROC INTPOINT more than once, PROC INTPOINT checks for consistency so that no conflict exists concerning the data values. For example, if the cost of some arc is seen to be one value and as more data are read, the cost of the same arc is seen to be another value, PROC INTPOINT issues an error message on the SAS log and stops. There are two exceptions to this:

- The bounds of arcs and nonarc variables, or the bounds of LP variables, are made as tight as possible. If several different values are given for the lower flow bound of an arc, the greatest value is used. If several different values are given for the lower bound of a nonarc or LP variable, the greatest value is used. If several different values are given for the capacity of an arc, the smallest value is used. If several different values are given for the upper bound of a nonarc or LP variable, the smallest value is used.
- Several values can be given for inequality constraint right-hand-sides. For a particular constraint, the lowest rhs value is used for the rhs if the constraint is of *less than or equal to* type. For a particular constraint, the greatest rhs value is used for the rhs if the constraint is of *greater than or equal to* type.

---

## Reasons for Infeasibility

Before optimization commences, PROC INTPOINT tests to ensure that the problem is not infeasible by ensuring that, with respect to supplies, demands, and arc flow bounds, flow conservation can be obeyed at each node:

- Let  $IN$  be the sum of lower flow bounds of arcs directed toward a node plus the node's supply. Let  $OUT$  be the sum of capacities of arcs directed from that node plus the node's demand. If  $IN$  exceeds  $OUT$ , not enough flow can leave the node.
- Let  $OUT$  be the sum of lower flow bounds of arcs directed from a node plus the node's demand. Let  $IN$  be the total capacity of arcs directed toward the node plus the node's supply. If  $OUT$  exceeds  $IN$ , not enough flow can arrive at the node.

Reasons why a network problem can be infeasible are similar to those previously mentioned but apply to a set of nodes rather than for an individual node.



Consider the network illustrated in Figure 4.10.

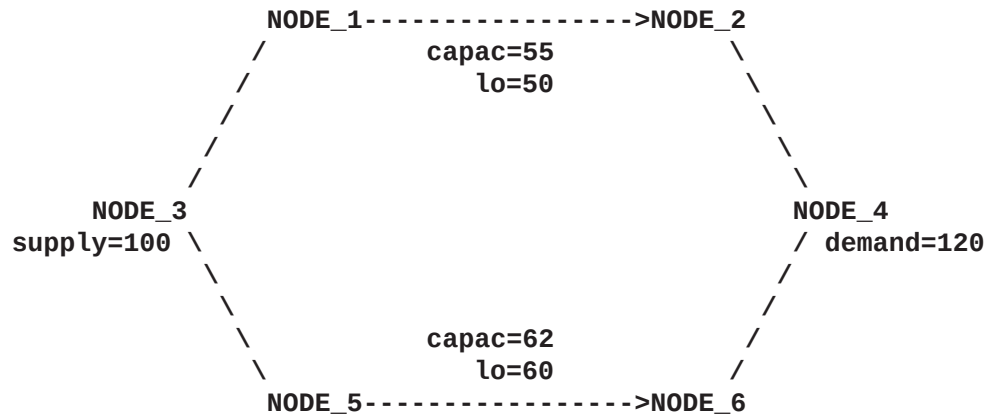


**Figure 4.10.** An Infeasible Network

The demand of NODE\_4 is 120. That can never be satisfied because the maximal flow through arcs (NODE\_1, NODE\_2) and (NODE\_5, NODE\_6) is 117. More specifically, the implicit supply of NODE\_2 and NODE\_6 is only 117, which is insufficient to satisfy the demand of other nodes (real or implicit) in the network.

Furthermore, the lower flow bounds of arcs (NODE\_1, NODE\_2) and (NODE\_5, NODE\_6) are greater than the flow that can reach the tail nodes of these arcs, that, by coincidence, is the total supply of the network. The implicit demand of nodes NODE\_1 and NODE\_5 is 110, which is greater than the amount of flow that can reach these nodes.

---

## Missing S Supply and Missing D Demand Values

In some models, you may want a node to be either a supply or demand node but you want the node to supply or demand the optimal number of flow units. To indicate that a node is such a supply node, use a missing S value in the SUPPLY list variable in the ARCDATA= data set or the SUPDEM list variable in the NODEDATA= data set. To indicate that a node is such a demand node, use a missing D value in the DEMAND list variable in the ARCDATA= data set or the SUPDEM list variable in the NODEDATA= data set.

Suppose the oil example in the “Introductory NPSC Example” section on page 75 is changed so that crude oil can be obtained from either the Middle East or U.S.A. in any amounts. You should specify that the node `middle east` is a supply node, but you do not want to stipulate that it supplies 100 units, as before. The node `u.s.a.` should also remain a supply node, but you do not want to stipulate that it supplies 80 units. You must specify that these nodes have missing S supply capabilities:

```

title 'Oil Industry Example';
title3 'Crude Oil can come from anywhere';
data miss_s;
  missing S;
  input _node_&$15. _sd_;
  datalines;
middle east          S
u.s.a.              S
servstn1 gas        -95
servstn1 diesel     -30
servstn2 gas        -40
servstn2 diesel     -15
;

```

The following PROC INTPOINT run uses the same ARCDATA= and CONDATA= data sets used in the “Introductory NPSC Example” section on page 75:

```

proc intpoint
  bytes=100000
  nodedata=miss_s      /* the supply (missing S) and */
                      /* demand data */
  arcdata=arcd1        /* the arc descriptions */
  condata=cond1        /* the side constraints */
  conout=solution;     /* the solution data set */
run;
proc print;
  var _from_ _to_ _cost_ _capac_ _lo_ _flow_ _fcost_;
  sum _fcost_;
run;

```

The following messages appear on the SAS log:

```

NOTE: Number of nodes= 14 .
NOTE: Number of supply nodes= 2 .
NOTE: Of these, 2 have unspecified (.S) supply capability.
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 0 , total demand= 180 .
NOTE: Number of arcs= 18 .
NOTE: Number of <= side constraints= 0 .
NOTE: Number of == side constraints= 2 .
NOTE: Number of >= side constraints= 2 .
NOTE: Number of side constraint coefficients= 8 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 17 .
NOTE: Number of >= constraints= 2 .
NOTE: Number of constraint coefficients= 48 .
NOTE: Number of variables= 20 .
NOTE: After preprocessing, number of <= constraints= 0.
NOTE: After preprocessing, number of == constraints= 8.

```

NOTE: After preprocessing, number of  $\geq$  constraints= 2.  
 NOTE: The preprocessor eliminated 9 constraints from the problem.  
 NOTE: The preprocessor eliminated 20 constraint coefficients from the problem.  
 NOTE: After preprocessing, number of variables= 11.  
 NOTE: The preprocessor eliminated 9 variables from the problem.  
 NOTE: 2 columns, 0 rows and 2 coefficients were added to the problem to handle unrestricted variables, variables that are split, and constraint slack or surplus variables.  
 NOTE: There are 21 nonzero elements in  $A * A$  transpose.  
 NOTE: Of the 10 rows and columns, 4 are sparse.  
 NOTE: There are 15 nonzero superdiagonal elements in the sparse rows of the factored  $A * A$  transpose. This includes fill-in.  
 NOTE: There are 5 operations of the form  

$$u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$$
 to factorize the sparse rows of  $A * A$  transpose.  
 NOTE: Bound feasibility attained by iteration 1.  
 NOTE: Dual feasibility attained by iteration 1.  
 NOTE: Constraint feasibility attained by iteration 2.  
 NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 7 iterations.  
 NOTE: Objective = 50075.  
 NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.  
 NOTE: There were 18 observations read from the data set WORK.ARC1.  
 NOTE: There were 6 observations read from the data set WORK.MISS\_S.  
 NOTE: There were 4 observations read from the data set WORK.COND1.  
 NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.

The CONOUT= data set is shown in Figure 4.11.

| Oil Industry Example             |             |                 |        |          |      |          |
|----------------------------------|-------------|-----------------|--------|----------|------|----------|
| Crude Oil can come from anywhere |             |                 |        |          |      |          |
| Obs                              | _from_      | _to_            | _cost_ | _capac_  | _lo_ | _FLOW_   |
| 1                                | refinery 1  | r1              | 200    | 175      | 50   | 145.00   |
| 2                                | refinery 2  | r2              | 220    | 100      | 35   | 35.00    |
| 3                                | r1          | ref1 diesel     | 0      | 75       | 0    | 36.25    |
| 4                                | r1          | ref1 gas        | 0      | 140      | 0    | 108.75   |
| 5                                | r2          | ref2 diesel     | 0      | 75       | 0    | 8.75     |
| 6                                | r2          | ref2 gas        | 0      | 100      | 0    | 26.25    |
| 7                                | middle east | refinery 1      | 63     | 95       | 20   | 20.00    |
| 8                                | u.s.a.      | refinery 1      | 55     | 99999999 | 0    | 125.00   |
| 9                                | middle east | refinery 2      | 81     | 80       | 10   | 10.00    |
| 10                               | u.s.a.      | refinery 2      | 49     | 99999999 | 0    | 25.00    |
| 11                               | ref1 diesel | servstn1 diesel | 18     | 99999999 | 0    | 30.00    |
| 12                               | ref2 diesel | servstn1 diesel | 36     | 99999999 | 0    | 0.00     |
| 13                               | ref1 gas    | servstn1 gas    | 15     | 70       | 0    | 68.75    |
| 14                               | ref2 gas    | servstn1 gas    | 17     | 35       | 5    | 26.25    |
| 15                               | ref1 diesel | servstn2 diesel | 17     | 99999999 | 0    | 6.25     |
| 16                               | ref2 diesel | servstn2 diesel | 23     | 99999999 | 0    | 8.75     |
| 17                               | ref1 gas    | servstn2 gas    | 22     | 60       | 0    | 40.00    |
| 18                               | ref2 gas    | servstn2 gas    | 31     | 99999999 | 0    | 0.00     |
|                                  |             |                 |        |          |      | =====    |
|                                  |             |                 |        |          |      | 50075.00 |

**Figure 4.11.** Missing S SUPDEM values in NODEDATA

The optimal supplies of nodes middle east and u.s.a. are 145 and 35 units, respectively. For this example, the same optimal solution is obtained if these nodes had supplies less than these values (each supplies 1 unit, for example) and the THRUNET option was specified in the PROC INTPOINT statement. With the THRUNET option active, when total supply exceeds total demand, the specified nonmissing demand values are the lowest number of flow units that must be absorbed by the corresponding node. This is demonstrated in the following PROC INTPOINT run. The missing S is most useful when nodes are to supply optimal numbers of flow units and it turns out that for some nodes, the optimal supply is zero:

```

data miss_s_x;
  missing S;
  input  _node_&$15. _sd_;
  datalines;
middle east          1
u.s.a.               1
servstn1 gas         -95
servstn1 diesel      -30
servstn2 gas         -40
servstn2 diesel      -15
;
proc intpoint
  bytes=100000
  thrunet
  nodedata=miss_s_x      /* No supply (missing S)      */
  arcdata=arcd1          /* the arc descriptions    */
  condata=cond1          /* the side constraints    */
  conout=solution;       /* the solution data set  */
run;
proc print;
  var _from_ _to_ _cost_ _capac_ _lo_ _flow_ _fcost_;
  sum _fcost_;
run;

```

The following messages appear on the SAS log. Note that the Total supply= 2, not zero as in the last run:

```

NOTE: Number of nodes= 14 .
NOTE: Number of supply nodes= 2 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 2 , total demand= 180 .
NOTE: Number of arcs= 18 .
NOTE: Number of <= side constraints= 0 .
NOTE: Number of == side constraints= 2 .
NOTE: Number of >= side constraints= 2 .
NOTE: Number of side constraint coefficients= 8 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 17 .
NOTE: Number of >= constraints= 2 .
NOTE: Number of constraint coefficients= 48 .
NOTE: Number of variables= 20 .
NOTE: After preprocessing, number of <= constraints= 0.
NOTE: After preprocessing, number of == constraints= 8.
NOTE: After preprocessing, number of >= constraints= 2.
NOTE: The preprocessor eliminated 9 constraints from the
      problem.
NOTE: The preprocessor eliminated 20 constraint
      coefficients from the problem.
NOTE: After preprocessing, number of variables= 11.
NOTE: The preprocessor eliminated 9 variables from the

```

problem.

NOTE: 2 columns, 0 rows and 2 coefficients were added to the problem to handle unrestricted variables, variables that are split, and constraint slack or surplus variables.

NOTE: There are 21 nonzero elements in  $A * A$  transpose.

NOTE: Of the 10 rows and columns, 4 are sparse.

NOTE: There are 15 nonzero superdiagonal elements in the sparse rows of the factored  $A * A$  transpose. This includes fill-in.

NOTE: There are 5 operations of the form  $u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$  to factorize the sparse rows of  $A * A$  transpose.

NOTE: Bound feasibility attained by iteration 1.

NOTE: Dual feasibility attained by iteration 1.

NOTE: Constraint feasibility attained by iteration 2.

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 7 iterations.

NOTE: Objective = 50075.

NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.

NOTE: There were 18 observations read from the data set WORK.ARC1.

NOTE: There were 6 observations read from the data set WORK.MISS\_S\_X.

NOTE: There were 4 observations read from the data set WORK.COND1.

NOTE: The data set WORK.SOLUTION has 18 observations and 14 variables.

If total supply exceeds total demand, any missing S values are ignored. If total demand exceeds total supply, any missing D values are ignored.

---

## Balancing Total Supply and Total Demand

### *When Total Supply Exceeds Total Demand*

When total supply of a network problem exceeds total demand, PROC INTPOINT adds an extra node (called the *excess node*) to the problem and sets the demand at that node equal to the difference between total supply and total demand. There are three ways that this excess node can be joined to the network. All three ways entail PROC INTPOINT generating a set of arcs (henceforth referred to as the *generated arcs*) that are directed toward the excess node. The total amount of flow in generated arcs equals the demand of the excess node. The generated arcs originate from one of three sets of nodes.

When you specify the THRUNET option, the set of nodes that generated arcs originate from are all demand nodes, even those demand nodes with unspecified demand capability. You indicate that a node has unspecified demand capability by using a missing D value instead of an actual value for demand data (discussed in the “Missing S Supply and Missing D Demand Values” section on page 139). The value specified as the demand of a demand node is in effect a lower bound of the number of flow units that node can actually demand. For missing D demand nodes, this lower bound is zero.

If you do not specify the THRUNET option, the way in which the excess node is joined to the network depends on whether there are demand nodes with unspecified demand capability (nodes with missing D demand) or not.

If there are missing D demand nodes, these nodes are the set of nodes that generated arcs originate from. The value specified as the demand of a demand node, if not missing D, is the number of flow units that node can actually demand. For a missing D demand node, the actual demand of that node may be zero or greater.

If there are no missing D demand nodes, the set of nodes that generated arcs originate from are the set of supply nodes. The value specified as the supply of a supply node is in effect an upper bound of the number of flow units that node can actually supply. For missing S supply nodes (discussed in the “Missing S Supply and Missing D Demand Values” section on page 139), this upper bound is zero, so missing S nodes when total supply exceeds total demand are transshipment nodes, that is, nodes that neither supply nor demand flow.

### **When Total Supply Is Less Than Total Demand**

When total supply of a network problem is less than total demand, PROC INTPOINT adds an extra node (called the *excess node*) to the problem and sets the supply at that node equal to the difference between total demand and total supply. There are three ways that this excess node can be joined to the network. All three ways entail PROC INTPOINT generating a set of arcs (henceforth referred to as the *generated arcs*) that originate from the excess node. The total amount of flow in generated arcs equals the supply of the excess node. The generated arcs are directed toward one of three sets of nodes.

When you specify the THRUNET option, the set of nodes that generated arcs are directed toward are all supply nodes, even those supply nodes with unspecified supply capability. You indicate that a node has unspecified supply capability by using a missing S value instead of an actual value for supply data (discussed in the “Missing S Supply and Missing D Demand Values” section on page 139). The value specified as the supply of a supply node is in effect a lower bound of the number of flow units that the node can actually supply. For missing S supply nodes, this lower bound is zero.

If you do not specify the THRUNET option, the way in which the excess node is joined to the network depends on whether there are supply nodes with unspecified supply capability (nodes with missing S supply) or not.

If there are missing S supply nodes, these nodes are the set of nodes that generated arcs are directed toward. The value specified as the supply of a supply node, if not missing S, is the number of flow units that the node can actually supply. For a missing S supply node, the actual supply of that node may be zero or greater.

If there are no missing S supply nodes, the set of nodes that generated arcs are directed toward are the set of demand nodes. The value specified as the demand of a demand node is in effect an upper bound of the number of flow units that node can actually demand. For missing D demand nodes (discussed in the “Missing S Supply and Missing D Demand Values” section on page 139), this upper bound is zero, so missing D nodes when total supply is less than total demand are transshipment nodes, that is, nodes that neither supply nor demand flow.

---

## How to Make the Data Read of PROC INTPOINT More Efficient

This section contains information that is useful when you want to solve large constrained network problems. However, much of this information is also useful if you have a large linear programming problem. All of the options described in this section that are not directly applicable to networks (options such as ARCS\_ONLY\_ARCDATA ARC\_SINGLE\_OBS, NNODES, and NARCS) can be specified to improve the speed at which LP data is read.

### **Large Constrained Network Problems**

Many of the models presented to PROC INTPOINT are enormous. They can be considered large by linear programming standards; problems with thousands, even millions, of variables and constraints. When dealing with side constrained network programming problems, models can have not only a linear programming component of that magnitude, but also a larger, possibly *much* larger, network component.

The majority of network problem's decision variables are arcs. Like an LP decision variable, an arc has an objective function coefficient, upper and lower value bounds, and a name. Arcs can have coefficients in constraints. Therefore, an arc is quite similar to an LP variable and places the same memory demands on optimization software as an LP variable. But a typical network model has many more arcs and nonarc variables than the typical LP model has variables. And arcs have tail and head nodes. Storing and processing node names require huge amounts of memory. To make matters worse, node names occupy memory at times when a lot of other data should reside in memory as well.

While memory requirements are lower for a model with embedded network component compared with the equivalent LP *once optimization starts*, the same is usually not true *during the data read*. Even though nodal flow conservation constraints in the LP should not be specified in the constrained network formulation, the memory requirements to read the latter are greater because each arc (unlike an LP variable) originates at one node and is directed toward another.

### **Paging**

PROC INTPOINT has facilities to read data when the available memory is insufficient to store all the data at once. PROC INTPOINT does this by allocating memory for different purposes; for example, to store an array or receive data read from an input SAS data set. After that memory has filled, the information is written to disk and PROC INTPOINT can resume filling that memory with new information. Often, information must be retrieved from disk so that data previously read can be examined or checked for consistency. Sometimes, to prevent any data from being lost, or to retain any changes made to the information in memory, the contents of the memory must be sent to disk before other information can take its place. This process of swapping information to and from disk is called paging. Paging can be very time consuming, so it is crucial to minimize the amount of paging performed.

There are several steps you can take to make PROC INTPOINT read the data of network and linear programming models more efficiently, particularly when memory is scarce and the amount of paging must be reduced. PROC INTPOINT will then be able to tackle large problems in what can be considered reasonable amounts of time.



### **The Order of Observations**

PROC INTPOINT is quite flexible in the ways data can be supplied to it. Data can be given by any reasonable means. PROC INTPOINT has convenient defaults that can save you work when generating the data. There can be several ways to supply the same piece of data, and some pieces of data can be given more than once. PROC INTPOINT reads everything, then merges it all together. However, this flexibility and convenience come at a price; PROC INTPOINT may not assume the data has a characteristic that, if possessed by the data, could save time and memory during the data read. Several options can indicate that the data has some exploitable characteristic.

For example, an arc cost can be specified once or several times in the ARCDATA= data set or the CONDATA= data set, or both. Every time it is given in the ARCDATA= data set, a check is made to ensure that the new value is the same as any corresponding value read in a previous observation of the ARCDATA= data set. Every time it is given in the CONDATA= data set, a check is made to ensure that the new value is the same as the value read in a previous observation of the CONDATA= data set, or previously in the ARCDATA= data set. PROC INTPOINT would save time if it knew that arc cost data would be encountered only once while reading the ARCDATA= data set, so performing the time-consuming check for consistency would not be necessary. Also, if you indicate that the CONDATA= data set contains data for constraints only, PROC INTPOINT will not expect any arc information, so memory will not be allocated to receive such data while reading the CONDATA= data set. This memory is used for other purposes and this might lead to a reduction in paging. If applicable, use the ARC\_SINGLE\_OBS or the CON\_SINGLE\_OBS option, or both, and the NON\_REPLIC=COEFS specification to improve how the ARCDATA= data set and the CONDATA= data set are read.

PROC INTPOINT allows the observations in input data sets to be in any order. However, major time savings can result if you are prepared to order observations in particular ways. Time spent by the SORT procedure to sort the input data sets, particularly the CONDATA= data set, may be more than made up for when PROC INTPOINT reads them, because PROC INTPOINT has in memory information possibly used when the previous observation was read. PROC INTPOINT can assume a piece of data is either similar to that of the last observation read or is new. In the first case, valuable information such as an arc or a nonarc variable number or a constraint number is retained from the previous observation. In the last case, checking the data with what has been read previously is not necessary.

Even if you do not sort the CONDATA= data set, grouping observations that contain data for the same arc or nonarc variable or the same row pays off. PROC INTPOINT establishes whether an observation being read is similar to the observation just read.

In practice, many input data sets for PROC INTPOINT have this characteristic, because it is natural for data for each constraint to be grouped together (when using the dense format of the CONDATA= data set) or data for each column to be grouped together (when using the sparse format of the CONDATA= data set). If data for each arc or nonarc is spread over more than one observation of the ARCDATA= data set, it is natural to group these observation together.

Use the GROUPED= parameter to indicate whether observations of the ARCDATA= data set, the CONDATA= data set, or both, are grouped in a way that can be exploited during data read.

You can save time if the type data for each row appears near the top of the CONDATA= data set, especially if it has the sparse format. Otherwise, when reading an observation, if PROC INTPOINT does not know if a row is a constraint or special row, the data is set aside. Once the data set has been completely read, PROC INTPOINT must reprocess the data it set aside. By then, it knows the type of each constraint or row or, if its type was not provided, it is assumed to have a default type.

### **Better Memory Utilization**

In order for PROC INTPOINT to make better utilization of available memory, you can now specify options that indicate the approximate size of the model. PROC INTPOINT then knows what to expect. For example, if you indicate that the problem has no nonarc variables, PROC INTPOINT will not allocate memory to store nonarc data. That memory is better utilized for other purposes. Memory is often allocated to receive or store data of some type. If you indicate that the model does not have much data of a particular type, the memory that would otherwise have been allocated to receive or store that data can be used to receive or store data of another type.

These are the problem size options:

- NNODES= approximate number of nodes
- NARCS= approximate number of arcs
- NNAS= approximate number of nonarc variables or LP variables
- NCONS= approximate number of NPSC side constraints or LP constraints
- NCOEFS= approximate number of NPSC side constraint coefficients or LP constraint coefficients

These options will sometimes be referred to as Nxxxx= options.

You do not need to specify all these options for the model, but the more you do, the better. If you do not specify some or all of these options, PROC INTPOINT guesses the size of the problem by using what it already knows about the model. Sometimes PROC INTPOINT guesses the size of the model by looking at the number of observations in the ARCDATA= and the CONDATA= data sets. However, PROC INTPOINT uses rough rules of thumb, that typical models are proportioned in certain ways (for example, if there are constraints, arcs, nonarc variables, or LP variables usually have about five constraint coefficients). If your model has an unusual shape or a disproportionate number of something, you are encouraged to use these options.

If you do use the options, if you do not know the exact values to specify, *overestimate* the values. For example, if you specify NARCS=10000 but the model has 10100 arcs, when dealing with the last 100 arcs, PROC INTPOINT might have to page out data for 10000 arcs each time one of the last arcs must be dealt with. Memory could have been allocated for all 10100 arcs without affecting (much) the rest of the data read, so NARCS=10000 could be more of a hindrance than a help.

The point of these Nxxxx= options is to indicate the model size when PROC INTPOINT does not know it. When PROC INTPOINT knows the “real” value, that value is used instead of Nxxxx=.

ARCS\_ONLY\_ARCDATA indicates that data for only arcs are in the ARCDATA= data set. Memory would not be wasted to receive data for nonarc variables.

Use the memory usage parameters:

- BYTES= size of PROC INTPOINT main working memory in number of bytes
- MEMREP indicates that memory usage report is to be displayed on the SAS log

Specifying an appropriate value for the BYTES= parameter is particularly important. Specify as large a number as possible, but not so large a number that will cause PROC INTPOINT (that is, the SAS System running underneath PROC INTPOINT) to run out of memory.

PROC INTPOINT reports its memory requirements on the SAS log if you specify the MEMREP option.

### ***Use Defaults to Reduce the Amount of Data***

Use the parameters that specify default values as much as possible. For example, if there are lots of arcs with the same cost value *c*, use DEFCOST=*c* for arcs that have that cost. Use missing values in the COST variable in the ARCDATA= data set instead of *c*. PROC INTPOINT ignores missing values, but must read, store, and process nonmissing values, even if they are equal to a default option or could have been equal to a default parameter had it been specified. Sometimes, using default parameters makes the need for some SAS variables in the ARCDATA= and the CONDATA= data sets no longer necessary, or reduces the quantity of data that must be read. The default options are:

- DEFCOST= default cost of arcs, objective function of nonarc variables or LP variables
- DEFMINFLOW= default lower flow bound of arcs, lower bound of nonarc variables or LP variables
- DEFCAPACITY= default capacity of arcs, upper bound of nonarc variables or LP variables
- DEFCTYPE= LE or DEFCTYPE= <=  
DEFCTYPE= EQ or DEFCTYPE= =  
DEFCTYPE= GE or DEFCTYPE= >=

DEFCTYPE=LE is the default.

The default options themselves have defaults. For example, you do not need to specify DEFCOST=0 in the PROC INTPOINT statement. You should still have missing values in the COST variable in the ARCDATA= data set for arcs that have zero costs.

If the network has only one supply node, one demand node, or both, use

- SOURCE= name of single node that has supply capability
- SUPPLY= the amount of supply of SOURCE

- SINK= name of single node that demands flow
- DEMAND= the amount of flow SINK demands

Do not specify that a constraint has zero right-hand-side values. That is the default. The only time it might be practical to specify a zero rhs is in observations of the CONDATA read early so that PROC INTPOINT can infer that a row is a constraint. This could prevent coefficient data from being put aside because PROC INTPOINT did not know the row was a constraint.

### **Names of Things**

To cut data read time and memory requirements, reduce the number of bytes in the longest node name, the longest arc name, the longest nonarc variable name, the longest LP variable name, and the longest constraint name to 8 bytes or less. The longer a name, the more bytes must be stored and compared with other names.

If an arc has no constraint coefficients, do not give it a name in the NAME list variable in the ARCDATA= data set. Names for such arcs serve no purpose.

PROC INTPOINT can have a default name for each arc. If an arc is directed from node *tailname* toward node *headname*, the default name for that arc is *tailname\_headname*. If you do not want PROC INTPOINT to use these default arc names, specify NAMECTRL=1. Otherwise, PROC INTPOINT must use memory for storing node names and these node names must be searched often.

If you want to use the default *tailname\_headname* name, that is, NAMECTRL=2 or NAMECTRL=3, do not use underscores in node names. If the CONDATA has a dense format and has a variable in the VAR list A\_B\_C\_D, or if the value A\_B\_C\_D is encountered as a value of the COLUMN list variable when reading the CONDATA= data set that has the sparse format, PROC INTPOINT first looks for a node named A. If it finds it, it looks for a node called B\_C\_D. It then looks for a node with the name A\_B and possibly a node with name C\_D. A search is then conducted for a node named A\_B\_C and possibly a node named D is done. Underscores could have caused PROC INTPOINT to look unnecessarily for nonexistent nodes. Searching for node names can be expensive, and the amount of memory to store node names is often large. It might be better to assign the arc name A\_B\_C\_D directly to an arc by having that value as a NAME list variable value for that arc in the ARCDATA= data set and specify NAMECTRL=1.

### **Other Ways to Speed-up Data Reads**

Arcs and nonarc variables, or LP variables, can have associated with them values or quantities that have no bearing with the optimization. This information is given in the ARCDATA= data set in the ID list variables. For example, in a distribution problem, information such as truck number and driver's name can be associated with each arc. This is useful when the optimal solution saved in the CONOUT= data set is analyzed. However, PROC INTPOINT needs to reserve memory to process this information when data is being read. For large problems when memory is scarce, it might be better to remove ancillary data from the ARCDATA. After PROC INTPOINT runs, use SAS software to merge this information into the CONOUT= data set that contains the optimal solution.

## Stopping Criteria

There are several reasons why PROC INTPOINT stops Interior Point optimization. Optimization stops when:

- the number of iteration equals MAXITERB= $m$
- the relative gap ( $dualitygap/(c^T x)$ ) between the primal and dual objectives is smaller than the value of the PDGAPTOL= option, and both the primal and dual problems are feasible. Duality gap is defined in the “Interior Point Algorithmic Details” section on page 58.

PROC INTPOINT may stop optimization when it detects that the rate at which the complementarity or duality gap is being reduced is too slow; that is, that there are consecutive iterations when the complementarity or duality gap has stopped getting smaller and the infeasibilities, if nonzero, have also stalled. Sometimes this indicates that the problem is infeasible.

The reasons to stop optimization outlined in the previous paragraph will be termed the *usual* stopping conditions in the following explanation.

However, when solving some problems, especially if the problems are large, the usual stopping criteria are inappropriate. PROC INTPOINT might stop optimizing prematurely. If it were allowed to perform additional optimization, a better solution would be found. On other occasions, PROC INTPOINT might do too much work. A sufficiently good solution might be reached several iterations before PROC INTPOINT eventually stops.

You can see PROC INTPOINT’s progress to the optimum by specifying PRINTLEVEL2=2. PROC INTPOINT will produce a table on the SAS log. A row of the table is generated during each iteration and consists of values of the affine step complementarity, the complementarity of the solution for the next iteration, the total bound infeasibility  $\sum_{i=1}^n infeas_{bi}$  (see the  $infeas_b$  array in the “Interior Point: Upper Bounds” section on page 63), and the total constraint infeasibility  $\sum_{i=1}^m infeas_{ci}$  (see the  $infeas_c$  array in the “Interior Point Algorithmic Details” section on page 58), the total dual infeasibility  $\sum_{i=1}^n infeas_{di}$  (see the  $infeas_d$  array in the “Interior Point Algorithmic Details” section on page 58). As optimization progresses, the values in all columns should converge to zero.

To tailor stopping criteria to your problem, you can use two sets of parameters: the STOP\_x and the KEEPGOING\_x parameters. The STOP\_x parameters (STOP\_C, STOP\_DG, STOP\_IB, STOP\_IC, STOP\_ID) are used to test for some condition at the beginning of each iteration and if met, to stop optimizing immediately. The KEEPGOING\_x parameters (KEEPGOING\_C, KEEPGOING\_DG, KEEPGOING\_IB, KEEPGOING\_IC, KEEPGOING\_ID) are used when PROC INTPOINT would ordinarily stop optimizing but does not if some conditions are not met.

For the sake of conciseness, a set of options might be referred to as the part of the option name they have in common followed by the suffix x. For example, STOP\_C, STOP\_DG, STOP\_IB, STOP\_IC, and STOP\_ID will collectively be referred to as STOP\_x.

At the beginning of each iteration, PROC INTPOINT will test whether complementarity is  $\leq$  STOP\_C (provided you specified a STOP\_C parameter) and if it is, PROC INTPOINT will stop optimizing. If the duality gap is  $\leq$  STOP\_DG (provided you specified a STOP\_DG parameter), PROC INTPOINT will stop optimizing immediately. This is true as well for the other STOP\_x parameters that are related to infeasibilities, STOP\_IB, STOP\_IC, and STOP\_ID.

For example, if you want PROC INTPOINT to stop optimizing for the usual stopping conditions, plus the additional condition, complementarity  $\leq$  100 or duality gap  $\leq$  0.001, then use:

```
proc intpoint stop_c=100 stop_dg=0.001
```

If you want PROC INTPOINT to stop optimizing for the usual stopping conditions, plus the additional condition, complementarity  $\leq$  1000 and duality gap  $\leq$  0.001 and constraint infeasibility  $\leq$  0.0001, then use:

```
proc intpoint  
  and_stop_c=1000 and_stop_dg=0.01 and_stop_ic=0.0001
```

Unlike the STOP\_x parameters that cause PROC INTPOINT to stop optimizing when any one of them is satisfied, the corresponding AND\_STOP\_x parameters (AND\_STOP\_C, AND\_STOP\_DG, AND\_STOP\_IB, AND\_STOP\_IC, and AND\_STOP\_ID) cause PROC INTPOINT to stop only if all (more precisely, all that are specified) options are satisfied. For example, if PROC INTPOINT should stop optimizing when

- complementarity  $\leq$  100 or duality gap  $\leq$  0.001 or
- complementarity  $\leq$  1000 and duality gap  $\leq$  0.001 and constraint infeasibility  $\leq$  0.000

then use:

```
proc intpoint  
  stop_c=100 stop_dg=0.001  
  and_stop_c=1000 and_stop_dg=0.01 and_stop_ic=0.0001
```

Just as the STOP\_x parameters have AND\_STOP\_x partners, the KEEPGOING\_x parameters have AND\_KEEPPGOING\_x partners. The role of the KEEPGOING\_x and AND\_KEEPPGOING\_x parameters is to prevent optimization from stopping too early, even though a usual stopping criteria is met.

When PROC INTPOINT detects that it should stop optimizing for a usual stopping condition,

- it will test whether complementarity is  $>$  KEEPGOING\_C (provided you specified a KEEPGOING\_C parameter), and if it is, PROC INTPOINT will perform more optimization.

- Otherwise, PROC INTPOINT will then test whether the primal-dual gap is  $> \text{KEEPGOING\_DG}$  (provided you specified a `KEEPGOING_DG` parameter), and if it is, PROC INTPOINT will perform more optimization.
- Otherwise, PROC INTPOINT will then test whether the total bound infeasibility  $\sum_{i=1}^n \text{infeas}_{bi} > \text{KEEPGOING\_IB}$  (provided you specified a `KEEPGOING_IB` parameter), and if it is, PROC INTPOINT will perform more optimization.
- Otherwise, PROC INTPOINT will then test whether the total constraint infeasibility  $\sum_{i=1}^m \text{infeas}_{ci} > \text{KEEPGOING\_IC}$  (provided you specified a `KEEPGOING_IC` parameter), and if it is, PROC INTPOINT will perform more optimization.
- Otherwise, PROC INTPOINT will then test whether the total dual infeasibility  $\sum_{i=1}^n \text{infeas}_{di} > \text{KEEPGOING\_ID}$  (provided you specified a `KEEPGOING_ID` parameter), and if it is, PROC INTPOINT will perform more optimization.
- Otherwise it will test whether complementarity is  $> \text{AND\_KEEPGOING\_C}$  (provided you specified an `AND\_KEEPGOING_C` parameter), *and* the primal-dual gap is  $> \text{AND\_KEEPGOING\_DG}$  (provided you specified an `AND\_KEEPGOING_DG` parameter), *and* the total bound infeasibility  $\sum_{i=1}^n \text{infeas}_{bi} > \text{AND\_KEEPGOING\_IB}$  (provided you specified an `AND\_KEEPGOING_IB` parameter), *and* the total constraint infeasibility  $\sum_{i=1}^m \text{infeas}_{ci} > \text{AND\_KEEPGOING\_IC}$  (provided you specified an `AND\_KEEPGOING_IC` parameter), *and* the total dual infeasibility  $\sum_{i=1}^n \text{infeas}_{di} > \text{AND\_KEEPGOING\_ID}$  (provided you specified an `AND\_KEEPGOING_ID` parameter), and if it is, PROC INTPOINT will perform more optimization.

If all these tests to decide whether more optimization should be performed are false, optimization is stopped.

The following PROC INTPOINT example will be used to illustrate how several stopping criteria options can be used together:

```
proc intpoint
  stop_c=1000
  and_stop_c=2000 and_stop_dg=0.01
  and_stop_ib=1 and_stop_ic=1 and_stop_id=1
  keepgoing_c=1500
  and_keepgoing_c=2500 and_keepgoing_dg=0.05
  and_keepgoing_ib=1 and_keepgoing_ic=1 and_keepgoing_id=1
```

At the beginning of each iteration, PROC INTPOINT will stop optimizing if

- complementarity  $\leq 1000$  or
- complementarity  $\leq 2000$  and duality gap  $\leq 0.01$  and the total bound, constraint, and dual infeasibilities are each  $\leq 1$

When PROC INTPOINT determines it should stop optimizing because a usual stopping condition is met, it will stop optimizing only if

- complementarity  $\leq 1500$  or
- complementarity  $\leq 2500$  and duality gap  $\leq 0.05$  and the total bound, constraint, and dual infeasibilities are each  $\leq 1$



## Examples

The following examples illustrate some of the capabilities of PROC INTPOINT. These examples, together with the other SAS/OR examples, can be found in the SAS sample library.

In order to illustrate variations in the use of the INTPOINT procedure, Example 4.1 through Example 4.5 use data from a company that produces two sizes of televisions. The company makes televisions with a diagonal screen measurement of either 19 inches or 25 inches. These televisions are made between March and May at both of the company's two factories. Each factory has a limit on the total number of televisions of each screen dimension that can be made during those months.

The televisions are distributed to one of two shops, stored at the factory where they were made, and sold later or shipped to the other factory. Some sets can be used to fill back-orders from the previous months. Each shop demands a number of each type of TV for the months March through May. The following network in Figure 4.12 illustrates the model. Arc costs can be interpreted as production costs, storage costs, back-order penalty costs, inter-factory transportation costs, and sales profits. The arcs can have capacities and lower flow bounds.

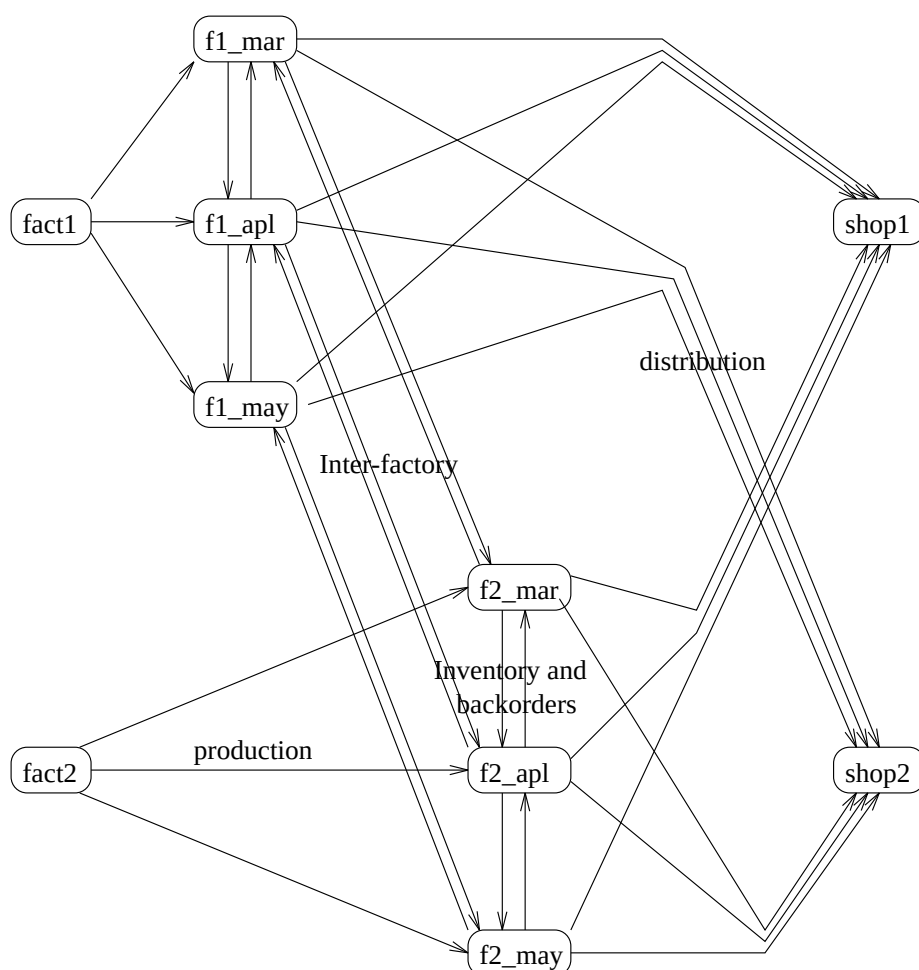


Figure 4.12. TV Problem

There are two similarly structured networks, one for the 19-inch televisions and the other for the 25-inch screen TVs. The minimum cost production, inventory, and distribution plan for both TV types can be determined in the same run of PROC INTPOINT. To ensure that node names are unambiguous, the names of nodes in the 19-inch network have suffix \_1, and the node names in the 25-inch network have suffix \_2.

### Example 4.1. Production, Inventory, Distribution Problem

The following code shows how to save a specific problem's data in data sets and solve the model with PROC INTPOINT:

```

title 'Production Planning/Inventory/Distribution';
title2 'Minimum Cost Flow problem';
title3;
data node0;
    input _node_ $ _supdem_ ;
    datalines;
fact1_1    1000
fact2_1     850
fact1_2    1000
fact2_2    1500
shop1_1    -900
shop2_1    -900
shop1_2    -900
shop2_2   -1450
;
data arc0;
    input _tail_ $ _head_ $ _cost_ _capac_ _lo_ diagonal factory
          key_id $10. mth_made $ _name_&$17. ;
    datalines;
fact1_1  f1_mar_1  127.9   500 50 19 1 production March prod f1 19 mar
fact1_1  f1_apr_1   78.6   600 50 19 1 production April prod f1 19 apl
fact1_1  f1_may_1   95.1   400 50 19 1 production May .
f1_mar_1 f1_apr_1   15      50 . 19 1 storage March .
f1_apr_1 f1_may_1   12      50 . 19 1 storage April .
f1_apr_1 f1_mar_1   28      20 . 19 1 backorder April back f1 19 apl
f1_may_1 f1_apr_1   28      20 . 19 1 backorder May back f1 19 may
f1_mar_1 f2_mar_1   11      . . 19 . f1_to_2 March .
f1_apr_1 f2_apr_1   11      . . 19 . f1_to_2 April .
f1_may_1 f2_may_1   16      . . 19 . f1_to_2 May .
f1_mar_1 shop1_1  -327.65  250 . 19 1 sales March .
f1_apr_1 shop1_1  -300     250 . 19 1 sales April .
f1_may_1 shop1_1  -285     250 . 19 1 sales May .
f1_mar_1 shop2_1  -362.74  250 . 19 1 sales March .
f1_apr_1 shop2_1  -300     250 . 19 1 sales April .
f1_may_1 shop2_1  -245     250 . 19 1 sales May .
fact2_1  f2_mar_1   88.0   450 35 19 2 production March prod f2 19 mar
fact2_1  f2_apr_1   62.4   480 35 19 2 production April prod f2 19 apl
fact2_1  f2_may_1  133.8   250 35 19 2 production May .
f2_mar_1 f2_apr_1   18      30 . 19 2 storage March .
f2_apr_1 f2_may_1   20      30 . 19 2 storage April .
f2_apr_1 f2_mar_1   17      15 . 19 2 backorder April back f2 19 apl
f2_may_1 f2_apr_1   25      15 . 19 2 backorder May back f2 19 may
f2_mar_1 f1_mar_1   10      40 . 19 . f2_to_1 March .
f2_apr_1 f1_apr_1   11      40 . 19 . f2_to_1 April .

```

```

f2_may_1 f1_may_1 13 40 . 19 . f2_to_1 May .
f2_mar_1 shop1_1 -297.4 250 . 19 2 sales March .
f2_apr_1 shop1_1 -290 250 . 19 2 sales April .
f2_may_1 shop1_1 -292 250 . 19 2 sales May .
f2_mar_1 shop2_1 -272.7 250 . 19 2 sales March .
f2_apr_1 shop2_1 -312 250 . 19 2 sales April .
f2_may_1 shop2_1 -299 250 . 19 2 sales May .
fact1_2 f1_mar_2 217.9 400 40 25 1 production March prod f1 25 mar
fact1_2 f1_apr_2 174.5 550 50 25 1 production April prod f1 25 apl
fact1_2 f1_may_2 133.3 350 40 25 1 production May .
f1_mar_2 f1_apr_2 20 40 . 25 1 storage March .
f1_apr_2 f1_may_2 18 40 . 25 1 storage April .
f1_apr_2 f1_mar_2 32 30 . 25 1 backorder April back f1 25 apl
f1_may_2 f1_apr_2 41 15 . 25 1 backorder May back f1 25 may
f1_mar_2 f2_mar_2 23 . . 25 . f1_to_2 March .
f1_apr_2 f2_apr_2 23 . . 25 . f1_to_2 April .
f1_may_2 f2_may_2 26 . . 25 . f1_to_2 May .
f1_mar_2 shop1_2 -559.76 . . 25 1 sales March .
f1_apr_2 shop1_2 -524.28 . . 25 1 sales April .
f1_may_2 shop1_2 -475.02 . . 25 1 sales May .
f1_mar_2 shop2_2 -623.89 . . 25 1 sales March .
f1_apr_2 shop2_2 -549.68 . . 25 1 sales April .
f1_may_2 shop2_2 -460.00 . . 25 1 sales May .
fact2_2 f2_mar_2 182.0 650 35 25 2 production March prod f2 25 mar
fact2_2 f2_apr_2 196.7 680 35 25 2 production April prod f2 25 apl
fact2_2 f2_may_2 201.4 550 35 25 2 production May .
f2_mar_2 f2_apr_2 28 50 . 25 2 storage March .
f2_apr_2 f2_may_2 38 50 . 25 2 storage April .
f2_apr_2 f2_mar_2 31 15 . 25 2 backorder April back f2 25 apl
f2_may_2 f2_apr_2 54 15 . 25 2 backorder May back f2 25 may
f2_mar_2 f1_mar_2 20 25 . 25 . f2_to_1 March .
f2_apr_2 f1_apr_2 21 25 . 25 . f2_to_1 April .
f2_may_2 f1_may_2 43 25 . 25 . f2_to_1 May .
f2_mar_2 shop1_2 -567.83 500 . 25 2 sales March .
f2_apr_2 shop1_2 -542.19 500 . 25 2 sales April .
f2_may_2 shop1_2 -461.56 500 . 25 2 sales May .
f2_mar_2 shop2_2 -542.83 500 . 25 2 sales March .
f2_apr_2 shop2_2 -559.19 500 . 25 2 sales April .
f2_may_2 shop2_2 -489.06 500 . 25 2 sales May .
;
proc intpoint
  bytes=1000000
  printlevel=2
  nodedata=node0
  arcdata=arc0
  conout=arc1;
run;

```

```

proc print data=arc1;
  var _from_ _to_ _cost_ _capac_ _lo_ _flow_ _fcost_
      diagonal factory key_id mth_made;
  sum _fcost_;
run;

```

The following notes appear on the SAS log:

```

NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 4 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 4350 , total demand= 4150 .
NOTE: Number of arcs= 64 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 21 .
NOTE: Number of >= constraints= 0 .
NOTE: Number of constraint coefficients= 136 .
NOTE: Number of variables= 68 .
NOTE: 0 columns, 0 rows and 0 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 54 nonzero elements in A * A transpose.
NOTE: Of the 21 rows and columns, 14 are sparse.
NOTE: There are 59 nonzero superdiagonal elements in the
      sparse rows of the factored A * A transpose. This
      includes fill-in.
NOTE: There are 80 operations of the form
      u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q] to factorize the
      sparse rows of A * A transpose.

```

| Iter | Complem_aff | Complem-ity | Duality_gap  | Tot_infeasb | Tot_infeasd  | Tot_infeasd |
|------|-------------|-------------|--------------|-------------|--------------|-------------|
| 0    | -1.000000   | 167086681   | 0.835362     | 52835       | 26238        | 34922       |
| 1    | 35446536    | 17335378    | 0.912917     | 1663.290349 | 825.996205   | 0           |
| 2    | 1957002     | 878825      | 0.416636     | 0           | 1.956907E-12 | 0           |
| 3    | 550439      | 237247      | 0.155174     | 0           | 0            | 0           |
| 4    | 126652      | 53958       | 0.039021     | 0           | 0            | 0           |
| 5    | 27526       | 18386       | 0.013499     | 0           | 0            | 0           |
| 6    | 9153.811245 | 3029.529789 | 0.002242     | 0           | 0            | 0           |
| 7    | 905.846996  | 614.526357  | 0.000455     | 0           | 0            | 0           |
| 8    | 219.526873  | 73.592013   | 0.000054542  | 0           | 0            | 0           |
| 9    | 11.383058   | 2.111334    | 0.000001565  | 0           | 0            | 0           |
| 10   | 0.089558    | 0.000361    | 2.672549E-10 | 0           | 0            | 0           |

```

NOTE: Primal-Dual Predictor-Corrector Interior point
      algorithm performed 10 iterations.
NOTE: Objective = -1281110.35.
NOTE: The data set WORK.ARC1 has 64 observations and 18
      variables.
NOTE: There were 64 observations read from the data set
      WORK.ARC0.
NOTE: There were 8 observations read from the data set
      WORK.NODE0.
NOTE: The data set WORK.ARC1 has 64 observations and 18
      variables.

```

The solution is given in the CONOUT=arc1 data sets. In the CONOUT= data set, shown in Output 4.1.1, the variables diagonal, factory, key\_id, and mth\_made form an implicit ID list. The diagonal variable has one of two values, 19 or 25. factory also has one of two values, 1 or 2, to denote the factory where either production or storage occurs, from where TVs are either sold to shops or satisfy back-orders. production, storage, sales, and backorder are values of the key\_id variable.

Other values of this variable, f1\_to\_2 and f2\_to\_1, are used when flow through arcs represents the transportation of TVs between factories. The mth\_made variable has values March, April, and May, the months when TVs that are modeled as flow through an arc were made (assuming that no televisions are stored for more than one month and none manufactured in May are used to fill March back-orders).

These ID variables can be used after the PROC INTPOINT run to produce reports and perform analysis on particular parts of the company's operation. For example, reports can be generated for production numbers for each factory; optimal sales figures for each shop; and how many TVs should be stored, used to fill back-orders, sent to the other factory, or any combination of these, for TVs with a particular screen, those produced in a particular month, or both.

**Output 4.1.1.** CONOUT=ARC1

| t<br>a<br>i<br>l | h<br>e<br>a<br>d | c<br>o<br>s<br>t | c<br>a<br>p<br>a<br>c<br>i<br>t<br>y | F<br>L<br>O<br>W | F<br>C<br>O<br>S<br>T | d<br>i<br>f<br>f<br>e<br>r<br>e<br>n<br>c<br>e | k<br>e<br>y | m<br>t<br>h |
|------------------|------------------|------------------|--------------------------------------|------------------|-----------------------|------------------------------------------------|-------------|-------------|
| fact1_1          | f1_apr_1         | 78.6             | 600 50 600.000                       | 47160.00         | 19 1                  | production                                     | April       |             |
| f1_mar_1         | f1_apr_1         | 15.0             | 50 0 0.000                           | 0.00             | 19 1                  | storage                                        | March       |             |
| f1_may_1         | f1_apr_1         | 28.0             | 20 0 0.000                           | 0.00             | 19 1                  | backorder                                      | May         |             |
| f2_apr_1         | f1_apr_1         | 11.0             | 40 0 0.000                           | 0.00             | 19 .                  | f2_to_1                                        | April       |             |
| fact1_2          | f1_apr_2         | 174.5            | 550 50 550.000                       | 95975.00         | 25 1                  | production                                     | April       |             |
| f1_mar_2         | f1_apr_2         | 20.0             | 40 0 0.000                           | 0.00             | 25 1                  | storage                                        | March       |             |
| f1_may_2         | f1_apr_2         | 41.0             | 15 0 15.000                          | 615.00           | 25 1                  | backorder                                      | May         |             |
| f2_apr_2         | f1_apr_2         | 21.0             | 25 0 0.000                           | 0.00             | 25 .                  | f2_to_1                                        | April       |             |
| fact1_1          | f1_mar_1         | 127.9            | 500 50 345.000                       | 44125.49         | 19 1                  | production                                     | March       |             |
| f1_apr_1         | f1_mar_1         | 28.0             | 20 0 20.000                          | 560.00           | 19 1                  | backorder                                      | April       |             |
| f2_mar_1         | f1_mar_1         | 10.0             | 40 0 40.000                          | 400.00           | 19 .                  | f2_to_1                                        | March       |             |
| fact1_2          | f1_mar_2         | 217.9            | 400 40 400.000                       | 87160.00         | 25 1                  | production                                     | March       |             |
| f1_apr_2         | f1_mar_2         | 32.0             | 30 0 30.000                          | 960.00           | 25 1                  | backorder                                      | April       |             |
| f2_mar_2         | f1_mar_2         | 20.0             | 25 0 25.000                          | 500.00           | 25 .                  | f2_to_1                                        | March       |             |
| fact1_1          | f1_may_1         | 95.1             | 400 50 50.000                        | 4755.01          | 19 1                  | production                                     | May         |             |
| f1_apr_1         | f1_may_1         | 12.0             | 50 0 50.000                          | 600.00           | 19 1                  | storage                                        | April       |             |
| f2_may_1         | f1_may_1         | 13.0             | 40 0 0.000                           | 0.00             | 19 .                  | f2_to_1                                        | May         |             |
| fact1_2          | f1_may_2         | 133.3            | 350 40 40.000                        | 5332.00          | 25 1                  | production                                     | May         |             |
| f1_apr_2         | f1_may_2         | 18.0             | 40 0 0.000                           | 0.00             | 25 1                  | storage                                        | April       |             |
| f2_may_2         | f1_may_2         | 43.0             | 25 0 0.000                           | 0.00             | 25 .                  | f2_to_1                                        | May         |             |
| f1_apr_1         | f2_apr_1         | 11.0             | 99999999 0 30.000                    | 330.00           | 19 .                  | f1_to_2                                        | April       |             |
| fact2_1          | f2_apr_1         | 62.4             | 480 35 480.000                       | 29952.00         | 19 2                  | production                                     | April       |             |
| f2_mar_1         | f2_apr_1         | 18.0             | 30 0 0.000                           | 0.00             | 19 2                  | storage                                        | March       |             |
| f2_may_1         | f2_apr_1         | 25.0             | 15 0 0.000                           | 0.00             | 19 2                  | backorder                                      | May         |             |
| f1_apr_2         | f2_apr_2         | 23.0             | 99999999 0 0.000                     | 0.00             | 25 .                  | f1_to_2                                        | April       |             |
| fact2_2          | f2_apr_2         | 196.7            | 680 35 680.000                       | 133756.00        | 25 2                  | production                                     | April       |             |
| f2_mar_2         | f2_apr_2         | 28.0             | 50 0 0.000                           | 0.00             | 25 2                  | storage                                        | March       |             |
| f2_may_2         | f2_apr_2         | 54.0             | 15 0 15.000                          | 810.00           | 25 2                  | backorder                                      | May         |             |
| f1_mar_1         | f2_mar_1         | 11.0             | 99999999 0 0.000                     | 0.00             | 19 .                  | f1_to_2                                        | March       |             |
| fact2_1          | f2_mar_1         | 88.0             | 450 35 290.000                       | 25520.00         | 19 2                  | production                                     | March       |             |
| f2_apr_1         | f2_mar_1         | 17.0             | 15 0 0.000                           | 0.00             | 19 2                  | backorder                                      | April       |             |
| f1_mar_2         | f2_mar_2         | 23.0             | 99999999 0 0.000                     | 0.00             | 25 .                  | f1_to_2                                        | March       |             |

| t<br>a<br>i<br>l | h<br>e<br>a<br>d | c<br>o<br>s<br>t | c<br>a<br>p<br>a<br>c<br>i<br>t<br>y | F<br>L<br>O<br>W | d<br>i<br>f<br>f<br>e<br>r<br>e<br>n<br>c<br>e | m<br>t<br>h      |
|------------------|------------------|------------------|--------------------------------------|------------------|------------------------------------------------|------------------|
| fact2_2          | f2_mar_2         | 182.00           | 650 35                               | 645.000          | 117390.00 25 2                                 | production March |
| f2_apr_2         | f2_mar_2         | 31.00            | 15 0                                 | 0.000            | 0.00 25 2                                      | backorder April  |
| f1_may_1         | f2_may_1         | 16.00            | 99999999                             | 0 100.000        | 1600.00 19 .                                   | f1_to_2 May      |
| fact2_1          | f2_may_1         | 133.80           | 250 35                               | 35.000           | 4683.00 19 2                                   | production May   |
| f2_apr_1         | f2_may_1         | 20.00            | 30 0                                 | 15.000           | 300.00 19 2                                    | storage April    |
| f1_may_2         | f2_may_2         | 26.00            | 99999999                             | 0 0.000          | 0.00 25 .                                      | f1_to_2 May      |
| fact2_2          | f2_may_2         | 201.40           | 550 35                               | 35.000           | 7049.00 25 2                                   | production May   |
| f2_apr_2         | f2_may_2         | 38.00            | 50 0                                 | 0.000            | 0.00 25 2                                      | storage April    |
| f1_mar_1         | shop1_1          | -327.65          | 250 0                                | 155.000          | -50785.73 19 1                                 | sales March      |
| f1_apr_1         | shop1_1          | -300.00          | 250 0                                | 250.000          | -75000.00 19 1                                 | sales April      |
| f1_may_1         | shop1_1          | -285.00          | 250 0                                | 0.000            | 0.00 19 1                                      | sales May        |
| f2_mar_1         | shop1_1          | -297.40          | 250 0                                | 250.000          | -74350.00 19 2                                 | sales March      |
| f2_apr_1         | shop1_1          | -290.00          | 250 0                                | 245.000          | -71050.02 19 2                                 | sales April      |
| f2_may_1         | shop1_1          | -292.00          | 250 0                                | 0.000            | 0.00 19 2                                      | sales May        |
| f1_mar_2         | shop1_2          | -559.76          | 99999999                             | 0 0.000          | 0.00 25 1                                      | sales March      |
| f1_apr_2         | shop1_2          | -524.28          | 99999999                             | 0 0.000          | 0.00 25 1                                      | sales April      |
| f1_may_2         | shop1_2          | -475.02          | 99999999                             | 0 25.000         | -11875.50 25 1                                 | sales May        |
| f2_mar_2         | shop1_2          | -567.83          | 500 0                                | 500.000          | -283915.00 25 2                                | sales March      |
| f2_apr_2         | shop1_2          | -542.19          | 500 0                                | 375.000          | -203321.25 25 2                                | sales April      |
| f2_may_2         | shop1_2          | -461.56          | 500 0                                | 0.000            | 0.00 25 2                                      | sales May        |
| f1_mar_1         | shop2_1          | -362.74          | 250 0                                | 250.000          | -90685.00 19 1                                 | sales March      |
| f1_apr_1         | shop2_1          | -300.00          | 250 0                                | 250.000          | -75000.00 19 1                                 | sales April      |
| f1_may_1         | shop2_1          | -245.00          | 250 0                                | 0.000            | 0.00 19 1                                      | sales May        |
| f2_mar_1         | shop2_1          | -272.70          | 250 0                                | 0.000            | 0.00 19 2                                      | sales March      |
| f2_apr_1         | shop2_1          | -312.00          | 250 0                                | 250.000          | -78000.00 19 2                                 | sales April      |
| f2_may_1         | shop2_1          | -299.00          | 250 0                                | 150.000          | -44850.00 19 2                                 | sales May        |
| f1_mar_2         | shop2_2          | -623.89          | 99999999                             | 0 455.000        | -283869.95 25 1                                | sales March      |
| f1_apr_2         | shop2_2          | -549.68          | 99999999                             | 0 535.000        | -294078.80 25 1                                | sales April      |
| f1_may_2         | shop2_2          | -460.00          | 99999999                             | 0 0.000          | 0.00 25 1                                      | sales May        |
| f2_mar_2         | shop2_2          | -542.83          | 500 0                                | 120.000          | -65139.60 25 2                                 | sales March      |
| f2_apr_2         | shop2_2          | -559.19          | 500 0                                | 320.000          | -178940.80 25 2                                | sales April      |
| f2_may_2         | shop2_2          | -489.06          | 500 0                                | 20.000           | -9781.20 25 2                                  | sales May        |

## Example 4.2. Altering Arc Data

This example examines the effect of changing some of the arc costs. The back-order penalty costs are increased by 20 percent. The sales profit of 25-inch TVs sent to the shops in May is increased by 30 units. The backorder penalty costs of 25-inch TVs manufactured in May for April consumption is decreased by 30 units. The production cost of 19-inch and 25-inch TVs made in May are decreased by 5 units and 20 units, respectively. How does the optimal solution of the network after these arc cost alterations compare with the optimum of the original network?

These SAS statements produce the new NODEDATA= and ARCDATA= data sets:

```

title2 'Minimum Cost Flow problem- Altered Arc Data';
data arc2;
  set arc1;
  oldcost=_cost_;
  oldfc=_fcost_;
  oldflow=_flow_;
  if key_id='backorder'
    then _cost_=_cost_*1.2;
  else if _tail_='f2_may_2' then _cost_=_cost_-30;
  if key_id='production' & mth_made='May' then
    if diagonal=19 then _cost_=_cost_-5;
    else _cost_=_cost_-20;

```

```

proc intpoint
  bytes=100000
  printlevel=2
  nodedata=node0
  arcdata=arc2
  conout=arc3;
run;

proc print data=arc3;
  var _tail_ _head_ _capac_ _lo_ _supply_ _demand_ _name_
     _cost_ _flow_ _fcost_ oldcost oldflow oldfc
     diagonal factory key_id mth_made;
  /* to get this variable order */
     sum oldfc _fcost_;
run;

```

The following notes appear on the SAS log:

```

NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 4 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 4350 , total demand= 4150 .
NOTE: Number of arcs= 64 .
NOTE: The following messages relate to the equivalent Linear
      Program solved by the Interior Point algorithm.
NOTE: Number of <= constraints= 0 .
NOTE: Number of == constraints= 21 .
NOTE: Number of >= constraints= 0 .
NOTE: Number of constraint coefficients= 136 .
NOTE: Number of variables= 68 .
NOTE: 0 columns, 0 rows and 0 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 54 nonzero elements in A * A transpose.
NOTE: Of the 21 rows and columns, 14 are sparse.
NOTE: There are 59 nonzero superdiagonal elements in the
      sparse rows of the factored A * A transpose. This
      includes fill-in.
NOTE: There are 80 operations of the form
      u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q] to factorize the
      sparse rows of A * A transpose.

```

| Iter | Complem_aff | Complem-ity | Duality_gap  | Tot_infeasb | Tot_infeasc  | Tot_infeasd |
|------|-------------|-------------|--------------|-------------|--------------|-------------|
| 0    | -1.000000   | 167898972   | 0.834344     | 52835       | 26238        | 35092       |
| 1    | 35575010    | 17420416    | 0.911474     | 1664.001693 | 826.349461   | 0           |
| 2    | 1969763     | 739523      | 0.369979     | 0           | 1.267542E-12 | 0           |
| 3    | 373828      | 189905      | 0.127240     | 0           | 0            | 0           |
| 4    | 117843      | 45042       | 0.032721     | 0           | 0            | 0           |
| 5    | 26090       | 15668       | 0.011493     | 0           | 0            | 0           |
| 6    | 8608.105391 | 2994.589912 | 0.002212     | 0           | 0            | 0           |
| 7    | 1477.385197 | 411.268625  | 0.000304     | 0           | 0            | 0           |
| 8    | 103.724839  | 24.733676   | 0.000018304  | 0           | 0            | 0           |
| 9    | 2.140932    | 0.002805    | 2.0758339E-9 | 0           | 0            | 0           |

```

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm
      performed 9 iterations.

```

```

NOTE: Objective = -1285086.45.

```

```

NOTE: The data set WORK.ARC3 has 64 observations and 21
      variables.

```

**NOTE:** There were 64 observations read from the data set **WORK.ARC2**.

**NOTE:** There were 8 observations read from the data set **WORK.NODE0**.

**NOTE:** The data set **WORK.ARC3** has 64 observations and 21 variables.

The solution is displayed in Output 4.2.1.



Output 4.2.1. CONOUT=ARC3

| _tail_    | _head_   | _capac_  | _lo_      | _SUPPLY_ | _DEMAND_ | _name_         | _cost_   | _FLOW_  |
|-----------|----------|----------|-----------|----------|----------|----------------|----------|---------|
| fact1_1   | f1_apr_1 | 600      | 50        | 1000     | .        | prod f1 19 apl | 78.6     | 540.000 |
| f1_mar_1  | f1_apr_1 | 50       | 0         | .        | .        |                | 15.0     | 0.000   |
| f1_may_1  | f1_apr_1 | 20       | 0         | .        | .        | back f1 19 may | 33.6     | 0.000   |
| f2_apr_1  | f1_apr_1 | 40       | 0         | .        | .        |                | 11.0     | 0.000   |
| fact1_2   | f1_apr_2 | 550      | 50        | 1000     | .        | prod f1 25 apl | 174.5    | 250.000 |
| f1_mar_2  | f1_apr_2 | 40       | 0         | .        | .        |                | 20.0     | 0.000   |
| f1_may_2  | f1_apr_2 | 15       | 0         | .        | .        | back f1 25 may | 49.2     | 15.000  |
| f2_apr_2  | f1_apr_2 | 25       | 0         | .        | .        |                | 21.0     | 0.000   |
| fact1_1   | f1_mar_1 | 500      | 50        | 1000     | .        | prod f1 19 mar | 127.9    | 340.000 |
| f1_apr_1  | f1_mar_1 | 20       | 0         | .        | .        | back f1 19 apl | 33.6     | 20.000  |
| f2_mar_1  | f1_mar_1 | 40       | 0         | .        | .        |                | 10.0     | 40.000  |
| fact1_2   | f1_mar_2 | 400      | 40        | 1000     | .        | prod f1 25 mar | 217.9    | 400.000 |
| f1_apr_2  | f1_mar_2 | 30       | 0         | .        | .        | back f1 25 apl | 38.4     | 30.000  |
| f2_mar_2  | f1_mar_2 | 25       | 0         | .        | .        |                | 20.0     | 25.000  |
| fact1_1   | f1_may_1 | 400      | 50        | 1000     | .        |                | 90.1     | 115.000 |
| f1_apr_1  | f1_may_1 | 50       | 0         | .        | .        |                | 12.0     | 0.000   |
| f2_may_1  | f1_may_1 | 40       | 0         | .        | .        |                | 13.0     | 0.000   |
| fact1_2   | f1_may_2 | 350      | 40        | 1000     | .        |                | 113.3    | 350.000 |
| f1_apr_2  | f1_may_2 | 40       | 0         | .        | .        |                | 18.0     | 0.000   |
| f2_may_2  | f1_may_2 | 25       | 0         | .        | .        |                | 13.0     | 0.000   |
| f1_apr_1  | f2_apr_1 | 99999999 | 0         | .        | .        |                | 11.0     | 20.000  |
| fact2_1   | f2_apr_1 | 480      | 35        | 850      | .        | prod f2 19 apl | 62.4     | 480.000 |
| f2_mar_1  | f2_apr_1 | 30       | 0         | .        | .        |                | 18.0     | 0.000   |
| f2_may_1  | f2_apr_1 | 15       | 0         | .        | .        | back f2 19 may | 30.0     | 0.000   |
| f1_apr_2  | f2_apr_2 | 99999999 | 0         | .        | .        |                | 23.0     | 0.000   |
| fact2_2   | f2_apr_2 | 680      | 35        | 1500     | .        | prod f2 25 apl | 196.7    | 680.000 |
| f2_mar_2  | f2_apr_2 | 50       | 0         | .        | .        |                | 28.0     | 0.000   |
| _FCOST_   | oldcost  | oldflow  | oldfc     | diagonal | factory  | key_id         | mth_made |         |
| 42443.99  | 78.6     | 600.000  | 47160.00  | 19       | 1        | production     | April    |         |
| 0.00      | 15.0     | 0.000    | 0.00      | 19       | 1        | storage        | March    |         |
| 0.00      | 28.0     | 0.000    | 0.00      | 19       | 1        | backorder      | May      |         |
| 0.00      | 11.0     | 0.000    | 0.00      | 19       | .        | f2_to_1        | April    |         |
| 43625.00  | 174.5    | 550.000  | 95975.00  | 25       | 1        | production     | April    |         |
| 0.00      | 20.0     | 0.000    | 0.00      | 25       | 1        | storage        | March    |         |
| 738.00    | 41.0     | 15.000   | 615.00    | 25       | 1        | backorder      | May      |         |
| 0.00      | 21.0     | 0.000    | 0.00      | 25       | .        | f2_to_1        | April    |         |
| 43486.01  | 127.9    | 345.000  | 44125.49  | 19       | 1        | production     | March    |         |
| 672.00    | 28.0     | 20.000   | 560.00    | 19       | 1        | backorder      | April    |         |
| 400.00    | 10.0     | 40.000   | 400.00    | 19       | .        | f2_to_1        | March    |         |
| 87160.00  | 217.9    | 400.000  | 87160.00  | 25       | 1        | production     | March    |         |
| 1152.00   | 32.0     | 30.000   | 960.00    | 25       | 1        | backorder      | April    |         |
| 500.00    | 20.0     | 25.000   | 500.00    | 25       | .        | f2_to_1        | March    |         |
| 10361.50  | 95.1     | 50.000   | 4755.01   | 19       | 1        | production     | May      |         |
| 0.00      | 12.0     | 50.000   | 600.00    | 19       | 1        | storage        | April    |         |
| 0.00      | 13.0     | 0.000    | 0.00      | 19       | .        | f2_to_1        | May      |         |
| 39655.00  | 133.3    | 40.000   | 5332.00   | 25       | 1        | production     | May      |         |
| 0.00      | 18.0     | 0.000    | 0.00      | 25       | 1        | storage        | April    |         |
| 0.00      | 43.0     | 0.000    | 0.00      | 25       | .        | f2_to_1        | May      |         |
| 220.00    | 11.0     | 30.000   | 330.00    | 19       | .        | f1_to_2        | April    |         |
| 29952.00  | 62.4     | 480.000  | 29952.00  | 19       | 2        | production     | April    |         |
| 0.00      | 18.0     | 0.000    | 0.00      | 19       | 2        | storage        | March    |         |
| 0.00      | 25.0     | 0.000    | 0.00      | 19       | 2        | backorder      | May      |         |
| 0.00      | 23.0     | 0.000    | 0.00      | 25       | .        | f1_to_2        | April    |         |
| 133756.00 | 196.7    | 680.000  | 133756.00 | 25       | 2        | production     | April    |         |
| 0.00      | 28.0     | 0.000    | 0.00      | 25       | 2        | storage        | March    |         |
| _tail_    | _head_   | _capac_  | _lo_      | _SUPPLY_ | _DEMAND_ | _name_         | _cost_   | _FLOW_  |
| f2_may_2  | f2_apr_2 | 15       | 0         | .        | .        | back f2 25 may | 64.8     | 0.000   |
| f1_mar_1  | f2_mar_1 | 99999999 | 0         | .        | .        |                | 11.0     | 0.000   |
| fact2_1   | f2_mar_1 | 450      | 35        | 850      | .        | prod f2 19 mar | 88.0     | 290.000 |
| f2_apr_1  | f2_mar_1 | 15       | 0         | .        | .        | back f2 19 apl | 20.4     | 0.000   |
| f1_mar_2  | f2_mar_2 | 99999999 | 0         | .        | .        |                | 23.0     | 0.000   |
| _FCOST_   | oldcost  | oldflow  | oldfc     | diagonal | factory  | key_id         | mth_made |         |
| 0.00      | 54.0     | 15.000   | 810.00    | 25       | 2        | backorder      | May      |         |
| 0.00      | 11.0     | 0.000    | 0.00      | 19       | .        | f1_to_2        | March    |         |
| 25520.00  | 88.0     | 290.000  | 25520.00  | 19       | 2        | production     | March    |         |
| 0.00      | 17.0     | 0.000    | 0.00      | 19       | 2        | backorder      | April    |         |
| 0.00      | 23.0     | 0.000    | 0.00      | 25       | .        | f1_to_2        | March    |         |

| _tail_     | _head_   | _capac_  | _lo_       | _SUPPLY_ | _DEMAND_ | _name_         | _cost_   | _FLOW_  |
|------------|----------|----------|------------|----------|----------|----------------|----------|---------|
| fact2_2    | f2_mar_2 | 650      | 35         | 1500     | .        | prod f2 25 mar | 182.00   | 635.000 |
| f2_apr_2   | f2_mar_2 | 15       | 0          | .        | .        | back f2 25 apr | 37.20    | 0.000   |
| f1_may_1   | f2_may_1 | 99999999 | 0          | .        | .        |                | 16.00    | 115.000 |
| fact2_1    | f2_may_1 | 250      | 35         | 850      | .        |                | 128.80   | 35.000  |
| f2_apr_1   | f2_may_1 | 30       | 0          | .        | .        |                | 20.00    | 0.000   |
| f1_may_2   | f2_may_2 | 99999999 | 0          | .        | .        |                | 26.00    | 335.000 |
| fact2_2    | f2_may_2 | 550      | 35         | 1500     | .        |                | 181.40   | 35.000  |
| f2_apr_2   | f2_may_2 | 50       | 0          | .        | .        |                | 38.00    | 0.000   |
| f1_mar_1   | shop1_1  | 250      | 0          | .        | 900      |                | -327.65  | 150.000 |
| f1_apr_1   | shop1_1  | 250      | 0          | .        | 900      |                | -300.00  | 250.000 |
| f1_may_1   | shop1_1  | 250      | 0          | .        | 900      |                | -285.00  | 0.000   |
| f2_mar_1   | shop1_1  | 250      | 0          | .        | 900      |                | -297.40  | 250.000 |
| f2_apr_1   | shop1_1  | 250      | 0          | .        | 900      |                | -290.00  | 250.000 |
| f2_may_1   | shop1_1  | 250      | 0          | .        | 900      |                | -292.00  | 0.000   |
| f1_mar_2   | shop1_2  | 99999999 | 0          | .        | 900      |                | -559.76  | 0.000   |
| f1_apr_2   | shop1_2  | 99999999 | 0          | .        | 900      |                | -524.28  | 0.000   |
| f1_may_2   | shop1_2  | 99999999 | 0          | .        | 900      |                | -475.02  | 0.000   |
| f2_mar_2   | shop1_2  | 500      | 0          | .        | 900      |                | -567.83  | 500.000 |
| f2_apr_2   | shop1_2  | 500      | 0          | .        | 900      |                | -542.19  | 400.000 |
| f2_may_2   | shop1_2  | 500      | 0          | .        | 900      |                | -491.56  | 0.000   |
| f1_mar_1   | shop2_1  | 250      | 0          | .        | 900      |                | -362.74  | 250.000 |
| f1_apr_1   | shop2_1  | 250      | 0          | .        | 900      |                | -300.00  | 250.000 |
| f1_may_1   | shop2_1  | 250      | 0          | .        | 900      |                | -245.00  | 0.000   |
| f2_mar_1   | shop2_1  | 250      | 0          | .        | 900      |                | -272.70  | 0.000   |
| f2_apr_1   | shop2_1  | 250      | 0          | .        | 900      |                | -312.00  | 250.000 |
| f2_may_1   | shop2_1  | 250      | 0          | .        | 900      |                | -299.00  | 150.000 |
| f1_mar_2   | shop2_2  | 99999999 | 0          | .        | 1450     |                | -623.89  | 455.000 |
| _FCOST_    | oldcost  | oldflow  | oldfc      | diagonal | factory  | key_id         | mth_made |         |
| 115570.00  | 182.00   | 645.000  | 117390.00  | 25       | 2        | production     | March    |         |
| 0.00       | 31.00    | 0.000    | 0.00       | 25       | 2        | backorder      | April    |         |
| 1840.00    | 16.00    | 100.000  | 1600.00    | 19       | .        | f1_to_2        | May      |         |
| 4508.00    | 133.80   | 35.000   | 4683.00    | 19       | 2        | production     | May      |         |
| 0.00       | 20.00    | 15.000   | 300.00     | 19       | 2        | storage        | April    |         |
| 8710.00    | 26.00    | 0.000    | 0.00       | 25       | .        | f1_to_2        | May      |         |
| 6349.00    | 201.40   | 35.000   | 7049.00    | 25       | 2        | production     | May      |         |
| 0.00       | 38.00    | 0.000    | 0.00       | 25       | 2        | storage        | April    |         |
| -49147.54  | -327.65  | 155.000  | -50785.73  | 19       | 1        | sales          | March    |         |
| -75000.00  | -300.00  | 250.000  | -75000.00  | 19       | 1        | sales          | April    |         |
| 0.00       | -285.00  | 0.000    | 0.00       | 19       | 1        | sales          | May      |         |
| -74350.00  | -297.40  | 250.000  | -74350.00  | 19       | 2        | sales          | March    |         |
| -72499.97  | -290.00  | 245.000  | -71050.02  | 19       | 2        | sales          | April    |         |
| 0.00       | -292.00  | 0.000    | 0.00       | 19       | 2        | sales          | May      |         |
| 0.00       | -559.76  | 0.000    | 0.00       | 25       | 1        | sales          | March    |         |
| 0.00       | -524.28  | 0.000    | 0.00       | 25       | 1        | sales          | April    |         |
| -0.02      | -475.02  | 25.000   | -11875.50  | 25       | 1        | sales          | May      |         |
| -283915.00 | -567.83  | 500.000  | -283915.00 | 25       | 2        | sales          | March    |         |
| -216875.98 | -542.19  | 375.000  | -203321.25 | 25       | 2        | sales          | April    |         |
| 0.00       | -461.56  | 0.000    | 0.00       | 25       | 2        | sales          | May      |         |
| -90685.00  | -362.74  | 250.000  | -90685.00  | 19       | 1        | sales          | March    |         |
| -75000.00  | -300.00  | 250.000  | -75000.00  | 19       | 1        | sales          | April    |         |
| 0.00       | -245.00  | 0.000    | 0.00       | 19       | 1        | sales          | May      |         |
| 0.00       | -272.70  | 0.000    | 0.00       | 19       | 2        | sales          | March    |         |
| -78000.00  | -312.00  | 250.000  | -78000.00  | 19       | 2        | sales          | April    |         |
| -44850.00  | -299.00  | 150.000  | -44850.00  | 19       | 2        | sales          | May      |         |
| -283869.95 | -623.89  | 455.000  | -283869.95 | 25       | 1        | sales          | March    |         |
| _tail_     | _head_   | _capac_  | _lo_       | _SUPPLY_ | _DEMAND_ | _name_         | _cost_   | _FLOW_  |
| f1_apr_2   | shop2_2  | 99999999 | 0          | .        | 1450     |                | -549.68  | 235.000 |
| f1_may_2   | shop2_2  | 99999999 | 0          | .        | 1450     |                | -460.00  | 0.000   |
| f2_mar_2   | shop2_2  | 500      | 0          | .        | 1450     |                | -542.83  | 110.000 |
| f2_apr_2   | shop2_2  | 500      | 0          | .        | 1450     |                | -559.19  | 280.000 |
| f2_may_2   | shop2_2  | 500      | 0          | .        | 1450     |                | -519.06  | 370.000 |
| _FCOST_    | oldcost  | oldflow  | oldfc      | diagonal | factory  | key_id         | mth_made |         |
| -129174.80 | -549.68  | 535.000  | -294078.80 | 25       | 1        | sales          | April    |         |
| 0.00       | -460.00  | 0.000    | 0.00       | 25       | 1        | sales          | May      |         |
| -59711.31  | -542.83  | 120.000  | -65139.60  | 25       | 2        | sales          | March    |         |
| -156573.22 | -559.19  | 320.000  | -178940.80 | 25       | 2        | sales          | April    |         |
| -192052.18 | -489.06  | 20.000   | -9781.20   | 25       | 2        | sales          | May      |         |

## Example 4.3. Adding Side Constraints

The manufacturer of Gizmo chips, which are parts needed to make televisions, can supply only 2,600 chips to factory 1 and 3,750 chips to factory 2 in time for production in each of the months March and April. However, Gizmo chips will not be in short supply in May. Three chips are required to make each 19-inch TV while the 25-inch TVs require four chips each. To limit the production of televisions produced at factory 1 in March so that the TVs have the correct number of chips, a side constraint called FACT1 MAR GIZMO is used. The form of this constraint is

```
3 * prod f1 19 mar + 4 * prod f1 25 mar <= 2600
```

prod f1 19 mar is the name of the arc directed from the node fact1\_1 toward node f1\_mar\_1 and, in the previous constraint, designates the flow assigned to this arc. The ARCDATA= and CONOUT= data sets have arc names in a variable called \_name\_.

The other side constraints (shown below) are called FACT2 MAR GIZMO, FACT1 APL GIZMO, and FACT2 APL GIZMO.

```
3 * prod f2 19 mar + 4 * prod f2 25 mar <= 3750
3 * prod f1 19 apl + 4 * prod f1 25 apl <= 2600
3 * prod f2 19 apl + 4 * prod f2 25 apl <= 3750
```

To maintain customer goodwill, the total number of backorders is not to exceed 50 sets. The side constraint TOTAL BACKORDER that models this restriction is

```
back f1 19 apl + back f1 25 apl +
back f2 19 apl + back f2 25 apl +
back f1 19 may + back f1 25 may +
back f2 19 may + back f2 25 may <= 50
```

The sparse CONDATA= data set format is used. All side constraints are of less than or equal type. Because this is the default type value for the DEFCONTYPE= option, type information is not necessary in the following CONDATA=con3. Also, DEFCONTYPE='<=' does not have to be specified in the PROC INTPOINT statement that follows. Notice that the \_column\_ variable value CHIP/BO LIMIT indicates that an observation of the con3 data set contains rhs information. Therefore, specify RHSOBS='CHIP/BO LIMIT':

```
title2 'Adding Side Constraints';
data con3;
  input _column_ &$14. _row_ &$15. _coef_ ;
  datalines;
prod f1 19 mar FACT1 MAR GIZMO 3
prod f1 25 mar FACT1 MAR GIZMO 4
CHIP/BO LIMIT FACT1 MAR GIZMO 2600
prod f2 19 mar FACT2 MAR GIZMO 3
prod f2 25 mar FACT2 MAR GIZMO 4
CHIP/BO LIMIT FACT2 MAR GIZMO 3750
prod f1 19 apl FACT1 APL GIZMO 3
```

```

prod f1 25 apl FACT1 APL GIZMO 4
CHIP/BO LIMIT FACT1 APL GIZMO 2600
prod f2 19 apl FACT2 APL GIZMO 3
prod f2 25 apl FACT2 APL GIZMO 4
CHIP/BO LIMIT FACT2 APL GIZMO 3750
back f1 19 apl TOTAL BACKORDER 1
back f1 25 apl TOTAL BACKORDER 1
back f2 19 apl TOTAL BACKORDER 1
back f2 25 apl TOTAL BACKORDER 1
back f1 19 may TOTAL BACKORDER 1
back f1 25 may TOTAL BACKORDER 1
back f2 19 may TOTAL BACKORDER 1
back f2 25 may TOTAL BACKORDER 1
CHIP/BO LIMIT TOTAL BACKORDER 50
;

```

The four pairs of data sets that follow can be used as ARCDATA= and NODEDATA= data sets in the following PROC INTPOINT run. The set used depends on which cost information the arcs are to have.

```

ARCDATA=arc0      NODEDATA=node0
ARCDATA=arc1      NODEDATA=node0
ARCDATA=arc2      NODEDATA=node0
ARCDATA=arc3      NODEDATA=node0

```

arc0, node0, and arc1 were created in Example 4.1. The first two data sets are the original input data sets.

In the previous example, arc2 was created by modifying arc1 to reflect different arc costs. arc2 and node0 can also be used as the ARCDATA= and NODEDATA= data sets in a PROC INTPOINT run.

If you are going to continue optimization using the changed arc costs, it is probably best to use arc3 and node0 as the ARCDATA= and NODEDATA= data sets.

PROC INTPOINT is used to find the changed cost network solution that obeys the chip limit and backorder side constraints. An explicit ID list has also been specified so that the variables oldcost, oldfc, and oldflow do not appear in the subsequent output data sets:

```

proc intpoint
  bytes=1000000
  print level2=2
  nodedata=node0 arcdata=arc3
  condata=con3 sparsesecondata rhsobs='CHIP/BO LIMIT'
  conout=arc4;
  id diagonal factory key_id mth_made;
run;

```

```

proc print data=arc4;
  var _tail_ _head_ _cost_ _capac_ _lo_ _flow_ _fcost_;
  /* to get this variable order */
  sum _fcost_;
run;

```

The following messages appear on the SAS log:

NOTE: The following variables in ARCDATA do not belong to any SAS variable list. These will be ignored.

```

_FLOW_
_Fcost_
_Rcost_
_ANUMB_
_TNUMB_
_STATUS_
oldcost
oldfc
oldflow

```

NOTE: Number of nodes= 20 .

NOTE: Number of supply nodes= 4 .

NOTE: Number of demand nodes= 4 .

NOTE: Total supply= 4350 , total demand= 4150 .

NOTE: Number of arcs= 64 .

NOTE: Number of <= side constraints= 5 .

NOTE: Number of == side constraints= 0 .

NOTE: Number of >= side constraints= 0 .

NOTE: Number of side constraint coefficients= 16 .

NOTE: The following messages relate to the equivalent Linear Program solved by the Interior Point algorithm.

NOTE: Number of <= constraints= 5 .

NOTE: Number of == constraints= 21 .

NOTE: Number of >= constraints= 0 .

NOTE: Number of constraint coefficients= 152 .

NOTE: Number of variables= 68 .

NOTE: 5 columns, 0 rows and 5 coefficients were added to the problem to handle unrestricted variables, variables that are split, and constraint slack or surplus variables.

NOTE: There are 82 nonzero elements in A \* A transpose.

NOTE: Of the 26 rows and columns, 14 are sparse.

NOTE: There are 80 nonzero superdiagonal elements in the sparse rows of the factored A \* A transpose. This includes fill-in.

NOTE: There are 68 operations of the form  $u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$  to factorize the sparse rows of A \* A transpose.

| Iter | Complem_aff | Complem-ity | Duality_gap | Tot_infeasb | Tot_infeasc  | Tot_infeasd |
|------|-------------|-------------|-------------|-------------|--------------|-------------|
| 0    | -1.000000   | 174740627   | 0.834344    | 52835       | 40217        | 46058       |
| 1    | 48947018    | 21567675    | 0.910848    | 2958.654286 | 2252.108767  | 2470.152448 |
| 2    | 4201633     | 1377647     | 0.520587    | 0           | 1.331557E-11 | 46.321197   |
| 3    | 377792      | 253074      | 0.163432    | 0           | 0            | 8.455959    |
| 4    | 124457      | 71148       | 0.051146    | 0           | 0            | 1.425193    |
| 5    | 46811       | 28865       | 0.021063    | 0           | 0            | 0.532792    |
| 6    | 11689       | 6377.059370 | 0.004710    | 0           | 0            | 0.092989    |
| 7    | 3201.780084 | 1923.406854 | 0.001424    | 0           | 0            | 0.019251    |
| 8    | 468.068267  | 245.400716  | 0.000182    | 0           | 0            | 0.002654    |

```
9    29.762950    5.252240  0.000003889          0          0          0
10   0.005544    0.000290 -4.924982E-9          0          0          0
```

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm  
performed 10 iterations.

NOTE: Objective = -1282708.625.

NOTE: The data set WORK.ARC4 has 64 observations and 14  
variables.

NOTE: There were 64 observations read from the data set  
WORK.ARC3.

NOTE: There were 8 observations read from the data set  
WORK.NODE0.

NOTE: There were 21 observations read from the data set  
WORK.CON3.

NOTE: The data set WORK.ARC4 has 64 observations and 14  
variables.

**Output 4.3.1.** CONOUT=ARC4

| Adding Side Constraints |          |          |         |          |      |         |            |
|-------------------------|----------|----------|---------|----------|------|---------|------------|
| Obs                     | _tail_   | _head_   | _cost_  | _capac_  | _lo_ | _FLOW_  | _FCOST_    |
| 1                       | fact1_1  | f1_apr_1 | 78.60   | 600      | 50   | 533.333 | 41920.00   |
| 2                       | f1_mar_1 | f1_apr_1 | 15.00   | 50       | 0    | 0.000   | 0.00       |
| 3                       | f1_may_1 | f1_apr_1 | 33.60   | 20       | 0    | 0.000   | 0.00       |
| 4                       | f2_apr_1 | f1_apr_1 | 11.00   | 40       | 0    | 0.000   | 0.00       |
| 5                       | fact1_2  | f1_apr_2 | 174.50  | 550      | 50   | 250.000 | 43625.00   |
| 6                       | f1_mar_2 | f1_apr_2 | 20.00   | 40       | 0    | 0.000   | 0.00       |
| 7                       | f1_may_2 | f1_apr_2 | 49.20   | 15       | 0    | 0.000   | 0.00       |
| 8                       | f2_apr_2 | f1_apr_2 | 21.00   | 25       | 0    | 0.000   | 0.00       |
| 9                       | fact1_1  | f1_mar_1 | 127.90  | 500      | 50   | 333.333 | 42633.33   |
| 10                      | f1_apr_1 | f1_mar_1 | 33.60   | 20       | 0    | 20.000  | 672.00     |
| 11                      | f2_mar_1 | f1_mar_1 | 10.00   | 40       | 0    | 40.000  | 400.00     |
| 12                      | fact1_2  | f1_mar_2 | 217.90  | 400      | 40   | 400.000 | 87160.00   |
| 13                      | f1_apr_2 | f1_mar_2 | 38.40   | 30       | 0    | 30.000  | 1152.00    |
| 14                      | f2_mar_2 | f1_mar_2 | 20.00   | 25       | 0    | 25.000  | 500.00     |
| 15                      | fact1_1  | f1_may_1 | 90.10   | 400      | 50   | 128.333 | 11562.83   |
| 16                      | f1_apr_1 | f1_may_1 | 12.00   | 50       | 0    | 0.000   | 0.00       |
| 17                      | f2_may_1 | f1_may_1 | 13.00   | 40       | 0    | 0.000   | 0.00       |
| 18                      | fact1_2  | f1_may_2 | 113.30  | 350      | 40   | 350.000 | 39655.00   |
| 19                      | f1_apr_2 | f1_may_2 | 18.00   | 40       | 0    | 0.000   | 0.00       |
| 20                      | f2_may_2 | f1_may_2 | 13.00   | 25       | 0    | 0.000   | 0.00       |
| 21                      | f1_apr_1 | f2_apr_1 | 11.00   | 99999999 | 0    | 13.333  | 146.67     |
| 22                      | fact2_1  | f2_apr_1 | 62.40   | 480      | 35   | 480.000 | 29952.00   |
| 23                      | f2_mar_1 | f2_apr_1 | 18.00   | 30       | 0    | 0.000   | 0.00       |
| 24                      | f2_may_1 | f2_apr_1 | 30.00   | 15       | 0    | 0.000   | 0.00       |
| 25                      | f1_apr_2 | f2_apr_2 | 23.00   | 99999999 | 0    | 0.000   | 0.00       |
| 26                      | fact2_2  | f2_apr_2 | 196.70  | 680      | 35   | 577.500 | 113594.25  |
| 27                      | f2_mar_2 | f2_apr_2 | 28.00   | 50       | 0    | 0.000   | 0.00       |
| 28                      | f2_may_2 | f2_apr_2 | 64.80   | 15       | 0    | 0.000   | 0.00       |
| 29                      | f1_mar_1 | f2_mar_1 | 11.00   | 99999999 | 0    | 0.000   | 0.00       |
| 30                      | fact2_1  | f2_mar_1 | 88.00   | 450      | 35   | 290.000 | 25520.00   |
| 31                      | f2_apr_1 | f2_mar_1 | 20.40   | 15       | 0    | 0.000   | 0.00       |
| 32                      | f1_mar_2 | f2_mar_2 | 23.00   | 99999999 | 0    | 0.000   | 0.00       |
| 33                      | fact2_2  | f2_mar_2 | 182.00  | 650      | 35   | 650.000 | 118300.00  |
| 34                      | f2_apr_2 | f2_mar_2 | 37.20   | 15       | 0    | 0.000   | 0.00       |
| 35                      | f1_may_1 | f2_may_1 | 16.00   | 99999999 | 0    | 115.000 | 1840.00    |
| 36                      | fact2_1  | f2_may_1 | 128.80  | 250      | 35   | 35.000  | 4508.00    |
| 37                      | f2_apr_1 | f2_may_1 | 20.00   | 30       | 0    | 0.000   | 0.00       |
| 38                      | f1_may_2 | f2_may_2 | 26.00   | 99999999 | 0    | 350.000 | 9100.00    |
| 39                      | fact2_2  | f2_may_2 | 181.40  | 550      | 35   | 122.500 | 22221.50   |
| 40                      | f2_apr_2 | f2_may_2 | 38.00   | 50       | 0    | 0.000   | 0.00       |
| 41                      | f1_mar_1 | shop1_1  | -327.65 | 250      | 0    | 143.333 | -46963.17  |
| 42                      | f1_apr_1 | shop1_1  | -300.00 | 250      | 0    | 250.000 | -75000.00  |
| 43                      | f1_may_1 | shop1_1  | -285.00 | 250      | 0    | 13.333  | -3800.00   |
| 44                      | f2_mar_1 | shop1_1  | -297.40 | 250      | 0    | 250.000 | -74350.00  |
| 45                      | f2_apr_1 | shop1_1  | -290.00 | 250      | 0    | 243.333 | -70566.67  |
| 46                      | f2_may_1 | shop1_1  | -292.00 | 250      | 0    | 0.000   | 0.00       |
| 47                      | f1_mar_2 | shop1_2  | -559.76 | 99999999 | 0    | 0.000   | 0.00       |
| 48                      | f1_apr_2 | shop1_2  | -524.28 | 99999999 | 0    | 0.000   | 0.00       |
| 49                      | f1_may_2 | shop1_2  | -475.02 | 99999999 | 0    | 0.000   | 0.00       |
| 50                      | f2_mar_2 | shop1_2  | -567.83 | 500      | 0    | 500.000 | -283915.00 |
| 51                      | f2_apr_2 | shop1_2  | -542.19 | 500      | 0    | 400.000 | -216876.00 |
| 52                      | f2_may_2 | shop1_2  | -491.56 | 500      | 0    | 0.000   | 0.00       |
| 53                      | f1_mar_1 | shop2_1  | -362.74 | 250      | 0    | 250.000 | -90685.00  |
| 54                      | f1_apr_1 | shop2_1  | -300.00 | 250      | 0    | 250.000 | -75000.00  |
| 55                      | f1_may_1 | shop2_1  | -245.00 | 250      | 0    | 0.000   | 0.00       |
| Adding Side Constraints |          |          |         |          |      |         |            |
| Obs                     | _tail_   | _head_   | _cost_  | _capac_  | _lo_ | _FLOW_  | _FCOST_    |
| 56                      | f2_mar_1 | shop2_1  | -272.70 | 250      | 0    | 0.0     | 0.00       |
| 57                      | f2_apr_1 | shop2_1  | -312.00 | 250      | 0    | 250.0   | -78000.00  |
| 58                      | f2_may_1 | shop2_1  | -299.00 | 250      | 0    | 150.0   | -44850.00  |
| 59                      | f1_mar_2 | shop2_2  | -623.89 | 99999999 | 0    | 455.0   | -283869.95 |
| 60                      | f1_apr_2 | shop2_2  | -549.68 | 99999999 | 0    | 220.0   | -120929.60 |
| 61                      | f1_may_2 | shop2_2  | -460.00 | 99999999 | 0    | 0.0     | 0.00       |
| 62                      | f2_mar_2 | shop2_2  | -542.83 | 500      | 0    | 125.0   | -67853.75  |
| 63                      | f2_apr_2 | shop2_2  | -559.19 | 500      | 0    | 177.5   | -99256.23  |
| 64                      | f2_may_2 | shop2_2  | -519.06 | 500      | 0    | 472.5   | -245255.85 |
| =====                   |          |          |         |          |      |         |            |
| -1282708.63             |          |          |         |          |      |         |            |

## Example 4.4. Using Constraints and More Alteration to Arc Data

Suppose the 25-inch screen TVs produced at factory 1 in May can be sold at either shop with an increased profit of 40 dollars each. What is the new optimal solution?

```

title2 'Using Constraints and Altering arc data';
data new_arc4;
  set arc4;
  oldcost=_cost_;
  oldflow=_flow_;
  oldfc=_fcost_;
  if _tail_='f1_may_2'
    & (_head_='shop1_2' | _head_='shop2_2')
    then _cost_=_cost_-40;

proc intpoint
  bytes=1000000
  printlevel2=2
  arcdata=new_arc4 nodedata=node0
  condata=con3 sparseseconddata rhsobs='CHIP/BO LIMIT'
  conout=arc5;
run;
proc print data=arc5 (drop = _status_ _rcost_);
  var _tail_ _head_ _cost_ _capac_ _lo_
      _supply_ _demand_ _name_
      _flow_ _fcost_ oldflow oldfc;
      /* to get this variable order */
  sum oldfc _fcost_;
run;

```

The following messages appear on the SAS log:

```

NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 4 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 4350 , total demand= 4150 .
NOTE: Number of arcs= 64 .
NOTE: Number of <= side constraints= 5 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of >= side constraints= 0 .
NOTE: Number of side constraint coefficients= 16 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 5 .
NOTE: Number of == constraints= 21 .
NOTE: Number of >= constraints= 0 .
NOTE: Number of constraint coefficients= 152 .
NOTE: Number of variables= 68 .
NOTE: 5 columns, 0 rows and 5 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 82 nonzero elements in A * A transpose.
NOTE: Of the 26 rows and columns, 14 are sparse.
NOTE: There are 80 nonzero superdiagonal elements in the

```



sparse rows of the factored  $A * A$  transpose. This includes fill-in.

NOTE: There are 68 operations of the form  $u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q]$  to factorize the sparse rows of  $A * A$  transpose.

| Iter | Complem_aff | Complem-ity | Duality_gap  | Tot_infeasb | Tot_infeasd  | Tot_infeasd |
|------|-------------|-------------|--------------|-------------|--------------|-------------|
| 0    | -1.000000   | 176072547   | 0.833846     | 52835       | 40217        | 46430       |
| 1    | 49323199    | 21763015    | 0.910646     | 2971.887954 | 2262.182150  | 2506.228441 |
| 2    | 4256602     | 1419262     | 0.526045     | 0           | 6.291145E-11 | 51.536868   |
| 3    | 371793      | 247491      | 0.159553     | 0           | 0            | 8.939364    |
| 4    | 119362      | 60394       | 0.043112     | 0           | 0            | 1.565229    |
| 5    | 27925       | 21537       | 0.015622     | 0           | 0            | 0.576834    |
| 6    | 10494       | 6617.424035 | 0.004839     | 0           | 0            | 0.124639    |
| 7    | 3367.128699 | 1357.529806 | 0.000996     | 0           | 0            | 0.001406    |
| 8    | 498.362201  | 156.077166  | 0.000115     | 0           | 0            | 0.000163    |
| 9    | 28.159576   | 1.000647    | 0.000000735  | 0           | 0            | 0           |
| 10   | 0.000549    | 0.000050153 | -8.10256E-11 | 0           | 0            | 0           |

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 10 iterations.

NOTE: Objective = -1295661.8.

NOTE: The data set WORK.ARC5 has 64 observations and 17 variables.

NOTE: There were 64 observations read from the data set WORK.NEW\_ARC4.

NOTE: There were 8 observations read from the data set WORK.NODE0.

NOTE: There were 21 observations read from the data set WORK.CON3.

NOTE: The data set WORK.ARC5 has 64 observations and 17 variables.

Output 4.4.1. CONOUT=ARC5

| _tail_         | _head_   | _cost_    | _capac_  | _lo_      | _SUPPLY_ | _DEMAND_ |
|----------------|----------|-----------|----------|-----------|----------|----------|
| fact1_1        | f1_apr_1 | 78.6      | 600      | 50        | 1000     | .        |
| f1_mar_1       | f1_apr_1 | 15.0      | 50       | 0         | .        | .        |
| f1_may_1       | f1_apr_1 | 33.6      | 20       | 0         | .        | .        |
| f2_apr_1       | f1_apr_1 | 11.0      | 40       | 0         | .        | .        |
| fact1_2        | f1_apr_2 | 174.5     | 550      | 50        | 1000     | .        |
| f1_mar_2       | f1_apr_2 | 20.0      | 40       | 0         | .        | .        |
| f1_may_2       | f1_apr_2 | 49.2      | 15       | 0         | .        | .        |
| f2_apr_2       | f1_apr_2 | 21.0      | 25       | 0         | .        | .        |
| fact1_1        | f1_mar_1 | 127.9     | 500      | 50        | 1000     | .        |
| f1_apr_1       | f1_mar_1 | 33.6      | 20       | 0         | .        | .        |
| f2_mar_1       | f1_mar_1 | 10.0      | 40       | 0         | .        | .        |
| fact1_2        | f1_mar_2 | 217.9     | 400      | 40        | 1000     | .        |
| f1_apr_2       | f1_mar_2 | 38.4      | 30       | 0         | .        | .        |
| f2_mar_2       | f1_mar_2 | 20.0      | 25       | 0         | .        | .        |
| fact1_1        | f1_may_1 | 90.1      | 400      | 50        | 1000     | .        |
| f1_apr_1       | f1_may_1 | 12.0      | 50       | 0         | .        | .        |
| f2_may_1       | f1_may_1 | 13.0      | 40       | 0         | .        | .        |
| fact1_2        | f1_may_2 | 113.3     | 350      | 40        | 1000     | .        |
| f1_apr_2       | f1_may_2 | 18.0      | 40       | 0         | .        | .        |
| f2_may_2       | f1_may_2 | 13.0      | 25       | 0         | .        | .        |
| f1_apr_1       | f2_apr_1 | 11.0      | 99999999 | 0         | .        | .        |
| fact2_1        | f2_apr_1 | 62.4      | 480      | 35        | 850      | .        |
| f2_mar_1       | f2_apr_1 | 18.0      | 30       | 0         | .        | .        |
| f2_may_1       | f2_apr_1 | 30.0      | 15       | 0         | .        | .        |
| f1_apr_2       | f2_apr_2 | 23.0      | 99999999 | 0         | .        | .        |
| fact2_2        | f2_apr_2 | 196.7     | 680      | 35        | 1500     | .        |
| f2_mar_2       | f2_apr_2 | 28.0      | 50       | 0         | .        | .        |
| _name_         | _FLOW_   | _FCOST_   | oldflow  | oldfc     |          |          |
| prod f1 19 apl | 533.333  | 41920.00  | 533.333  | 41920.00  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| back f1 19 may | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| prod f1 25 apl | 250.000  | 43625.00  | 250.000  | 43625.00  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| back f1 25 may | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| prod f1 19 mar | 333.333  | 42633.33  | 333.333  | 42633.33  |          |          |
| back f1 19 apl | 20.000   | 672.00    | 20.000   | 672.00    |          |          |
|                | 40.000   | 400.00    | 40.000   | 400.00    |          |          |
| prod f1 25 mar | 400.000  | 87160.00  | 400.000  | 87160.00  |          |          |
| back f1 25 apl | 30.000   | 1152.00   | 30.000   | 1152.00   |          |          |
|                | 25.000   | 500.00    | 25.000   | 500.00    |          |          |
|                | 128.333  | 11562.83  | 128.333  | 11562.83  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 350.000  | 39655.00  | 350.000  | 39655.00  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 13.333   | 146.67    | 13.333   | 146.67    |          |          |
| prod f2 19 apl | 480.000  | 29952.00  | 480.000  | 29952.00  |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| back f2 19 may | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| prod f2 25 apl | 550.000  | 108185.00 | 577.500  | 113594.25 |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| _tail_         | _head_   | _cost_    | _capac_  | _lo_      | _SUPPLY_ | _DEMAND_ |
| f2_may_2       | f2_apr_2 | 64.8      | 15       | 0         | .        | .        |
| f1_mar_1       | f2_mar_1 | 11.0      | 99999999 | 0         | .        | .        |
| fact2_1        | f2_mar_1 | 88.0      | 450      | 35        | 850      | .        |
| f2_apr_1       | f2_mar_1 | 20.4      | 15       | 0         | .        | .        |
| f1_mar_2       | f2_mar_2 | 23.0      | 99999999 | 0         | .        | .        |
| fact2_2        | f2_mar_2 | 182.0     | 650      | 35        | 1500     | .        |
| f2_apr_2       | f2_mar_2 | 37.2      | 15       | 0         | .        | .        |
| _name_         | _FLOW_   | _FCOST_   | oldflow  | oldfc     |          |          |
| back f2 25 may | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| prod f2 19 mar | 290.000  | 25520.00  | 290.000  | 25520.00  |          |          |
| back f2 19 apl | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
|                | 0.000    | 0.00      | 0.000    | 0.00      |          |          |
| prod f2 25 mar | 650.000  | 118300.00 | 650.000  | 118300.00 |          |          |
| back f2 25 apl | 0.000    | 0.00      | 0.000    | 0.00      |          |          |

| _tail_   | _head_   | _cost_     | _capac_  | _lo_       | _SUPPLY_ | _DEMAND_ |
|----------|----------|------------|----------|------------|----------|----------|
| f1_may_1 | f2_may_1 | 16.00      | 99999999 | 0          | .        | .        |
| fact2_1  | f2_may_1 | 128.80     | 250      | 35         | 850      | .        |
| f2_apr_1 | f2_may_1 | 20.00      | 30       | 0          | .        | .        |
| f1_may_2 | f2_may_2 | 26.00      | 99999999 | 0          | .        | .        |
| fact2_2  | f2_may_2 | 181.40     | 550      | 35         | 1500     | .        |
| f2_apr_2 | f2_may_2 | 38.00      | 50       | 0          | .        | .        |
| f1_mar_1 | shop1_1  | -327.65    | 250      | 0          | .        | 900      |
| f1_apr_1 | shop1_1  | -300.00    | 250      | 0          | .        | 900      |
| f1_may_1 | shop1_1  | -285.00    | 250      | 0          | .        | 900      |
| f2_mar_1 | shop1_1  | -297.40    | 250      | 0          | .        | 900      |
| f2_apr_1 | shop1_1  | -290.00    | 250      | 0          | .        | 900      |
| f2_may_1 | shop1_1  | -292.00    | 250      | 0          | .        | 900      |
| f1_mar_2 | shop1_2  | -559.76    | 99999999 | 0          | .        | 900      |
| f1_apr_2 | shop1_2  | -524.28    | 99999999 | 0          | .        | 900      |
| f1_may_2 | shop1_2  | -515.02    | 99999999 | 0          | .        | 900      |
| f2_mar_2 | shop1_2  | -567.83    | 500      | 0          | .        | 900      |
| f2_apr_2 | shop1_2  | -542.19    | 500      | 0          | .        | 900      |
| f2_may_2 | shop1_2  | -491.56    | 500      | 0          | .        | 900      |
| f1_mar_1 | shop2_1  | -362.74    | 250      | 0          | .        | 900      |
| f1_apr_1 | shop2_1  | -300.00    | 250      | 0          | .        | 900      |
| f1_may_1 | shop2_1  | -245.00    | 250      | 0          | .        | 900      |
| f2_mar_1 | shop2_1  | -272.70    | 250      | 0          | .        | 900      |
| f2_apr_1 | shop2_1  | -312.00    | 250      | 0          | .        | 900      |
| f2_may_1 | shop2_1  | -299.00    | 250      | 0          | .        | 900      |
| f1_mar_2 | shop2_2  | -623.89    | 99999999 | 0          | .        | 1450     |
| f1_apr_2 | shop2_2  | -549.68    | 99999999 | 0          | .        | 1450     |
| f1_may_2 | shop2_2  | -500.00    | 99999999 | 0          | .        | 1450     |
| _name_   | _FLOW_   | _FCOST_    | oldflow  | oldfc      |          |          |
|          | 115.000  | 1840.00    | 115.000  | 1840.00    |          |          |
|          | 35.000   | 4508.00    | 35.000   | 4508.00    |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 0.000    | 0.00       | 350.000  | 9100.00    |          |          |
|          | 150.000  | 27210.00   | 122.500  | 22221.50   |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 143.333  | -46963.17  | 143.333  | -46963.17  |          |          |
|          | 250.000  | -75000.00  | 250.000  | -75000.00  |          |          |
|          | 13.333   | -3800.00   | 13.333   | -3800.00   |          |          |
|          | 250.000  | -74350.00  | 250.000  | -74350.00  |          |          |
|          | 243.333  | -70566.67  | 243.333  | -70566.67  |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 350.000  | -180257.00 | 0.000    | 0.00       |          |          |
|          | 500.000  | -283915.00 | 500.000  | -283915.00 |          |          |
|          | 50.000   | -27109.50  | 400.000  | -216876.00 |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 250.000  | -90685.00  | 250.000  | -90685.00  |          |          |
|          | 250.000  | -75000.00  | 250.000  | -75000.00  |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
|          | 250.000  | -78000.00  | 250.000  | -78000.00  |          |          |
|          | 150.000  | -44850.00  | 150.000  | -44850.00  |          |          |
|          | 455.000  | -283869.95 | 455.000  | -283869.95 |          |          |
|          | 220.000  | -120929.60 | 220.000  | -120929.60 |          |          |
|          | 0.000    | 0.00       | 0.000    | 0.00       |          |          |
| _tail_   | _head_   | _cost_     | _capac_  | _lo_       | _SUPPLY_ | _DEMAND_ |
| f2_mar_2 | shop2_2  | -542.83    | 500      | 0          | .        | 1450     |
| f2_apr_2 | shop2_2  | -559.19    | 500      | 0          | .        | 1450     |
| f2_may_2 | shop2_2  | -519.06    | 500      | 0          | .        | 1450     |
| _name_   | _FLOW_   | _FCOST_    | oldflow  | oldfc      |          |          |
|          | 125.000  | -67853.75  | 125.000  | -67853.75  |          |          |
|          | 500.000  | -279595.00 | 177.500  | -99256.23  |          |          |
|          | 150.000  | -77859.00  | 472.500  | -245255.85 |          |          |

---

### Example 4.5. Nonarc Variables in the Side Constraints

You can verify that the FACT2 MAR GIZMO constraint has a left-hand-side activity of 3,470, which is not equal to the `_RHS_` of this constraint. Not all of the 3,750 chips that can be supplied to factory 2 for March production are used. It is suggested that all the possible chips be obtained in March and those not used be saved for April production. Because chips must be kept in an air-controlled environment, it costs one dollar to store each chip purchased in March until April. The maximum number of chips that can be stored in this environment at each factory is 150. In addition, a search of the parts inventory at factory 1 turned up 15 chips available for their March production.

Nonarc variables are used in the side constraints that handle the limitations of supply of Gizmo chips. A nonarc variable called `f1 unused mar` has as a value the number of chips that are not used at factory 1 in March. Another nonarc variable, `f2 unused mar`, has as a value the number of chips that are not used at factory 2 in March. `f1 chips from mar` has as a value the number of chips left over from March used for production at factory 1 in April. Similarly, `f2 chips from mar` has as a value the number of chips left over from March used for April production at factory 2 in April. The last two nonarc variables have objective function coefficients of 1 and upper bounds of 150. The Gizmo side constraints are

```
3*prod f1 19 mar + 4*prod f1 25 mar + f1 unused chips = 2615
3*prod f2 19 apl + 4*prod f2 25 apl + f2 unused chips = 3750
3*prod f1 19 apl + 4*prod f1 25 apl - f1 chips from mar = 2600
3*prod f2 19 apl + 4*prod f2 25 apl - f2 chips from mar = 3750
f1 unused chips + f2 unused chips -
f1 chips from mar - f2 chips from mar >= 0
```

The last side constraint states that the number of chips not used in March is not less than the number of chips left over from March and used in April. Here, this constraint is called `CHIP LEFTOVER`.

The following SAS code creates a new data set containing constraint data. It seems that most of the constraints are now equalities, so you specify `DEFCONTYPE=EQ` in the `PROC INTPOINT` statements from now on and provide constraint type data for constraints that are not “equal to” type, using the default `TYPEOBS` value `_TYPE_` as the `_COLUMN_` variable value to indicate observations that contain constraint type data. Also, from now on, the default `RHSOBS` value is used:

```

title2 'Nonarc Variables in the Side Constraints';
data con6;
  input _column_ &$17. _row_ &$15. _coef_ ;
  datalines;
prod f1 19 mar      FACT1 MAR GIZMO  3
prod f1 25 mar      FACT1 MAR GIZMO  4
f1 unused chips     FACT1 MAR GIZMO  1
_RHS_              FACT1 MAR GIZMO 2615
prod f2 19 mar      FACT2 MAR GIZMO  3
prod f2 25 mar      FACT2 MAR GIZMO  4
f2 unused chips     FACT2 MAR GIZMO  1
_RHS_              FACT2 MAR GIZMO 3750
prod f1 19 apl      FACT1 APL GIZMO  3
prod f1 25 apl      FACT1 APL GIZMO  4
f1 chips from mar   FACT1 APL GIZMO -1
_RHS_              FACT1 APL GIZMO 2600
prod f2 19 apl      FACT2 APL GIZMO  3
prod f2 25 apl      FACT2 APL GIZMO  4
f2 chips from mar   FACT2 APL GIZMO -1
_RHS_              FACT2 APL GIZMO 3750
f1 unused chips     CHIP LEFTOVER  1
f2 unused chips     CHIP LEFTOVER  1
f1 chips from mar   CHIP LEFTOVER -1
f2 chips from mar   CHIP LEFTOVER -1
_TYPE_             CHIP LEFTOVER  1
back f1 19 apl      TOTAL BACKORDER 1
back f1 25 apl      TOTAL BACKORDER 1
back f2 19 apl      TOTAL BACKORDER 1
back f2 25 apl      TOTAL BACKORDER 1
back f1 19 may      TOTAL BACKORDER 1
back f1 25 may      TOTAL BACKORDER 1
back f2 19 may      TOTAL BACKORDER 1
back f2 25 may      TOTAL BACKORDER 1
_TYPE_             TOTAL BACKORDER -1
_RHS_              TOTAL BACKORDER 50
;

```

The nonarc variables f1 chips from mar and f2 chips from mar have objective function coefficients of 1 and upper bounds of 150. There are various ways in which this information can be furnished to PROC INTPOINT. If there were a TYPE list variable in the CONDATA= data set, observations could be in the form:

| <u>COLUMN</u>     | <u>TYPE</u> | <u>ROW</u> | <u>COEF</u> |
|-------------------|-------------|------------|-------------|
| f1 chips from mar | objfn       | .          | 1           |
| f1 chips from mar | upperbd     | .          | 150         |
| f2 chips from mar | objfn       | .          | 1           |
| f2 chips from mar | upperbd     | .          | 150         |

It is desirable to assign ID list variable values to all the nonarc variables:

```
data arc6;
    set arc5;
    drop oldcost oldfc oldflow _flow_ _fcost_ _status_ _rcost_;
data arc6_b;
    input _name_ &$17. _cost_ _capac_ factory key_id $ ;
    datalines;
f1 unused chips      .      . 1 chips
f2 unused chips      .      . 2 chips
f1 chips from mar    1 150 1 chips
f2 chips from mar    1 150 2 chips
;
proc append force
    base=arc6 data=arc6_b;
proc intpoint
    bytes=1000000
    printlevel2=2
    nodedata=node0 arcdata=arc6
    condata=con6 defcontype=eq sparsesecondata
    conout=arc7;
run;
```

The following messages appear on the SAS log:

```
NOTE: Number of nodes= 20 .
NOTE: Number of supply nodes= 4 .
NOTE: Number of demand nodes= 4 .
NOTE: Total supply= 4350 , total demand= 4150 .
NOTE: Number of arcs= 64 .
NOTE: Number of nonarc variables= 4 .
NOTE: Number of <= side constraints= 1 .
NOTE: Number of == side constraints= 4 .
NOTE: Number of >= side constraints= 1 .
NOTE: Number of side constraint coefficients= 24 .
NOTE: The following messages relate to the equivalent
      Linear Program solved by the Interior Point
      algorithm.
NOTE: Number of <= constraints= 1 .
NOTE: Number of == constraints= 25 .
NOTE: Number of >= constraints= 1 .
NOTE: Number of constraint coefficients= 160 .
NOTE: Number of variables= 72 .
NOTE: 2 columns, 0 rows and 2 coefficients were added to
      the problem to handle unrestricted variables,
      variables that are split, and constraint slack or
      surplus variables.
NOTE: There are 86 nonzero elements in A * A transpose.
NOTE: Of the 27 rows and columns, 16 are sparse.
NOTE: There are 102 nonzero superdiagonal elements in the
      sparse rows of the factored A * A transpose. This
      includes fill-in.
NOTE: There are 160 operations of the form
      u[i,j]=u[i,j]-u[q,j]*u[q,i]/u[q,q] to factorize the
      sparse rows of A * A transpose.
Iter Complen_aff Complen-ity Duality_gap Tot_infeasb Tot_infeasc Tot_infeasd
0 -1.000000 180211988 0.837584 55030 38331 44219
```

|    |             |             |              |             |                |             |
|----|-------------|-------------|--------------|-------------|----------------|-------------|
| 1  | 54358286    | 27998179    | 0.909216     | 4949.974964 | 3447.853782    | 6457.752523 |
| 2  | 9372834     | 2493303     | 0.657738     |             | 0 8.594938E-10 | 196.155157  |
| 3  | 360076      | 311567      | 0.192309     |             | 0              | 24.426434   |
| 4  | 133734      | 91250       | 0.064112     |             | 0              | 5.827344    |
| 5  | 66155       | 36320       | 0.026230     |             | 0              | 0           |
| 6  | 18053       | 8903.419999 | 0.006517     |             | 0              | 0           |
| 7  | 3897.577387 | 1910.615008 | 0.001402     |             | 0              | 0           |
| 8  | 847.313077  | 362.303033  | 0.000266     |             | 0              | 0           |
| 9  | 146.082127  | 45.122412   | 0.000033136  |             | 0              | 0           |
| 10 | 6.217057    | 0.574649    | 0.000000422  |             | 0              | 0           |
| 11 | 0.002810    | 0.000029061 | 2.152507E-11 |             | 0              | 0           |

NOTE: Primal-Dual Predictor-Corrector Interior point algorithm performed 11 iterations.

NOTE: Objective = -1295542.742.

NOTE: The data set WORK.ARC7 has 68 observations and 14 variables.

NOTE: There were 68 observations read from the data set WORK.ARC6.

NOTE: There were 8 observations read from the data set WORK.NODE0.

NOTE: There were 31 observations read from the data set WORK.CON6.

NOTE: The data set WORK.ARC7 has 68 observations and 14 variables.

The optimal solution data set, CONOUT=ARC7 in Output 4.5.1 follow:

```
proc print data=arc7;
  var _tail_ _head_ _name_ _cost_ _capac_ _lo_
    _flow_ _fcost_;
  sum _fcost_;
run;
```

**Output 4.5.1.** CONOUT=ARC7

| Nonarc Variables in the Side Constraints |          |          |                |         |          |      |         |           |  |
|------------------------------------------|----------|----------|----------------|---------|----------|------|---------|-----------|--|
| Obs                                      | _tail_   | _head_   | _name_         | _cost_  | _capac_  | _lo_ | _FLOW_  | _FCOST_   |  |
| 1                                        | fact1_1  | f1_apr_1 | prod f1 19 apl | 78.60   | 600      | 50   | 540.000 | 42444.00  |  |
| 2                                        | f1_mar_1 | f1_apr_1 |                | 15.00   | 50       | 0    | 0.000   | 0.00      |  |
| 3                                        | f1_may_1 | f1_apr_1 | back f1 19 may | 33.60   | 20       | 0    | 0.000   | 0.00      |  |
| 4                                        | f2_apr_1 | f1_apr_1 |                | 11.00   | 40       | 0    | 0.000   | 0.00      |  |
| 5                                        | fact1_2  | f1_apr_2 | prod f1 25 apl | 174.50  | 550      | 50   | 250.000 | 43625.00  |  |
| 6                                        | f1_mar_2 | f1_apr_2 |                | 20.00   | 40       | 0    | 0.000   | 0.00      |  |
| 7                                        | f1_may_2 | f1_apr_2 | back f1 25 may | 49.20   | 15       | 0    | 0.000   | 0.00      |  |
| 8                                        | f2_apr_2 | f1_apr_2 |                | 21.00   | 25       | 0    | 25.000  | 525.00    |  |
| 9                                        | fact1_1  | f1_mar_1 | prod f1 19 mar | 127.90  | 500      | 50   | 338.333 | 43272.83  |  |
| 10                                       | f1_apr_1 | f1_mar_1 | back f1 19 apl | 33.60   | 20       | 0    | 20.000  | 672.00    |  |
| 11                                       | f2_mar_1 | f1_mar_1 |                | 10.00   | 40       | 0    | 40.000  | 400.00    |  |
| 12                                       | fact1_2  | f1_mar_2 | prod f1 25 mar | 217.90  | 400      | 40   | 400.000 | 87160.00  |  |
| 13                                       | f1_apr_2 | f1_mar_2 | back f1 25 apl | 38.40   | 30       | 0    | 30.000  | 1152.00   |  |
| 14                                       | f2_mar_2 | f1_mar_2 |                | 20.00   | 25       | 0    | 25.000  | 500.00    |  |
| 15                                       | fact1_1  | f1_may_1 |                | 90.10   | 400      | 50   | 116.667 | 10511.67  |  |
| 16                                       | f1_apr_1 | f1_may_1 |                | 12.00   | 50       | 0    | 0.000   | 0.00      |  |
| 17                                       | f2_may_1 | f1_may_1 |                | 13.00   | 40       | 0    | 0.000   | 0.00      |  |
| 18                                       | fact1_2  | f1_may_2 |                | 113.30  | 350      | 40   | 350.000 | 39655.00  |  |
| 19                                       | f1_apr_2 | f1_may_2 |                | 18.00   | 40       | 0    | 0.000   | 0.00      |  |
| 20                                       | f2_may_2 | f1_may_2 |                | 13.00   | 25       | 0    | 0.000   | 0.00      |  |
| 21                                       | f1_apr_1 | f2_apr_1 |                | 11.00   | 99999999 | 0    | 20.000  | 220.00    |  |
| 22                                       | fact2_1  | f2_apr_1 | prod f2 19 apl | 62.40   | 480      | 35   | 480.000 | 29952.00  |  |
| 23                                       | f2_mar_1 | f2_apr_1 |                | 18.00   | 30       | 0    | 0.000   | 0.00      |  |
| 24                                       | f2_may_1 | f2_apr_1 | back f2 19 may | 30.00   | 15       | 0    | 0.000   | 0.00      |  |
| 25                                       | f1_apr_2 | f2_apr_2 |                | 23.00   | 99999999 | 0    | 0.000   | 0.00      |  |
| 26                                       | fact2_2  | f2_apr_2 | prod f2 25 apl | 196.70  | 680      | 35   | 577.500 | 113594.25 |  |
| 27                                       | f2_mar_2 | f2_apr_2 |                | 28.00   | 50       | 0    | 0.000   | 0.00      |  |
| 28                                       | f2_may_2 | f2_apr_2 | back f2 25 may | 64.80   | 15       | 0    | 0.000   | 0.00      |  |
| 29                                       | f1_mar_1 | f2_mar_1 |                | 11.00   | 99999999 | 0    | 0.000   | 0.00      |  |
| 30                                       | fact2_1  | f2_mar_1 | prod f2 19 mar | 88.00   | 450      | 35   | 290.000 | 25520.00  |  |
| 31                                       | f2_apr_1 | f2_mar_1 | back f2 19 apl | 20.40   | 15       | 0    | 0.000   | 0.00      |  |
| 32                                       | f1_mar_2 | f2_mar_2 |                | 23.00   | 99999999 | 0    | 0.000   | 0.00      |  |
| 33                                       | fact2_2  | f2_mar_2 | prod f2 25 mar | 182.00  | 650      | 35   | 650.000 | 118300.00 |  |
| 34                                       | f2_apr_2 | f2_mar_2 | back f2 25 apl | 37.20   | 15       | 0    | 0.000   | 0.00      |  |
| 35                                       | f1_may_1 | f2_may_1 |                | 16.00   | 99999999 | 0    | 115.000 | 1840.00   |  |
| 36                                       | fact2_1  | f2_may_1 |                | 128.80  | 250      | 35   | 35.000  | 4508.00   |  |
| 37                                       | f2_apr_1 | f2_may_1 |                | 20.00   | 30       | 0    | 0.000   | 0.00      |  |
| 38                                       | f1_may_2 | f2_may_2 |                | 26.00   | 99999999 | 0    | 0.000   | 0.00      |  |
| 39                                       | fact2_2  | f2_may_2 |                | 181.40  | 550      | 35   | 122.500 | 22221.50  |  |
| 40                                       | f2_apr_2 | f2_may_2 |                | 38.00   | 50       | 0    | 0.000   | 0.00      |  |
| 41                                       | f1_mar_1 | shop1_1  |                | -327.65 | 250      | 0    | 148.333 | -48601.42 |  |
| 42                                       | f1_apr_1 | shop1_1  |                | -300.00 | 250      | 0    | 250.000 | -75000.00 |  |
| 43                                       | f1_may_1 | shop1_1  |                | -285.00 | 250      | 0    | 1.667   | -475.00   |  |
| 44                                       | f2_mar_1 | shop1_1  |                | -297.40 | 250      | 0    | 250.000 | -74350.00 |  |
| 45                                       | f2_apr_1 | shop1_1  |                | -290.00 | 250      | 0    | 250.000 | -72500.00 |  |



The optimal value of the nonarc variable f2 unused chips is 280. This means that although there are 3,750 chips that can be used at factory 2 in March, only 3,470 are used. As the optimal value of f1 unused chips is zero, all chips available for production in March at factory 1 are used. The nonarc variable f2 chips from mar also has zero optimal value. This means that the April production at factory 2 does not need any chips that could have been held in inventory since March. However, the nonarc variable f1 chips from mar has value of 20. Thus, 3,490 chips should be ordered for factory 2 in March. Twenty of these chips should be held in inventory until April, then sent to factory 1.

## Example 4.6. Solving an LP Problem with Data in MPS Format

In this example, PROC INTPOINT is ultimately used to solve an LP. But prior to that, there is SAS code that is used to read a MPS format file and initialize an input SAS data set. MPS was an optimization package developed for IBM computers many years ago and the format by which data had to be supplied to that system became the industry standard for other optimization software packages, including those developed recently. The MPS format is described in Murtagh (1981). If you have an LP with has data in MPS format in a file /your-directorys/your-filename.dat, then the following SAS code should be run:

```
filename w '/your-directorys/your-filename.dat';
data raw;
  infile w lrecl=80 pad;
  input field1 $ 2-3 field2 $ 5-12 field3 $ 15-22
        field4 25-36 field5 $ 40-47 field6 50-61;
run;
%sasmpsxs;
data lp;
  set;
  if _type_="FREE" then _type_="MIN";
  if lag(_type_)="*HS" then _type_="RHS";
run;
proc sort data=lp;by _col_;run;

proc intpoint
  arcdata=lp
  condata=lp sparseseconddata rhsobs=rhs grouped=condata
  conout=solutn /* SAS data set for the optimal solution */
  bytes=20000000
  nnas=1700 ncoefs=4000 ncons=700
  printlevel2=2 memrep;
run;

proc lp
  data=lp sparsedata
  endpause time=3600 maxit1=100000 maxit2=100000;
run;
show status;
quit;
```

You will have to specify the appropriate path and file name in which your MPS format data resides.

`%sasmpsxs;` is a SAS macro provided within SAS/OR software. The MPS format resembles the sparse format of the `CONDATA=` data set for `PROC INTPOINT`. The SAS macro `%sasmpsxs;` examines the MPS data and transfers it into a SAS data set while automatically taking into account how the MPS format differs slightly from `PROC INTPOINT`'s sparse format.

The parameters `NNAS==1700`, `NCOEFS=4000`, and `NCONS=700` indicate the approximate (over-estimated) number of variables, coefficients and constraints this model has. You must change these to your problems dimensions. Knowing these, `PROC INTPOINT` is able to utilize memory better and read the data faster. These parameters are optional.

The `PROC SORT` preceding `PROC INTPOINT` is not necessary, but sorting the SAS data set can speed up `PROC INTPOINT` when it reads the data. After the sort, data for each column is grouped together. `GROUPEd=condata` can be specified.

For small problems, presorting and specifying those additional options is not going to greatly influence `PROC INTPOINT`'s run time. However, when problems are large, presorting and specifying those additional options can be very worthwhile.

If you generate the model yourself, you will be familiar enough with it to know what to specify for the `RHSOBS=` parameter. If the value of the SAS variable in the `COLUMN` list is equal to the character string specified as the `RHSOBS=` option, the data in that observation is interpreted as right-hand-side data as opposed to coefficient data. If you do not know what to specify for the `RHSOBS=rhs-charstr`, you should first run `PROC LP` and optionally set `maxit1=1` and `maxit2=1`. `PROC LP` will output a Problem Summary that includes the line

```
Rhs Variable      rhs-charstr.
```

`BYTES=20000000` is the size of working memory `PROC INTPOINT` is allowed.

The options `PRINTLEVEL2=2` and `MEMREP` indicate that you want to see an iteration log and messages about memory usage. Specifying these options are optional.

---

## References

- Lustig, I.J., Marsten, R.E., and Shanno, D.F. (1992), "On Implementing Mehrotra's Predictor-Corrector Interior-Point Method for Linear Programming," *SIAM Journal of Optimization* 2 (3), 435-449
- Murtagh, B.A. (1981), "Advanced Linear Programming," McGraw-Hill International Book Company.
- Reid, J.K. (1975), "A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases," *Harwell Report CSS 20*, Atomic Energy Research Establishment, Didcot, Oxfordshire, England.
- Roos, C., Terlaky, T., and Vial, J.-Ph. (1997), "Theory and Algorithms for Linear Optimization," Chichester, England: John Wiley & Sons.
- Wright, S.J. (1996), "Primal-Dual Interior Point Algorithms," Philadelphia: SIAM.
- Ye, Y. (1996), "Interior Point Algorithms: Theory and Analysis," Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons.



# Chapter 5

## The LP Procedure

### Chapter Table of Contents

---

|                             |     |
|-----------------------------|-----|
| <b>OVERVIEW</b> . . . . .   | 185 |
| <b>SYNTAX</b> . . . . .     | 185 |
| PROC LP Statement . . . . . | 185 |



# Chapter 5

## The LP Procedure

---

### Overview

A new option has been added to the LP procedure that enables you to choose a generally more efficient LU decomposition algorithm. While the current default algorithm for PROC LP, Reid's algorithm, is very reliable, you can now choose a relatively new and very efficient algorithm by Gilbert and Peierls. It is several times faster than Reid's algorithm on many large linear or integer programming problems.

---

### Syntax

---

#### PROC LP Statement

The following new option is available in the LP procedure statement:

**LU=***i*

specifies an LU decomposition algorithm.

|      |                                                                     |
|------|---------------------------------------------------------------------|
| LU=1 | selects the LU decomposition algorithm by Reid, 1976                |
| LU=2 | selects the LU decomposition algorithm by Gilbert and Peierls, 1988 |

The default value is 1.





# Chapter 6

## The PM Procedure

### Chapter Table of Contents

---

|                                             |     |
|---------------------------------------------|-----|
| <b>OVERVIEW</b> . . . . .                   | 189 |
| <b>DETAILS</b> . . . . .                    | 189 |
| Microsoft Project Data Conversion . . . . . | 189 |



# Chapter 6

## The PM Procedure

---

### Overview

MP2KTOPM is a new SAS macro that converts Microsoft® Project 2000 data saved in MDB format to a form that is readable by the PM procedure. The converted data can also be read by the CPM procedure.

The new options added to the CPM procedure are also available with PROC PM.

---

### Details

---

#### Microsoft Project Data Conversion

There are two SAS macros, MDBTOPM and MP2KTOPM, that are available for converting Microsoft® Project data to a form that is readable by the PM procedure. MDBTOPM converts Microsoft Project 98 data and MP2KTOPM converts Microsoft Project 2000 data. In the event of a successful conversion, each of these macros proceeds to invoke an instance of the PM procedure using values for the relevant options that were determined during the course of the data conversion. Execution of these macros requires SAS/ACCESS software.

MDBTOPM is a SAS macro that converts Microsoft Project 98 data saved in MDB format to a form that is readable by the PM procedure. The MDBTOPM macro has two arguments: one that specifies the location of the .MDB file and another that specifies the location of the directory for storing the SAS data sets.

MP2KTOPM is a SAS macro that converts Microsoft Project 2000 data saved in MDB format to a form that is readable by the PM procedure. The MP2KTOPM macro has three arguments: one that specifies the location of the .MDB file, one that specifies the location of the directory for storing the SAS data sets, and an optional numeric third argument that controls the mode of the PM procedure invocation. A nonzero value invokes the PM procedure in the default interactive mode and a value of zero invokes the PM procedure in noninteractive mode.

The following SAS data sets are generated by the conversion macros:

- Activity data set
- Calendar data set
- Holiday data set
- Workday data set
- Resource data set

- Preferences data set

The MDBTOPM and MP2KTOPM macros convert the hierarchical relationships, precedence relationships, time constraints, resource availabilities, resource requirements, project calendars, resource calendars, holiday information, workshift information, actual start and finish times, and baseline start and finish times. In addition, the notes and cost fields are also extracted and stored in the Activity data set.

**Note:** In order to retain the values of the options used to invoke the PM procedure, you will need to save the preferences by choosing Preferences->Save from the “Project” pull-down menu.

# Subject Index

## A

- affine step
  - INTPOINT procedure, 62, 64
- arc names
  - INTPOINT procedure, 74, 77, 101, 116, 119, 126
- ARC DATA= option
  - INTPOINT procedure, 74–76, 81–82, 89, 101, 106, 115, 117–118, 121, 125

## B

- bill of material (BOM), 3
  - indented, 3
  - modularized, 27
  - multilevel, 3
  - planning, 27
  - single-level, 3
  - summarized, 3
- blending constraints
  - INTPOINT procedure, 69
- BOM
  - See bill of material
- BOM data set
  - Indented, 6, 13
  - Single-level, 3, 12
  - Summarized, 9, 13
- BOM examples
  - bills of material verification, 33
  - planning bill of material, 27
  - with lead time information, 21
- BOM procedure
  - computer resource requirements, 21
  - Indented BOM data set, 13
  - input data sets, 15
  - missing values, 19
  - \_ORBOM\_ macro variable, 20
  - output data sets, 17–18
  - overview, 3
  - Single-level BOM data set, 12
  - Summarized BOM data set, 13
- BYTES= option
  - INTPOINT procedure, 89

## C

- \_CAPAC\_ variable
  - ARC DATA data set (INTPOINT), 116
  - CONOUT data set (INTPOINT), 135
- case sensitivity
  - INTPOINT procedure, 106, 136
- centering step

- INTPOINT procedure, 62, 65
- central path
  - INTPOINT procedure, 60
- CHART variables
  - Schedule data set (GANTT), 51
- Cholesky factorization
  - INTPOINT procedure, 62, 79
- \_COE, SAS variables with names that begin with
  - CON DATA data set (INTPOINT), 116
- COEF statement
  - INTPOINT procedure, 122, 129
- color specification
  - milestone (GANTT), 51
- COLORS= option, GOPTIONS statement
  - GANTT procedure, 51
- COLUMN statement
  - INTPOINT procedure, 102, 105, 107, 128
- \_COLUMN\_ variable
  - CON DATA data set (INTPOINT), 117
- complementarity
  - INTPOINT procedure, 60, 63
- Component variables
  - Single-level BOM data set, 13
- computer resource requirements
  - BOM procedure, 12, 21
- \_CON, SAS variables with names that begin with
  - CON DATA data set (INTPOINT), 120
- \_CON\_ variable
  - CON DATA data set (INTPOINT), 120
- CON DATA= option
  - INTPOINT procedure, 74–75, 77, 81–82, 89, 101, 105–107, 116, 119, 122, 124, 126
- CONOUT= option
  - INTPOINT procedure, 75, 82, 118, 135
- converting Microsoft Project data to SAS, 189
- \_COST\_ variable
  - ARC DATA data set (INTPOINT), 117
- \_COST\_ variable
  - CONOUT data set (INTPOINT), 135
- CPM examples
  - finish milestone, 41
- CPM procedure
  - details, 40
  - finish milestone, 39–41
  - fix actual times, 40
- CPU requirement
  - See computer resource requirements

## D

data set options  
 INTPOINT procedure, 96

data storage requirements  
 See computer resource requirements

DEFCONTYPE= option  
 INTPOINT procedure, 87, 122, 129

DEMAND statement  
 INTPOINT procedure, 139

\_DEMAND\_ variable  
 ARCDATA data set (INTPOINT), 117  
 CONOUT data set (INTPOINT), 136

DEMAND= option  
 INTPOINT procedure, 101

dense input format  
 INTPOINT procedure, 75, 82, 98, 100, 102, 106,  
 119–120, 122, 124, 126, 147, 150

dependent demand, 16

dependent demand process, 16, 20

details  
 INTPOINT procedure, 125

distribution problem  
 INTPOINT procedure, 66

dual problem  
 INTPOINT procedure, 59

## E

embedded networks  
 INTPOINT procedure, 72

end item, 13

examples  
 See also BOM examples  
 BOM procedure, 21  
 INTPOINT procedure, 155

excess node  
 INTPOINT procedure, 144

## F

\_FCOST\_ variable  
 CONOUT data set (INTPOINT), 80, 84, 136

finish milestone  
 CPM procedure, 39–41  
 example (CPM), 41

fix actual times  
 CPM procedure, 40

flow conservation constraints  
 INTPOINT procedure, 56, 68–69, 72, 79, 132, 138,  
 146

\_FLOW\_ variable  
 CONOUT data set (INTPOINT), 80, 84, 136

\_FROM\_ variable  
 ARCDATA data set (INTPOINT), 121  
 CONOUT data set (INTPOINT), 135

functional summary  
 INTPOINT procedure, 93

## G

GANTT procedure  
 description of, 51  
 Imagemap data set, 52

overview, 51

getting started  
 INTPOINT procedure, 74

GOPTIONS statement  
 COLORS= option, 51

graphics version  
 options specific to (GANTT), 51

GROUPED= option  
 INTPOINT procedure, 97

## H

\_HEAD\_ variable  
 ARCDATA data set (INTPOINT), 118

HEADNODE statement  
 INTPOINT procedure, 119

\_HL\_ variable  
 ARCDATA data set (INTPOINT), 116

HTML= option  
 Schedule data set (GANTT), 52

## I

ID variables  
 Single-level BOM data set, 13  
 Summarized BOM data set, 18

indented bill of material, 3

Indented BOM data set, 7, 13  
 variables, 17

independent demand, 15

infeasibility  
 INTPOINT procedure, 60, 64, 138

INFINITY= option  
 INTPOINT procedure, 97–98, 101

input data sets  
 See also Single-level BOM data set  
 BOM procedure, 15  
 INTPOINT procedure, 96, 125

Interior Point algorithm  
 INTPOINT procedure, 55, 58

INTPOINT procedure  
 affine step, 62, 64  
 arc names, 74, 77, 101, 116, 119, 126  
 ARCDATA= option, 74–76, 81–82, 89, 101, 106,  
 115, 117–118, 121, 125  
 blending constraints, 69  
 BYTES= option, 89  
 case sensitivity, 106, 136  
 centering step, 62, 65  
 central path, 60  
 Cholesky factorization, 62, 79  
 COEF statement, 122, 129  
 COLUMN statement, 102, 105, 107, 128  
 complementarity, 60, 63  
 CONDATA= option, 74–75, 77, 81–82, 89, 101,  
 105–107, 116, 119, 122, 124, 126  
 CONOUT= option, 75, 82, 118, 135  
 data set options, 96  
 DEFCONTYPE= option, 87, 122, 129  
 DEMAND statement, 139  
 DEMAND= option, 101

- dense input format, 75, 82, 98, 100, 102, 106, 119–120, 122, 124, 126, 147, 150
  - details, 125
  - distribution problem, 66
  - dual problem, 59
  - embedded networks, 72
  - examples, 155
  - excess node, 144
  - flow conservation constraints, 56, 68–69, 72, 79, 132, 138, 146
  - functional summary, 93
  - getting started, 74
  - GROUPED= option, 97
  - HEADNODE statement, 119
  - infeasibility, 60, 64, 138
  - INFINITY= option, 97–98, 101
  - input data sets, 96, 125
  - Interior Point algorithm, 55, 58
  - INTPOINT statement, 89
  - introductory example, 126, 128
  - introductory LP example, 83
  - introductory NPSC example, 75
  - inventory problem, 66
  - Karush-Kuhn-Tucker (KKT) conditions, 59, 63
  - linear programming problem, 58, 81, 132, 146
  - loop arcs, 137
  - LP, 58
  - LP variable names, 82
  - mathematical description of LP, 58
  - mathematical description of NPSC, 56
  - MAXFLOW option, 99, 101, 105–106
  - maximal flow problem, 97, 105–106
  - MAXIMIZE option, 86
  - minimum flow problem, 101
  - missing S supply and missing D demand, 139
  - multicommodity problems, 70
  - multiple arcs, 137
  - multiprocess, multiproduct example, 71
  - NAME statement, 101, 106, 116–118, 126
  - NAMECTRL= option, 101, 119
  - network problems, 66
  - network programming problem with side constraints, 56
  - NODE statement, 120
  - NODEDATA= option, 74–76, 89, 106, 119–120
  - nonarc variables, 72
  - NPSC, 56
  - options classified by function, 93
  - output data sets, 96, 135
  - preprocessing, 59, 75, 82, 110
  - Primal-Dual with Predictor-Corrector algorithm, 58, 62
  - production-inventory-distribution problem, 66
  - proportional constraints, 67, 77
  - QUIT statement, 90
  - RHS statement, 126
  - RHSOBS= option, 116, 128
  - ROW statement, 116, 124, 126
  - RUN statement, 90
  - shortest path problem, 105–106
  - SHORTPATH option, 99, 101, 106
  - side constraints, 56, 72
  - SINK= option, 97, 99, 101, 105
  - SOURCE= option, 97, 101, 105–106
  - sparse input format, 75, 82, 87, 98, 100, 102, 105–107, 116, 120, 122, 128, 137, 147, 150, 165
  - SPARSECONDATA option, 128
  - special rows, 116, 122, 127, 131
  - stopping criteria, 112, 151
  - summary of PROC INTPOINT Options, 93
  - SUPDEM statement, 139
  - SUPPLY statement, 139
  - supply-chain problem, 66
  - SUPPLY= option, 101
  - symbolic factorization, 62
  - syntax, 91
  - table of syntax elements, 93
  - TAILNODE statement, 119, 121
  - THRUNET option, 144
  - TYPE statement, 126
  - TYPEOBS= option, 116, 128
  - upper bounds, 59, 63
  - VAR statement, 102, 122, 124, 126
  - INTPOINT statement
    - INTPOINT procedure, 89
  - introductory example
    - INTPOINT procedure, 126, 128
  - introductory LP example
    - INTPOINT procedure, 83
  - introductory NPSC example
    - INTPOINT procedure, 75
  - inventory problem
    - INTPOINT procedure, 66
  - item master file
    - See part master data file
- K**
- Karush-Kuhn-Tucker (KKT) conditions
    - INTPOINT procedure, 59, 63
- L**
- lead time, 14
  - LeadTime variable
    - Indented BOM data set, 14, 17
    - Single-level BOM data set, 14
  - \_LENGTH\_ variable
    - ARCDATA data set (INTPOINT), 117
  - \_Level\_ variable
    - Indented BOM data set, 7, 17
  - linear programming problem
    - INTPOINT procedure, 58, 81, 132, 146
  - list of, BOM procedure
    - variables, 15
  - \_LO\_ variable
    - ARCDATA data set (INTPOINT), 118
    - CONOUT data set (INTPOINT), 135
  - loop arcs

INTPOINT procedure, 137  
 Low\_Code variable  
     Summarized BOM data set, 10, 19  
 \_LOWER\_ variable  
     ARCDATA data set (INTPOINT), 118  
 \_LOWERBD variable  
     ARCDATA data set (INTPOINT), 118  
 LP  
     INTPOINT procedure, 58  
 LP variable names  
     INTPOINT procedure, 82

**M**

macro variable  
     \_ORBOM\_, 20  
 master production schedule, 16  
 master schedule item, 10, 15  
 mathematical description of LP  
     INTPOINT procedure, 58  
 mathematical description of NPSC  
     INTPOINT procedure, 56  
 MAXFLOW option  
     INTPOINT procedure, 99, 101, 105–106  
 maximal flow problem  
     INTPOINT procedure, 97, 105–106  
 MAXIMIZE option  
     INTPOINT procedure, 86  
 MDB format  
     Microsoft Project, 189  
 MDBTOPM Macro  
     PM procedure, 189  
 memory requirements  
     See computer resource requirements  
 Microsoft Project  
     converting to PM, 189  
     MDB format, 189  
 milestones  
     color, 51  
 \_MINFLOW variable  
     ARCDATA data set (INTPOINT), 118  
 minimum flow problem  
     INTPOINT procedure, 101  
 missing S supply and missing D demand  
     INTPOINT procedure, 139  
 missing values  
     BOM procedure, 19  
 modularized bill of material construction, 27  
 MP2KTOPM Macro  
     PM procedure, 189  
 MPS  
     See master production schedule  
 MSI  
     See master schedule item  
 multicommodity problems  
     INTPOINT procedure, 70  
 multilevel bill of material, 3  
 multiple arcs  
     INTPOINT procedure, 137  
 multiprocess, multiproduct example

INTPOINT procedure, 71

## N

NAME statement  
     INTPOINT procedure, 101, 106, 116–118, 126  
 \_NAME\_ variable  
     ARCDATA data set (INTPOINT), 119  
     CONOUT data set (INTPOINT), 135  
 NAMECTRL= option  
     INTPOINT procedure, 101, 119  
 Net\_Req variable  
     Summarized BOM data set, 10, 19  
 network problems  
     INTPOINT procedure, 66  
 network programming problem with side constraints  
     INTPOINT procedure, 56  
 NODE statement  
     INTPOINT procedure, 120  
 \_NODE\_ variable  
     NODEDATA data set (INTPOINT), 119  
 NODEDATA= option  
     INTPOINT procedure, 74–76, 89, 106, 119–120  
 nonarc variables  
     INTPOINT procedure, 72  
 NPSC  
     INTPOINT procedure, 56

## O

On\_Hand variable  
     Summarized BOM data set, 14  
 option overplanning, 29  
 options classified by function  
     See functional summary  
 \_ORBOM\_ macro variable, 20  
 output data sets  
     See also Indented BOM data set  
     See also Summarized BOM data set  
     BOM procedure, 17–18  
     INTPOINT procedure, 96, 135  
 overview  
     BOM procedure, 3  
     GANTT procedure, 51

## P

Paren\_ID variable  
     Indented BOM data set, 8, 17  
 \_Parent\_ variable  
     Indented BOM data set, 7, 17  
 part master data file, 16  
 \_Part\_ variable  
     Indented BOM data set, 7  
 Part variable  
     Indented BOM data set, 14, 17  
     Single-level BOM data set, 14  
     Summarized BOM data set, 14, 18  
 Part\_ID variable  
     Indented BOM data set, 7, 17  
 phantom bill of material  
     See planning bill of material



phantom part, 27  
 planning bill of material, 27  
 PM procedure  
   details, 189  
   MDBTOPM Macro, 189  
   MP2KTOPM Macro, 189  
 preprocessing  
   INTPOINT procedure, 59, 75, 82, 110  
 Primal-Dual with Predictor-Corrector algorithm  
   INTPOINT procedure, 58, 62  
 problem size specification  
   BOM procedure, 21  
   number of parts (BOM), 12  
   size of symbolic table (BOM), 12  
   utility data sets(BOM), 12  
 \_Prod\_ variable  
   Indented BOM data set, 8, 17  
 production-inventory-distribution problem  
   INTPOINT procedure, 66  
 proportional constraints  
   INTPOINT procedure, 67, 77  
 pseudo bill of material  
   See planning bill of material

**Q**  
 QtyOnHand variable  
   Single-level BOM data set, 14  
   Summarized BOM data set, 18  
 QtyPer variables  
   See Quantity variables  
 Qty\_Prod variable  
   Indented BOM data set, 8, 17  
 Quantity variables  
   Indented BOM data set, 14, 17  
   Single-level BOM data set, 14  
 QUIT statement  
   INTPOINT procedure, 90

**R**  
 \_RCOST\_ variable  
   CONOUT data set (INTPOINT), 136  
 references  
   INTPOINT procedure, 181  
 Requirement variable  
   Single-level BOM data set, 15  
   Summarized BOM data set, 18  
 RHS statement  
   INTPOINT procedure, 126  
 \_RHS\_ variable  
   CONDATA data set (INTPOINT), 119  
 RHSOBS= option  
   INTPOINT procedure, 116, 128  
 \_ROW, SAS variables with names that begin with  
   CONDATA data set (INTPOINT), 120  
 ROW statement  
   INTPOINT procedure, 116, 124, 126  
 \_ROW\_ variable  
   CONDATA data set (INTPOINT), 120  
 RUN statement

INTPOINT procedure, 90

**S**  
 schedule computation  
   finish milestone, 40  
 \_SD\_ variable  
   NODEDATA data set (INTPOINT), 121  
 shortest path problem  
   INTPOINT procedure, 105–106  
 SHORTPATH option  
   INTPOINT procedure, 99, 101, 106  
 side constraints  
   INTPOINT procedure, 56, 72  
 single-level bill of material, 3  
 Single-level BOM data set, 3, 12  
   missing values, 19  
   variables, 15  
 single-level where-used list, 8  
 SINK= option  
   INTPOINT procedure, 97, 99, 101, 105  
 SOURCE= option  
   INTPOINT procedure, 97, 101, 105–106  
 sparse input format  
   INTPOINT procedure, 75, 82, 87, 98, 100, 102,  
     105–107, 116, 120, 122, 128, 137, 147,  
     150, 165  
 SPARSECONDATA option  
   INTPOINT procedure, 128  
 special rows  
   INTPOINT procedure, 116, 122, 127, 131  
 \_STATUS\_ variable  
   CONOUT data set (INTPOINT), 136  
 stopping criteria  
   INTPOINT procedure, 112, 151  
 summarized bill of material, 3  
 Summarized BOM data set, 9, 13  
   variables, 18  
 summary of PROC INTPOINT Options  
   INTPOINT procedure, 93  
 SUPDEM statement  
   INTPOINT procedure, 139  
 \_SUPDEM\_ variable  
   NODEDATA data set (INTPOINT), 121  
 super bill of material  
   See planning bill of material  
 SUPPLY statement  
   INTPOINT procedure, 139  
 \_SUPPLY\_ variable  
   ARCDATA data set (INTPOINT), 121  
   CONOUT data set (INTPOINT), 136  
 supply-chain problem  
   INTPOINT procedure, 66  
 SUPPLY= option  
   INTPOINT procedure, 101  
 symbolic factorization  
   INTPOINT procedure, 62  
 syntax  
   INTPOINT procedure, 91

**T**

table of syntax elements

See functional summary

TAILNODE statement

INTPOINT procedure, 119, 121

\_TAIL\_ variable

ARCDATA data set (INTPOINT), 90, 121

THRUNET option

INTPOINT procedure, 144

\_TO\_ variable

ARCDATA data set (INTPOINT), 118

CONOUT data set (INTPOINT), 135

Tot\_Lead variable

Indented BOM data set, 14, 17

TYPE statement

INTPOINT procedure, 126

\_TYPE\_ variable

CONDATA data set (INTPOINT), 122

TYPEOBS= option

INTPOINT procedure, 116, 128

**U**

upper bounds

INTPOINT procedure, 59, 63

\_UPPER\_ variable

ARCDATA data set (INTPOINT), 116

\_UPPERBD variable

ARCDATA data set (INTPOINT), 116

**V**

VAR statement

INTPOINT procedure, 102, 122, 124, 126

variables

list of, BOM procedure, 15

treatment of missing values (BOM), 19

**W**

WEB variable

Schedule data set (GANTT), 52

Web-enabled Gantt charts, 51–52

where-used list

single-level, 8

# Syntax Index

## A

- ACTUAL statement
  - CPM procedure, 40
- AND\_KEEPPGOING\_C= option
  - PROC INTPOINT statement, 114
- AND\_KEEPPGOING\_DG= option
  - PROC INTPOINT statement, 115
- AND\_KEEPPGOING\_IB= option
  - PROC INTPOINT statement, 115
- AND\_KEEPPGOING\_IC= option
  - PROC INTPOINT statement, 115
- AND\_KEEPPGOING\_ID= option
  - PROC INTPOINT statement, 115
- AND\_STOP\_C= option
  - PROC INTPOINT statement, 113
- AND\_STOP\_DG= option
  - PROC INTPOINT statement, 113
- AND\_STOP\_IB= option
  - PROC INTPOINT statement, 113
- AND\_STOP\_IC= option
  - PROC INTPOINT statement, 113
- AND\_STOP\_ID= option
  - PROC INTPOINT statement, 114
- ARC\_SINGLE\_OBS option
  - PROC INTPOINT statement, 97
- ARCDATA= option
  - PROC INTPOINT statement, 96
- ARCNAME statement
  - INTPOINT procedure, 119
- ARCS\_ONLY\_ARCDATA option
  - PROC INTPOINT statement, 97

## B

- BOM procedure
  - PROC BOM statement, 12
  - STRUCTURE statement, 13
- BPD= option
  - PROC INTPOINT statement, 97
- BYPASSDIV= option
  - PROC INTPOINT statement, 97
- BYPASSDIVIDE= option
  - PROC INTPOINT statement, 97
- BYTES= option
  - PROC INTPOINT statement, 98

## C

- CAPAC statement
  - INTPOINT procedure, 115
- CAPACITY statement

- INTPOINT procedure, 115
- CHART statement (GANTT), 51
  - graphics options, 51
- CMILE= option
  - CHART statement (GANTT), 51
- COEF statement
  - INTPOINT procedure, 116
- COLUMN statement
  - INTPOINT procedure, 116
- COMP= option
  - See COMPONENT= option
- COMPONENT= option
  - STRUCTURE statement (BOM), 13
- CON\_SINGLE\_OBS option
  - PROC INTPOINT statement, 98
- CONDATA= option
  - PROC INTPOINT statement, 96
- CONOUT= option
  - PROC INTPOINT statement, 96
- CONTYPE statement
  - INTPOINT procedure, 122
- COST statement
  - INTPOINT procedure, 117
- COUT= option
  - PROC INTPOINT statement, 96
- CPM procedure, 39
  - ACTUAL statement, 40
  - PROC CPM statement, 39
  - syntax, 39

## D

- DATA= option
  - PROC BOM statement, 12
- DC= option
  - PROC INTPOINT statement, 98
- DCT= option
  - PROC INTPOINT statement, 99
- DEFCAPACITY= option
  - PROC INTPOINT statement, 98
- DEFCONTYPE= option
  - PROC INTPOINT statement, 99
- DEFCOST= option
  - PROC INTPOINT statement, 99
- DEFMINFLOW= option
  - PROC INTPOINT statement, 99
- DEFTYPE= option
  - PROC INTPOINT statement, 99
- DEMAND statement
  - INTPOINT procedure, 117

DEMAND= option  
     PROC INTPOINT statement, 99  
 DMF= option  
     PROC INTPOINT statement, 99  
 DUR= option (BOM)  
     See LEADTIME= option

**E**

ENDITEM= option  
     STRUCTURE statement (BOM), 13

**F**

FIXASTART option  
     ACTUAL statement (CPM), 40  
 FROM statement  
     INTPOINT procedure, 121  
 FROMNODE statement  
     INTPOINT procedure, 121

**G**

GANTT procedure, 51  
     CHART statement, 51  
     syntax, 51  
 GROSSREQ= option  
     See REQUIREMENT= option  
 GROUPEd= option  
     PROC INTPOINT statement, 99

**H**

HEAD statement  
     INTPOINT procedure, 117  
 HEADNODE statement  
     INTPOINT procedure, 117  
 HTML= option  
     CHART statement (GANTT), 51–52

**I**

ID statement  
     INTPOINT procedure, 118  
 ID= option  
     STRUCTURE statement (BOM), 13  
 IMAGEMAP= option  
     PROC GANTT statement, 52  
 INF= option  
     PROC INTPOINT statement, 101  
 INFINITY= option  
     PROC INTPOINT statement, 101  
 INVENTORY= option  
     See QTYONHAND= option  
 ITEM= option  
     See PART= option

**K**

KEEPGOING\_C= option  
     PROC INTPOINT statement, 114  
 KEEPGOING\_DG= option  
     PROC INTPOINT statement, 114  
 KEEPGOING\_IB= option

    PROC INTPOINT statement, 114  
 KEEPGOING\_IC= option  
     PROC INTPOINT statement, 114  
 KEEPGOING\_ID= option  
     PROC INTPOINT statement, 114

**L**

LEADTIME= option  
     STRUCTURE statement (BOM), 14  
 LO statement  
     INTPOINT procedure, 118  
 LOWERBD statement  
     INTPOINT procedure, 118  
 LP procedure, 185  
     LP statement, 185  
     syntax, 185  
 LP statement  
     LP procedure, 185  
 LU= option  
     PROC LP statement, 185

**M**

MAX option  
     PROC INTPOINT statement, 101  
 MAXFLOW option  
     PROC INTPOINT statement, 101  
 MAXIMIZE option  
     PROC INTPOINT statement, 101  
 MEMREPOption  
     PROC INTPOINT statement, 101  
 MF option  
     PROC INTPOINT statement, 101  
 MINFLOW statement  
     INTPOINT procedure, 118

**N**

NAME statement  
     INTPOINT procedure, 119  
 NAMECTRL=option  
     PROC INTPOINT statement, 101  
 NARCS= option  
     PROC INTPOINT statement, 104  
 NCOEFS= option  
     PROC INTPOINT statement, 104  
 NCONS= option  
     PROC INTPOINT statement, 104  
 NNames= option  
     PROC BOM statement, 12  
 NNAS= option  
     PROC INTPOINT statement, 104  
 NNODES= option  
     PROC INTPOINT statement, 104  
 NODE statement  
     INTPOINT procedure, 119  
 NODATA= option  
     PROC INTPOINT statement, 96  
 NON\_REPLIC= option  
     PROC INTPOINT statement, 104  
 NOUTIL option

PROC BOM statement, 12  
 NPARTS= option  
   PROC BOM statement, 12

## O

OBJFN statement  
   INTPOINT procedure, 117  
 OUT= option  
   PROC BOM statement, 13

## P

PART= option  
   STRUCTURE statement (BOM), 14  
 PRINTLEVEL2= option  
   PROC INTPOINT statement, 111  
 PROC BOM statement  
   DATA= option, 12  
   NNAMES= option, 12  
   NOUTIL option, 12  
   NPARTS= option, 12  
   OUT= option, 13  
   statement options, 12  
   SUMMARYOUT= option, 13  
 PROC CPM statement  
   statement options, 39  
 PROC INTPOINT statement  
   INTPOINT procedure, 93, 96  
 PROC LP statement  
   LP procedure, 185

## Q

QTYONHAND= option  
   STRUCTURE statement (BOM), 14  
 QTYPER= option  
   See QUANTITY= option  
 QUANTITY= option  
   STRUCTURE statement (BOM), 14  
 QUIT statement  
   INTPOINT procedure, 119

## R

REQUIREMENT= option  
   STRUCTURE statement (BOM), 15  
 RHS statement  
   INTPOINT procedure, 119  
 RHSOBS= option  
   PROC INTPOINT statement, 105  
 ROW statement  
   INTPOINT procedure, 120  
 RUN statement  
   INTPOINT procedure, 120

## S

SCALE= option  
   PROC INTPOINT statement, 105  
 SCDATA option  
   PROC INTPOINT statement, 106  
 SETFINISHMILESTONE option

PROC CPM statement, 39  
 SHORTPATH option  
   PROC INTPOINT statement, 105  
 SINK= option  
   PROC INTPOINT statement, 105  
 SINKNODE= option  
   PROC INTPOINT statement, 105  
 SOURCE= option  
   PROC INTPOINT statement, 106  
 SOURCENODE= option  
   PROC INTPOINT statement, 106  
 SP option  
   PROC INTPOINT statement, 105  
 SPARSECONDATA option  
   PROC INTPOINT statement, 106  
 STOP\_C= option  
   PROC INTPOINT statement, 112  
 STOP\_DG= option  
   PROC INTPOINT statement, 112  
 STOP\_IB= option  
   PROC INTPOINT statement, 112  
 STOP\_IC= option  
   PROC INTPOINT statement, 113  
 STOP\_ID= option  
   PROC INTPOINT statement, 113  
 STRUCTURE statement (BOM)  
   COMPONENT= option, 13  
   ENDITEM= option, 13  
   ID= option, 13  
   QTYONHAND= option, 14  
   QUANTITY= option, 14  
   REQUIREMENT= option, 15  
   BOM procedure, 13  
 SUMMARYOUT= option  
   PROC BOM statement, 13  
 SUPDEM statement  
   INTPOINT procedure, 120  
 SUPPLY statement  
   INTPOINT procedure, 121  
 SUPPLY= option  
   PROC INTPOINT statement, 106

## T

TAIL statement  
   INTPOINT procedure, 121  
 TAILNODE statement  
   INTPOINT procedure, 121  
 THRUNET option  
   PROC INTPOINT statement, 106  
 TO statement  
   INTPOINT procedure, 117  
 TONODE statement  
   INTPOINT procedure, 117  
 TYPE statement  
   INTPOINT procedure, 122  
 TYPEOBS= option  
   PROC INTPOINT statement, 107

## **U**

UPPERBD statement  
    INTPOINT procedure, 115

## **V**

VAR statement  
    INTPOINT procedure, 124  
VARIABLES statement (BOM)  
    See STRUCTURE statement  
VARNAME statement  
    INTPOINT procedure, 119  
VERBOSE= option  
    PROC INTPOINT, or INTPOINT statement, 107

## **W**

WEB= option  
    CHART statement (GANTT), 51–52

## **Z**

Z2= option  
    PROC INTPOINT, or INTPOINT statement, 107  
ZERO2= option  
    PROC INTPOINT, or INTPOINT statement, 107  
ZEROTOL= option  
    PROC INTPOINT, or INTPOINT statement, 108

# Your Turn

---

If you have comments or suggestions about *SAS/OR<sup>®</sup> Software: Changes and Enhancements, Release 8.2*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute Inc.  
SAS Publishing  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute Inc.  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [suggest@sas.com](mailto:suggest@sas.com)





*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS<sup>®</sup> Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

- 1.** Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
- 2.** Leave the "Subject" line blank.
- 3.** Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.



SAS Publishing

TECHNICAL REPORT

# **SAS/STAT<sup>®</sup> Software: Changes and Enhancements, Release 8.2**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/STAT® Software: Changes and Enhancements, Release 8.2*, Cary, NC: SAS Institute Inc., 2001

**SAS/STAT® Software: Changes and Enhancements, Release 8.2**  
Copyright © 2001 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-850-6

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2001

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Table of Contents

---

|                                                 |     |
|-------------------------------------------------|-----|
| Chapter 1. The BOXPLOT Procedure . . . . .      | 1   |
| Chapter 2. The CATMOD Procedure . . . . .       | 9   |
| Chapter 3. The FACTOR Procedure . . . . .       | 13  |
| Chapter 4. The FREQ Procedure . . . . .         | 17  |
| Chapter 5. The GAM Procedure . . . . .          | 21  |
| Chapter 6. The LIFEREG Procedure . . . . .      | 75  |
| Chapter 7. The LOESS Procedure . . . . .        | 105 |
| Chapter 8. The LOGISTIC Procedure . . . . .     | 115 |
| Chapter 9. The MI Procedure . . . . .           | 129 |
| Chapter 10. The MIANALYZE Procedure . . . . .   | 201 |
| Chapter 11. The NPAR1WAY Procedure . . . . .    | 235 |
| Chapter 12. The PROBIT Procedure . . . . .      | 239 |
| Chapter 13. The REG Procedure . . . . .         | 299 |
| Chapter 14. The SURVEYMEANS Procedure . . . . . | 303 |
| Subject Index . . . . .                         | 315 |
| Syntax Index . . . . .                          | 319 |



# Chapter 1

## The BOXPLOT Procedure

### Chapter Table of Contents

---

|                                                     |   |
|-----------------------------------------------------|---|
| <b>OVERVIEW</b> . . . . .                           | 3 |
| <b>SYNTAX</b> . . . . .                             | 3 |
| <b>DETAILS</b> . . . . .                            | 4 |
| <b>EXAMPLES</b> . . . . .                           | 5 |
| Example 1.1 Putting Overlays on Box Plots . . . . . | 5 |





# Chapter 1

## The BOXPLOT Procedure

---

### Overview

You can now specify variables from the input data set to be plotted as overlays on the box plot of an analysis variable. You can control the appearance of the symbols used to plot the overlay variables and the lines connecting the plotted points.

You can also specify uniform resource locators (URLs) to be associated with box-and-whisker plots and overlay points. When graphics output is directed to HTML, these URLs can provide links to other Web pages.

---

### Syntax

The PLOT statement supports the following new options.

**CCOVERLAY=**(*color-list*)

specifies colors for the line segments connecting points on overlays. Colors in the CCOVERLAY= list are matched with variables in the corresponding positions in the OVERLAY= list. By default, points are connected by line segments of the same color as the plotted points. You can specify the value NONE to suppress the line segments connecting points on an overlay.

**COVERLAY=**(*color-list*)

specifies the colors used to plot overlay variables. Colors in the COVERLAY= list are matched with variables in the corresponding positions in the OVERLAY= list.

**HTML=**(*variable*)

specifies uniform resource locators (URLs) as values of the specified character variable, for formatted values of a numeric variable. These URLs are associated with box-and-whisker plots when graphics output is directed into HTML. The value of the HTML= variable should be the same for each observation with a given value of the group variable.

**LOVERLAY=**(*linetypes*)

specifies line types for the line segments connecting points on overlays. Line types in the LOVERLAY= list are matched with variables in the corresponding positions in the OVERLAY= list.

**NOOVERLAYLEGEND**

suppresses the legend for overlay variables that is displayed by default when the OVERLAY= option is specified.

**OVERLAY=**(*variable-list*)

specifies variables to be plotted as overlays on a box plot. A point is plotted for each

overlay variable at each group for which it has a non-missing value. The value of a particular overlay variable should be the same for each observation in the input data set with a given value of the group variable.

**OVERLAYHTML=(variable-list)**

specifies variables whose values are URLs to be associated with points on overlays. These URLs are associated with points on an overlay plot when graphics output is directed into HTML. Variables in the OVERLAYHTML= list are matched with variables in the corresponding positions in the OVERLAY= list. The value of the OVERLAYHTML= variable should be the same for each observation with a given value of the group variable.

**OVERLAYLEGLAB='label'**

specifies the label displayed to the left of the overlay legend. The label can be up to 16 characters long and must be enclosed in quotes.

**OVERLAYSYM=(symbol-list)**

specifies symbols used to plot points on overlays. Symbols in the OVERLAYSYM= list are matched with variables in the corresponding positions in the OVERLAY= list.

**OVERLAYSYMHT=(value-list)**

specifies the heights of symbols used to plot points on overlays. Heights in the OVERLAYSYMHT= list are matched with variables in the corresponding positions in the OVERLAY= list.

**WOVERLAY=(value-list)**

specifies the widths in pixels of the line segments connecting points on overlays. Widths in the WOVERLAY= list are matched with variables in the corresponding positions in the OVERLAY= list.

---

## Details

Values specified with the COVERLAY=, OVERLAYSYM=, and OVERLAYSYMHT= options are matched with overlay variables by position in the option value lists. When one of these options is not specified, the associated values are taken from the SYMBOL definitions currently in effect. If an option is specified but lists fewer values than the number of overlay variables, the overlays for the remaining unmatched variables use values from the SYMBOL definitions. SYMBOL1 is used in the box plot, so the SYMBOL definitions associated by default with the overlay plots start with SYMBOL2.

The vertical axis of a box plot is scaled to accommodate each box-and-whisker plot and all the points on any overlaid plots. If the range of overlay variable values is much greater than the range of analysis variable values, the box-and-whisker plots may be compressed and difficult to read.

---

## Examples

---

### Example 1.1. Putting Overlays on Box Plots

---

The following statements create two data sets. The first, called `Scores`, contains student scores for each of four exams given during the spring 2000 semester. The `History` data set contains average scores made on the exams by classes in the two previous spring semesters. The `History` data set is merged with `Scores` to produce an input data set for the `BOXPLOT` procedure.

```
proc format;
  value examfmt
    1='Midterm 1'
    2='Midterm 2'
    3='Midterm 3'
    4='Final'
  ;

data Scores;
  input exam @;
  do i=1 to 18;
    input score @;
    output;
  end;
  drop i;
  datalines;
1 78 92 83 81 65 90 72 80 63 88 85 79 94 71 76 84 77 82
2 82 88 94 90 73 79 75 84 68 82 86 73 89 75 77 98 81 86
3 88 85 95 91 76 77 78 88 80 76 83 84 91 75 83 88 85 85
4 85 90 91 93 68 82 82 97 77 83 85 89 95 77 84 90 89 89
;
run;

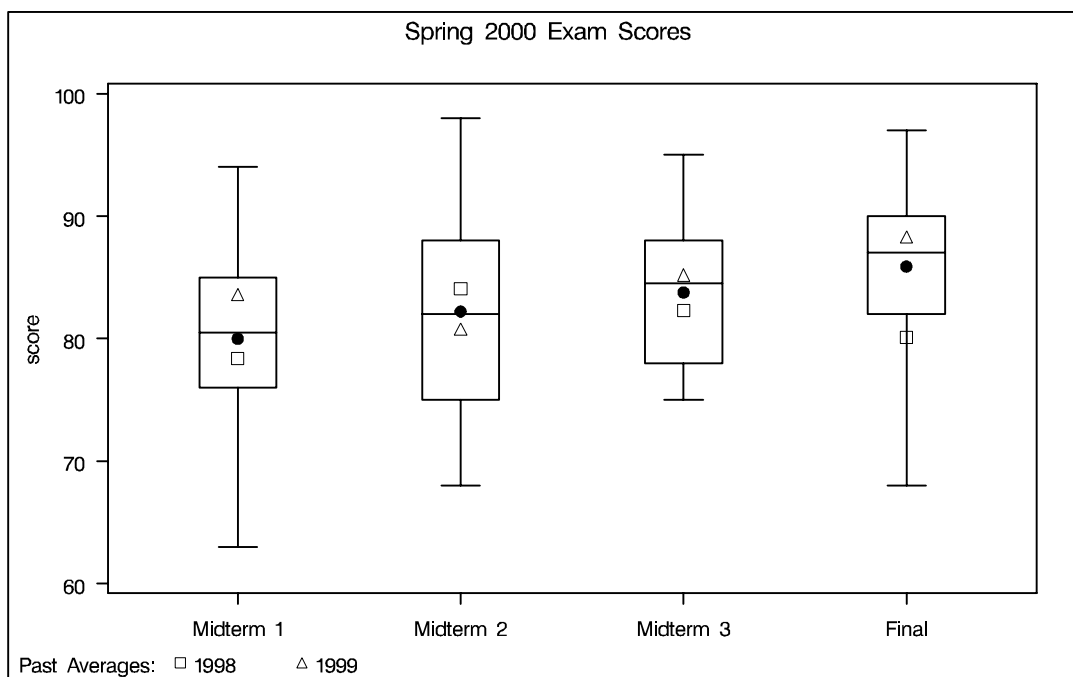
data History;
  label avg98 = '1998';
  label avg99 = '1999';
  input exam avg98 avg99;
  datalines;
1 78.4 83.6
2 84.1 80.8
3 82.3 85.2
4 80.1 88.3
;
run;

data Scores;
  merge scores history;
  by exam;
run;
```

You can create a box plot of scores on the four exams from 2000, overlaid with averages from the two previous years. The following statements produce the box plot with overlays shown in Output 1.1.1.

```
symbol1 v=dot c=salmon;
title 'Spring 2000 Exam Scores';
proc boxplot data=Scores;
  plot score * exam /
    cframe      = vlight
    cboxes      = dark
    cboxfill    = yellow
    overlay     = (avg98 avg99)
    coverlay    = (vlight vlight)
    overlaysym  = (square circle)
    ccoverlay   = (none none)
    overlaylab  = 'Past Averages:'
    nohlabel
  ;
  format exam examfmt.;
run;
```

**Output 1.1.1.** Box Plot of Exam Scores



The OVERLAY= option specifies the variables to plot as overlays on the box plot. The COVERLAY= and OVERLAYSYM= options select the colors and shapes of the symbols used to plot the overlays. CCOVERLAY= specifies the colors of the line segments used to connect the overlay points. In this case the keyword NONE is specified for each overlay, suppressing the connecting lines.

The order in which variables appear in the OVERLAY= list determines the values from related option lists that are used to plot the overlays. The variable `avg98` appears first in the OVERLAY= list, so it is plotted using the first values from the related lists: COVERLAY= VIGB, OVERLAYSYM= SQUARE, and CCOVERLAY= NONE. The values associated with the second overlay variable, `avg99`, are VIG, CIRCLE, and NONE.



# Chapter 2

## The CATMOD Procedure

### Chapter Table of Contents

---

|                           |    |
|---------------------------|----|
| <b>OVERVIEW</b> . . . . . | 11 |
| <b>SYNTAX</b> . . . . .   | 11 |
| MODEL Statement . . . . . | 11 |





## Chapter 2

# The CATMOD Procedure

---

### Overview

You can now produce Wald confidence limits for the parameter estimates with the CLPARM option in the MODEL statement; the ALPHA= option enables you to set the significance level.

---

### Syntax

---

#### MODEL Statement

The following options have been added to the MODEL statement.

**ALPHA=***value*

sets the significance level for the Wald confidence intervals for parameter estimates. The value must be between 0 and 1. The default value of 0.05 results in the calculation of a 95% confidence interval. This option has no effect unless the CLPARM option is also specified.

**CLPARM**

produces Wald confidence limits for the parameter estimates. The confidence coefficient can be specified with the ALPHA= option.

The  $100(1 - \alpha)\%$  Wald confidence interval for a parameter  $\beta$  is defined as

$$\hat{\beta} \pm z_{1-\alpha/2} \hat{\sigma}$$

where  $z_p$  is the 100 $p$ th percentile of the standard normal distribution,  $\hat{\beta}$  is the parameter estimate, and  $\hat{\sigma}$  is the estimate of its standard error.



# Chapter 3

## The FACTOR Procedure

### Chapter Table of Contents

---

|                                 |    |
|---------------------------------|----|
| <b>SYNTAX</b> . . . . .         | 15 |
| PROC FACTOR Statement . . . . . | 15 |



## Chapter 3

# The FACTOR Procedure

---

### Syntax

---

#### PROC FACTOR Statement

---

The following options have been restored.

**FLAG= $p$**

flags absolute values larger than  $p$  with an asterisk when used with the ROUND option. The default value of the FLAG= option is the root mean square of all the values in the matrix being printed. Negative values are not allowed for the FLAG= option. The FLAG= option is not effective when standard errors or confidence intervals are also printed.

**FUZZ= $p$**

prints correlations and factor loadings with absolute values less than  $p$  as missing. For partial correlations, the FUZZ= value is divided by 2. For residual correlations, the FUZZ= value is divided by 4. The exact values in any matrix can be obtained from the OUTSTAT= and ODS output data sets. Negative values are not allowed for the FUZZ= option. The FUZZ= option is not effective when standard errors or confidence intervals are also printed.

**ROUND**

prints correlation and loading matrices with entries multiplied by 100 and rounded to the nearest integer. The exact values can be obtained from the OUTSTAT= and ODS output data sets. The ROUND option is not effective when standard errors or confidence intervals are also printed. See also the FLAG= option.



# Chapter 4

## The FREQ Procedure

### Chapter Table of Contents

---

|                            |    |
|----------------------------|----|
| <b>OVERVIEW</b> . . . . .  | 19 |
| <b>SYNTAX</b> . . . . .    | 19 |
| EXACT Statement . . . . .  | 19 |
| TABLES Statement . . . . . | 20 |





## Chapter 4

# The FREQ Procedure

---

### Overview

The new POINT option in the EXACT statement requests exact point probabilities for the test statistics.

The BINOMIAL option in the TABLES statement has a new LEVEL= suboption to specify which variable level PROC FREQ uses to compute the binomial proportion for one-way tables. By default, PROC FREQ computes the binomial proportion for the first variable level that appears in the output.

The new BINOMIALC option in the TABLES statement computes the BINOMIAL option statistics, but includes a continuity correction in the asymptotic confidence limits and asymptotic test. Similarly, the new RISKDIFFC option in the TABLES statement computes the RISKDIFF statistics for  $2 \times 2$  tables, but includes a continuity correction in the asymptotic confidence limits.

---

### Syntax

---

#### EXACT Statement

**EXACT** *statistic-options* < / *computation-options* > ;

The EXACT statement requests exact tests or confidence limits for the specified statistics. Optionally, PROC FREQ computes Monte Carlo estimates of the exact *p*-values. The *statistic-options* specify the statistics for which to provide exact tests or confidence limits. The *computation-options* specify options for the computation of exact statistics.

The following new *computation-option* is now available in the EXACT statement after the slash (/):

#### POINT

requests exact point probabilities for the test statistics.

The POINT option is available for all the EXACT statement *statistic-options* except the OR option, which provides exact confidence limits as opposed to an exact test. The POINT option is not available with the MC option, which provides Monte Carlo estimation of exact *p*-values instead of direct exact *p*-value computation.

---

## TABLES Statement

The BINOMIAL option has a new LEVEL= suboption. The BINOMIALC and RISKDIFFC options have been added to the TABLES statement.

**BINOMIAL** < (**P=** *value*) | (**LEVEL=** *level-number* | *level-value*) >

requests the binomial proportion for one-way tables. The BINOMIAL option also provides the asymptotic standard error, asymptotic and exact confidence intervals, and the asymptotic test for the binomial proportion. To request an exact test for the binomial proportion, use the BINOMIAL option in the EXACT statement.

To specify the null hypothesis proportion for the test, use P=. If you omit P=*value*, PROC FREQ uses 0.5 as the default for the test. By default, the BINOMIAL option computes the proportion of observations for the first variable level that appears in the output. To specify a different level, use LEVEL=*level-number* or LEVEL=*level-value*, where *level-number* is the variable level's number or order in the output, and *level-value* is the formatted value of the variable level.

To include a continuity correction in the asymptotic confidence interval and test, use the BINOMIALC option instead of the BINOMIAL option.

**BINOMIALC** < (**P=** *value*) | (**LEVEL=** *level-number* | *level-value*) >

requests the BINOMIAL option statistics for one-way tables, but includes a continuity correction in the asymptotic confidence interval and the asymptotic test. The BINOMIAL option statistics include the binomial proportion, the asymptotic standard error, asymptotic and exact confidence intervals, and the asymptotic test for the binomial proportion. To request an exact test for the binomial proportion, use the BINOMIAL option in the EXACT statement.

To specify the null hypothesis proportion for the test, use P=. If you omit P=*value*, PROC FREQ uses 0.5 as the default for the test. By default, the BINOMIALC option computes the proportion of observations for the first variable level that appears in the output. To specify a different level, use LEVEL=*level-number* or LEVEL=*level-value*, where *level-number* is the variable level's number or order in the output, and *level-value* is the formatted value of the variable level.

### RISKDIFFC

requests the RISKDIFF option statistics for  $2 \times 2$  tables, but includes a continuity correction in the asymptotic confidence limits. The RISKDIFF option statistics include the column 1 and 2 risks (or binomial proportions), risk differences, and their confidence limits.

# Chapter 5

## The GAM Procedure

### Chapter Table of Contents

---

|                                                                  |    |
|------------------------------------------------------------------|----|
| <b>OVERVIEW</b>                                                  | 23 |
| <b>GETTING STARTED</b>                                           | 23 |
| <b>SYNTAX</b>                                                    | 28 |
| PROC GAM Statement                                               | 28 |
| BY Statement                                                     | 28 |
| CLASS Statement                                                  | 29 |
| FREQ Statement                                                   | 29 |
| ID Statement                                                     | 30 |
| MODEL Statement                                                  | 30 |
| OUTPUT Statement                                                 | 32 |
| SCORE Statement                                                  | 33 |
| <b>DETAILS</b>                                                   | 33 |
| Nonparametric Regression                                         | 33 |
| Additive Models and Generalized Additive Models                  | 34 |
| Backfitting and Local Scoring Algorithms                         | 35 |
| Smoothers                                                        | 38 |
| Selection of Smoothing Parameters                                | 39 |
| Confidence Intervals for Smoothers                               | 41 |
| Distribution Family and Canonical Link                           | 41 |
| Forms of Additive Models                                         | 42 |
| ODS Table Names                                                  | 43 |
| <b>EXAMPLES</b>                                                  | 43 |
| Example 5.1 Generalized Additive Model with Binary Data          | 43 |
| Example 5.2 Comparing PROC GAM with PROC TPSPLINE                | 50 |
| Example 5.3 Poisson Regression Analysis of Component Reliability | 55 |
| Example 5.4 Comparing PROC GAM with PROC LOESS                   | 63 |
| <b>REFERENCES</b>                                                | 72 |



## Chapter 5

# The GAM Procedure

---

### Overview

The GAM procedure is a new procedure that fits generalized additive models as those models are defined by Hastie and Tibshirani (1990). This procedure provides an array of powerful tools for data analysis, based on nonparametric regression and smoothing techniques.

Nonparametric regression relaxes the usual assumption of linearity and enables you to uncover structure in the relationship between the independent variables and the dependent variable that might otherwise be missed. The SAS System provides many procedures for nonparametric regression, such as the LOESS procedure for local regression and the TPSPLINE procedure for thin-plate smoothing splines. The generalized additive models fit by the GAM procedure combine

- an additive assumption (Stone 1985) that enables relatively many nonparametric relationships to be explored simultaneously with
- the distributional flexibility of generalized linear models (Nelder 1972)

Thus, you can use the GAM procedure when you have multiple independent variables whose effect you want to model nonparametrically, or when the dependent variable is not normally distributed. See the “Nonparametric Regression” section on page 33 for more details on the form of generalized additive models.

The GAM procedure

- provides nonparametric estimates for additive models
- supports the use of multidimensional data
- supports multiple SCORE statements
- fits both generalized semiparametric additive models and generalized additive models
- enables you to choose a particular model by specifying the model degrees of freedom or smoothing parameter

---

### Getting Started

The following example illustrates the use of the GAM procedure to explore in a nonparametric way how two factors affect a response. The data come from a study (Sackett et al. 1987) of the factors affecting patterns of insulin-dependent diabetes mellitus

in children. The objective is to investigate the dependence of the level of serum C-peptide on various other factors in order to understand the patterns of residual insulin secretion. The response measurement is the logarithm of C-peptide concentration (pmol/ml) at diagnosis, and the predictor measurements are age and base deficit (a measure of acidity):

```

title 'Patterns of Diabetes';
data diabetes;
  input Age BaseDeficit CPeptide @@;
  logCP = log(CPeptide);
  datalines;
    5.2   -8.1  4.8   8.8  -16.1  4.1  10.5   -0.9  5.2
  10.6   -7.8  5.5  10.4  -29.0  5.0   1.8  -19.2  3.4
  12.7  -18.9  3.4  15.6  -10.6  4.9   5.8   -2.8  5.6
    1.9 -25.0  3.7   2.2   -3.1  3.9   4.8   -7.8  4.5
    7.9 -13.9  4.8   5.2   -4.5  4.9   0.9  -11.6  3.0
   11.8   -2.1  4.6   7.9   -2.0  4.8  11.5   -9.0  5.5
   10.6 -11.2  4.5   8.5   -0.2  5.3  11.1   -6.1  4.7
   12.8   -1.0  6.6  11.3   -3.6  5.1   1.0   -8.2  3.9
   14.5   -0.5  5.7  11.9   -2.0  5.1   8.1   -1.6  5.2
   13.8 -11.9  3.7  15.5   -0.7  4.9   9.8   -1.2  4.8
   11.0 -14.3  4.4  12.4   -0.8  5.2  11.1  -16.8  5.1
    5.1   -5.1  4.6   4.8   -9.5  3.9   4.2  -17.0  5.1
    6.9   -3.3  5.1  13.2   -0.7  6.0   9.9   -3.3  4.9
   12.5 -13.6  4.1  13.2   -1.9  4.6   8.9  -10.0  4.9
   10.8 -13.5  5.1
  ;
run;

```

The following statements perform the desired analysis. The PROC GAM statement invokes the procedure and specifies the `diabetes` data set as input. The MODEL statement specifies `logCP` as the response variable and requests that univariate B-splines with the default of 4 degrees of freedom be used to model the effect of `Age` and `BaseDeficit`. The OUTPUT statement specifies that partial prediction curves are to be saved in the data set `estimates`:

```

title 'Patterns of Diabetes';
proc gam data=diabetes;
  model logCP = spline(age) spline(BaseDeficit);
  output out=estimates p;
run;

```

The results are shown in Figure 5.1 and Figure 5.2.

| Patterns of Diabetes                                          |              |
|---------------------------------------------------------------|--------------|
| The GAM Procedure                                             |              |
| Dependent Variable: logCP                                     |              |
| Smoothing Model Component(s): spline(Age) spline(BaseDeficit) |              |
| Summary of Input Data Set                                     |              |
| Number of Observations                                        | 43           |
| Number of Missing Observations                                | 0            |
| Distribution                                                  | Gaussian     |
| Link Function                                                 | Identity     |
| Iteration Summary and Fit Statistics                          |              |
| Final Number of Backfitting Iterations                        | 5            |
| Final Backfitting Criterion                                   | 5.542745E-10 |
| The Deviance of the Final Estimate                            | 0.4180791724 |

**Figure 5.1.** Summary Statistics

Figure 5.1 shows two tables. The first table summarizes the input data set and the distributional family used for the model, and the second one summarizes the convergence criterion for backfitting.

| Patterns of Diabetes                                          |                     |                |            |                |
|---------------------------------------------------------------|---------------------|----------------|------------|----------------|
| The GAM Procedure                                             |                     |                |            |                |
| Dependent Variable: logCP                                     |                     |                |            |                |
| Smoothing Model Component(s): spline(Age) spline(BaseDeficit) |                     |                |            |                |
| Regression Model Analysis                                     |                     |                |            |                |
| Parameter Estimates                                           |                     |                |            |                |
| Parameter                                                     | Parameter Estimate  | Standard Error | t Value    | Pr >  t        |
| Intercept                                                     | 1.48141             | 0.05120        | 28.93      | <.0001         |
| Linear(Age)                                                   | 0.01437             | 0.00437        | 3.28       | 0.0024         |
| Linear(BaseDeficit)                                           | 0.00807             | 0.00247        | 3.27       | 0.0025         |
| Smoothing Model Analysis                                      |                     |                |            |                |
| Fit Summary for Smoothing Components                          |                     |                |            |                |
| Component                                                     | Smoothing Parameter | DF             | GCV        | Num Unique Obs |
| Spline(Age)                                                   | 0.995582            | 4.000000       | 0.011675   | 37             |
| Spline(BaseDeficit)                                           | 0.995299            | 4.000000       | 0.012437   | 39             |
| Smoothing Model Analysis                                      |                     |                |            |                |
| Analysis of Deviance                                          |                     |                |            |                |
| Source                                                        | DF                  | Sum of Squares | Chi-Square | Pr > ChiSq     |
| Spline(Age)                                                   | 4.00000             | 0.150761       | 12.2605    | 0.0155         |
| Spline(BaseDeficit)                                           | 4.00000             | 0.081273       | 6.6095     | 0.1580         |

**Figure 5.2.** Analysis of Model

Figure 5.2 displays summary statistics for the model. It consists of three tables. The first is the “Parameter Estimates” table for the parametric part of the model. It indicates that the linear trends for both Age and BaseDeficit are highly significant with  $p$ -values of 0.0024 and 0.0025. The second table is the summary of smoothing components of the nonparametric part of the model. Since the GAM fit used the default  $DF = 4$ , the main point of this table is to present the smoothing parameter values that yield this  $DF$  for each component. Finally, the third table shows the “Analysis of Deviance” table for the nonparametric component of the model.

The `P` option in the `OUTPUT` statement puts the partial predictions for Age and BaseDeficit in the output data set. You can compute the entire partial prediction effect for each factor by adding the estimated linear terms to the respective partial predictions, as in the following statement:

```
data estimates; set estimates;
  P2_age          = P_age          + 0.01437*age;
  P2_BaseDeficit = P_BaseDeficit + 0.00807*BaseDeficit;
run;
```

Plotting the partial predictions is one way to explore the overall shape of the relationship between each factor and the response. First of all, the following statements set up the graphics options:

```
legend1 frame cframe=ligr cborder=black label=none position=center;
axis1 label=(angle=90 rotate=0) minor=none;
axis2 minor=none order=(0 to 16 by 4);
symbol1 color=red interpol=join value=none line=1;
symbol2 color=blue interpol=join value=none line=2;
```

Then the following statements plot the partial prediction curves for Age and BaseDeficit:

```
proc sort data=estimates;
  by age;

title;
proc gplot data=estimates;
  plot P_age *age = 1
       P2_Age*Age = 2 /overlay legend=legend1 frame
       cframe=ligr name='gam1' vaxis=axis1 haxis=axis2;
run;

proc sort data=estimates;
  by BaseDeficit;

axis2 minor=none;
proc gplot data=estimates;
  plot P_BaseDeficit *BaseDeficit = 1
       P2_BaseDeficit*BaseDeficit = 2 /
       overlay legend=legend1 frame cframe=ligr name='gam2'
       vaxis=axis1 haxis=axis2;
run;
```



Finally, the following statements redisplay the curves side by side for easy comparison:

```

goptions display;
proc greplay tc=tempcat nofs;
  igout gseg;
  tdef newtwo des='two plots of equal size'
  1/llx=0    lly=0
    ulx=0    uly=100
    urx=50   ury=100
    lrx=50   lry=0
  2/llx=50   lly=0
    ulx=50   uly=100
    urx=100  ury=100
    lrx=100  lry=0
;
template newtwo;
  treplay 1:gam1
          2:gam2;
run; quit;

```

The resulting plots for each predictor with (dotted lines) and without (solid lines) the linear term are shown in Figure 5.3.

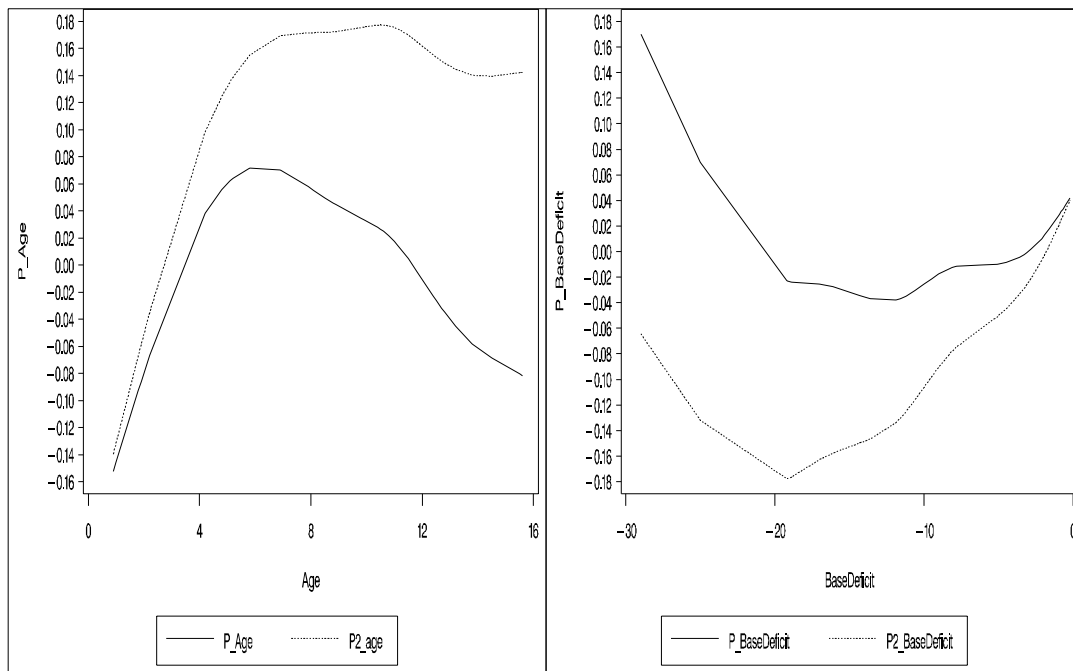


**Figure 5.3.** Partial Prediction for Each Predictor

Both plots show a strong quadratic pattern, with a possible indication of higher-order behavior. Further investigation is required to determine whether these patterns are real or not.

---

## Syntax

```

PROC GAM < option > ;
  CLASS variables ;
  MODEL dependent = < PARAM(effects) >
                        smoothing effects < /options > ;
  SCORE data=SAS-data-set out=SAS-data-set ;
  OUTPUT < out=SAS-data-set > keyword < ...keyword > < /option > ;
  BY variables ;
  ID variables ;
  FREQ variable ;

```

The syntax of the GAM procedure is similar to that of other regression procedures in the SAS System. The PROC GAM and MODEL statements are required. The SCORE statement can appear multiple times; all other statements appear only once.

The syntax for PROC GAM is described in the following sections in alphabetical order after the description of the PROC GAM statement.

---

### PROC GAM Statement

```
PROC GAM< option > ;
```

The PROC GAM statement invokes the procedure. You can specify the following option.

**DATA=SAS-data-set**

specifies the SAS data set to be read by PROC GAM. The default value is the most recently created data set.

---

### BY Statement

```
BY variables ;
```

You can specify a BY statement with PROC GAM to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in the order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the GAM procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (ac-

cording to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.

- Create an index for the BY variables using the DATASETS procedure.

For more information on the BY statement, refer to the discussion in *SAS Language Reference: Concepts, Version 8*. For more information on the DATASETS procedure, refer to the discussion in the *SAS Procedures Guide, Version 8*.

---

## CLASS Statement

**CLASS** *variables* ;

The CLASS statement names the classification variables to be used in the analysis. Typical class variables are TREATMENT, SEX, RACE, GROUP, and REPLICATION. If the CLASS statement is used, it must appear before the MODEL statement.

Classification variables can be either character or numeric. Class levels are determined from the formatted values of the CLASS variables. Thus, you can use formats to group values into levels. Refer to the discussion of the FORMAT procedure in the *SAS Procedures Guide, Version 8*, and the discussions for the FORMAT statement and SAS formats in *SAS Language Reference: Dictionary, Version 8*.

---

## FREQ Statement

**FREQ** *variable* ;

The FREQ statement names a variable that provides frequencies for each observation in the DATA= data set. Specifically, if  $n$  is the value of the FREQ variable for a given observation, then that observation is used  $n$  times.

The analysis produced using a FREQ statement reflects the expanded number of observations. You can produce the same analysis (without the FREQ statement) by first creating a new data set that contains the expanded number of observations. For example, if the value of the FREQ variable is 5 for the first observation, the first five observations in the new data set are identical. Each observation in the old data set is replicated  $n_i$  times in the new data set, where  $n_i$  is the value of the FREQ variable for that observation.

If the value of the FREQ variable is missing or is less than 1, the observation is not used in the analysis. If the value is not an integer, only the integer portion is used.

The FREQ statement is not available when a loess smoother is included in the model.

---

## ID Statement

**ID** *variables* ;

The variables in the ID statement are copied from the input data set to the OUT= data set. If you omit the ID statement, only the variables used in the MODEL statement and requested statistics are included in the output data set.

---

## MODEL Statement

**MODEL** *dependent*=<PARAM(*effects*)> <*smoothing effects*> </options>;

**MODEL** *event/trials*=<PARAM(*effects*)> <*smoothing effects*> </options> ;

The MODEL statement specifies the dependent variable and the independent effects you want to use to model its values. Specify the independent parametric variables inside the parentheses of PARAM( ). The parametric variables can be either CLASS variables or continuous variables. Class variables must be declared with a CLASS statement. Interactions between variables can also be included as parametric effects. The syntax for the specification of effects is the same as for the GLM procedure.

Any number of smoothing effects can be specified, as follows:

| Smoothing Effect                         | Meaning                                                   |
|------------------------------------------|-----------------------------------------------------------|
| SPLINE(variable <, df=number>)           | fit smoothing spline with the variable and with DF=number |
| LOESS(variable <, df=number>)            | fit local regression with the variable and with DF=number |
| SPLINE2(variable, variable <,df=number>) | fit bivariate thin-plate smoothing spline with DF=number  |

If you do not specify the DF=number option with a smoothing effect, DF=4 is used by default, unless you specify the METHOD=GCV model option.

Both parametric effects and smoothing effects are optional, but at least one of them must be present.

If only parametric variables are present, PROC GAM fits a parametric linear model using the terms inside the parentheses of PARAM( ). If only smoothing effects are present, PROC GAM fits a nonparametric additive model. If both types of effect are present, PROC GAM fits a semiparametric model using the parametric effects as the linear part of the model.

The following table shows how to specify various models for a dependent variable  $y$  and independent variables  $x$ ,  $x_1$ , and  $x_2$ .

**Table 5.1.** Syntax for Common GAM Models

| Type of Model     | Syntax                                               | Mathematical Form                       |
|-------------------|------------------------------------------------------|-----------------------------------------|
| Parametric        | model $y = \text{param}(x)$ ;                        | $E(y) = \beta_0 + \beta_1 x$            |
| Nonparametric     | model $y = \text{spline}(x)$ ;                       | $E(y) = \beta_0 + s(x)$                 |
| Nonparametric     | model $y = \text{loess}(x)$ ;                        | $E(y) = \beta_0 + s(x)$                 |
| Semiparametric    | model $y = \text{param}(x_1) \text{ spline}(x_2)$ ;  | $E(y) = \beta_0 + \beta_1 x_1 + s(x_2)$ |
| Additive          | model $y = \text{spline}(x_1) \text{ spline}(x_2)$ ; | $E(y) = \beta_0 + s_1(x_1) + s_2(x_2)$  |
| Thin-plate spline | model $y = \text{spline}(x_1, x_2)$ ;                | $E(y) = \beta_0 + s(x_1, x_2)$          |

You can specify the following options in the MODEL statement.

**ALPHA=number**

specifies the significance level  $\alpha$  of the confidence limits on the final nonparametric component estimates when you request confidence limits to be included in the output data set. Specify *number* as a value between 0 and 1. The default value is 0.05. See the “OUTPUT Statement” section on page 32 for more information on the OUTPUT statement.

**DIST=distribution-id**

specifies the distribution family used in the model. The *distribution-id* can be either GAUSSIAN, BINOMIAL, BINARY, GAMMA, or POISSON. The canonical link is used with those distributions. Although theoretically, alternative links are possible, with nonparametric models the final fit is relatively insensitive to the precise choice of link function. Therefore, only the canonical link for each distribution family is implemented in PROC GAM. The loess smoother is not available for the Binomial distribution when the number of trials is greater than 1.

**EPSILON=number**

specifies the convergence criterion for the backfitting algorithm. The default value is  $1.0e - 8$ .

**EPSSCORE=number**

specifies the convergence criterion for the local score algorithm. The default value is  $1.0e - 8$ .

**ITPRINT**

produces an iteration summary table for the smoothing effects.

**MAXITER=number**

specifies the maximum number of iterations for the backfitting algorithm. The default value is 50.

**MAXITSCORE=number**

specifies the maximum number of iterations for the local score algorithm. The default value is 100.

**METHOD=GCV**

specifies that the value of the smoothing parameter should be selected by generalized cross validation. If you specify both **METHOD=GCV** and the **DF=** option for the smoothing effects, the user-specified **DF=** is used, and the **METHOD=GCV** option is ignored. See the “Selection of Smoothing Parameters” section on page 39 for more details on the GCV method.

**NOTEST**

requests that the procedure not produce the “Analysis of Deviance” table. This option reduces the running time of the procedure.

---

**OUTPUT Statement**

**OUTPUT** *OUT=SAS-data-set* < *keyword* . . . *keyword* > ;

The **OUTPUT** statement creates a new SAS data set containing diagnostic measures calculated after fitting the model.

You can request a variety of diagnostic measures that are calculated for each observation in the data set. The new data set contains the variables specified in the **MODEL** statement in addition to the requested variables. If no *keyword* is present, the data set contains only the predicted values.

Details on the specifications in the **OUTPUT** statement are as follows.

**OUT=SAS-data-set**

specifies the name of the new data set to contain the diagnostic measures. This specification is required.

*keyword*

specifies the statistics to include in the output data set. The keywords and the statistics they represent are as follows:

|              |                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------|
| <b>PRED</b>  | predicted values for each smoothing component and overall predicted values at design points            |
| <b>UCLM</b>  | upper confidence limits for each predicted smoothing component                                         |
| <b>LCLM</b>  | lower confidence limits for each predicted smoothing component                                         |
| <b>ADIAG</b> | diagonal element of the hat matrix associated with the observation for each smoothing spline component |
| <b>RESID</b> | residual standardized by its weights                                                                   |
| <b>STD</b>   | standard deviation of the prediction for each smoothing component                                      |
| <b>ALL</b>   | implies all preceding keywords                                                                         |

The names of the new variables that contain the statistics are formed by using a prefix of one or more characters that identify the statistic, followed by an underscore (\_), followed by the variable name.

The prefixes of those new variables are as follows:

| Keywords | Prefix                             |
|----------|------------------------------------|
| PRED     | P_                                 |
| UCLM     | UCLM_                              |
| LCLM     | LCLM_                              |
| ADIAG    | ADIAG_                             |
| RESID    | R_                                 |
| STD      | STD_ for spline<br>STDP_ for loess |

For example, suppose that you have a dependent variable  $y$  and an independent smoothing variable  $x$ , and you specify the keywords PRED and ADIAG. In this case, the output SAS data set will contain the variables P\_ $y$ , P\_ $x$ , and ADIAG\_ $x$ .

---

## SCORE Statement

**SCORE** *DATA=SAS-data-set OUT=SAS-data-set ;*

The SCORE statement calculates predicted values for a new data set. The variables generated by the SCORE statement use the same naming conventions with prefixes as the OUTPUT statement. If you have multiple data sets to predict, you can specify multiple SCORE statements. You must use a SCORE statement for each data set.

The following options must be specified in the SCORE statement.

**DATA=SAS-data-set**

specifies an input SAS data set containing all the variables included in independent effects in the MODEL statement. The predicted response is computed for each observation in the SCORE DATA= data set.

**OUT=SAS-data-set**

specifies the name of the SAS data set to contain the predictions.

---

## Details

---

### Nonparametric Regression

Nonparametric regression relaxes the usual assumption of linearity and enables you to explore the data more flexibly, uncovering structure in the data that might otherwise be missed.

However, many forms of nonparametric regression do not perform well when the number of independent variables in the model is large. The sparseness of data in this setting causes the variances of the estimates to be unacceptably large. The problem of rapidly increasing variance for increasing dimensionality is sometimes referred to as the “curse of dimensionality.” Interpretability is another problem with nonparametric regression based on kernel and smoothing spline estimates. The information

these estimates contain about the relationship between the dependent and independent variables is often difficult to comprehend.

To overcome these difficulties, Stone (1985) proposed additive models. These models estimate an additive approximation to the multivariate regression function. The benefits of an additive approximation are at least twofold. First, since each of the individual additive terms is estimated using a univariate smoother, the curse of dimensionality is avoided, at the cost of not being able to approximate universally. Second, estimates of the individual terms explain how the dependent variable changes with the corresponding independent variables.

To extend the additive model to a wide range of distribution families, Hastie and Tibshirani (1990) proposed generalized additive models. These models enable the mean of the dependent variable to depend on an additive predictor through a non-linear link function. The models permit the response probability distribution to be any member of the exponential family of distributions. Many widely used statistical models belong to this general class; they include additive models for Gaussian data, nonparametric logistic models for binary data, and nonparametric log-linear models for Poisson data.

---

## Additive Models and Generalized Additive Models

This section describes the methodology and the fitting procedure behind generalized additive models.

Let  $Y$  be a response random variable and  $X_1, X_2, \dots, X_p$  be a set of predictor variables. A regression procedure can be viewed as a method for estimating the expected value of  $Y$  given the values of  $X_1, X_2, \dots, X_p$ . The standard linear regression model assumes a linear form for the conditional expectation

$$E(Y|X_1, X_2, \dots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

Given a sample, estimates of  $\beta_0, \beta_1, \dots, \beta_p$  are usually obtained by the least squares method.

The additive model generalizes the linear model by modeling the conditional expectation as

$$E(Y|X_1, X_2, \dots, X_p) = s_0 + s_1(X_1) + s_2(X_2) + \dots + s_p(X_p)$$

where  $s_i(X)$ ,  $i = 1, 2, \dots, p$  are smooth functions.

In order to be estimable, the smooth functions  $s_i$  have to satisfy standardized conditions such as  $E s_j(X_j) = 0$ . These functions are not given a parametric form but instead are estimated in a nonparametric fashion.

While traditional linear models and additive models can be used in most statistical data analysis, there are types of problems for which they are not appropriate. For example, the normal distribution may not be adequate for modeling discrete responses such as counts or bounded responses such as proportions.



Generalized additive models address these difficulties, extending additive models to many other distributions besides just the normal. Thus, generalized additive models can be applied to a much wider range of data analysis problems.

Similar to generalized linear models, generalized additive models consist of a random component, an additive component, and a link function relating the two components. The response  $Y$ , the random component, is assumed to have exponential family density

$$f_Y(y; \theta; \phi) = \exp \left\{ \frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi) \right\}$$

where  $\theta$  is called the natural parameter and  $\phi$  is the scale parameter. The mean of the response variable  $\mu$  is related to the set of covariates  $X_1, X_2, \dots, X_p$  by a link function  $g$ . The quantity

$$\eta = s_0 + \sum_{i=1}^p s_i(X_i)$$

where  $s_1(\cdot), \dots, s_p(\cdot)$  are smooth functions defines the additive component, and the relationship between  $\mu$  and  $\eta$  is defined by  $g(\mu) = \eta$ . The most commonly used link function is the canonical link, for which  $\eta = \theta$ .

Generalized additive models and generalized linear models can be applied in similar situations, but they serve different analytic purposes. Generalized linear models emphasize estimation and inference for the parameters of the model, while generalized additive models focus on exploring data nonparametrically. Generalized additive models are more suitable for exploring the data set and visualizing the relationship between the dependent variable and the independent variables.

---

## Backfitting and Local Scoring Algorithms

Consider the estimation of the smoothing terms  $s_0, s_1(\cdot), \dots, s_p(\cdot)$  in the additive model

$$\eta(X) = s_0 + \sum_{i=1}^p s_i(X_i)$$

where  $E[s_j(X_j)] = 0$  for every  $j$ . Since the algorithm for additive models is the basis for fitting generalized additive models, the algorithm for additive models is discussed first.

Many ways are available to approach the formulation and estimation of additive models. The backfitting algorithm is a general algorithm that can fit an additive model using any regression-type fitting mechanisms.

Define the  $j$ th set of partial residuals as

$$R_j = Y - s_0 - \sum_{k \neq j} s_k(X_k)$$

then  $E(R_j|X_j) = s_j(X_j)$ . This observation provides a way for estimating each smoothing function  $s_j(\cdot)$  given estimates  $\{\hat{s}_i(\cdot), i \neq j\}$  for all the others. The resulting iterative procedure is known as the backfitting algorithm (Friedman and Stuetzle 1981).

### The Backfitting Algorithm

**1. Initialization:**

$$s_0 = E(Y), s_1^1 = s_2^1 = \cdots = s_p^1 = 0, m = 0.$$

**2. Iterate:**

$$m = m + 1$$

for  $j = 1$  to  $p$  do:

$$R_j = Y - s_0 - \sum_{k=1}^{j-1} s_k^m(X_k) - \sum_{k=j+1}^p s_k^{m-1}(X_k)$$

$$S_j^m = E(R_j|X_j).$$

**3. Until:**

RSS = Avg( $Y - s_0 - \sum_{j=1}^p s_j^m(X_j)$ )<sup>2</sup> fails to decrease, or satisfies the convergence criterion.

In the preceding notation,  $s_j^m(\cdot)$  denotes the estimate of  $s_j(\cdot)$  at the  $m$ th iteration. It can be shown that RSS never increases at any step, which implies that the algorithm always converges.

The GAM procedure uses the following condition as the convergence criterion for the backfitting algorithm:

$$\frac{\sum_{j=1}^n \sum_{i=1}^k w(x_i)(s_i^{m-1}(x_j) - s_i^m(x_j))^2}{\sum_{j=1}^n \sum_{i=1}^k w(x_i)(s_i^{m-1}(x_j))^2} \leq \epsilon$$

where  $\epsilon = 10^{-8}$  by default; you can change this with EPSILON= option on the MODEL statement. A weighted backfitting algorithm has the same form as for the unweighted case, except that the smoothers are weighted. The weights might represent the relative precision of each observation or might arise as part of another iterative procedure. For example, weights are used in the local scoring procedure described later in this section.

The algorithm so far described fits just additive models. The algorithm for generalized additive models is a little more complicated. Generalized additive models extend generalized linear models in the same manner that additive models extend linear regression models, that is, by replacing form  $\alpha + \sum_j X_j \beta_j$  with the additive form  $\alpha + \sum_j f_j(X_j)$ . Thus, it is helpful to review the iteratively reweighted least-square procedure for computing the maximum likelihood estimates in a generalized linear model.

For generalized linear models, the maximum likelihood estimate of  $\beta$  is defined by the score equations

$$\sum_{i=1}^n x_{ij} \left( \frac{\partial \mu_i}{\partial \eta_i} \right) V_i^{-1}(y_i - \mu_i), j = 0, 1, \dots, p$$

where  $V_i$  is the variance matrix for  $Y_i$ . The Fisher scoring procedure is the standard method for solving these equations. It involves a Newton-Raphson algorithm using the expected information matrix. An equivalent procedure that is convenient for this problem is called dependent variable regression and is a form of iteratively reweighted least squares. Given a current coefficient vector  $\beta^0$ , with corresponding linear predictor  $\eta^0$  and fitted values  $\mu^0$ , construct the adjusted dependent variable

$$z_i = \eta_i^0 + (y_i - \mu_i^0) \left( \frac{\partial \eta_i}{\partial \mu_i} \right)_0$$

Define weights  $w_i$  by

$$w_i^{-1} = \left( \frac{\partial \eta_i}{\partial \mu_i} \right)_0^2 V_i^0$$

The algorithm proceeds by regressing  $z_i$  on  $x$  with weight  $w_i$  to obtain a revised estimate  $\beta$ . Then a new  $\mu^0$  and  $\eta^0$  are computed, new  $z_i$ s are computed, and the process is repeated until the change in the deviance

$$D(y; \hat{\mu}) = 2(l(\mu_{max}; y) - l(\hat{\mu}'y))$$

is sufficiently small.

Some adjusted dependent variables and weights for commonly used models are listed in the following table.

| Distribution         | Link        | Adjusted Dependent(Z)            | Weights(w)      |
|----------------------|-------------|----------------------------------|-----------------|
| Normal               | identity    | $y$                              | 1               |
| $\text{Bin}(n, \mu)$ | logit       | $\eta + (y - \mu)/n\mu(1 - \mu)$ | $n\mu(1 - \mu)$ |
| Gamma                | -reciprocal | $\eta + (y - \mu)/\mu^2$         | $1/\mu^2$       |
| Poisson              | log         | $\eta + (y - \mu)/\mu$           | $\mu$           |

Generalized additive models differ from generalized linear models in that an additive predictor replaces the linear predictor. Estimation of the additive terms is accomplished by replacing the weighted linear regression in the adjusted dependent variable regression by the weighted backfitting algorithm for fitting a weighted additive mode. This results in the following algorithm described as the *local scoring algorithm*. The name “local scoring” derives from the fact that local averaging is used to generalize the Fisher scoring procedure.

### **The General Local Scoring Algorithm**

#### **1. Initialization:**

$$s_i = g(E(y)), s_1^0 = s_2^0 = \cdots = s_p^0 = 0, m = 0.$$

#### **2. Iterate:**

$$m = m + 1$$

Form the adjusted dependent variable, predictor, and mean based on the previous iteration using following formulas:

$$\begin{aligned}
Z &= \eta^{m-1} + (Y - \mu^{m-1})(\partial\eta/\partial\mu^{m-1}) \\
\eta^{m-1} &= s_0 + \sum_{j=1}^p s_j^{m-1}(X_j) \\
\mu^{m-1} &= g^{-1}(\eta^{m-1}).
\end{aligned}$$

Form the weights

$$w_i = (\partial\mu^{m-1}/\partial\eta^{m-1})^2 V_i^{-1}.$$

Fit an additive model to  $Z$  using the backfitting algorithm with weights  $W$  to obtain estimated functions  $s_j^m(\cdot)$ .

**3. Until:**

$\text{Avg}(D(Y, \mu^m))$  fails to decrease, where  $\text{Avg}(D(Y, \mu^m))$  is an average of the deviance of estimate  $\mu^m$ ; or satisfies the convergence criterion.

The GAM procedure uses the following condition as the convergence criterion for local scoring:

$$\frac{\sum_{j=1}^n \sum_{i=1}^k w(x_i) (s_i^{m-1}(x_j) - s_i^m(x_j))^2}{\sum_{j=1}^n \sum_{i=1}^k w(x_i) (s_i^{m-1}(x_j))^2} \leq \epsilon^s$$

where  $\epsilon^s = 10^{-8}$  by default; you can change this with the `EPSSCORE=` option on the `MODEL` statement.

The estimating procedure for generalized additive models consists of two loops. Inside each step of the local scoring algorithm (outer loop), a weighted backfitting algorithm (inner loop) is used until convergence, that is, until the RSS fails to decrease. Then, based on the estimates from this weighted backfitting algorithm, a new set of weights is calculated and the next iteration of the scoring algorithm starts. The scoring algorithm stops when the deviance of the estimates ceases to decrease.

---

## Smoothers

A smoother is a tool for summarizing the trend of a response measurement  $Y$  as a function of one or more predictor measurements  $X_1, \dots, X_p$ . It produces an estimate of the trend that is less variable than  $Y$  itself. An important property of a smoother is its nonparametric nature. It doesn't assume a rigid form for the dependence of  $Y$  on  $X_1, \dots, X_p$ . This section gives a brief overview of the smoothers that can be used with the GAM procedure.

### Cubic Smoothing Spline

A smoothing spline is the solution to the following optimization problem: among all functions  $\eta(x)$  with two continuous derivatives, find one that minimizes the penalized least square

$$\sum_{i=1}^n (y_i - \eta(x_i))^2 + \lambda \int_a^b (\eta''(t))^2 dt$$

where  $\lambda$  is a fixed constant, and  $a \leq x_1 \leq \dots \leq x_n \leq b$ . The first term measures closeness to the data while the second term penalizes curvature in the function. It

can be shown that there exists an explicit, unique minimizer, and that minimizer is a natural cubic spline with knots at the unique values of  $x_i$ .

The parameter  $\lambda$  is the smoothing parameter. Large values of  $\lambda$  produce smoother curves while smaller values produce wiggly curves.

### Local Regression

Local regression is proposed by Cleveland, Devlin, and Grosse (1988). The idea of local regression is that at a predictor  $x$ , the regression function  $\eta(x)$  can be locally approximated by the value of a function in some specified parametric class. Such a local approximation is obtained by fitting a regression surface to the data points within a chosen neighborhood of the point  $x$ . A weighted least squares algorithm is used to fit linear or quadratic functions of the predictors at the centers of neighborhoods. The radius of each neighborhood is chosen so that the neighborhood contains a specified percentage of the data points. The smoothing parameter for the local regression procedure, which controls the smoothness of the estimated curve, is the fraction of the data in each local neighborhood. Data points in a given local neighborhood are weighted by a smooth decreasing function of their distance from the center of the neighborhood. Refer to “The LOESS Procedure” in *SAS/STAT User’s Guide, Version 8* for more details.

### Thin-Plate Smoothing Spline

The thin-plate smoothing spline is a multivariate version of the cubic smoothing spline. The theoretical foundations for the thin-plate smoothing spline are described in Duchon (1976, 1977) and Meinguet (1979). Further results and applications are given in Wahba and Wendelberger (1980). Refer to “The TPSPLINE Procedure” in *SAS/STAT User’s Guide, Version 8* for more details.

---

## Selection of Smoothing Parameters

### CV and GCV

The smoothers discussed here have a single smoothing parameter. In choosing the smoothing parameter, cross validation can be used. Cross validation works by leaving points  $(x_i, y_i)$  out one at a time, estimating the squared residual for smooth function at  $x_i$  based on the remaining  $n - 1$  data points, and choosing the smoother to minimize the sum of those squared residuals. This mimics the use of training and test samples for prediction. The cross validation function is defined as

$$CV(\lambda) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{\eta}_{\lambda}^{-i}(x_i))^2$$

where  $\hat{\eta}_{\lambda}^{-i}(x_i)$  indicates the fit at  $x_i$ , computed by leaving out the  $i$ th data point. The quantity  $nCV(\lambda)$  is sometimes called the prediction sum of squares or *PRESS* (Allen 1974).

All of the smoothers fit by the GAM procedure can be formulated as a linear combination of the sample responses

$$\hat{\eta}(x) = A(\lambda)Y$$

for some matrix  $A(\lambda)$ , which depends on  $\lambda$ . (The matrix  $A(\lambda)$  depends on  $x$  and the sample data, as well, but this dependence is suppressed in the preceding equation.) Let  $a_{ii}$  be the diagonal elements of the  $A(\lambda)$ . Then the  $CV$  function can be expressed as

$$CV(\lambda) = \frac{1}{n} \sum_{i=1}^n \left( \frac{(y_i - \hat{\eta}_\lambda(x_i))}{1 - a_{ii}} \right)^2$$

In most cases, it is very time consuming to compute the quantity  $a_{ii}$ . To solve this computational problem, Wahba (1990) has proposed the generalized cross validation function ( $GCV$ ) that can be used to solve a wide variety of problems involving selection of a parameter to minimize the prediction risk.

The  $GCV$  function is defined as

$$GCV(\lambda) = \frac{\sum_{i=1}^n (y_i - \hat{\eta}_\lambda(x_i))^2}{(n - \text{tr}(A(\lambda)))^2}$$

The  $GCV$  formula simply replaces the  $a_{ii}$  with  $\text{tr}(A(\lambda))/n$ . Therefore, it can be viewed as a weighted version of  $CV$ . In most of the cases of interest,  $GCV$  is closely related to  $CV$  but much easier to compute. The GAM procedure uses the  $GCV$  function as the criterion for choosing the smoothing parameters.

### Degrees of Freedom

The estimated GAM model can be expressed as

$$\hat{\eta}(X) = \hat{s}_0 + \sum_{i=1}^p A_i(y, \lambda) Y$$

Because the weights are calculated based on previous iteration during the local scoring iteration, the matrices  $A_i$  may depend on  $Y$  for non-Gaussian data. However, for the final iteration, the  $A_i$  matrix has the same role as the projection matrix in linear regression; therefore, nonparametric degrees of freedom (DF) for the  $i$ th smoother can be defined as

$$DF(\text{smoother}) = \text{tr}(A_i(y, \lambda))$$

The GAM procedure gives you the option of specifying the degrees of freedom for each individual smoothing component. If you choose a particular value for the degrees of freedom, then during every local scoring algorithm the corresponding smoothing parameter  $\lambda$  will be set to satisfying

$$\text{tr}(A_i(y, \lambda)) = df$$

The final estimate for the smoother during this local scoring iteration will be based on this  $\lambda$ .

---

## Confidence Intervals for Smoothers

In the GAM procedure, curvewise confidence intervals for smoothing splines and pointwise confidence intervals for loess are provided in the output dataset.

### **Curvewise Confidence Interval for Smoothing Spline**

Viewing the spline model as a Bayesian model, Wahba (1983) proposed Bayesian confidence intervals for smoothing spline estimates as follows:

$$\hat{s}_\lambda(x_i) \pm z_{\alpha/2} \sqrt{\hat{\sigma}^2 a_{ii}(\lambda)}$$

where  $a_{ii}(\lambda)$  is the  $i$ th diagonal element of the  $A(\lambda)$  matrix and  $z_{\alpha/2}$  is the  $\alpha/2$  point of the normal distribution. The confidence intervals are interpreted as intervals “across the function” as opposed to point-wise intervals.

Suppose that you fit a spline estimate to experimental data that consists of a true function  $f$  and a random error term,  $\epsilon_i$ . In repeated experiments, it is likely that about  $100(1 - \alpha)\%$  of the confidence intervals cover the corresponding true values, although some values are covered every time and other values are not covered by the confidence intervals most of the time. This effect is more pronounced when the true response curve or surface has small regions of particularly rapid change.

### **Pointwise Confidence Interval for Loess**

As defined in Cleveland et al. (1988), a standardized residual for a loess smoother follows a  $t$  distribution with  $\rho$  degrees of freedom, where  $\rho$  is called the “lookup degrees of freedom”, defined as

$$\rho = \delta_1^2 / \delta_2$$

where  $\delta_1 = \text{Trace}((I - A(\lambda))^T (I - A(\lambda)))$  and  $\delta_2 = \text{Trace}((I - A(\lambda))^T (I - A(\lambda)))^2$ . Therefore an approximate pointwise confidence interval at  $x_i$  is

$$\hat{s}_\lambda(x_i) \pm t_{\alpha/2; \rho} \hat{\sigma}(x_i)$$

where  $\hat{\sigma}(x_i)$  is the estimate of the standard deviation.

---

## Distribution Family and Canonical Link

For each distribution, more than one link can exist. Different link functions may result in slight differences in estimates for parametric models. However, the difference will be less pronounced for nonparametric models because of the flexibility of nonparametric model forms. To simplify the calculation, the GAM procedure uses the canonical link.

The GAM procedure can fit the data from the Gaussian and binomial distributions.

### **The Gaussian Model**

With this model, the link function is the identity function, and the generalized additive model is the additive model.

**The Binomial and Logistic Models**

A binomial response model assumes that the proportion of successes  $Y$  is such that  $Y$  has a  $Bin(n, p(x))$  distribution. The  $Bin(n, p(x))$  refers to the binomial distribution with parameters  $n$  and  $p(x)$ . Often the data are binary, in which case  $n = 1$ . The canonical link is

$$g(p) = \log \frac{p}{n - p} = \eta$$

**The Poisson Model**

The link function for the Poisson model is the log function. Assume the mean of the Poisson distribution is  $\mu(x)$ , the dependence of  $\mu(x)$  and independent variable  $x_1, \dots, x_k$  is

$$g(\mu) = \log(\mu) = \eta$$

**The Gamma Model**

Let the mean of the Gamma distribution be  $\mu(x)$ . The canonical link function for the Gamma distribution is  $-1/\mu(x)$ . Therefore, the relationship between  $\mu(x)$  and the independent variable  $x_1, \dots, x_k$  is

$$g(\mu) = -\frac{1}{\mu} = \eta$$

---

**Forms of Additive Models**

Suppose that  $y$  is a continuous variable, and  $x_1$  and  $x_2$  are two explanatory variables of interest. To fit an additive model, you can use a MODEL statement similar to that used in many regression procedures in the SAS system:

```
model y = spline(x1) spline(x2);
```

This model statement requires the procedure to fit the following model:

$$\eta(x_1, x_2) = \text{Intercept} + s_1(x_1) + s_2(x_2)$$

where the  $s_i(\cdot)$  terms denote nonparametric spline functions of the respective explanatory variables.

The GAM procedure can fit semiparametric models. The following MODEL statement assumes a linear relation with  $x_1$  and an unknown functional relation with  $x_2$ :

```
model y = param(x1) spline(x2);
```



If you want to fit a model containing a functional two-way interaction between  $x_1$  and  $x_2$ , you can use the following MODEL statement:

```
model y = spline2(x1,x2);
```

In this case, the GAM procedure fits a model equivalent to that of PROC TPSPLINE.

---

## ODS Table Names

PROC GAM assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, refer to the chapter “Using the Output Delivery System” in *SAS/STAT User’s Guide, Version 8*.

**Table 5.2.** ODS Tables Produced by PROC GAM

| ODS Table Name     | Description                                        | Statement | Option  |
|--------------------|----------------------------------------------------|-----------|---------|
| ANODEV             | Analysis of Deviance table for smoothing variables | PROC      | default |
| ClassSummary       | Summary of class variables                         | PROC      | default |
| ConvergenceStatus  | Convergence status of the local score algorithm    | PROC      | default |
| InputSummary       | Input data summary                                 | PROC      | default |
| IterHistory        | Iteration history table                            | MODEL     | ITPRINT |
| IterSummary        | Iteration summary                                  | PROC      | default |
| FitSummary         | Fit parameters and fit summary                     | PROC      | default |
| ParameterEstimates | Parameter estimation for regression variables      | PROC      | default |

By referring to the names of such tables, you can use the ODS OUTPUT statement to place one or more of these tables in output data sets.

---

## Examples

---

### Example 5.1. Generalized Additive Model with Binary Data

The following example illustrates the capabilities of the GAM procedure and compares it to the GENMOD procedure.

The data used in this example are based on a study by Bell et al. (1989). Bell and his associates studied the result of multiple-level thoracic and lumbar laminectomy, a corrective spinal surgery commonly performed on children. The data in the study consist of retrospective measurements on 83 patients. The specific outcome of interest is the presence (1) or absence (0) of kyphosis, defined as a forward flexion of the spine of at least 40 degrees from vertical. The available predictor variables are Age in months at time of the operation, the starting of vertebrae levels involved in the operation (StartVert), and the number of levels involved (NumVert). The goal of this analysis is to identify risk factors for kyphosis. PROC GENMOD can be used to

investigate the relationship among kyphosis and the predictors. The following DATA step creates the data kyphosis:

```

title 'Comparing PROC GAM with PROC GENMOD';
data kyphosis;
  input Age StartVert NumVert Kyphosis @@;
  datalines;
71 5 3 0      158 14 3 0      128 5 4 1
2 1 5 0      1 15 4 0      1 16 2 0
61 17 2 0     37 16 3 0     113 16 2 0
59 12 6 1     82 14 5 1     148 16 3 0
18 2 5 0      1 12 4 0     243 8 8 0
168 18 3 0    1 16 3 0     78 15 6 0
175 13 5 0    80 16 5 0    27 9 4 0
22 16 2 0    105 5 6 1     96 12 3 1
131 3 2 0    15 2 7 1      9 13 5 0
12 2 14 1    8 6 3 0      100 14 3 0
4 16 3 0     151 16 2 0     31 16 3 0
125 11 2 0   130 13 5 0    112 16 3 0
140 11 5 0   93 16 3 0      1 9 3 0
52 6 5 1     20 9 6 0      91 12 5 1
73 1 5 1     35 13 3 0     143 3 9 0
61 1 4 0     97 16 3 0     139 10 3 1
136 15 4 0   131 13 5 0    121 3 3 1
177 14 2 0   68 10 5 0      9 17 2 0
139 6 10 1   2 17 2 0      140 15 4 0
72 15 5 0    2 13 3 0     120 8 5 1
51 9 7 0     102 13 3 0     130 1 4 1
114 8 7 1    81 1 4 0     118 16 3 0
118 16 4 0   17 10 4 0    195 17 2 0
159 13 4 0   18 11 4 0     15 16 5 0
158 15 4 0   127 12 4 0    87 16 4 0
206 10 4 0   11 15 3 0     178 15 4 0
157 13 3 1   26 13 7 0    120 13 2 0
42 6 7 1     36 13 4 0
;

proc genmod;
  model Kyphosis = Age StartVert NumVert
                  / link=logit dist=binomial;
run;

```

**Output 5.1.1.** GENMOD Analysis: Partial Output

| Comparing PROC GAM with PROC GENMOD                                                                                                                                                     |    |          |                |                            |        |            |            |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----------|----------------|----------------------------|--------|------------|------------|
| The GENMOD Procedure                                                                                                                                                                    |    |          |                |                            |        |            |            |
| PROC GENMOD is modeling the probability that Kyphosis='0'. One way to change this to model the probability that Kyphosis='1' is to specify the DESCENDING option in the PROC statement. |    |          |                |                            |        |            |            |
| Analysis Of Parameter Estimates                                                                                                                                                         |    |          |                |                            |        |            |            |
| Parameter                                                                                                                                                                               | DF | Estimate | Standard Error | Wald 95% Confidence Limits |        | Chi-Square | Pr > ChiSq |
| Intercept                                                                                                                                                                               | 1  | 1.2497   | 1.2424         | -1.1853                    | 3.6848 | 1.01       | 0.3145     |
| Age                                                                                                                                                                                     | 1  | -0.0061  | 0.0055         | -0.0170                    | 0.0048 | 1.21       | 0.2713     |
| StartVert                                                                                                                                                                               | 1  | 0.1972   | 0.0657         | 0.0684                     | 0.3260 | 9.01       | 0.0027     |
| NumVert                                                                                                                                                                                 | 1  | -0.3031  | 0.1790         | -0.6540                    | 0.0477 | 2.87       | 0.0904     |
| Scale                                                                                                                                                                                   | 0  | 1.0000   | 0.0000         | 1.0000                     | 1.0000 |            |            |
| NOTE: The scale parameter was held fixed.                                                                                                                                               |    |          |                |                            |        |            |            |

The GENMOD analysis of the independent variable effects is shown in Output 5.1.1. Based on these results, the only significant factor is **StartVert** with a log odds ratio of  $-0.1972$ . The variable **NumVert** has a  $p$ -value of 0.0904 with a log odds ratio of 0.3031.

The GENMOD procedure assumes a strict linear relationship between the response and the predictors. The following SAS statements use PROC GAM to investigate a less restrictive model, with moderately flexible spline terms for each of the predictors:

```

title 'Comparing PROC GAM with PROC GENMOD';
proc gam data=kyphosis;
    model Kyphosis=spline(Age,df=3) spline(StartVert,df=3)
           spline(NumVert,df=3) /dist = binomial;
    output out=estimate p;
run;

```

The MODEL statement requests an additive model using a univariate B-spline for each term. The option `dist=binomial` with binary responses specifies a logistic model. Each term is fitted using a smoothing spline with three degrees of freedom. Although this might seem to be an unduly modest amount of flexibility, it is better to be conservative with a data set this small. An output data set `estimate` containing predicted values is requested by the OUTPUT statement.

Output 5.1.2 and Output 5.1.3 list the output from PROC GAM.

Output 5.1.2. Summary Statistics

| Comparing PROC GAM with PROC GENMOD                                         |              |
|-----------------------------------------------------------------------------|--------------|
| The GAM Procedure                                                           |              |
| Dependent Variable: Kyphosis                                                |              |
| Smoothing Model Component(s): spline(Age) spline(StartVert) spline(NumVert) |              |
| Summary of Input Data Set                                                   |              |
| Number of Observations                                                      | 83           |
| Number of Missing Observations                                              | 0            |
| Distribution                                                                | Binomial     |
| Link Function                                                               | Logit        |
| Iteration Summary and Fit Statistics                                        |              |
| Number of local score iterations                                            | 9            |
| Local score convergence criterion                                           | 2.6635758E-9 |
| Final Number of Backfitting Iterations                                      | 1            |
| Final Backfitting Criterion                                                 | 5.2326788E-9 |
| The Deviance of the Final Estimate                                          | 46.610922317 |

Output 5.1.3. Model Fit Statistics

| Comparing PROC GAM with PROC GENMOD                                         |                     |                |            |                |
|-----------------------------------------------------------------------------|---------------------|----------------|------------|----------------|
| The GAM Procedure                                                           |                     |                |            |                |
| Dependent Variable: Kyphosis                                                |                     |                |            |                |
| Smoothing Model Component(s): spline(Age) spline(StartVert) spline(NumVert) |                     |                |            |                |
| Regression Model Analysis                                                   |                     |                |            |                |
| Parameter Estimates                                                         |                     |                |            |                |
| Parameter                                                                   | Parameter Estimate  | Standard Error | t Value    | Pr >  t        |
| Intercept                                                                   | -2.01533            | 1.45620        | -1.38      | 0.1706         |
| Linear(Age)                                                                 | 0.01213             | 0.00794        | 1.53       | 0.1308         |
| Linear(StartVert)                                                           | -0.18615            | 0.07628        | -2.44      | 0.0171         |
| Linear(NumVert)                                                             | 0.38347             | 0.19102        | 2.01       | 0.0484         |
| Smoothing Model Analysis                                                    |                     |                |            |                |
| Fit Summary for Smoothing Components                                        |                     |                |            |                |
| Component                                                                   | Smoothing Parameter | DF             | GCV        | Num Unique Obs |
| Spline(Age)                                                                 | 0.999996            | 3.000000       | 328.512864 | 66             |
| Spline(StartVert)                                                           | 0.999551            | 3.000000       | 317.646703 | 16             |
| Spline(NumVert)                                                             | 0.921758            | 3.000000       | 20.144058  | 10             |
| Smoothing Model Analysis                                                    |                     |                |            |                |
| Analysis of Deviance                                                        |                     |                |            |                |
| Source                                                                      | DF                  | Sum of Squares | Chi-Square | Pr > ChiSq     |
| Spline(Age)                                                                 | 3.00000             | 10.494369      | 16.4358    | 0.0009         |
| Spline(StartVert)                                                           | 3.00000             | 5.494968       | 8.6060     | 0.0350         |
| Spline(NumVert)                                                             | 3.00000             | 2.184518       | 3.4213     | 0.3311         |

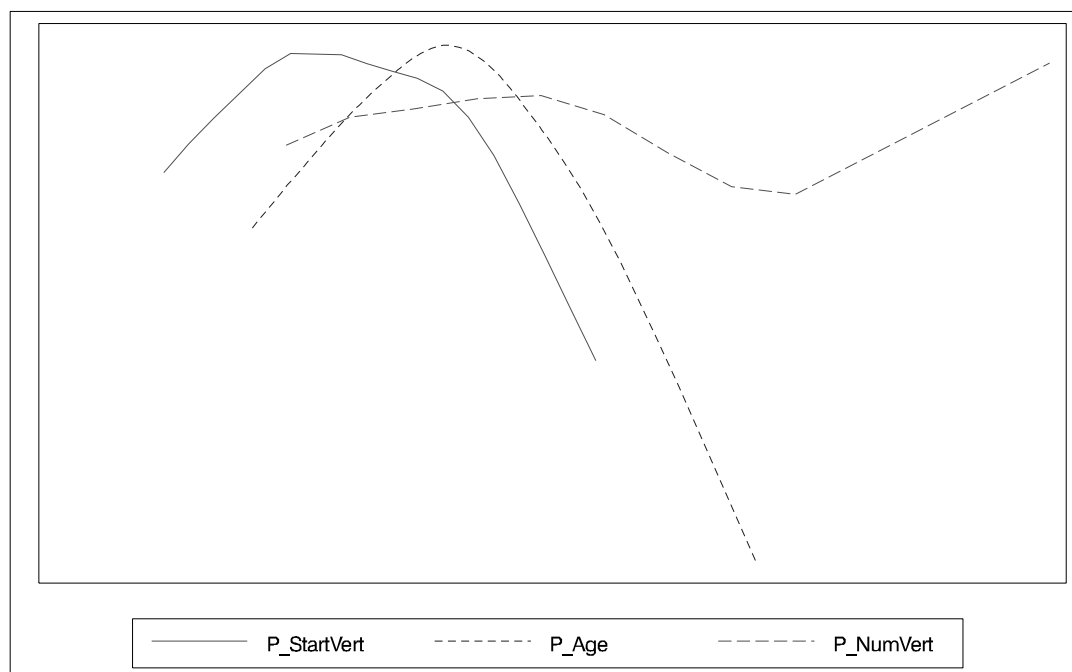
The critical part of the GAM results is the “Analysis of Deviance” table, shown in Output 5.1.3. For each smoothing effect in the model, this table gives an  $\chi^2$ -test comparing the deviance between the full model and the model without this variable. In this case, the analysis of deviance results indicates that the effect of Age and StartVert are significant, while the effect of NumVert is insignificant. Plots of predictions against predictor can be used to investigate why PROC GAM and PROC GENMOD produce different results.

The GAM statement requests an output data set of predicted values to be created. Since the estimate of the generalized additive model is the sum of functional estimates of individual predictors, plus a constant, the output data set will contain a column of partial prediction for each predictor. If requested, a Bayesian confidence interval or a point-wise standard-error band, as defined in Hastie and Tibshirani (1990), can be produced in the output data set.

Using the following statements, the data set estimate is plotted in Output 5.1.4:

```
proc sort data=estimate(keep=StartVert P_StartVert)
    out =StartVert;
    by StartVert;
proc sort data=estimate(keep=Age          P_Age          )
    out =Age;
    by Age;
proc sort data=estimate(keep=NumVert      P_NumVert      )
    out =NumVert;
    by NumVert;
data Plot; merge StartVert Age NumVert;
proc standard m=0 s=1 data=Plot out=Plot;
    var StartVert Age NumVert;
run;

legend1 frame cframe=ligr cborder=black label=none
    position=center;
axis1 label=none minor=none major=none value=none;
axis2 label=none minor=none major=none value=none;
symbol1 color=red   interpol=join value=none line=1;
symbol2 color=blue  interpol=join value=none line=2;
symbol3 color=green interpol=join value=none line=3;
proc gplot data=Plot;
    title;
    plot P_StartVert*StartVert=1
        P_Age          *Age          =2
        P_NumVert      *NumVert      =3 / overlay legend=legend1
        frame cframe=ligr vaxis=axis1 haxis=axis2;
run;
```

**Output 5.1.4.** Partial Prediction for Each Predictor

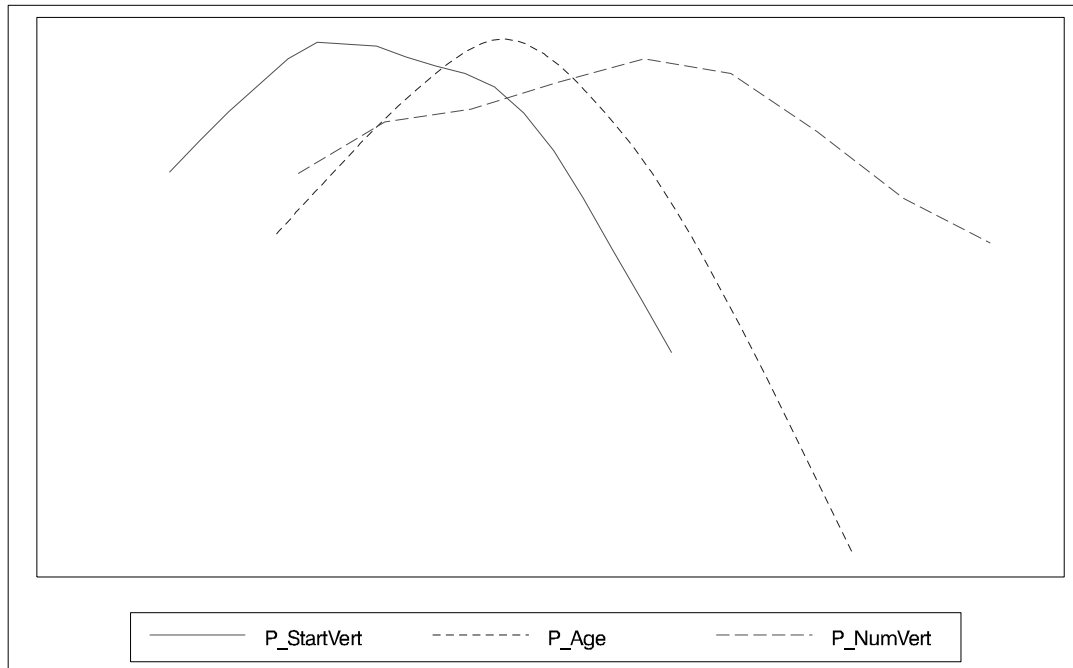
The plot shows that the partial predictions corresponding to both Age and StartVert have a strong quadratic pattern, while NumVert has a more complicated but weaker pattern. However, in the plot for NumVert, notice that about half the vertical range of the function is determined by the point at the upper extreme. It would be a good idea, therefore, to re-run the analysis without this point, to see how much it affects the conclusions. You can do this by simply including a WHERE clause when specifying the data set for the GAM procedure, as in the following code:

```
title 'Comparing PROC GAM with PROC GENMOD';
proc gam data=kyphosis(where=(NumVert^=14));
  model Kyphosis=spline(Age,df=3) spline(StartVert,df=3)
           spline(NumVert, df=3) /dist = binomial;
  output out=estimate p;
run;
```

**Output 5.1.5.** Analysis After Removing NumVert=14

| Comparing PROC GAM with PROC GENMOD                                         |         |                |            |            |
|-----------------------------------------------------------------------------|---------|----------------|------------|------------|
| The GAM Procedure                                                           |         |                |            |            |
| Dependent Variable: Kyphosis                                                |         |                |            |            |
| Smoothing Model Component(s): spline(Age) spline(StartVert) spline(NumVert) |         |                |            |            |
| Smoothing Model Analysis                                                    |         |                |            |            |
| Analysis of Deviance                                                        |         |                |            |            |
| Source                                                                      | DF      | Sum of Squares | Chi-Square | Pr > ChiSq |
| Spline(Age)                                                                 | 3.00000 | 10.587557      | 16.9032    | 0.0007     |
| Spline(StartVert)                                                           | 3.00000 | 5.477093       | 8.7443     | 0.0329     |
| Spline(NumVert)                                                             | 3.00000 | 3.209089       | 5.1234     | 0.1630     |

The analysis of deviance table from this re-analysis is shown in Output 5.1.5, and Output 5.1.6 shows the re-computed partial predictor plots.

**Output 5.1.6.** Partial Prediction After Removing NumVert=14

After removing data point NumVert=14, the predictors `age` and `StartVert` are still significant and the variable `NumVert` still insignificant. Based on the plot in Output 5.1.6, the removed point has almost no effect on estimates of curve shape for variables `Age` and `StartVert`. But the removal has a dramatic effect on the variable `NumVert`: this curve for this variable `NumVert` now also seems quadratic, though it is much less pronounced than for the other two variables.

Having used the GAM procedure to discover an appropriate form of the dependence of `Kyphosis` on each of the three independent variables, you can use the GENMOD procedure to fit and assess the corresponding parametric model. The following code

fits a GENMOD model with quadratic terms for all three variables, including tests for the joint linear and quadratic effects of each variable. The resulting contrast tests are shown in Output 5.1.7.

```

title 'Comparing PROC GAM with PROC GENMOD';
proc genmod data=kyphosis(where=(NumVert<=14));
  model kyphosis = Age      Age      *Age
                  StartVert StartVert*StartVert
                  NumVert   NumVert  *NumVert
                  /link=logit dist=binomial;
  contrast 'Age'      Age      1, Age*Age      1;
  contrast 'StartVert' StartVert 1, StartVert*StartVert 1;
  contrast 'NumVert'   NumVert  1, NumVert*NumVert  1;
run;

```

**Output 5.1.7.** Joint Linear and Quadratic Tests

| Comparing PROC GAM with PROC GENMOD                                                                                                                                                     |    |            |            |      |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|------------|------------|------|
| The GENMOD Procedure                                                                                                                                                                    |    |            |            |      |
| PROC GENMOD is modeling the probability that Kyphosis='0'. One way to change this to model the probability that Kyphosis='1' is to specify the DESCENDING option in the PROC statement. |    |            |            |      |
| Contrast Results                                                                                                                                                                        |    |            |            |      |
| Contrast                                                                                                                                                                                | DF | Chi-Square | Pr > ChiSq | Type |
| Age                                                                                                                                                                                     | 2  | 13.63      | 0.0011     | LR   |
| StartVert                                                                                                                                                                               | 2  | 15.41      | 0.0005     | LR   |
| NumVert                                                                                                                                                                                 | 2  | 3.56       | 0.1684     | LR   |

The results for the quadratic GENMOD model are now quite consistent with the GAM results.

From this example, you can see that PROC GAM is very useful in visualizing the data and detecting the nonlinearity among the variables.

## Example 5.2. Comparing PROC GAM with PROC TPSPLINE

This example compares the GAM procedure with the TPSPLINE procedure, another nonparametric procedure that fits a smooth surface to multivariate data. It does not assume additivity of the model and uses very general basis functions for model fitting, making the TPSPLINE procedure much slower than the GAM procedure. For more details about the TPSPLINE procedure, refer to “The TPSPLINE Procedure” in *SAS/STAT User’s Guide, Version 8*.

The data used here is also analyzed in “The TPSPLINE Procedure” in *SAS/STAT User’s Guide, Version 8*. It presents age-adjusted melanoma incidences for 37 years from the Connecticut Tumor Registry (Houghton, Flannery, and Viola 1980):



```

title 'Comparing PROC GAM with PROC TPSPLINE';
data melanoma;
  input year incidences @@;
  datalines;
1936    0.9    1937    0.8    1938    0.8    1939    1.3
1940    1.4    1941    1.2    1942    1.7    1943    1.8
1944    1.6    1945    1.5    1946    1.5    1947    2.0
1948    2.5    1949    2.7    1950    2.9    1951    2.5
1952    3.1    1953    2.4    1954    2.2    1955    2.9
1956    2.5    1957    2.6    1958    3.2    1959    3.8
1960    4.2    1961    3.9    1962    3.7    1963    3.3
1964    3.7    1965    3.9    1966    4.1    1967    3.8
1968    4.7    1969    4.4    1970    4.8    1971    4.8
1972    4.8
;
run;

```

The variable `incidences` records the number of melanoma cases per 100,000 people for the years 1936 to 1972.

Four to five degrees of freedom for each nonparametric term in a generalized additive model fits most data well. However, to select DF more objectively you can use the GCV option to minimize the generalized cross validation function, as shown in the following PROC GAM code:

```

title 'Comparing PROC GAM with PROC TPSPLINE';
proc gam data=melanoma;
  model incidences = spline(year) /method = GCV;
  output out=gam p;
run;

```

The results are listed in Output 5.2.1 and Output 5.2.2.

#### Output 5.2.1. Summary Statistics

| Comparing PROC GAM with PROC TPSPLINE      |              |
|--------------------------------------------|--------------|
| The GAM Procedure                          |              |
| Dependent Variable: incidences             |              |
| Smoothing Model Component(s): spline(year) |              |
| Summary of Input Data Set                  |              |
| Number of Observations                     | 37           |
| Number of Missing Observations             | 0            |
| Distribution                               | Gaussian     |
| Link Function                              | Identity     |
| Iteration Summary and Fit Statistics       |              |
| Final Number of Backfitting Iterations     | 2            |
| Final Backfitting Criterion                | 0            |
| The Deviance of the Final Estimate         | 1.2242517494 |

Output 5.2.2. Model Fit Statistics

| Comparing PROC GAM with PROC TPSPLINE      |                     |                |            |                |
|--------------------------------------------|---------------------|----------------|------------|----------------|
| The GAM Procedure                          |                     |                |            |                |
| Dependent Variable: incidences             |                     |                |            |                |
| Smoothing Model Component(s): spline(year) |                     |                |            |                |
| Regression Model Analysis                  |                     |                |            |                |
| Parameter Estimates                        |                     |                |            |                |
| Parameter                                  | Parameter Estimate  | Standard Error | t Value    | Pr >  t        |
| Intercept                                  | -212.69706          | 7.00491        | -30.36     | <.0001         |
| Linear(year)                               | 0.11029             | 0.00358        | 30.77      | <.0001         |
| Smoothing Model Analysis                   |                     |                |            |                |
| Fit Summary for Smoothing Components       |                     |                |            |                |
| Component                                  | Smoothing Parameter | DF             | GCV        | Num Unique Obs |
| Spline(year)                               | 0.634903            | 13.414936      | 0.088803   | 37             |
| Smoothing Model Analysis                   |                     |                |            |                |
| Analysis of Deviance                       |                     |                |            |                |
| Source                                     | DF                  | Sum of Squares | Chi-Square | Pr > ChiSq     |
| Spline(year)                               | 13.41494            | 2.736763       | 50.4880    | <.0001         |

Based on the summary of the model, the final model has a DF = 13.414936 and the nonparametric trend is highly significant. Note that this DF is much greater than the default value of 4, indicating that there is a great deal of structure in the yearly incidence rates of melanoma. A prediction plot should reveal the nature of this structure:

```

legend1 frame cframe=ligr cborder=black label=none
      position=center;
axis1  label=(angle=90 rotate=0);
axis2  minor=none;
symbol1 color=red interpol=join value=none line=1;

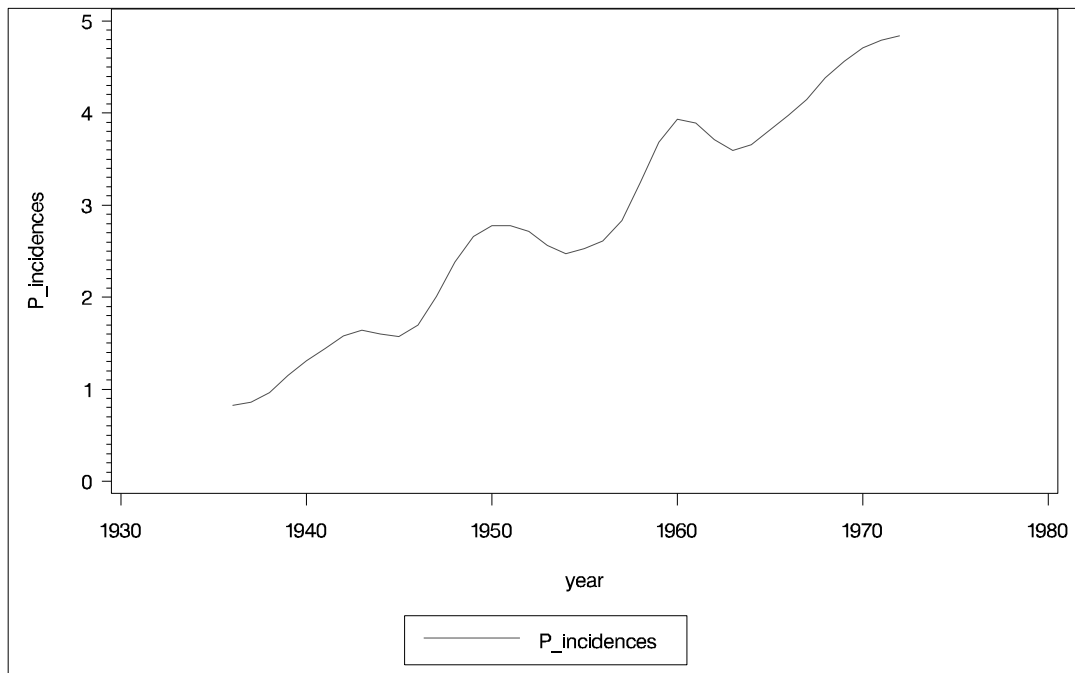
proc sort data=gam; by year;
proc gplot data=gam;
  title;
  plot p_incidences*year = 1 /overlay legend=legend1
    frame cframe=ligr vaxis=axis1 haxis=axis2;
run;

```

Output 5.2.3 shows the predicted melanoma rate over time. Two features stand out on this plot:

- Melanoma incidence on the whole increases over the period of the study.
- A strong periodic effect is evident, with a period of a little more than a decade. This has been attributed to the 11-year sunspot cycle: the more sunspots there are, the more melanoma cases there are likely to be.

**Output 5.2.3.** Predicted Melanoma Incidence Rates By Year



Since PROC TPSPLINE also fits a nonparametric model, PROC TPSPLINE and PROC GAM fits should be very similar for this univariate case. The following code produces the TPSPLINE analysis shown in Output 5.2.4:

```
title 'Comparing PROC GAM with PROC TPSPLINE';
proc tpspline data=melanoma;
  model incidences = (year);
  output out=tpspline p;
run;
```

**Output 5.2.4.** Analysis from PROC TPSPLINE

| Comparing PROC GAM with PROC TPSPLINE     |         |
|-------------------------------------------|---------|
| The TPSPLINE Procedure                    |         |
| Dependent Variable: incidences            |         |
| Summary of Input Data Set                 |         |
| Number of Non-Missing Observations        | 37      |
| Number of Missing Observations            | 0       |
| Unique Smoothing Design Points            | 37      |
| Summary of Final Model                    |         |
| Number of Regression Variables            | 0       |
| Number of Smoothing Variables             | 1       |
| Order of Derivative in the Penalty        | 2       |
| Dimension of Polynomial Space             | 2       |
| Summary Statistics<br>of Final Estimation |         |
| log10(n*Lambda)                           | -0.0607 |
| Smoothing Penalty                         | 0.5171  |
| Residual SS                               | 1.2243  |
| Tr(I-A)                                   | 22.5852 |
| Model DF                                  | 14.4148 |
| Standard Deviation                        | 0.2328  |

The TPSPLINE model analysis shows that the DF for the model is 14.4148. This is consistent with the GAM model because the DF value in GAM excludes the degree of freedom of the linear Linear(year) term. The OUTPUT statements in the PROC GAM and PROC TPSPLINE code create gam and tpspline data sets containing the predicted values for the respective procedures. You can use the following code to look at the values for the two procedures side by side:

```

title 'Comparing PROC GAM with PROC TPSPLINE';
data both; merge gam      (rename=(p_incidences=gam      ))
                tpspline(rename=(p_incidences=tpspline));
proc print data=both;
  var year gam tpspline;
run;

```

The results, the first ten of which are displayed in Output 5.2.5, show that PROC GAM and PROC TPSPLINE give essentially the same predictions for this problem.

**Output 5.2.5.** Melanoma Predictions for First Ten Years

| Comparing PROC GAM with PROC TPSPLINE |     |      |         |          |
|---------------------------------------|-----|------|---------|----------|
|                                       | Obs | year | gam     | tpspline |
|                                       | 1   | 1936 | 0.82425 | 0.82424  |
|                                       | 2   | 1937 | 0.85580 | 0.85580  |
|                                       | 3   | 1938 | 0.96379 | 0.96379  |
|                                       | 4   | 1939 | 1.15046 | 1.15046  |
|                                       | 5   | 1940 | 1.31044 | 1.31044  |
|                                       | 6   | 1941 | 1.43881 | 1.43881  |
|                                       | 7   | 1942 | 1.58218 | 1.58218  |
|                                       | 8   | 1943 | 1.64382 | 1.64382  |
|                                       | 9   | 1944 | 1.60148 | 1.60148  |
|                                       | 10  | 1945 | 1.57498 | 1.57499  |

### Example 5.3. Poisson Regression Analysis of Component Reliability

In this example, the number of maintenance repairs on a complex system are modeled as realizations of Poisson random variables. The system under investigation has a large number of components, which occasionally break down and are replaced or repaired. During a four-year period, the system was observed to be in a state of steady operation, meaning that the rate of operation remained approximately constant. A monthly maintenance record is available for that period, which tracks the number of components removed for maintenance each month. The data are listed in the following statements that create a SAS data set.

```

title 'Analysis of Component Reliability';
data equip;
  input year month removals @@;
  datalines;
1987 1 2 1987 2 4 1987 3 3
1987 4 3 1987 5 3 1987 6 8
1987 7 2 1987 8 6 1987 9 3
1987 10 9 1987 11 4 1987 12 10
1988 1 4 1988 2 6 1988 3 4
1988 4 4 1988 5 3 1988 6 5
1988 7 3 1988 8 4 1988 9 5
1988 10 3 1988 11 6 1988 12 3
1989 1 2 1989 2 6 1989 3 1
1989 4 5 1989 5 5 1989 6 4
1989 7 2 1989 8 2 1989 9 2
1989 10 5 1989 11 1 1989 12 10
1990 1 3 1990 2 8 1990 3 12
1990 4 7 1990 5 3 1990 6 2
1990 7 4 1990 8 3 1990 9 0
1990 10 6 1990 11 6 1990 12 6
;
run;

```

For planning purposes, it is of interest to understand the long- and short-term trends in the maintenance needs of the system. Over the long term, it is suspected that

the quality of new components and repair work improves over time, so the number of component removals would tend to decrease from year to year. It is not known whether the robustness of the system is affected by seasonal variations in the operating environment, but this possibility is also of interest.

Because the maintenance record is in the form of counts, the number of removals are modeled as realizations of Poisson random variables. Denote by  $\lambda_{ij}$  the unobserved component removal rate for year  $i$  and month  $j$ . Since the data were recorded at regular intervals (from a system operating at a constant rate), each  $\lambda_{ij}$  is assumed to be a function of year and month only.

A preliminary two-way analysis is performed using PROC GENMOD to make broad inferences on repair trends. A log-link is specified for the model

$$\log \lambda_{ij} = \mu + \text{Year}_i + \text{MONTH}_j,$$

where  $\mu$  is a grand mean,  $\text{YEAR}_i$  is the effect of the  $i$ th year, and  $\text{MONTH}_j$  is the effect of the  $j$ th month. A CLASS statement declares the variables `year` and `month` as categorical. Type III sum of squares are requested to test whether there is an overall effect of year and/or month.

```

title2 'Two-way model';
proc genmod data=equip;
  class year month;
  model removals=year month
    / dist=Poisson link=log type3;
run;

```

**Output 5.3.1.** PROC GENMOD Listing for Type III Analysis

| Analysis of Component Reliability |    |            |            |
|-----------------------------------|----|------------|------------|
| Two-way model                     |    |            |            |
| The GENMOD Procedure              |    |            |            |
| LR Statistics For Type 3 Analysis |    |            |            |
| Source                            | DF | Chi-Square | Pr > ChiSq |
| year                              | 3  | 2.63       | 0.4527     |
| month                             | 11 | 21.12      | 0.0321     |

Output 5.3.1 displays the listed Type III statistics for the fitted model. With the test for year effects yielding a  $p$ -value of 0.4527, there is no evidence of a long-term trend in maintenance rates. Apparently, the quality of new or repaired components did not change between 1987 and 1990. However, the test for monthly trends does yield a small  $p$ -value of 0.0321, indicating that seasonal trends are just barely significant at the  $\alpha = 0.05$  level.

The Type III tests indicate that the  $YEAR_i$  term may be dropped from the model. The focus of the analysis is now on identifying the form of the underlying seasonal trend, which is a task that PROC GAM is especially suited for. PROC GAM will be used to fit both a reduced categorical model, with  $YEAR_i$  eliminated, and a nonparametric spline model. Although PROC GENMOD also has the capability to fit categorical models, as demonstrated above, PROC GAM will be used to fit both models for a better comparison.

The following PROC GAM statements specify the reduced categorical model. For this part of the analysis, a CLASS statement is again used to specify that month is a categorical variable. In the follow-up, the seasonal effect will be treated as a nonparametric function of month.

```

title2 'One-way model';
proc gam data=equip;
  class month;
  model removals=param(month)
              / dist=Poisson;
  output out=est p;
run;

```

The following statements generate a plot of the estimated seasonal trend. Note that the predicted values in the output data set correspond to the *logarithms* of the  $\lambda_{ij}$ , and so the exponential function is applied to put them on the scale of the original data. The plot is displayed in Output 5.3.2.

```

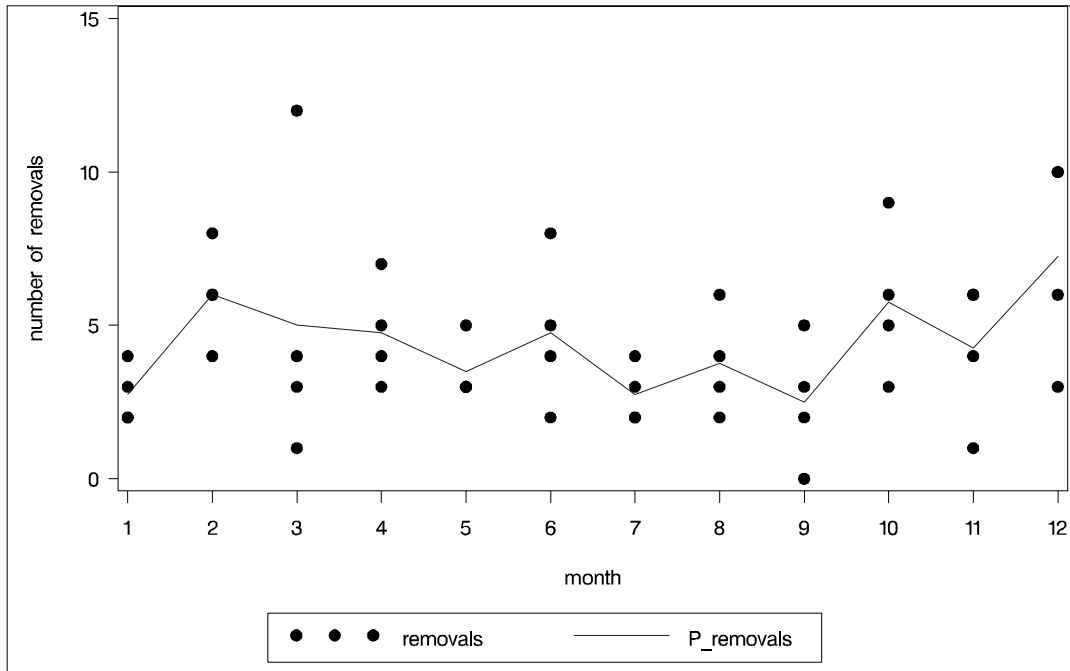
proc sort data=est out=plot;
  by month;
run;

data plot;
  set plot;
  P_removals = exp(P_removals);
run;

legend1 frame cframe=ligr cborder=black label=none
        position=center;
axis1  minor=none order=(0 to 15 by 5)
        label=(angle=90 rotate=0 "number of removals");
axis2  minor=none label=("month");
symbol1 color=black interpol=none value=dot;
symbol2 color=blue  interpol=join value=none line=1;

title;
proc gplot data=plot;
  plot removals*month=1 P_removals*month=2
        / overlay cframe=ligr legend=legend1 frame
        vaxis=axis1 haxis=axis2;
run; quit;

```

**Output 5.3.2.** Predicted Seasonal Trend from a Parametric Model Fit Using a CLASS Statement

The predicted repair rates shown in Output 5.3.2 form a jagged seasonal pattern. Ignoring the month-to-month fluctuations, which are difficult to explain and may be artifacts of random noise, the general removal rate trend starts by increasing at the beginning of the year; the trend flattens out in February and then decreases through August; it flattens out again in September and begins an increasing trend which continues throughout the rest of the year.

One advantage of nonparametric regression is its ability to highlight general trends in the data, such as those described above, and attribute local fluctuations to unexplained random noise. The nonparametric regression model used by PROC GAM specifies that the underlying removal rates  $\lambda_j$  are of the form

$$\log \lambda_j = \beta_0 + \beta_1 x_j + s(x_j)$$

where  $x_j$  is the value of month at month  $j$ ,  $\beta_1$  is a linear coefficient, and  $s$  is a nonparametric regression function.  $\beta_1$  and  $s$  define the linear and nonparametric parts, respectively, of the seasonal trend.

The following statements request that PROC GAM fit a cubic spline model to the monthly repair data. The output listing is displayed in Output 5.3.3.



```

title 'Analysis of Component Reliability';
title2 'Spline model';
proc gam data=equip;
  model removals=spline(month)
    / dist=Poisson method=gcv;
  output out=est p lclm uclm;
run;

```

The METHOD=GCV option is used to determine an appropriate level of smoothing. The keywords LCLM and UCLM in the OUTPUT statement requests that lower and upper 95% confidence bounds on each  $s(x_{ij})$  be included in the output data set.

**Output 5.3.3.** PROC GAM Listing for Cubic Spline Regression Using the METHOD=GCV Option

| Spline model                                |              |
|---------------------------------------------|--------------|
| The GAM Procedure                           |              |
| Dependent Variable: removals                |              |
| Smoothing Model Component(s): spline(month) |              |
| Summary of Input Data Set                   |              |
| Number of Observations                      | 48           |
| Number of Missing Observations              | 0            |
| Distribution                                | Poisson      |
| Link Function                               | Log          |
| Iteration Summary and Fit Statistics        |              |
| Number of local score iterations            | 5            |
| Local score convergence criterion           | 2.546171E-11 |
| Final Number of Backfitting Iterations      | 1            |
| Final Backfitting Criterion                 | 5.894286E-11 |
| The Deviance of the Final Estimate          | 56.901544117 |

**Output 5.3.4.** Model Fit Statistics

| Spline model                                |                     |                |            |                |
|---------------------------------------------|---------------------|----------------|------------|----------------|
| The GAM Procedure                           |                     |                |            |                |
| Dependent Variable: removals                |                     |                |            |                |
| Smoothing Model Component(s): spline(month) |                     |                |            |                |
| Regression Model Analysis                   |                     |                |            |                |
| Parameter Estimates                         |                     |                |            |                |
| Parameter                                   | Parameter Estimate  | Standard Error | t Value    | Pr >  t        |
| Intercept                                   | 1.34594             | 0.14509        | 9.28       | <.0001         |
| Linear(month)                               | 0.02274             | 0.01893        | 1.20       | 0.2362         |
| Smoothing Model Analysis                    |                     |                |            |                |
| Fit Summary for Smoothing Components        |                     |                |            |                |
| Component                                   | Smoothing Parameter | DF             | GCV        | Num Unique Obs |
| Spline(month)                               | 0.901512            | 2.879980       | 0.115848   | 12             |
| Smoothing Model Analysis                    |                     |                |            |                |
| Analysis of Deviance                        |                     |                |            |                |
| Source                                      | DF                  | Sum of Squares | Chi-Square | Pr > ChiSq     |
| Spline(month)                               | 2.87998             | 8.877763       | 6.8836     | 0.0693         |

Notice in the listing of Output 5.3.4 that the DF value chosen by minimizing GCV is about 2.88, which is smaller than the default value of 4. This indicates that the spline model of the seasonal trend is relatively simple. As indicated by the “Analysis of Deviance” table, it is a non-significant feature of the data: the table lists a  $p$ -value of 0.0693 for the hypothesis of no seasonal trend. Note also that the “Parameter Estimates” table lists a non-significant  $p$ -value of 0.2362 for the hypothesis of no linear factor in the seasonal trend.

Using the following code, you can plot the predicted repair rates as displayed in Output 5.3.5.

```
proc sort data=est out=plot;
  by month;
run;

data plot;
  set plot;
  P_removals = exp(P_removals);
run;

legend1 frame cframe=ligr cborder=black label=none
  position=center;
axis1 minor=none order=(0 to 15 by 5)
```

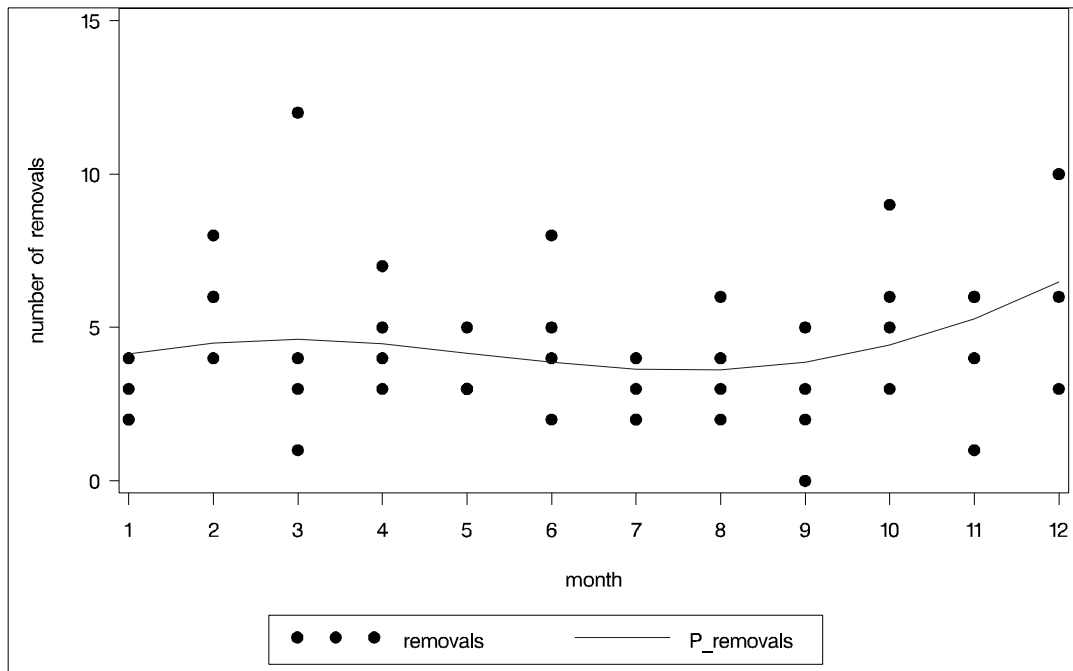
```

        label=(angle=90 rotate=0 "number of removals");
axis2 minor=none label=("month");
symbol1 color=black interpol=none value=dot;
symbol2 color=blue interpol=join value=none line=1;

title;
proc gplot data=plot;
    plot removals*month=1 P_removals*month=2
        / overlay cframe=ligr legend=legend1 frame
          vaxis=axis1 haxis=axis2;
run; quit;

```

**Output 5.3.5.** Predicted Seasonal Trend from a Cubic Spline Model



In Output 5.3.5, it is apparent that the pattern of repair rates follows the general pattern observed in Output 5.3.2. However, the plot of Output 5.3.5 is much cleaner as the month-to-month fluctuations are smoothed out to reveal the broader seasonal trend.

Both Output 5.3.2 and Output 5.3.5 show plots of the predicted dependent variable removals (after applying the exponential function) given in the output data set. You can also investigate trends in the data by examining the fit for an independent variable. For the variable month, the output statistics  $P\_month$ ,  $LCLM\_month$ , and  $UCLM\_month$  are the fitted values and confidence limits for the nonlinear component of the effect of month on removals. This component is the function  $s$  in the nonparametric regression model.

The following statements produce a plot of each  $s(x_{ij})$  along with 95% confidence bounds. The plot is displayed in Output 5.3.6.

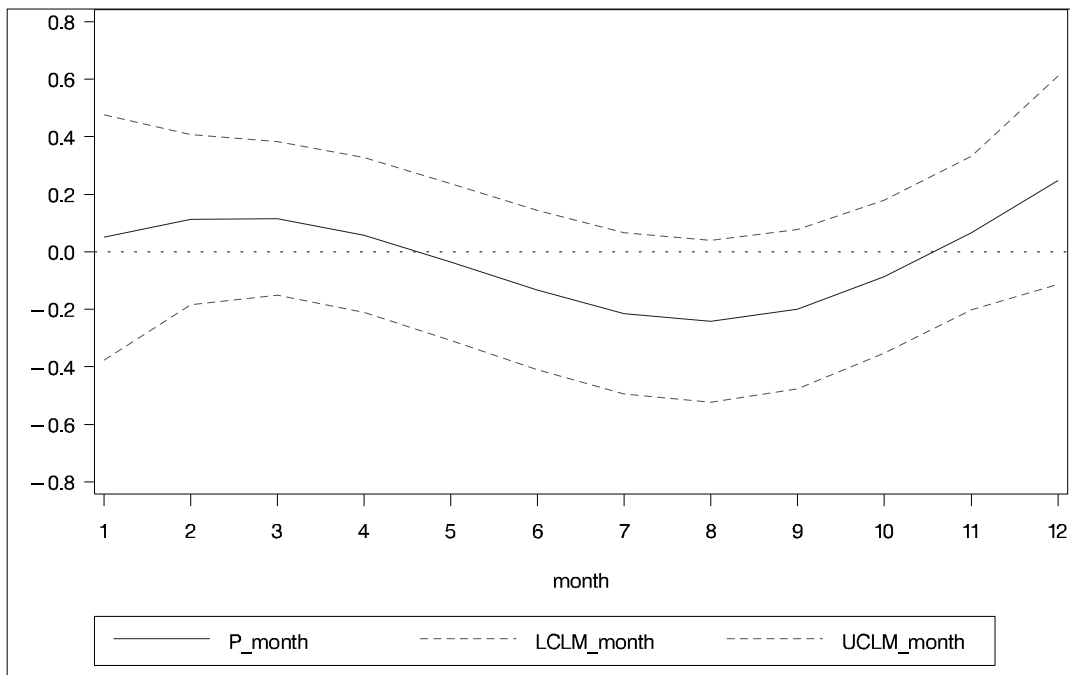
```

legend1 frame cframe=ligr cborder=black
      label=none position=center;
axis1 minor=none order=(-0.8 to 0.8 by 0.2) label=none;
axis2 minor=none label=("month");
symbol1 color=black interpol=none value=dot;
symbol2 color=blue interpol=join value=none line=1;
symbol3 color=red interpol=join value=none line=4;

proc gplot data=plot;
  plot P_month*month=2
      LCLM_month*month=3 UCLM_month*month=3
      / overlay cframe=ligr legend=legend1 frame
      vref=0 lv=34 vaxis=axis1 haxis=axis2;
run;

```

**Output 5.3.6.** Estimated Nonparametric Factor of Seasonal Trend (Solid Line), Along with 95% Confidence Bounds (Dashed Lines)



The small  $p$ -value in Output 5.3.2 of 0.0321 for the hypothesis of no seasonal trend indicates that the data exhibit significant seasonal structure. However, Output 5.3.6 is a graphical illustration of a degree of indistinctness in that structure. For instance, the horizontal reference line at zero is entirely within the 95% confidence band, that is, the estimated nonlinear part of the trend is relatively flat. Thus, despite evidence of seasonality based on the parametric model, it is difficult to narrow down its significant effects to a specific part of the year.

## Example 5.4. Comparing PROC GAM with PROC LOESS

In an analysis of simulated data from a hypothetical chemistry experiment, additive nonparametric regression performed by PROC GAM is compared to the unrestricted multidimensional procedure of PROC LOESS.

In each repetition of the experiment, a catalyst is added to a chemical solution, thereby inducing synthesis of a new material. The data are measurements of the temperature of the solution, the amount of catalyst added, and the yield of the chemical reaction. The following code reads and plots the raw data.

```
data ExperimentA;
  format Temperature f4.0 Catalyst f6.3 Yield f8.3;
  input Temperature Catalyst Yield @@;
  datalines;
80 0.005 6.039 80 0.010 4.719 80 0.015 6.301
80 0.020 4.558 80 0.025 5.917 80 0.030 4.365
80 0.035 6.540 80 0.040 5.063 80 0.045 4.668
80 0.050 7.641 80 0.055 6.736 80 0.060 7.255
80 0.065 5.515 80 0.070 5.260 80 0.075 4.813
80 0.080 4.465 90 0.005 4.540 90 0.010 3.553
90 0.015 5.611 90 0.020 4.586 90 0.025 6.503
90 0.030 4.671 90 0.035 4.919 90 0.040 6.536
90 0.045 4.799 90 0.050 6.002 90 0.055 6.988
90 0.060 6.206 90 0.065 5.193 90 0.070 5.783
90 0.075 6.482 90 0.080 5.222 100 0.005 5.042
100 0.010 5.551 100 0.015 4.804 100 0.020 5.313
100 0.025 4.957 100 0.030 6.177 100 0.035 5.433
100 0.040 6.139 100 0.045 6.217 100 0.050 6.498
100 0.055 7.037 100 0.060 5.589 100 0.065 5.593
100 0.070 7.438 100 0.075 4.794 100 0.080 3.692
110 0.005 6.005 110 0.010 5.493 110 0.015 5.107
110 0.020 5.511 110 0.025 5.692 110 0.030 5.969
110 0.035 6.244 110 0.040 7.364 110 0.045 6.412
110 0.050 6.928 110 0.055 6.814 110 0.060 8.071
110 0.065 6.038 110 0.070 6.295 110 0.075 4.308
110 0.080 7.020 120 0.005 5.409 120 0.010 7.009
120 0.015 6.160 120 0.020 7.408 120 0.025 7.123
120 0.030 7.009 120 0.035 7.708 120 0.040 5.278
120 0.045 8.111 120 0.050 8.547 120 0.055 8.279
120 0.060 8.736 120 0.065 6.988 120 0.070 6.283
120 0.075 7.367 120 0.080 6.579 130 0.005 7.629
130 0.010 7.171 130 0.015 5.997 130 0.020 6.587
130 0.025 7.335 130 0.030 7.209 130 0.035 8.259
130 0.040 6.530 130 0.045 8.400 130 0.050 7.218
130 0.055 9.167 130 0.060 9.082 130 0.065 7.680
130 0.070 7.139 130 0.075 7.275 130 0.080 7.544
140 0.005 4.860 140 0.010 5.932 140 0.015 3.685
140 0.020 5.581 140 0.025 4.935 140 0.030 5.197
140 0.035 5.559 140 0.040 4.836 140 0.045 5.795
140 0.050 5.524 140 0.055 7.736 140 0.060 5.628
140 0.065 6.644 140 0.070 3.785 140 0.075 4.853
140 0.080 6.006
;
```

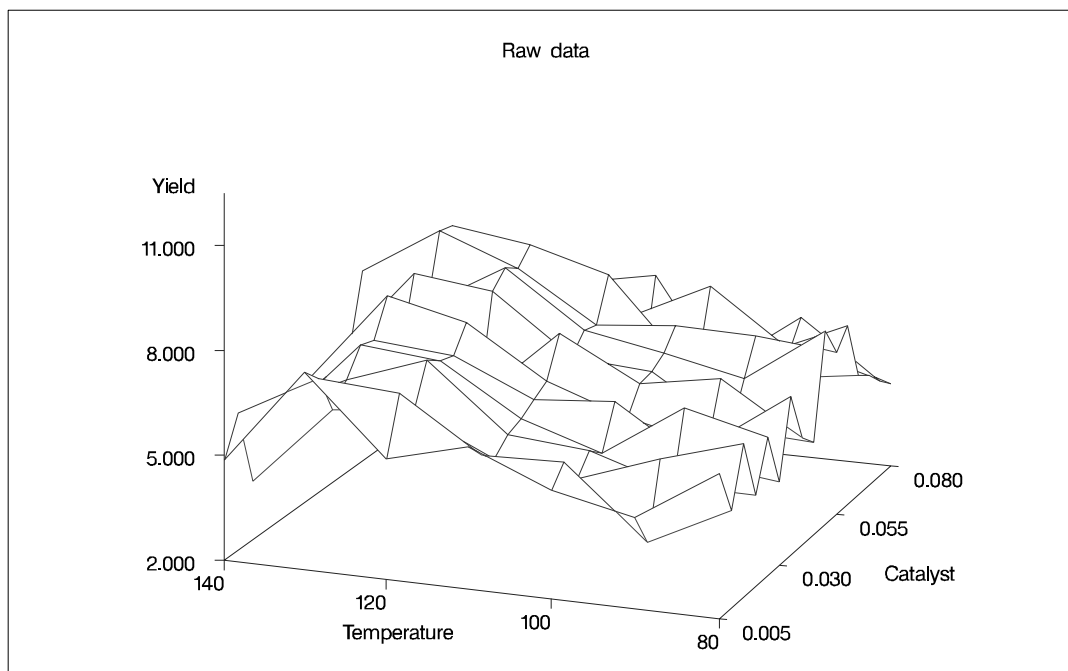
```

title2 'Raw data';
proc g3d data=ExperimentA;
  plot Temperature*Catalyst=Yield
        / zmin=2 zmax=11;
run;

```

The plot is displayed in Output 5.4.1.

**Output 5.4.1.** Surface Plot of Yield by Temperature and Amount of Catalyst



A surface fitted to the plot of Output 5.4.1 by PROC LOESS will be of a very general (and flexible) type, since the procedure requires only weak assumptions about the structure of the dependencies among the data. PROC GAM, on the other hand, makes stronger structural assumptions by restricting the fitted surface to an additive form. These differences will be demonstrated in this example.

The following code requests that both PROC LOESS and PROC GAM fit surfaces to the data.

```

ods output OutputStatistics=PredLOESS;
proc loess data=ExperimentA;
  model Yield = Temperature Catalyst
        / scale=sd degree=2 select=gcv;
run;
ods output close;

proc gam data=ExperimentA;
  model Yield = loess(Temperature) loess(Catalyst)
        / method=gcv;
  output out=PredGAM;
run;

```

In both cases the smoothing parameter was chosen as the value that minimizes GCV. This is performed automatically by PROC LOESS and PROC GAM.

The following code generates plots of the predicted yields, which are displayed in Output 5.4.2.

```

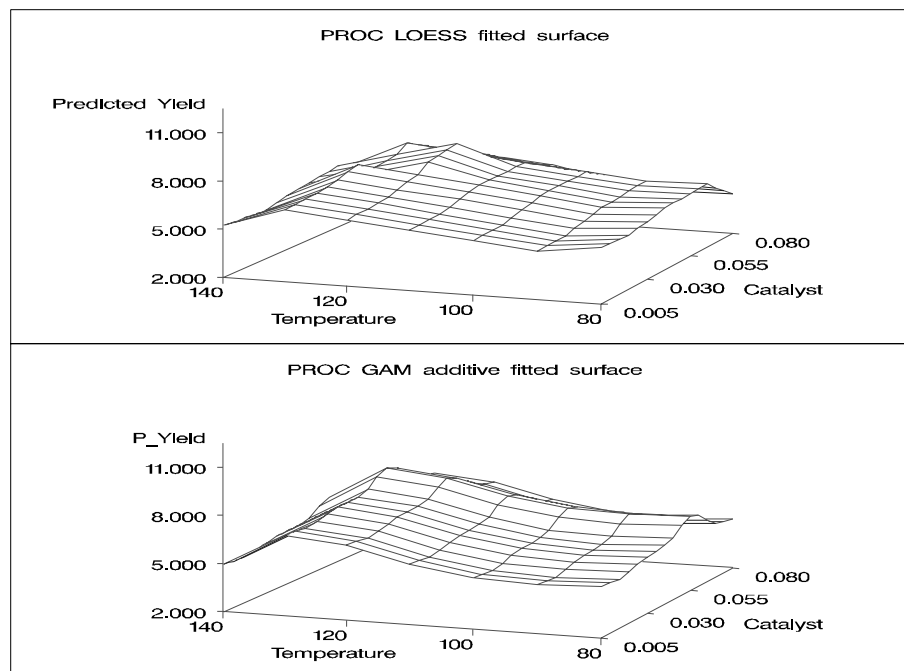
title2 'PROC LOESS fitted surface';
proc g3d data=PredLOESS;
  format pred f6.3;
  plot Temperature*Catalyst=pred
        / name='LOESSA' zmin=2 zmax=11;
run;

title2 'PROC GAM additive fitted surface';
proc g3d data=PredGAM;
  format P_Yield f6.3;
  plot Temperature*Catalyst=P_Yield
        / name='GAMA' zmin=2 zmax=11;
run;

options display;
proc greplay nofs tc=sashelp.templt template=v2;
  igout=gseg;
  treplay 1:loessa 2:gama;
run; quit;

```

**Output 5.4.2.** Fitted Regression Surfaces



Though both PROC LOESS and PROC GAM use the statistical technique loess, it is apparent from Output 5.4.2 that the manner in which it is applied is very different.

By smoothing out the data in local neighborhoods, PROC LOESS essentially fits a surface to the data in pieces, one neighborhood at a time. The local regions are treated independently, so separate areas of the fitted surface are only weakly related. PROC GAM imposes structure by requiring that cross-sections of the fitted surface always have the same shape, thereby relating regions that have a common value of the same individual regressor variable. Under that restriction, the loess technique need not be applied to the entire multidimensional scatter plot, but only to one-dimensional cross-sections of the data.

The advantage of using additive model fitting is that its statistical power is directed toward univariate smoothing, and so it is able to discern the finer details of any underlying structure in the data. Regression data may be very sparse when viewed in the context of multidimensional space, even when every individual set of regressor values densely covers its range. This is the familiar curse of dimensionality. Sparse data greatly restricts the effectiveness of nonparametric procedures, but additive model fitting, when appropriate, is one way to overcome this limitation.

To examine these properties, plots of cross-sections of unrestricted (PROC LOESS) and additive (PROC GAM) fitted surfaces for the variable *Catalyst* are generated by the following code. The code for creating the cross-section plots and overlaying them is somewhat complicated, so a macro %XPlot is employed to make it easy to create this plot for the results of each procedure.

```
axis1 minor=none order=(2 to 11 by 2)
      label=(angle=90 rotate=0 "Predicted Yield");
axis2 minor=none order=(0.005 to 0.080 by 0.025)
      label=("Catalyst");
symbol1 color=blue interpol=join value=none
      line=1 width=1;

%macro XPLOT(proc=,name=);

proc sort data=Pred&proc;
  by Catalyst Temperature;
run;

data PredX&proc;
  keep Pred80 Pred90 Pred100 Pred110 Pred120 Pred130
      Pred140 Catalyst;
  array xPred{8:14} Pred80 Pred90 Pred100 Pred110
      Pred120 Pred130 Pred140;
  retain Pred80 Pred90 Pred100 Pred110 Pred120
      Pred130 Pred140;
  set Pred&proc;
  %if &proc=LOESS %then %do;
    xPred{Temperature/10} = pred;
  %end;
  %else %if &proc=GAM %then %do;
    xPred{Temperature/10} = P_Yield;
  %end;
  if abs(Temperature-140)<1 then output;
run;
```



```

proc gplot data=PredX&proc;
  plot Pred140*Catalyst=1 Pred130*Catalyst=1
       Pred120*Catalyst=1 Pred110*Catalyst=1
       Pred100*Catalyst=1 Pred90*Catalyst=1
       Pred80*Catalyst=1
       / overlay cframe=ligr name=&name
       vaxis=axis1 haxis=axis2;
run; quit;

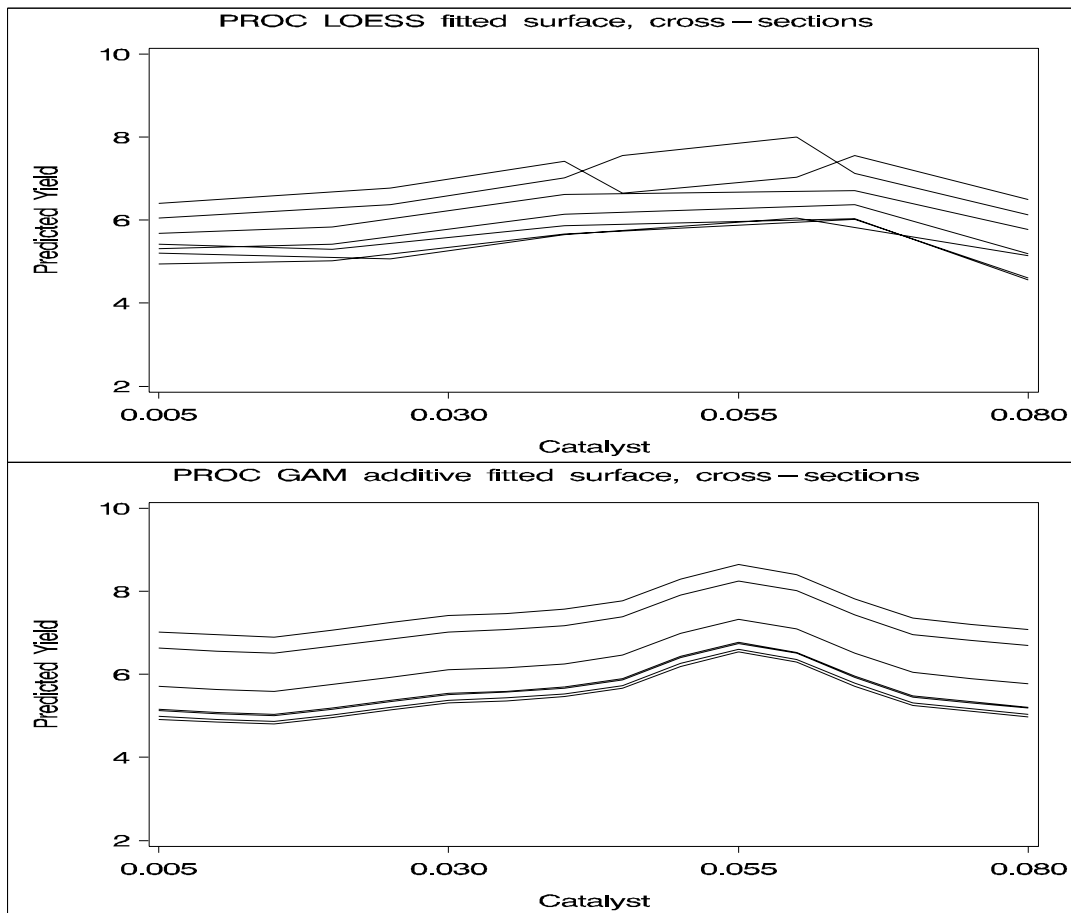
%mend;

title;
title2 'PROC LOESS fitted surface, cross-sections';
%XPLOT(proc=LOESS,name='XLOESSA');
title2 'PROC GAM additive fitted surface, cross-sections';
%XPLOT(proc=GAM,name='XGAMA');

goptions display;
proc greplay nofs tc=sashelp.templt template=v2;
  igout=gseg;
  treplay 1:xloessa 2:xgama;
run; quit;

```

The plots are displayed in Output 5.4.3.

**Output 5.4.3.** Cross-sections of Fitted Regression Surfaces

Notice that the graphs in the top panel (PROC LOESS) of Output 5.4.3 have varying shapes, while every graph in the bottom panel (PROC GAM) is the same curve shifted vertically. This illustrates precisely the kind of structural differences that distinguish additive models. A second important comparison to make in Output 5.4.2 and Output 5.4.3 is the level of detail in the fitted regression surfaces. Cross-sections of the PROC LOESS surface are rather flat, but those of the additive surface have a clear shape. In particular, the ridge near Catalyst=0.055 is only vaguely evident in the PROC LOESS surface, but it is plainly revealed by the additive procedure.

For an example of a situation where unrestricted multidimensional fitting is preferred over additive regression, consider the following simulated data from a similar experiment. The following code creates another SAS data set and plot.

```
data ExperimentB;
  format Temperature f4.0 Catalyst f6.3 Yield f8.3;
  input Temperature Catalyst Yield @@;
  datalines;
  80 0.005 9.115 80 0.010 9.275 80 0.015 9.160
  80 0.020 7.065 80 0.025 6.054 80 0.030 4.899
  80 0.035 4.504 80 0.040 4.238 80 0.045 3.232
  80 0.050 3.135 80 0.055 5.100 80 0.060 4.802
```

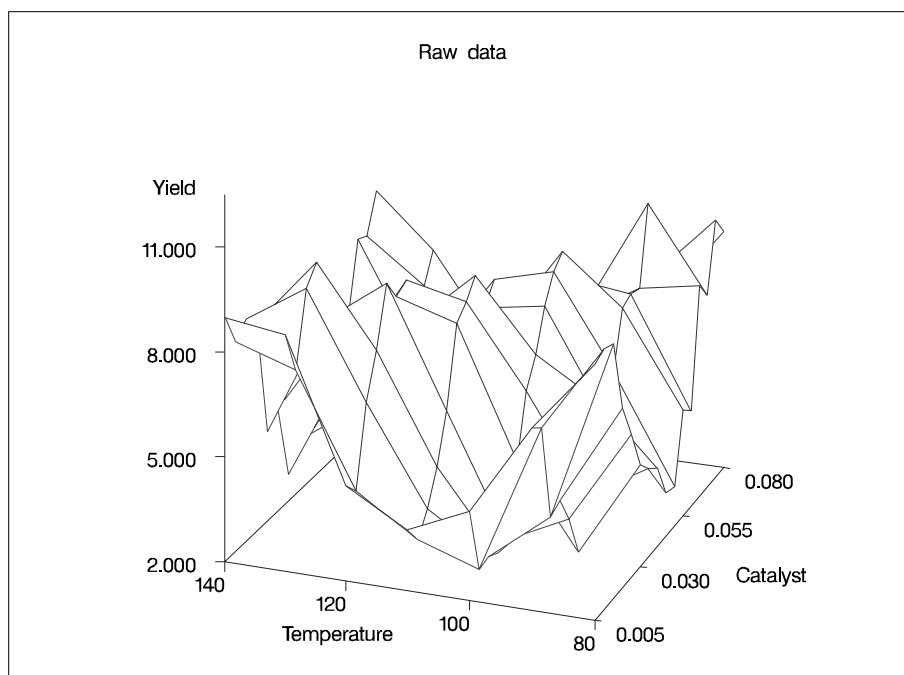
```

80 0.065 8.218 80 0.070 7.679 80 0.075 9.669
80 0.080 9.071 90 0.005 7.085 90 0.010 6.814
90 0.015 4.009 90 0.020 4.199 90 0.025 3.377
90 0.030 2.141 90 0.035 3.500 90 0.040 5.967
90 0.045 5.268 90 0.050 6.238 90 0.055 7.847
90 0.060 7.992 90 0.065 7.904 90 0.070 10.184
90 0.075 7.914 90 0.080 6.842 100 0.005 4.497
100 0.010 2.565 100 0.015 2.637 100 0.020 2.436
100 0.025 2.525 100 0.030 4.474 100 0.035 6.238
100 0.040 7.029 100 0.045 8.183 100 0.050 8.939
100 0.055 9.283 100 0.060 8.246 100 0.065 6.927
100 0.070 7.062 100 0.075 5.615 100 0.080 4.687
110 0.005 3.706 110 0.010 3.154 110 0.015 3.726
110 0.020 4.634 110 0.025 5.970 110 0.030 8.219
110 0.035 8.590 110 0.040 9.097 110 0.045 7.887
110 0.050 8.480 110 0.055 6.818 110 0.060 7.666
110 0.065 4.375 110 0.070 3.994 110 0.075 3.630
110 0.080 2.685 120 0.005 4.697 120 0.010 4.268
120 0.015 6.507 120 0.020 7.747 120 0.025 9.412
120 0.030 8.761 120 0.035 8.997 120 0.040 7.538
120 0.045 7.003 120 0.050 6.010 120 0.055 3.886
120 0.060 4.897 120 0.065 2.562 120 0.070 2.714
120 0.075 3.141 120 0.080 5.081 130 0.005 8.729
130 0.010 7.460 130 0.015 9.549 130 0.020 10.049
130 0.025 8.131 130 0.030 7.553 130 0.035 6.191
130 0.040 6.272 130 0.045 4.649 130 0.050 3.884
130 0.055 2.522 130 0.060 4.366 130 0.065 3.272
130 0.070 4.906 130 0.075 6.538 130 0.080 7.380
140 0.005 8.991 140 0.010 8.029 140 0.015 8.417
140 0.020 8.049 140 0.025 4.608 140 0.030 5.025
140 0.035 2.795 140 0.040 3.123 140 0.045 3.407
140 0.050 4.183 140 0.055 3.750 140 0.060 6.316
140 0.065 5.799 140 0.070 7.992 140 0.075 7.835
140 0.080 8.985
;
run;

title2 'Raw data';
proc g3d data=ExperimentB;
  plot Temperature*Catalyst=Yield
    / zmin=2 zmax=11;
run;

```

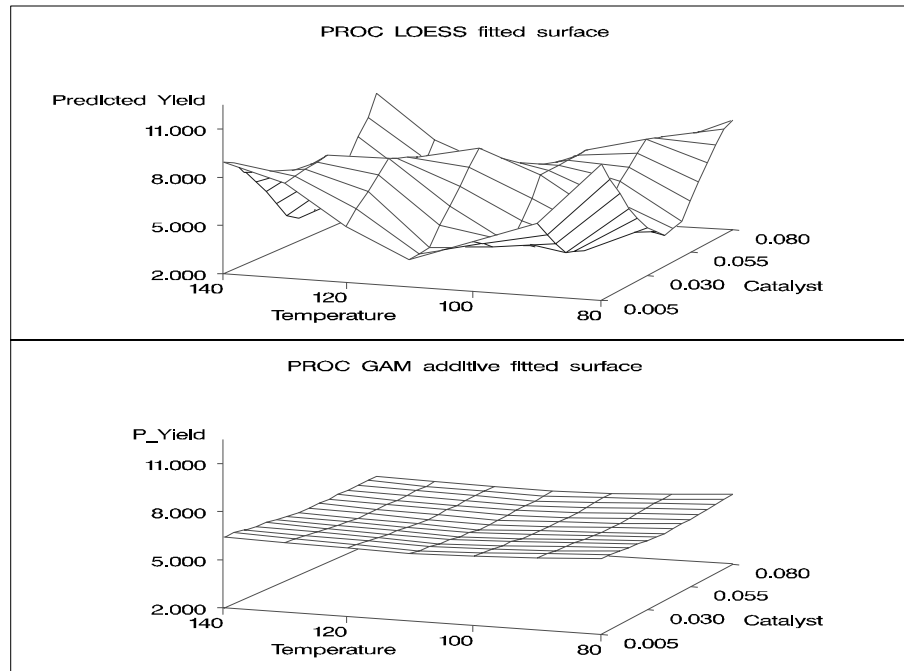
A plot of the raw data is displayed in Output 5.4.4.

**Output 5.4.4.** Raw Data from Experiment B

Though the surface displayed in Output 5.4.4 is quite jagged, a distinct feature of the plot is a large ridge that runs diagonally across its surface. One would expect that the ridge would appear in the fitted regression surface of an appropriate nonparametric procedure. Nevertheless, between PROC LOESS and PROC GAM, only PROC LOESS is able to capture this significant feature.

The SAS code for fitting the new data is essentially the same as that for the data set from the first experiment. Similar to Output 5.4.2, multivariate and additive fitted surfaces for these data are displayed in Output 5.4.5.

**Output 5.4.5.** Fitted Regression Surfaces



It is clear from Output 5.4.5 that the results of PROC LOESS and PROC GAM are completely different. While the plot in the top panel resembles the raw data plot in Output 5.4.4, the plot in the bottom panel is essentially featureless.

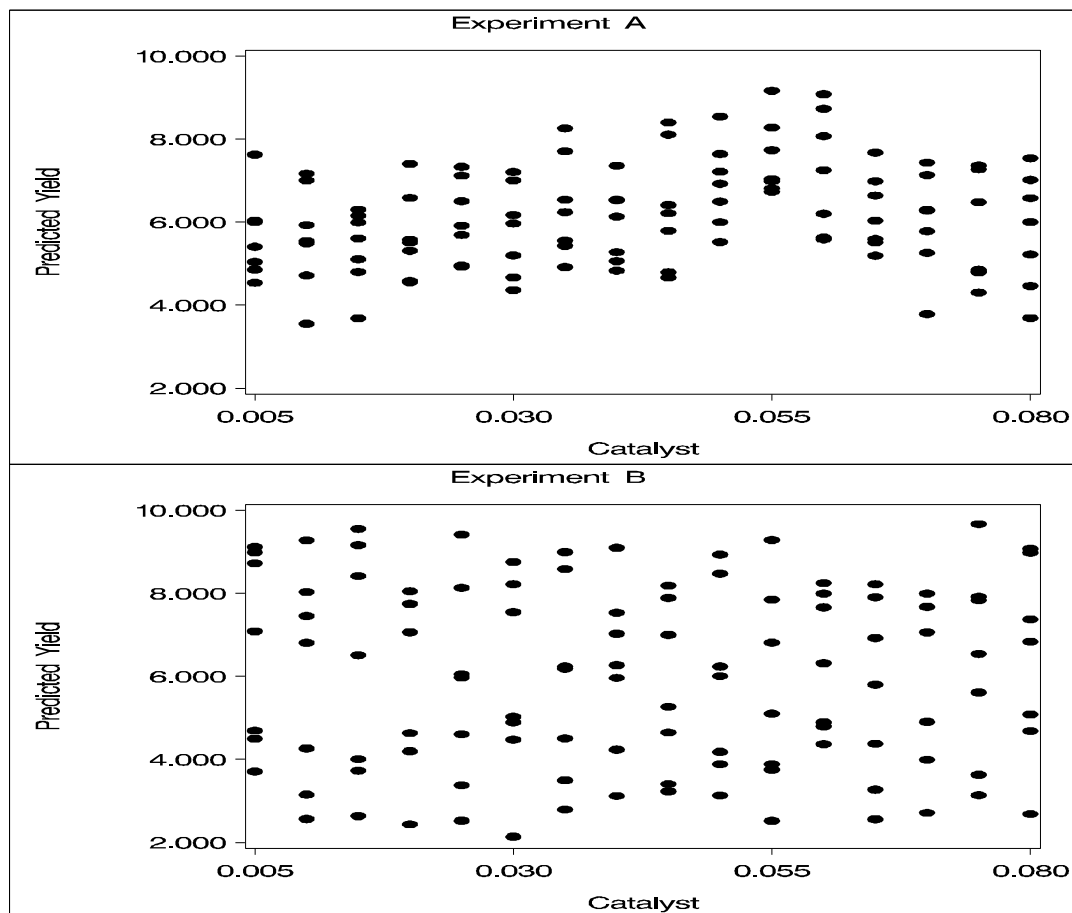
To understand what is happening, compare the scatter plots of Yield by Catalyst for the two data sets in this example. These are generated by the following code and displayed in Output 5.4.6.

```
axis1 minor=none order=(2 to 11 by 2)
      label=(angle=90 rotate=0 "Predicted Yield");
axis2 minor=none order=(0.005 to 0.080 by 0.025) label=("Catalyst");
symbol2 c=yellow v=dot i=none;

title2 'Experiment A';
proc gplot data=ExperimentA;
  plot Yield*Catalyst=2
    / cframe=ligr name='ExptA'
    vaxis=axis1 haxis=axis2;
run; quit;

title2 'Experiment B';
proc gplot data=ExperimentB;
  plot Yield*Catalyst=2
    / cframe=ligr name='ExptB'
    vaxis=axis1 haxis=axis2;
run; quit;

proc greplay nofs tc=sashelp.templt template=v2;
  igout=gseg;
  treplay 1:expta 2:exptb;
run; quit;
```

**Output 5.4.6.** Scatterplots of Yield by Catalyst

The top panel of Output 5.4.6 hints at the same kind of structure exhibited in the fitted cross-sections of Output 5.4.3. In PROC GAM, the additive model component corresponding to Catalyst is fit to a similar scatter plot, with the partial residuals computed in the backfitting algorithm, so it is able to capture the trend seen here. In contrast, when the second data set is viewed from the perspective of Output 5.4.4, the diagonal ridge apparent in Output 5.4.4 is washed out, and no clear structure shows up in the scatter plot. As a result, the additive model fit produced by PROC GAM is relatively featureless.

---

## References

- Allen, D.M. (1974), "The Relationship Between Variable Selection and Data Augmentation and a Method of Prediction," *Technometrics*, 16, 125–127.
- Bell, D., Walker, J., O'Connor, G., Orrel, J. and Tibshirani, R. (1989), "Spinal Deformation Following Multi-Level Thoracic and Lumbar Laminectomy in Children." Submitted for publication.
- Cleveland, W.S., Devlin, S.J., and Grosse, E. (1988), "Regression by Local Fitting," *Journal of Econometrics*, 37, 87–114.

- Duchon, J. (1976), "Fonctions-Spline et Esperances Conditionnelles de Champs Gaussiens," *Ann. Sci. Univ. Clermont Ferrand II Math.*, 14, 19–27.
- Duchon, J. (1977), "Splines Minimizing Rotation-Invariant Semi-Norms in Sobolev Spaces," in *Constructive Theory of Functions of Several Variables*, eds. W. Schempp and K. Zeller, New York: Springer-Verlag, 85–100.
- Friedman, J.H. and Stuetzle, W. (1981), "Projection Pursuit Regression," *Journal of the American Statistical Association*, 76, 817–823.
- Hastie, T.J. and Tibshirani, R.J. (1990), *Generalized Additive Models*, New York: Chapman and Hall.
- Houghton, A.N., Flannery, J., and Viola, M.V. (1980), "Malignant Melanoma in Connecticut and Denmark," *International Journal of Cancer*, 25, 95–104.
- Meinguet, J. (1979), "Multivariate Interpolation at Arbitrary Points Made Simple," *Journal of Applied Mathematics and Physics (ZAMP)*, 30, 292–304.
- Nelder, J.A. and Wedderburn, R.W.M. (1972), "Generalized Linear Models," *Journal of the Royal Statistical Society, Series A*, 135, 370–384.
- SAS Institute Inc. (1999), *SAS Language Reference: Concepts, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS Language Reference: Dictionary, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS Procedures Guide, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS/STAT User's Guide, Version 8*, Cary, NC: SAS Institute Inc.
- Sockett, E.B., Daneman, D., Clarson, C., and Ehrich, R.M. (1987), "Factors Affecting and Patterns of Residual Insulin Secretion During the First Year of Type I (Insulin Dependent) Diabetes Mellitus in Children," *Diabet*, 30, 453–459.
- Stone, C.J. (1985), "Additive Regression and Other Nonparametric Models," *Annals of Statistics*, 13, 689–705.
- Wahba, G. (1983), "Bayesian 'Confidence Intervals' for the Cross Validated Smoothing Spline," *Journal of the Royal Statistical Society, Series B*, 45, 133–150.
- Wahba, G. (1990), *Spline Models for Observational Data*, Philadelphia: Society for Industrial and Applied Mathematics.
- Wahba, G. and Wendelberger, J. (1980), "Some New Mathematical Methods for Variational Objective Analysis Using Splines and Cross Validation," *Monthly Weather Review*, 108, 1122–1145.





# Chapter 6

## The LIFEREG Procedure

### Chapter Table of Contents

---

|                                                                                      |     |
|--------------------------------------------------------------------------------------|-----|
| <b>OVERVIEW</b> . . . . .                                                            | 77  |
| <b>SYNTAX</b> . . . . .                                                              | 77  |
| PROC LIFEREG Statement . . . . .                                                     | 77  |
| INSET Statement . . . . .                                                            | 78  |
| MODEL Statement . . . . .                                                            | 80  |
| OUTPUT Statement . . . . .                                                           | 80  |
| PROBPLOT Statement . . . . .                                                         | 81  |
| <b>DETAILS</b> . . . . .                                                             | 87  |
| Confidence Intervals . . . . .                                                       | 87  |
| Probability Plotting . . . . .                                                       | 88  |
| INEST= Data Set . . . . .                                                            | 92  |
| XDATA= Data Set . . . . .                                                            | 93  |
| <b>EXAMPLES</b> . . . . .                                                            | 94  |
| Example 6.1 Analysis of Arbitrarily Censored Data with Interaction Effects . . . . . | 94  |
| Example 6.2 Probability Plotting–Right Censoring . . . . .                           | 98  |
| Example 6.3 Probability Plotting–Arbitrarily Censoring . . . . .                     | 100 |
| <b>REFERENCES</b> . . . . .                                                          | 103 |



## Chapter 6

# The LIFEREG Procedure

---

### Overview

You can specify interaction terms among any of the specified explanatory variables in the MODEL statement in the same way as in the GLM procedure. Because of the more complicated models you can specify, only a single MODEL statement is now allowed for each PROC LIFEREG statement. If more than one MODEL statement is specified, only the last one is used.

You can use the new PROBLOT statement to construct probability plots for complete or censored data. You can display the fitted distribution and confidence intervals along with nonparametric estimates of the CDF on the probability plots. In addition, you can control graphical features such as plot layout, colors, plotting symbols, line styles, and the text fonts used in the probability plot.

You can use the new XDATA= data set to provide the values for effects when the predicted probability and quantile and their confidence limits are computed. By default, the LIFEREG procedure uses the overall mean for effects containing continuous a variable (or variables) and the highest level of a single classification variable as the reference level.

You can use the new INEST= data set to provide the initial values for parameters in the model. The structure of this data set is similar to that of the OUTEST= data set. This data set will overwrite the original INITIAL= option in the MODEL statement.

The OUTEST= data set is extended to all models whether there is a CLASS statement or not. The OUTEST= data set does not include the fixed scale nor the fixed shape parameter.

You can now obtain standardized and Cox-Snell residuals in the OUPUT= data set.

---

### Syntax

---

#### PROC LIFEREG Statement

```
PROC LIFEREG < options > ;
```

**GOUT=***graphics-catalog*

specifies a graphics catalog in which to save graphics output.

**INEST=** *SAS-data-set*

specifies an input SAS data set that contains initial estimates for all the parameters in

the model. See the section “INSET= Data Set” on page 92 for a detailed description of the contents of the INSET= data set.

**NAMELEN=*n***

specifies the length of effect names in tables and output data sets to be *n* characters, where *n* is a value between 20 and 200. The default length is 20 characters.

**XDATA= *SAS-data-set***

specifies an input SAS data set that contains values for all the independent variables in the MODEL statement and variables in the CLASS statement for probability plotting. If there are covariates specified in a MODEL statement and a probability plot is requested with a PROBPLOT statement, you specify fixed values for the effects in the MODEL statement with the XDATA= data set. See the section “XDATA= Data Set” on page 93 for a detailed description of the contents of the XDATA= data set.

---

## INSET Statement

**INSET** < *keyword-list* > < / *options* >;

The box or table of summary information produced on plots made with the PROBPLOT statement is called an *inset*. You can use the INSET statement to customize the information that is printed in the inset box as well as to customize the appearance of the inset box. To supply the information that is displayed in the inset box, you specify *keywords* corresponding to the information that you want shown. For example, the following statements produce a probability plot with the number of observations, the number of right-censored observations, the name of the distribution, and the estimated Weibull shape parameter in the inset.

```
proc lifereg data=epidemic;
  model life = dose / dist = weibull;
  probplot ;
  inset nobs right dist shape;
run;
```

By default, inset entries are identified with appropriate labels. However, you can provide a customized label by specifying the *keyword* for that entry followed by the equal sign (=) and the label in quotes. For example, the following INSET statement produces an inset containing the number of observations and the name of the distribution, labeled “Sample Size” and “Distribution” in the inset.

```
inset nobs='Sample Size' dist='Distribution';
```

If you specify a keyword that does not apply to the plot you are creating, then the keyword is ignored.

If you specify more than one INSET statement, only the first one is used.

The following table lists keywords available in the INSET statement to display summary statistics, distribution parameters, and distribution fitting information.

**Table 6.1.** INSET Statement Keywords

|            |                                                     |
|------------|-----------------------------------------------------|
| CONFIDENCE | confidence coefficient for all confidence intervals |
| DIST       | name of the distribution                            |
| INTERVAL   | number of interval-censored observations            |
| LEFT       | number of left-censored observations                |
| NOBS       | number of observations                              |
| NMISS      | number of observations with missing values          |
| RIGHT      | number of right-censored observations               |
| SCALE      | value of the scale parameter                        |
| SHAPE      | value of the shape parameter                        |
| UNCENSORED | number of uncensored observations                   |

The following *options* control the appearance of the box. All *options* are specified after the slash (/) in the INSET statement.

**CFILL=***color*

specifies the color for the filling box.

**CFILLH=***color*

specifies the color for the filling box header.

**CFRAME=***color*

specifies the color for the frame.

**CHEADER=***color*

specifies the color for text in the header.

**CTEXT=***color*

specifies the color for the text.

**FONT=***font*

specifies the software font for the text.

**HEIGHT=***value*

specifies the height of the text.

**HEADER=**'*quoted string*'

specifies text for the header or box title.

**NOFRAME**

omits the frame around the box.

**POS=** *value* <**DATA** | **PERCENT**>

determines the position of the inset. The *value* can be a compass point (N, NE, E, SE, S, SW, W, NW) or a pair of coordinates (x, y) enclosed in parentheses. The coordinates can be specified in screen percent units or axis data units. The default is screen percent units.

**REFPOINT=** *name*

specifies the reference point for an inset that is positioned by a pair of coordinates

with the POS= option. You use the REFPOINT= option in conjunction with the POS= coordinates. The REFPOINT= option specifies which corner of the inset frame you have specified with coordinates (x, y), and it can take the value of BR (bottom right), BL (bottom left), TR (top right), or TL (top left). The default is REFPOINT=BL. If the inset position is specified as a compass point, then the REFPOINT= option is ignored.

---

## MODEL Statement

*<label:> MODEL response<\*censor(list)>=effects < / options > ;*

*<label:> MODEL (lower,upper)=effects < / options > ;*

*<label:> MODEL events/trials=effects < / options > ;*

Only a single MODEL statement can be used with one invocation of the LIFEREG procedure. If multiple MODEL statements are present, only the last is used. The optional *label* is used to label the model estimates in the output SAS data set and OUTEST= data set.

### ALPHA=value

sets the significance level for the confidence intervals for regression parameters and estimated survival probabilities. The value must be between 0 and 1. By default, ALPHA = 0.05.

### CONVERGE=value

The default is changed to CONVERGE=1E-8.

---

## OUTPUT Statement

**OUTPUT** <OUT=SAS-data-set> keyword=name <...keyword=name> ;

Two new keywords are available.

CRESIDUAL | CRES specifies a variable to contain the Cox-Snell residuals

$$-\log(S(w_i))$$

where  $S$  is the standard survival function and

$$w_i = \frac{t_i - \mathbf{x}_i' \mathbf{b}}{\sigma}$$

If the response variable in the corresponding model statement is binomial, then the residuals are not computed, and this variable contains missing values.

SRESIDUAL | SRES specifies a variable to contain the standardized residuals

$$\frac{t_i - \mathbf{x}_i' \mathbf{b}}{\sigma}$$

If the response variable in the corresponding model statement is binomial, then the residuals are not computed, and this variable contains missing values.

---

## PROBPLOT Statement

**PROBPLOT | PLOT** < / options > ;

You can use the PROBPLOT statement to create a probability plot from lifetime data. The data can be uncensored, right-censored, or arbitrarily censored. You can specify any number of PROBPLOT statements after a MODEL statement. The syntax used for the response in the MODEL statement determines the type of censoring assumed in creating the probability plot. The model fit with the MODEL statement is plotted along with the data. If there are covariates in the model, they are set to constant values specified in the XDATA= data set when creating the probability plot. If no XDATA= data set is specified, continuous variables are set to their overall mean values and categorical variables specified in the CLASS statement are set to their highest levels.

You can specify the following options to control the content, layout, and appearance of a probability plot.

**ANNOTATE=SAS-data-set**

**ANNO=SAS-data-set**

specifies an ANNOTATE data set, as described in *SAS/GRAPH Software: Reference*, that enables you to add features to the probability plot. The ANNOTATE= data set you specify in the PROBPLOT statement is used for all plots created by the statement.

**CAXIS=color**

**CAXES=color**

specifies the color used for the axes and tick marks. This option overrides any COLOR= specifications in an AXIS statement. The default is the first color in the device color list.

**CCENSOE=color**

specifies the color for filling the censor plot area. The default is the first color in the device color list.

**CENBIN**

plots censored data as frequency counts (rounding-off for non-integer frequency) rather than as individual points.

**CENCOLOR=color**

specifies the color for the censor symbol. The default is the first color in the device color list.

**CENSYMBOL=***symbol* | (*symbol list*)

specifies symbols for censored values. The *symbol* is one of the symbol names (plus, star, square, diamond, triangle, hash, paw, point, dot, and circle) or a letter (A–Z). If you do not specify the CENSYMBOL= option, the symbol used for censored values is the same as for failures.

**CFIT=***color*

specifies the color for the fitted probability line and confidence curves. The default is the first color in the device color list.

**CFRAME=***color***CFR=***color*

specifies the color for the area enclosed by the axes and frame. This area is not shaded by default.

**CGRID=***color*

specifies the color for grid lines. The default is the first color in the device color list.

**CHREF=***color***CH=***color*

specifies the color for lines requested by the HREF= option. The default is the first color in the device color list.

**CTEXT=***color*

specifies the color for tick mark values and axis labels. The default is the color specified for the CTEXT= option in the most recent GOPTIONS statement.

**CVREF=***color***CV=***color*

specifies the color for lines requested by the VREF= option. The default is the first color in the device color list.

**DESCRIPTION=**'*string*'**DES=**'*string*'

specifies a description, up to 40 characters, that appears in the PROC GREPLAY master menu. The default is the variable name.

**FONT=***font*

specifies a software font for reference line and axis labels. You can also specify fonts for axis labels in an AXIS statement. The FONT= font takes precedence over the FTEXT= font specified in the most recent GOPTIONS statement. Hardware characters are used by default.

**HCL**

computes and draws confidence limits for the predicted probabilities in the horizontal direction.

**HEIGHT=***value*

specifies the height of text used outside framed areas. The default value is 3.846 (in percentage).

**HLOWER=***value*

specifies the lower limit on the lifetime axis scale. The HLOWER= option specifies



*value* as the lower lifetime axis tick mark. The tick mark interval and the upper axis limit are determined automatically.

**HOFFSET=***value*

specifies the offset for the horizontal axis. The default value is 1.

**HUPPER=***value*

specifies *value* as the upper lifetime axis tick mark. The tick mark interval and the lower axis limit are determined automatically.

**HREF < (INTERSECT) > =***value-list*

requests reference lines perpendicular to the horizontal axis. If (INTERSECT) is specified, a second reference line perpendicular to the vertical axis is drawn that intersects the fit line at the same point as the horizontal axis reference line. If a horizontal axis reference line label is specified, the intersecting vertical axis reference line is labeled with the vertical axis value. See also the CHREF=, HREFLABELS=, and LHREF= options.

**HREFLABELS=**'*label1*' ... '*labeln*'

**HREFLABEL=**'*label1*' ... '*labeln*'

**HREFLAB=**'*label1*' ... '*labeln*'

specifies labels for the lines requested by the HREF= option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**HREFLABPOS=***n*

specifies the vertical position of labels for HREF= lines. The following table shows the valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement         |
|----------|-------------------------|
| 1        | top                     |
| 2        | staggered from top      |
| 3        | bottom                  |
| 4        | staggered from bottom   |
| 5        | alternating from top    |
| 6        | alternating from bottom |

**INBORDER**

requests a border around probability plots.

**INTERTILE=***value*

specifies the distance between tiles.

**ITPRINTEM**

displays the iteration history for the Turnbull algorithm.

**JITTER=***value*

specifies the amount to jitter overlaying plot symbols, in units of symbol width.

**LFIT=***linetype*

specifies a line style for fitted curves and confidence limits. By default, fitted curves are drawn by connecting solid lines (*linetype* = 1), and confidence limits are drawn by connecting dashed lines (*linetype* = 3).

**LGRID=linetype**specifies a line style for all grid lines. *linetype* is between 1 and 46. The default is 35.**LHREF=linetype****LH=linetype**

specifies the line type for lines requested by the HREF= option. The default is 2, which produces a dashed line.

**LVREF=linetype****LV=linetype**

specifies the line type for lines requested by the VREF= option. The default is 2, which produces a dashed line.

**MAXITEM=n1 <,n2> n1**specifies the maximum number of iterations allowed for the Turnbull algorithm. Iteration history will be printed in increments of *n2* if requested with the ITPRINTEM option. See the section “Arbitrarily Censored Data” on page 90 for details.**NAME='string'**

specifies a name for the plot, up to eight characters, that appears in the PROC GREPLAY master menu. The default is 'LIFEREG'.

**NOCENPLOT**

suppresses the plotting of censored data points.

**NOCONF**

suppresses the default percentile confidence bands on the probability plot.

**NODATA**

suppresses plotting of the estimated empirical probability plot.

**NOFIT**

suppresses the fitted probability (percentile) line and confidence bands.

**NOFRAME**

suppresses the frame around plotting areas.

**NOGRID**

suppresses grid lines.

**NOHLABEL**

suppresses horizontal labels.

**NOHTICK**

suppresses horizontal tick marks.

**NOPOLISH**

suppresses setting small interval probabilities to zero in the Turnbull algorithm.

**NOVLABEL**

suppresses vertical labels.

**NOVTICK**

suppresses vertical tick marks.

**NPINTERVALS=***interval type*

specifies one of the two kinds of confidence limits for the estimated cumulative probabilities, pointwise (NPINTERVALS=POINT) or simultaneous (NPINTERVALS=SIMUL), requested by the PPOUT option to be printed in the tabular output.

**PCTLIST=***value-list*

specifies the list of percentages for which to compute percentile estimates. *value-list* must be a list of values separated by blanks or commas. Each value in the list must be between 0 and 100.

**PLOWER=***value*

specifies the lower limit on the probability axis scale. The PLOWER= option specifies *value* as the lower probability axis tick mark. The tick mark interval and the upper axis limit are determined automatically.

**PRINTPROBS**

prints intervals and associated probabilities for the Turnbull algorithm.

**PUPPER=***value*

specifies the upper limit on the probability axis scale. The PUPPER= option specifies *value* as the upper probability axis tick mark. The tick mark interval and the lower axis limit are determined automatically.

**PPOS=***character-list*

specifies the plotting position type. See the section “Probability Plotting” on page 88 for details.

| <b>PPOS</b> | <b>Method</b>                   |
|-------------|---------------------------------|
| EXPRANK     | expected ranks                  |
| MEDRANK     | median ranks                    |
| MEDRANK1    | median ranks (exact formula)    |
| KM          | Kaplan-Meier                    |
| MKM         | modified Kaplan-Meier (default) |

**PPOUT**

specifies that a table of the cumulative probabilities plotted on the probability plot be printed. Kaplan-Meier estimates of the cumulative probabilities are also printed, along with standard errors and confidence limits. The confidence limits can be pointwise or simultaneous, as specified by the NPINTERVALS= option.

**PROBLIST=***value-list*

specifies the list of initial values for the Turnbull algorithm.

**ROTATE**

requests probability plots with probability scale on the horizontal axis.

**SQUARE**

makes the layout of the probability plots square.

**TOLLIKE=***value*

specifies the criterion for convergence in the Turnbull algorithm.

**TOLPROB=***value*

specifies the criterion for setting the interval probability to zero in the Turnbull algorithm.

**VAXISLABEL=***'string'*

specifies a label for the vertical axis.

**VREF=***value-list*

requests reference lines perpendicular to the vertical axis. If (INTERSECT) is specified, a second reference line perpendicular to the horizontal axis is drawn that intersects the fit line at the same point as the vertical axis reference line. If a vertical axis reference line label is specified, the intersecting horizontal axis reference line is labeled with the horizontal axis value. See also the entries for the CVREF=, LVREF=, and VREFLABELS= options.

**VREFLABELS=***'label1' ... 'labeln'***VREFLABEL=***'label1' ... 'labeln'***VREFLAB=***'label1' ... 'labeln'*

specifies labels for the lines requested by the VREF= option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**VREFLABPOS=***n*

specifies the horizontal position of labels for VREF= lines. The valid values for *n* and the corresponding label placements are shown in the following table.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | left            |
| 2        | right           |

**WAXIS=***n*

specifies line thickness for axes and frame. The default value is 1.

**WFIT=***n*

specifies line thickness for fitted curves. The default value is 1.

**WGRID=***n*

specifies line thickness for grids. The default value is 1.

**WREFL=***n*

specifies line thickness for reference lines. The default value is 1.

## Details

### Confidence Intervals

Confidence intervals are computed for all model parameters and are reported in the “Analysis of Parameter Estimates” table. The confidence coefficient can be specified with the ALPHA= $\alpha$  MODEL statement option, resulting in a  $(1 - \alpha) \times 100\%$  two-sided confidence coefficient. The default confidence coefficient is 95%, corresponding to  $\alpha = .05$ .

#### Regression Parameters

A two-sided  $(1 - \alpha) \times 100\%$  confidence interval  $[\beta_{iL}, \beta_{iU}]$  for the regression parameter  $\beta_i$  is based on the asymptotic normality of the maximum likelihood estimator  $\hat{\beta}_i$  and is computed by

$$\beta_{iL} = \hat{\beta}_i - z_{1-\alpha/2}(\text{SE}_{\hat{\beta}_i})$$

$$\beta_{iU} = \hat{\beta}_i + z_{1-\alpha/2}(\text{SE}_{\hat{\beta}_i})$$

where  $\text{SE}_{\hat{\beta}_i}$  is the estimated standard error of  $\hat{\beta}_i$ , and  $z_p$  is the  $p \times 100\%$  percentile of the standard normal distribution.

#### Scale Parameter

A two-sided  $(1 - \alpha) \times 100\%$  confidence interval  $[\sigma_L, \sigma_U]$  for the scale parameter  $\sigma$  in the location-scale model is based on the asymptotic normality of the logarithm of the maximum-likelihood estimator  $\log(\hat{\sigma})$ , and is computed by

$$\sigma_L = \hat{\sigma} / \exp[z_{1-\alpha/2}(\text{SE}_{\hat{\sigma}})/\hat{\sigma}]$$

$$\sigma_U = \hat{\sigma} \exp[z_{1-\alpha/2}(\text{SE}_{\hat{\sigma}})/\hat{\sigma}]$$

Refer to Meeker and Escobar (1998) for more information.

#### Weibull Scale and Shape Parameters

The Weibull distribution scale parameter  $\eta$  and shape parameter  $\beta$  are obtained by transforming the extreme value location parameter  $\mu$  and scale parameter  $\sigma$ :

$$\eta = \exp(\mu)$$

$$\beta = 1/\sigma$$

Consequently, two-sided  $(1 - \alpha) \times 100\%$  confidence intervals for the Weibull scale and shape parameters are computed as

$$[\eta_L, \eta_U] = [\exp(\mu_L), \exp(\mu_U)]$$

$$[\beta_L, \beta_U] = [1/\sigma_U, 1/\sigma_L]$$

**Gamma Shape Parameter**

A two-sided  $(1 - \alpha) \times 100\%$  confidence interval for the 3-parameter gamma shape parameter  $\delta$  is computed by

$$[\delta_L, \delta_U] = [\hat{\delta} - z_{1-\alpha/2}(\text{SE}_{\hat{\delta}}), \hat{\delta} + z_{1-\alpha/2}(\text{SE}_{\hat{\delta}})]$$

---

**Probability Plotting**

Probability plots are useful tools for the display and analysis of lifetime data. Probability plots use an inverse distribution scale so that a cumulative distribution function (CDF) plots as a straight line. A nonparametric estimate of the CDF of the lifetime data will plot approximately as a straight line, thus providing a visual assessment of goodness-of-fit.

You can use the PROBPLOT statement in LIFEREG to create probability plots of data that are complete, right-censored, interval-censored, or a combination of censoring types (arbitrarily censored). A line representing the maximum likelihood fit from the MODEL statement and pointwise parametric confidence bands for the cumulative probabilities are also included on the plot.

A random variable  $Y$  belongs to a *location-scale* family of distributions if its CDF  $F$  is of the form

$$\Pr\{Y \leq y\} = F(y) = G\left(\frac{y - \mu}{\sigma}\right)$$

where  $\mu$  is the location parameter and  $\sigma$  is the scale parameter. Here,  $G$  is a CDF that cannot depend on any unknown parameters, and  $G$  is the CDF of  $Y$  if  $\mu = 0$  and  $\sigma = 1$ . For example, if  $Y$  is a normal random variable with mean  $\mu$  and standard deviation  $\sigma$ ,

$$G(u) = \Phi(u) = \int_{-\infty}^u \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right) du$$

and

$$F(y) = \Phi\left(\frac{y - \mu}{\sigma}\right)$$

The normal, extreme value, and logistic distributions are location-scale models. The 3-parameter gamma distribution is a location-scale model if the shape parameter  $\delta$  is fixed. If  $T$  has a lognormal, Weibull, or log-logistic distribution, then  $\log(T)$  has a distribution that is a location-scale model. Probability plots are constructed for lognormal, Weibull, and log-logistic distributions by using  $\log(T)$  instead of  $T$  in the plots.

Let  $y_{(1)} \leq y_{(2)} \leq \dots \leq y_{(n)}$  be ordered observations of a random sample with distribution function  $F(y)$ . A probability plot is a plot of the points  $y_{(i)}$  against  $m_i = G^{-1}(a_i)$ , where  $a_i = \hat{F}(y_{(i)})$  is an estimate of the CDF  $F(y_{(i)}) = G\left(\frac{y_{(i)} - \mu}{\sigma}\right)$ .

The nonparametric CDF estimates  $a_i$  are sometimes called *plotting positions*. The axis on which the points  $m_i$  are plotted is usually labeled with a probability scale (the scale of  $a_i$ ).

If  $F$  is one of the location-scale distributions, then  $y$  is the lifetime; otherwise, the log of the lifetime is used to transform the distribution to a location-scale model.

If the data actually have the stated distribution, then  $\hat{F} \approx F$ ,

$$m_i = G^{-1}(\hat{F}(y_i)) \approx G^{-1}\left(G\left(\frac{y_{(i)} - \mu}{\sigma}\right)\right) = \frac{y_{(i)} - \mu}{\sigma}$$

and points  $(y_{(i)}, m_i)$  should fall approximately on a straight line.

There are several ways to compute the nonparametric CDF estimates used in probability plots from lifetime data. These are discussed in the next two sections.

### **Complete and Right-Censored Data**

The censoring times must be taken into account when you compute plotting positions for right-censored data. The modified Kaplan-Meier method described in the following section is the default method for computing nonparametric CDF estimates for display on probability plots. Refer to Abernethy (1996), Meeker and Escobar (1998), and Nelson (1982) for discussions of the methods described in the following sections.

### **Expected Ranks, Kaplan-Meier, and Modified Kaplan-Meier Methods**

Let  $y_{(1)} \leq y_{(2)} \leq \dots \leq y_{(n)}$  be ordered observations of a random sample including failure times and censor times. Order the data in increasing order. Label all the data with reverse ranks  $r_i$ , with  $r_1 = n, \dots, r_n = 1$ . For the lifetime (not censoring time) corresponding to reverse rank  $r_i$ , compute the survival function estimate

$$S_i = \left[ \frac{r_i}{r_i + 1} \right] S_{i-1}$$

with  $S_0 = 1$ . The expected rank plotting position is computed as  $a_i = 1 - S_i$ . The option PPOS=EXPRANK specifies the expected rank plotting position.

For the Kaplan-Meier method,

$$S_i = \left[ \frac{r_i - 1}{r_i} \right] S_{i-1}$$

The Kaplan-Meier plotting position is then computed as  $a'_i = 1 - S_i$ . The option PPOS=KM specifies the Kaplan-Meier plotting position.

For the modified Kaplan-Meier method, use

$$S'_i = \frac{S_i + S_{i-1}}{2}$$

where  $S_i$  is computed from the Kaplan-Meier formula with  $S_0 = 1$ . The plotting position is then computed as  $a''_i = 1 - S'_i$ . The option PPOS=MKM specifies the

modified Kaplan-Meier plotting position. If the PPOS option is not specified, the modified Kaplan-Meier plotting position is used as the default method.

For complete samples,  $a_i = i/(n+1)$  for the expected rank method,  $a'_i = i/n$  for the Kaplan-Meier method, and  $a''_i = (i - .5)/n$  for the modified Kaplan-Meier method. If the largest observation is a failure for the Kaplan-Meier estimator, then  $F_n = 1$  and the point is not plotted.

### Median Ranks

Let  $y_{(1)} \leq y_{(2)} \leq \dots \leq y_{(n)}$  be ordered observations of a random sample including failure times and censor times. A failure order number  $j_i$  is assigned to the  $i$ th failure:  $j_i = j_{i-1} + \Delta$ , where  $j_0 = 0$ . The increment  $\Delta$  is initially 1 and is modified when a censoring time is encountered in the ordered sample. The new increment is computed as

$$\Delta = \frac{(n+1) - \text{previous failure order number}}{1 + \text{number of items beyond previous censored item}}$$

The plotting position is computed for the  $i$ th failure time as

$$a_i = \frac{j_i - .3}{n + .4}$$

For complete samples, the failure order number  $j_i$  is equal to  $i$ , the order of the failure in the sample. In this case, the preceding equation for  $a_i$  is an approximation to the median plotting position computed as the median of the  $i$ th-order statistic from the uniform distribution on (0, 1). In the censored case,  $j_i$  is not necessarily an integer, but the preceding equation still provides an approximation to the median plotting position. The PPOS=MEDRANK option specifies the median rank plotting position.

### Arbitrarily Censored Data

The LIFEREG procedure can create probability plots for data that consists of combinations of exact, left-censored, right-censored, and interval-censored lifetimes, that is, arbitrarily censored data. The LIFEREG procedure uses an iterative algorithm developed by Turnbull (1976) to compute a nonparametric maximum likelihood estimate of the cumulative distribution function for the data. Since the technique is maximum likelihood, standard errors of the cumulative probability estimates are computed from the inverse of the associated Fisher information matrix. This algorithm is an example of the expectation-maximization (EM) algorithm. The default initial estimate assigns equal probabilities to each interval. You can specify different initial values with the PROBLIST= option. Convergence is determined if the change in the log likelihood between two successive iterations is less than delta, where the default value of delta is  $10^{-8}$ . You can specify a different value for delta with the TOLLIKE= option. Iterations will be terminated if the algorithm does not converge after a fixed number of iterations. The default maximum number of iterations is 1000. Some data may require more iterations for convergence. You can specify the maximum allowed number of iterations with the MAXITEM= option in the PROBPLOT statement. The iteration history of the log likelihood is displayed if you specify the ITPRINTEM option. The iteration history of the estimated interval probabilities are also displayed if you specify both options ITPRINTEM and PRINTPROBS.



If an interval probability is smaller than a tolerance ( $10^{-6}$  by default) after convergence, the probability is set to zero, the interval probabilities are renormalized so that they add to one, and iterations are restarted. Usually the algorithm converges in just a few more iterations. You can change the default value of the tolerance with the TOLPROB= option. You can specify the NOPOLISH option to avoid setting small probabilities to zero and restarting the algorithm.

If you specify the ITPRINTEM option, a table summarizing the Turnbull estimate of the interval probabilities is displayed. The columns labeled “Reduced Gradient” and “Lagrange Multiplier” are used in checking final convergence of the maximum likelihood estimate. The Lagrange multipliers must all be greater than or equal to zero, or the solution is not maximum likelihood. Refer to Gentleman and Geyer (1994) for more details of the convergence checking. Also refer to Meeker and Escobar (1998, chap. 3) for more information.

See Example 6.3 on page 100 for an illustration.

### Nonparametric Confidence Intervals

You can use the PPOUT option in the PROBPLOT statement to create a table containing the nonparametric CDF estimates computed by the selected method, Kaplan-Meier CDF estimates, standard errors of the Kaplan-Meier estimator, and nonparametric confidence limits for the CDF. The confidence limits are either pointwise or simultaneous, depending on the value of the NPINTERVALS= option in the PROBPLOT statement. The method used in the LIFEREG procedure for computation of approximate pointwise and simultaneous confidence intervals for cumulative failure probabilities relies on the Kaplan-Meier estimator of the cumulative distribution function of failure time and approximate standard deviation of the Kaplan-Meier estimator. For the case of arbitrarily censored data, the Turnbull algorithm, discussed previously, provides an extension of the Kaplan-Meier estimator. Both the Kaplan-Meier and the Turnbull estimators provide an estimate of the standard error of the CDF estimator,  $se_{\hat{F}}$ , that is used in computing confidence intervals.

#### Pointwise Confidence Intervals

Approximate  $(1 - \alpha)100\%$  pointwise confidence intervals are computed as in Meeker and Escobar (1998, section 3.6) as

$$[F_L, F_U] = \left[ \frac{\hat{F}}{\hat{F} + (1 - \hat{F})w}, \frac{\hat{F}}{\hat{F} + (1 - \hat{F})/w} \right]$$

where

$$w = \exp \left[ \frac{z_{1-\alpha/2} se_{\hat{F}}}{(\hat{F}(1 - \hat{F}))} \right]$$

where  $z_p$  is the  $p$ th quantile of the standard normal distribution.

#### Simultaneous Confidence Intervals

Approximate  $(1 - \alpha)100\%$  simultaneous confidence bands valid over the lifetime interval  $(t_a, t_b)$  are computed as the “Equal Precision” case of Nair (1984) and Meeker

and Escobar (1998, section 3.8) as

$$[F_L, F_U] = \left[ \frac{\hat{F}}{\hat{F} + (1 - \hat{F})w}, \frac{\hat{F}}{\hat{F} + (1 - \hat{F})/w} \right]$$

where

$$w = \exp \left[ \frac{e_{a,b,1-\alpha/2} \mathbf{se}_{\hat{F}}}{(\hat{F}(1 - \hat{F}))} \right]$$

where the factor  $x = e_{a,b,1-\alpha/2}$  is the solution of

$$x \exp(-x^2/2) \log \left[ \frac{(1-a)b}{(1-b)a} \right] / \sqrt{8\pi} = \alpha/2$$

The time interval  $(t_a, t_b)$  over which the bands are valid depends in a complicated way on the constants  $a$  and  $b$  defined in Nair (1984),  $0 < a < b < 1$ .  $a$  and  $b$  are chosen by default so that the confidence bands are valid between the lowest and highest times corresponding to failures in the case of multiply censored data, or, to the lowest and highest intervals for which probabilities are computed for arbitrarily censored data. You can optionally specify  $a$  and  $b$  directly with the `NPINTERVALS=SIMULTANEOUS( $a, b$ )` option in the `PROBPLOT` statement.

---

## INEST= Data Set

If specified, the `INEST=` data set specifies initial estimates for all the parameters in the model. The `INEST=` data set must contain the intercept variable (named `Intercept`) and all independent variables in the `MODEL` statement.

If `BY` processing is used, the `INEST=` data set should also include the `BY` variables, and there must be at least one observation for each `BY` group. If there is more than one observation in one `BY` group, the first one read is used for that `BY` group.

If the `INEST=` data set also contains the `_TYPE_` variable, only observations with `_TYPE_` value 'PARMS' are used as starting values. Combining the `INEST=` data set and the option `MAXITER=` in the `MODEL` statement, partial scoring can be done, such as predicting on a validation data set by using the model built from a training data set.

You can specify starting values for the iterative algorithm in the `INEST=` data set. This data set overwrites the `INITIAL=` option in the `MODEL` statement, which is a little difficult to use for models including multilevel interaction effects. The `INEST=` data set has the same structure as the `OUTEST=` data set but is not required to have all the variables or observations that appear in the `OUTEST=` data set. One simple use of the `INEST=` option is passing the previous `OUTEST=` data set directly to the next model as an `INEST=` data set, assuming that the two models have the same parameterization.

---

## XDATA= Data Set

The XDATA= data set is used for plotting the predicted probability when there are covariates specified in a MODEL statement and a probability plot is specified with a PROBLOT statement. See Example 6.1 on page 94 for an illustration.

The XDATA= data set is an input SAS data set that contains values for all the independent variables in the MODEL statement and variables in the CLASS statement. The XDATA= data set has the same structure as the DATA= data set but is not required to have all the variables or observations that appear in the DATA= data set.

The XDATA= data set must contain all the independent variables in the MODEL statement and variables in the CLASS statement. Even though variables in the CLASS statement may not be used, valid values are required for these variables in the XDATA= data set. Missing values are not allowed. Missing values are not allowed in the XDATA= data set for any of the independent variables either. Missing values are allowed for the dependent variables and other variables if they are included in the XDATA= data set.

If BY processing is used, the XDATA= data set should also include the BY variables, and there must be at least one valid observation for each BY group. If there is more than one valid observation in a BY group, the last one read is used for that BY group.

If there is no XDATA= data set in the PROC LIFEREG statement, by default, the LIFEREG procedure will use the overall mean for effects containing a continuous variable (or variables) and the highest level of a single classification variable as reference level. The rules are summarized as follows:

- If the effect contains a continuous variable (or variables), the overall mean of this effect (not the variables) is used.
- If the effect is a single classification variable, the highest level of the variable is used.

## Examples

### Example 6.1. Analysis of Arbitrarily Censored Data with Interaction Effects

The following artificial data are for a study of the natural recovery time of mice after injection of a certain toxin. 20 mice were grouped by sex (sex: 1 = Male, 2 = Female) with equal sizes. Their ages (in days) were recorded at the injection. Their recovery times (in minutes) were also recorded. Toxin density in blood was used to decide whether a mouse recovered. Mice were checked at two times for recovery. If a mouse had recovered at the first time, the observation is left-censored, and no further measurement is made. The variable `time1` is set to missing and `time2` is set to the measurement time to indicate left-censoring. If a mouse had not recovered at the first time, it was checked later at a second time. If it had recovered by the second measurement time, the observation is interval-censored and the variable `time1` is set to the first measurement time and `time2` is set to the second measurement time. If there was no recovery at the second measurement, the observation is right-censored, and `time1` is set to the second measurement time and `time2` is set to missing to indicate right-censoring.

The following statements create a SAS data set containing the data from the experiment and fit a Weibull model with age, sex, and age and sex interaction as covariates.

```

title 'Natural Recovery Time';
data mice;
  input sex age time1 time2 ;
  datalines;
1 57 631 631
1 45 . 170
1 54 227 227
1 43 143 143
1 64 916 .
1 67 691 705
1 44 100 100
1 59 730 .
1 47 365 365
1 74 1916 1916
2 79 1326 .
2 75 837 837
2 84 1200 1235
2 54 . 365
2 74 1255 1255
2 71 1823 .
2 65 537 637
2 33 583 683
2 77 955 .
2 46 577 577
;

data xrow1;
  input sex age time1 time2 ;

```

```

      datalines;
1  50  .  .
;

data xrow2;
  input sex age time1 time2 ;
  datalines;
2  60.6  .  .
;

proc lifereg data=mice xdata=xrow1;
  class sex ;
  model (time1, time2) = age sex age*sex / dist=Weibull;

  probplot / nodata
    font = swiss
    plower=.5
    vref(intersect) = 75
    vreflab = '75 Percent'
    vreflabpos = 2
    cfit=blue
    cframe=ligr
  ;
  inset / cfill = white
    ctext = blue;
run; quit;

```

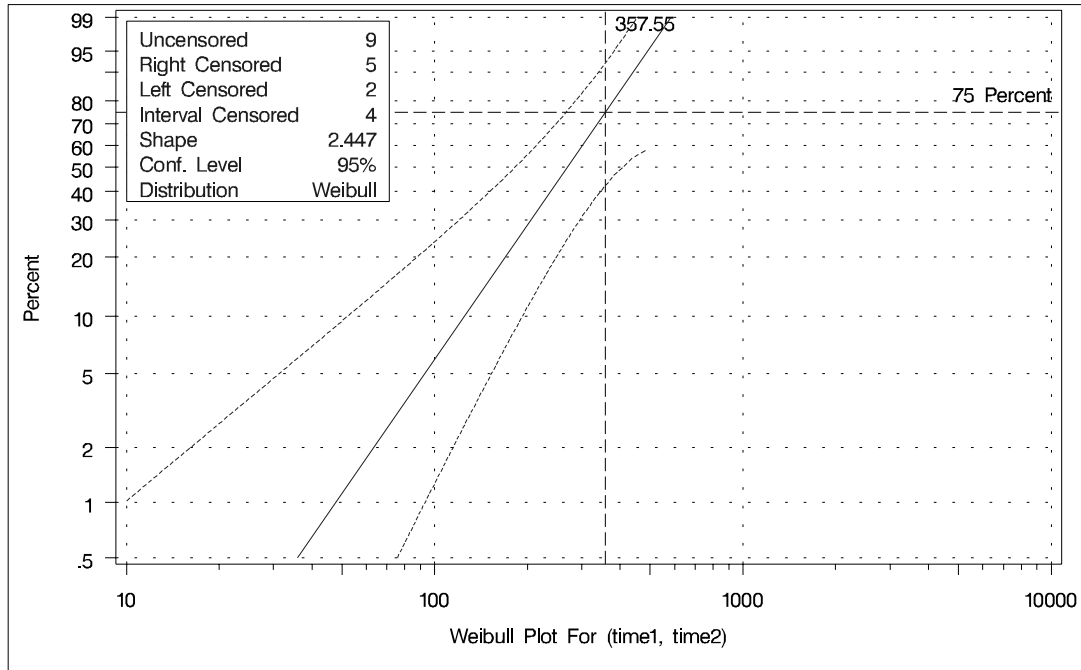
Standard output is shown in Output 6.1.1. Tables containing general model information, Type III tests for the main effects and interaction terms, and parameter estimates are created.

**Output 6.1.1.** Parameter Estimates for the Interaction Model

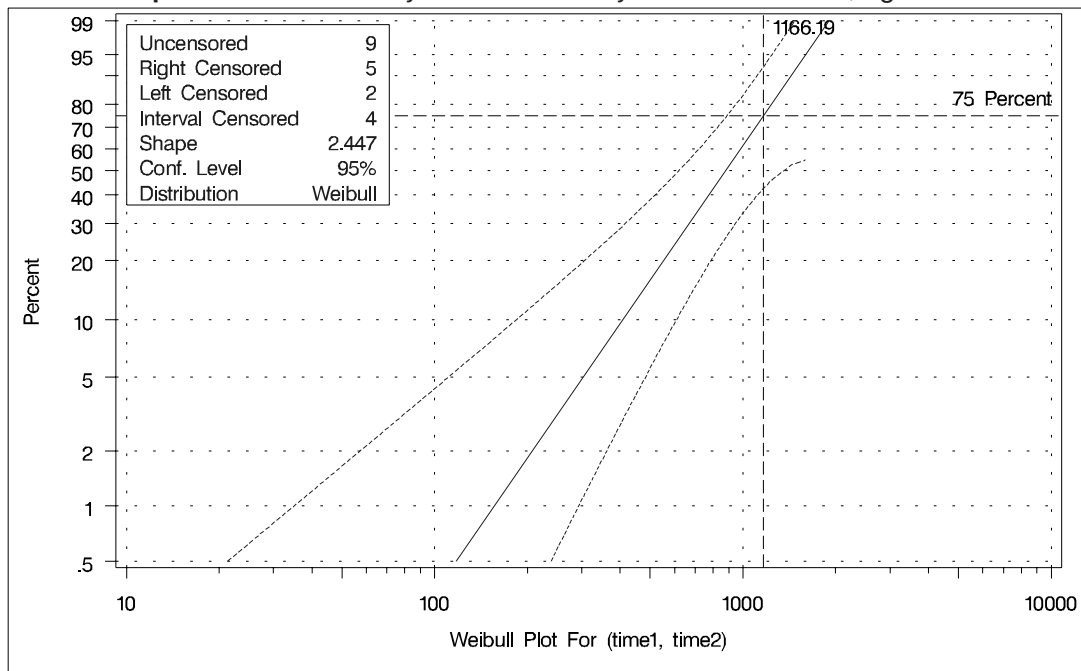
| Natural Recovery Time           |          |                |                       |             |            |            |  |
|---------------------------------|----------|----------------|-----------------------|-------------|------------|------------|--|
| The LIFEREG Procedure           |          |                |                       |             |            |            |  |
| Model Information               |          |                |                       |             |            |            |  |
| Data Set                        |          |                |                       | WORK.MICE   |            |            |  |
| Dependent Variable              |          |                |                       | Log(time1)  |            |            |  |
| Dependent Variable              |          |                |                       | Log(time2)  |            |            |  |
| Number of Observations          |          |                |                       | 20          |            |            |  |
| Noncensored Values              |          |                |                       | 9           |            |            |  |
| Right Censored Values           |          |                |                       | 5           |            |            |  |
| Left Censored Values            |          |                |                       | 2           |            |            |  |
| Interval Censored Values        |          |                |                       | 4           |            |            |  |
| Name of Distribution            |          |                |                       | Weibull     |            |            |  |
| Log Likelihood                  |          |                |                       | -25.9103295 |            |            |  |
| Type III Analysis of Effects    |          |                |                       |             |            |            |  |
| Effect                          | DF       | Wald           |                       | Pr > ChiSq  |            |            |  |
|                                 |          | Chi-Square     |                       |             |            |            |  |
| age                             | 1        | 33.8496        |                       | <.0001      |            |            |  |
| sex                             | 1        | 14.0245        |                       | 0.0002      |            |            |  |
| age*sex                         | 1        | 10.7196        |                       | 0.0011      |            |            |  |
| Analysis of Parameter Estimates |          |                |                       |             |            |            |  |
| DF                              | Estimate | Standard Error | 95% Confidence Limits |             | Chi-Square | Pr > ChiSq |  |
| 1                               | 5.4110   | 0.5549         | 4.3234                | 6.4986      | 95.08      | <.0001     |  |
| 1                               | 0.0250   | 0.0086         | 0.0081                | 0.0419      | 8.42       | 0.0037     |  |
| 1                               | -3.9808  | 1.0630         | -6.0643               | -1.8974     | 14.02      | 0.0002     |  |
| 2                               | 0.0000   | 0.0000         | 0.0000                | 0.0000      | .          | .          |  |
| 1                               | 0.0613   | 0.0187         | 0.0246                | 0.0980      | 10.72      | 0.0011     |  |
| 2                               | 0.0000   | 0.0000         | 0.0000                | 0.0000      | .          | .          |  |
| 1                               | 0.4087   | 0.0900         | 0.2654                | 0.6294      |            |            |  |
| 1                               | 2.4468   | 0.5391         | 1.5887                | 3.7682      |            |            |  |

The following two plots display the predicted probability against the recovery time for two different populations. Output 6.1.2 is created with the PROBLOT statement with the option XDATA= xrow1, which specifies the population with sex = 1, age = 50. Although the SAS statements are not shown, Output 6.1.3 is created with the PROBLOT statement with the option XDATA= xrow2, which specifies the population with sex = 2, age = 60.6. These are the default values that the LIFEREG procedure would use for the probability plot if the XDATA= option had not been specified. Reference lines are used to display specified predicted probability points and their relative locations on the plot.

**Output 6.1.2.** Probability Plot for Recovery Time with sex = 1, age = 50



**Output 6.1.3.** Probability Plot for Recovery Time with sex = 2, age = 60.6



## Example 6.2. Probability Plotting–Right Censoring

The following statements create a SAS data set containing observed and right-censored lifetimes of 70 diesel engine fans (Nelson 1982, p. 318).

```

title 'Engine Fan Lifetime Study';
data fan;
  input lifetime censor@@;
  lifetime = lifetime / 1000;
  label lifetime = Lifetime;
  datalines;
450 0    460 1    1150 0    1150 0    1560 1
1600 0    1660 1    1850 1    1850 1    1850 1
1850 1    1850 1    2030 1    2030 1    2030 1
2070 0    2070 0    2080 0    2200 1    3000 1
3000 1    3000 1    3000 1    3100 0    3200 1
3450 0    3750 1    3750 1    4150 1    4150 1
4150 1    4150 1    4300 1    4300 1    4300 1
4300 1    4600 0    4850 1    4850 1    4850 1
4850 1    5000 1    5000 1    5000 1    6100 1
6100 0    6100 1    6100 1    6300 1    6450 1
6450 1    6700 1    7450 1    7800 1    7800 1
8100 1    8100 1    8200 1    8500 1    8500 1
8500 1    8750 1    8750 0    8750 1    9400 1
9900 1    10100 1    10100 1    10100 1    11500 1
;
run;

```

Some of the fans had not failed at the time the data were collected, and the unfailed units have right-censored lifetimes. The variable LIFETIME represents either a failure time or a censoring time in thousands of hours. The variable CENSOR is equal to 0 if the value of LIFETIME is a failure time, and it is equal to 1 if the value is a censoring time. The following statements use the LIFEREG procedure to produce the probability plot with an inset for the engine lifetimes.

```

symbol v=dot c=white;
proc lifereg;
  model lifetime*censor( 1 ) = / d = weibull;
  probplot
    cencolor = red
    cframe   = ligr
    cfit     = blue
  ppout
    npintervals=simul
  ;
  inset /
    cfill = white
    ctext = blue;
run; quit;

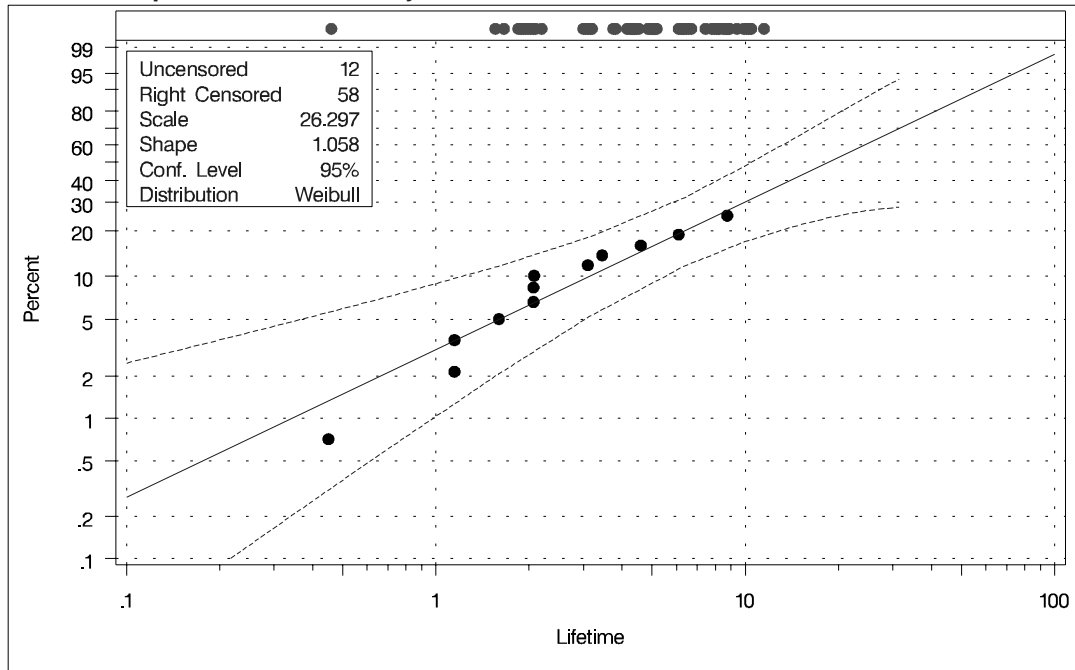
```

The resulting graphical output is shown in Output 6.2.1. The estimated CDF, a line representing the maximum-likelihood fit, and pointwise parametric confidence bands



are plotted in the body of Output 6.2.1. The values of right-censored observations are plotted along the top of the graph. The “Cumulative Probability Estimates” table is also shown in Output 6.2.2.

**Output 6.2.1.** Probability Plot for the Fan Data



**Output 6.2.2.** CDF Estimates

| Engine Fan Lifetime Study        |                        |                                    |        |                       |                             |
|----------------------------------|------------------------|------------------------------------|--------|-----------------------|-----------------------------|
| The LIFEREG Procedure            |                        |                                    |        |                       |                             |
| Cumulative Probability Estimates |                        |                                    |        |                       |                             |
| Lifetime                         | Cumulative Probability | Simultaneous 95% Confidence Limits |        | Kaplan-Meier Estimate | Kaplan-Meier Standard Error |
|                                  |                        | Lower                              | Upper  |                       |                             |
| 0.45                             | 0.0071                 | 0.0007                             | 0.2114 | 0.0143                | 0.0142                      |
| 1.15                             | 0.0215                 | 0.0033                             | 0.2114 | 0.0288                | 0.0201                      |
| 1.15                             | 0.0360                 | 0.0073                             | 0.2168 | 0.0433                | 0.0244                      |
| 1.6                              | 0.0506                 | 0.0125                             | 0.2304 | 0.0580                | 0.0282                      |
| 2.07                             | 0.0666                 | 0.0190                             | 0.2539 | 0.0751                | 0.0324                      |
| 2.07                             | 0.0837                 | 0.0264                             | 0.2760 | 0.0923                | 0.0361                      |
| 2.08                             | 0.1008                 | 0.0344                             | 0.2972 | 0.1094                | 0.0392                      |
| 3.1                              | 0.1189                 | 0.0436                             | 0.3223 | 0.1283                | 0.0427                      |
| 3.45                             | 0.1380                 | 0.0535                             | 0.3471 | 0.1477                | 0.0460                      |
| 4.6                              | 0.1602                 | 0.0653                             | 0.3844 | 0.1728                | 0.0510                      |
| 6.1                              | 0.1887                 | 0.0791                             | 0.4349 | 0.2046                | 0.0581                      |
| 8.75                             | 0.2488                 | 0.0884                             | 0.6391 | 0.2930                | 0.0980                      |

### Example 6.3. Probability Plotting–Arbitrarily Censoring

Table 6.2 contains microprocessor failure data from Nelson (1990). Units were inspected at predetermined time intervals. The data consist of inspection interval endpoints (in hours) and the number of units failing in each interval. A missing (.) lower endpoint indicates left censoring, and a missing upper endpoint indicates right censoring. These can be thought of as semi-infinite intervals with a lower (upper) endpoint of negative (positive) infinity for left (right) censoring.

**Table 6.2.** Interval-Censored Data

| Lower Endpoint | Upper Endpoint | Number Failed |
|----------------|----------------|---------------|
| .              | 6              | 6             |
| 6              | 12             | 2             |
| 24             | 48             | 2             |
| 24             | .              | 1             |
| 48             | 168            | 1             |
| 48             | .              | 839           |
| 168            | 500            | 1             |
| 168            | .              | 150           |
| 500            | 1000           | 2             |
| 500            | .              | 149           |
| 1000           | 2000           | 1             |
| 1000           | .              | 147           |
| 2000           | .              | 122           |

The following SAS program will compute the Turnbull estimate and create a lognormal probability plot.

```

data micro;
    input t1 t2 f ;
    datalines;
. 6 6
6 12 2
12 24 0
24 48 2
24 . 1
48 168 1
48 . 839
168 500 1
168 . 150
500 1000 2
500 . 149
1000 2000 1
1000 . 147
2000 . 122
;
run;

```

```

title;
symbol v=dot c=white;
proc lifereg data=micro;
  model ( t1 t2 ) = / d=lognormal intercept=25 scale=5;
  weight f;
  probplot
    cframe = ligr
    cfit    = blue
    pupper = 10
    itprintem
    printprobs
    maxitem = (1000,25)
    ppout;
  inset / cfill = white;
run; quit;

```

The two initial values INTERCEPT= 25 and SCALE= 5 in the MODEL statement are used to aid convergence in the model-fitting algorithm.

The following tables are created by the PROBPLOT statement in addition to the standard tabular output from the MODEL statement. Output 6.3.1 shows the iteration history for the Turnbull estimate of the CDF for the microprocessor data. With both options ITPRINTEM and PRINTPROBS specified in the PROBPLOT statement, this table contains the log likelihoods and interval probabilities for every 25th iteration and the last iteration. It would only contain the log likelihoods if the option PRINTPROBS were not specified.

**Output 6.3.1.** Iteration History for the Turnbull Estimate

| The LIFEREG Procedure                                  |               |                          |                          |                          |                          |
|--------------------------------------------------------|---------------|--------------------------|--------------------------|--------------------------|--------------------------|
| Iteration History for the Turnbull Estimate of the CDF |               |                          |                          |                          |                          |
| Iteration                                              | Loglikelihood | (., 6)<br>(168, 500)     | (6, 12)<br>(500, 1000)   | (24, 48)<br>(1000, 2000) | (48, 168)<br>(2000, .)   |
| 0                                                      | -1133.4051    | 0.125<br>0.125           | 0.125<br>0.125           | 0.125<br>0.125           | 0.125<br>0.125           |
| 25                                                     | -104.16622    | 0.00421644<br>0.00237846 | 0.00140548<br>0.00846094 | 0.00140648<br>0.04565407 | 0.00173338<br>0.93474475 |
| 50                                                     | -101.15151    | 0.00421644<br>0.00234891 | 0.00140548<br>0.00727679 | 0.00140648<br>0.01174486 | 0.00173293<br>0.96986811 |
| 75                                                     | -101.06641    | 0.00421644<br>0.00234891 | 0.00140548<br>0.00727127 | 0.00140648<br>0.00835638 | 0.00173293<br>0.9732621  |
| 100                                                    | -101.06534    | 0.00421644<br>0.00234891 | 0.00140548<br>0.00727125 | 0.00140648<br>0.00801814 | 0.00173293<br>0.97360037 |
| 125                                                    | -101.06533    | 0.00421644<br>0.00234891 | 0.00140548<br>0.00727125 | 0.00140648<br>0.00798438 | 0.00173293<br>0.97363413 |
| 130                                                    | -101.06533    | 0.00421644<br>0.00234891 | 0.00140548<br>0.00727125 | 0.00140648<br>0.007983   | 0.00173293<br>0.97363551 |

**Output 6.3.2.** Summary for the Turnbull Algorithm

| The LIFEREG Procedure |                   |             |                     |                        |
|-----------------------|-------------------|-------------|---------------------|------------------------|
| Lower<br>Lifetime     | Upper<br>Lifetime | Probability | Reduced<br>Gradient | Lagrange<br>Multiplier |
| .                     | 6                 | 0.0042      | 0                   | 0                      |
| 6                     | 12                | 0.0014      | 0                   | 0                      |
| 24                    | 48                | 0.0014      | 0                   | 0                      |
| 48                    | 168               | 0.0017      | 0                   | 0                      |
| 168                   | 500               | 0.0023      | 0                   | 0                      |
| 500                   | 1000              | 0.0073      | -7.219342E-9        | 0                      |
| 1000                  | 2000              | 0.0080      | -0.037063236        | 0                      |
| 2000                  | .                 | 0.9736      | 0.0003038877        | 0                      |

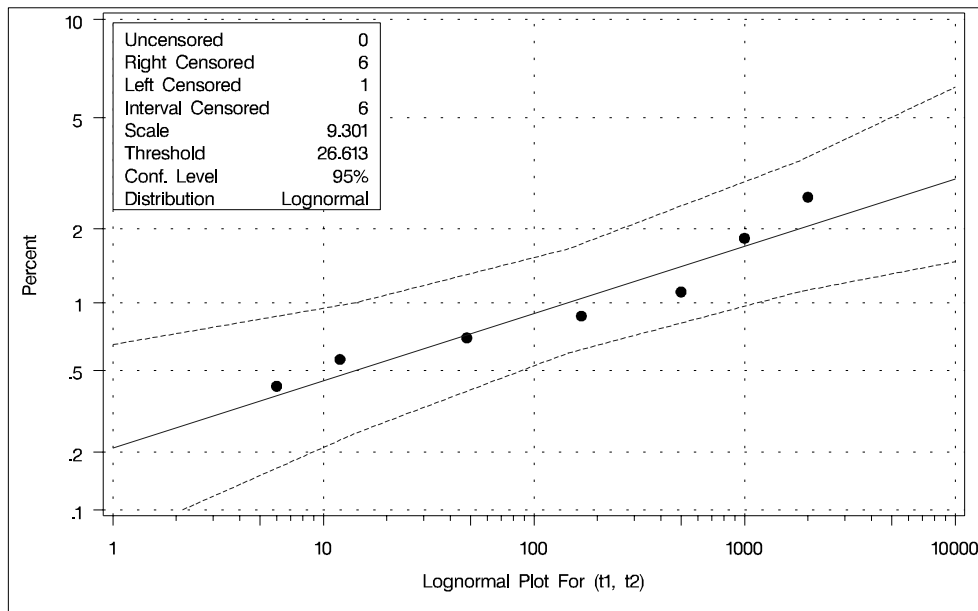
The table in Output 6.3.2 summarizes the Turnbull estimates of the interval probabilities, the reduced gradients, and Lagrange multipliers as described in the section “Arbitrarily Censored Data” on page 90.

**Output 6.3.3.** Final CDF Estimates for Turnbull Algorithm

| The LIFEREG Procedure            |                   |                           |                                       |        |                   |
|----------------------------------|-------------------|---------------------------|---------------------------------------|--------|-------------------|
| Cumulative Probability Estimates |                   |                           |                                       |        |                   |
| Lower<br>Lifetime                | Upper<br>Lifetime | Cumulative<br>Probability | Pointwise 95%<br>Confidence<br>Limits |        | Standard<br>Error |
|                                  |                   |                           | Lower                                 | Upper  |                   |
| 6                                | 6                 | 0.0042                    | 0.0019                                | 0.0094 | 0.0017            |
| 12                               | 24                | 0.0056                    | 0.0028                                | 0.0112 | 0.0020            |
| 48                               | 48                | 0.0070                    | 0.0038                                | 0.0130 | 0.0022            |
| 168                              | 168               | 0.0088                    | 0.0047                                | 0.0164 | 0.0028            |
| 500                              | 500               | 0.0111                    | 0.0058                                | 0.0211 | 0.0037            |
| 1000                             | 1000              | 0.0184                    | 0.0094                                | 0.0357 | 0.0063            |
| 2000                             | 2000              | 0.0264                    | 0.0124                                | 0.0553 | 0.0101            |

Output 6.3.3 shows the final estimate of the CDF, along with standard errors and nonparametric confidence limits. Two kinds of nonparametric confidence limits, pointwise or simultaneous, are available. The default is the pointwise nonparametric confidence limits. You can specify the simultaneous nonparametric confidence limits by the NPINTERVALS= SIMUL option.

Output 6.3.4 shows the CDF estimates, the maximum likelihood fit, and the pointwise parametric confidence limits plotted on a lognormal probability plot.

**Output 6.3.4.** Lognormal Probability Plot for the Microprocessor Data


---

## References

- Abernethy, R.B. (1996), *The New Weibull Handbook*, Second Edition, North Palm Beach, FL: Robert B. Abernethy.
- Allison, P.D. (1995), *Survival Analysis Using the SAS System: A Practical Guide*, Cary, NC: SAS Institute.
- Cox, D.R. (1972), "Regression Models and Life Tables (with discussion)," *Journal of the Royal Statistical Society, Series B*, 34, 187–220.
- Cox, D.R. and Oakes, D. (1984), *Analysis of Survival Data*, London: Chapman and Hall.
- Elandt-Johnson, R.C. and Johnson, N.L. (1980), *Survival Models and Data Analysis*, New York: John Wiley & Sons, Inc.
- Gentleman, R. and Geyer, C.J. (1994), "Maximum Likelihood for Interval Censored Data: Consistency and Computation," *Biometrika*, 81 (3), 618–623.
- Green, W.H. (1993), *Econometric Analysis*, Second Edition, New York: Cambridge University Press.
- Gross, A.J. and Clark, V.A. (1975), *Survival Distributions: Reliability Applications in the Biomedical Sciences*, New York: John Wiley & Sons, Inc.
- Kalbfleisch, J.D. and Prentice, R.L. (1980), *The Statistical Analysis of Failure Time Data*, New York: John Wiley & Sons, Inc.
- Klein, J.P. and Moeschberger, M.L. (1997), *Survival Analysis: Techniques for Censored and Truncated Data*, Berlin: Springer-Verlag.

- Lawless, J.E. (1982), *Statistical Models and Methods for Lifetime Data*, New York: John Wiley & Sons, Inc.
- Lee, E.T. (1980), *Statistical Methods for Survival Data Analysis*, Belmont, CA: Lifetime Learning Publications.
- Maddala, G.S. (1983), *Limited-Dependent and Qualitative Variables in Econometrics*, New York: Cambridge University Press.
- Meeker, W.Q. and Escobar, L.A. (1998), *Statistical Methods for Reliability Data*, New York: John Wiley & Sons.
- Mroz, T.A. (1987), "The Sensitivity of an Empirical Model of Married Women's Work to Economic and Statistical Assumptions," *Econometrica* 55, 765–799.
- Nair, V.N. (1984), "Confidence Bands for Survival Functions With Censored Data: A Comparative Study," *Technometrics*, 26 (3), 265–275.
- Nelson, W. (1982), *Applied Life Data Analysis*, New York: John Wiley & Sons.
- Nelson, W. (1990), *Accelerated Testing: Statistical Models, Test Plans, and Data Analyses*, New York: John Wiley & Sons.
- Rao, C.R. (1973), *Linear Statistical Inference and Its Applications*, New York: John Wiley & Sons, Inc.
- Tobin, J. (1958), "Estimation of Relationships for Limited Dependent Variables," *Econometrica*, 26, 24–36.
- Turnbull, B.W. (1976), "The Empirical Distribution Function with Arbitrarily Grouped, Censored and Truncated Data," *Journal of the Royal Statistical Society, Series B*, 38, 290–295.

# Chapter 7

## The LOESS Procedure

### Chapter Table of Contents

---

|                                                                         |     |
|-------------------------------------------------------------------------|-----|
| <b>OVERVIEW</b> . . . . .                                               | 107 |
| <b>SYNTAX</b> . . . . .                                                 | 107 |
| MODEL Statement . . . . .                                               | 107 |
| <b>DETAILS</b> . . . . .                                                | 109 |
| Automatic Smoothing Parameter Selection Based on Degrees of Freedom . . | 109 |
| <b>EXAMPLE</b> . . . . .                                                | 109 |
| Example 7.1 Melanoma Incidences . . . . .                               | 109 |
| <b>REFERENCES</b> . . . . .                                             | 112 |





## Chapter 7

# The LOESS Procedure

---

### Overview

The `SELECT=` option in the `MODEL` statement now enables you to select the smoothing parameter so that an appropriate measure of the model degrees of freedom matches a given target value.

### Syntax

---

#### MODEL Statement

The following updated option is available in the `MODEL` statement after a slash (/).

**SELECT=criterion** < ( <GLOBAL> <STEPS> <RANGE(lower,upper)> ) >

**SELECT=DFTYPE** ( target <GLOBAL> <STEPS> <RANGE(lower,upper)> )

specifies that automatic smoothing parameter selection be done using the named *criterion* or *DFTYPE*. Valid values for the *criterion* are

AICC specifies the  $AIC_C$  criterion (Hurvich, Simonoff, and Tsai 1998).

AICC1 specifies the  $AIC_{C_1}$  criterion (Hurvich, Simonoff, and Tsai 1998).

GCV specifies the generalized cross-validation criterion (Craven and Wahba 1979).

The *DFTYPE* specifies the measure to be used to estimate the model degrees of freedom. The measures implemented in PROC LOESS all depend on the prediction matrix  $L$  relating the observed and predicted values of the dependent variable. Valid values are

DF1 specifies  $\text{Trace}(L)$ .

DF2 specifies  $\text{Trace}(L^T L)$ .

DF3 specifies  $2\text{Trace}(L) - \text{Trace}(L^T L)$ .

For both types of selection, the smoothing parameter value is selected to yield a minimum of an optimization criterion. If you specify *criterion* as one of AICC, AICC1, or GCV, the optimization criterion is the specified *criterion*. If you specify *DFTYPE* as one of DF1, DF2, or DF3, the optimization criterion is  $|DFTYPE - target|$ , where *target* is a specified target degree of freedom value. Note that if you specify a *DFTYPE*, then you must also subsequently specify a target value in parentheses. See the section “Automatic Smoothing Parameter Selection Based on Degrees of Freedom” on page 109 for definitions and properties of the selection criteria.

The selection is done as follows:

- If you specify the `SMOOTH=value-list` option, then PROC LOESS selects the value in this list that yields the global minimum of the specified optimization criterion, with ties going to the larger value.
- If you do not specify the `SMOOTH=` option, then PROC LOESS finds a local minimum of the specified optimization criterion using a golden section search.

You can specify the following modifiers in parentheses after the specified criterion to alter the behavior of the `SELECT=` option:

|                             |                                                                                                                                                                                                                        |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GLOBAL                      | specifies that a global minimum be found within the range of smoothing parameter values examined. This modifier has no effect if you also specify the <code>SMOOTH=</code> option in the <code>MODEL</code> statement. |
| STEPS                       | specifies that all models evaluated in the selection process be displayed.                                                                                                                                             |
| RANGE( <i>lower,upper</i> ) | specifies that only smoothing parameter values greater than or equal to <i>lower</i> and less than or equal to <i>upper</i> be examined.                                                                               |

For models with one dependent variable, if you specify neither the `SELECT=` nor the `SMOOTH=` options in the `MODEL` statement, then PROC LOESS uses `SELECT=AICC`.

The following table summarizes how the smoothing parameter values are chosen for various combinations of the `SMOOTH=` option, the `SELECT=` option, and the `SELECT=` option modifiers.

**Table 7.1.** Smoothing Parameter Value(s) Used for Combinations of `SMOOTH=` and `SELECT=` OPTIONS for Models with One Dependent Variable

| Syntax                                                           | Search Method             | Search Domain                                |
|------------------------------------------------------------------|---------------------------|----------------------------------------------|
| <i>default</i>                                                   | golden section using AICC | (0, 1]                                       |
| <code>SMOOTH=list</code>                                         | no selection              | values in <i>list</i>                        |
| <code>SMOOTH=list SELECT=criterion</code>                        | global                    | values in <i>list</i>                        |
| <code>SMOOTH=list SELECT=criterion ( RANGE(<i>l, u</i>) )</code> | global                    | values in <i>list</i> within [ <i>l, u</i> ] |
| <code>SELECT=criterion</code>                                    | golden section            | (0, 1]                                       |
| <code>SELECT=criterion ( RANGE(<i>l,u</i>) )</code>              | golden section            | [ <i>l, u</i> ]                              |
| <code>SELECT=criterion ( GLOBAL )</code>                         | global                    | (0, 1]                                       |
| <code>SELECT=criterion ( GLOBAL RANGE(<i>l, u</i>) )</code>      | global                    | [ <i>l, u</i> ]                              |

Some examples of using the `SELECT=` option are

|                              |                                                                         |
|------------------------------|-------------------------------------------------------------------------|
| <code>SELECT=GCV</code>      | specifies selection using the GCV <i>criterion</i> .                    |
| <code>SELECT=DF1(6.3)</code> | specifies selection using the DF1 <i>DFT</i> ype with target value 6.3. |

SELECT=AICC(STEPS) specifies selection using the AICC *criterion* showing all step details.

SELECT=DF2(7 GLOBAL) specifies selection using the DF2 *DFT*ype with target value 7 using a global search algorithm.

**Note:** The SELECT= option cannot be used for models with more than one dependent variable.

---

## Details

---

### Automatic Smoothing Parameter Selection Based on Degrees of Freedom

PROC LOESS can now perform automatic smoothing parameter selection by setting (as near as possible) an approximate measure of model degrees of freedom to a specified target value. These methods are useful for making meaningful comparisons between loess fits and other parametric and nonparametric fits. There are three commonly used measures of model degrees of freedom in nonparametric models. All of them are functions of the prediction matrix  $L$  such that

$$\hat{y} = Ly$$

where  $y$  is the vector of observed values and  $\hat{y}$  is the corresponding vector of predicted values of the dependent variable. The measures are:

$$\begin{aligned} \text{DF1} &= \text{Trace}(L) \\ \text{DF2} &= \text{Trace}(L^T L) \\ \text{DF3} &= 2\text{Trace}(L) - \text{Trace}(L^T L) \end{aligned}$$

A discussion of their properties can be found in Hastie and Tibshirani (1990). You invoke these methods by specifying the SELECT=*DFT*ype(*target*) option in the MODEL statement, where *DFT*ype is one of DF1, DF2, or DF3. The criterion that is minimized is  $|DFT\text{ype} - \text{target}|$ .

---

## Example

---

### Example 7.1. Melanoma Incidences

The following data from the Connecticut Tumor Registry presents age-adjusted numbers of melanoma incidences per 100,000 people for 37 years from 1936 to 1972 (Houghton, Flannery, and Viola 1980).

```
data Melanoma;
  input Year Incidences @@;
  format Year d4.0;
datalines;
1936    0.9   1937    0.8   1938    0.8   1939    1.3
```

```

1940    1.4    1941    1.2    1942    1.7    1943    1.8
1944    1.6    1945    1.5    1946    1.5    1947    2.0
1948    2.5    1949    2.7    1950    2.9    1951    2.5
1952    3.1    1953    2.4    1954    2.2    1955    2.9
1956    2.5    1957    2.6    1958    3.2    1959    3.8
1960    4.2    1961    3.9    1962    3.7    1963    3.3
1964    3.7    1965    3.9    1966    4.1    1967    3.8
1968    4.7    1969    4.4    1970    4.8    1971    4.8
1972    4.8
;

```

The following statements fit a model with a target value of 2 for the DF2 criterion.

```

proc loess data=Melanoma;
  model Incidences=Year / select=DF2(2);
  ods output outputStatistics=out2;
run;

```

The optimal smoothing parameter selection table is shown in Output 7.1.1.

**Output 7.1.1.** Optimal Smoothing Criterion Table from PROC LOESS

| The LOESS Procedure            |                     |
|--------------------------------|---------------------|
| Dependent Variable: Incidences |                     |
| Optimal Smoothing Criterion    |                     |
| DF2                            | Smoothing Parameter |
| 2.30235                        | 1.00000             |

Note that the actual value of DF2 achieved is not exactly the target value of 2. This occurs because even though the smoothing parameter is a continuous variable, smoothing parameter values in the range  $(0, 1]$  correspond to a finite set of local models. Among this finite set of models, the model with a smoothing parameter value of 1 yields the DF2 measure that is closest to the specified target value of 2.

Recall that the smoothing parameter value specifies the fraction of the data to be used in each local model. Thus, the selected smoothing parameter value 1 corresponds to using all the data points in each local linear model. This is to be expected, as a LOESS model with about two degrees of freedom has enough latitude to determine one common slope and intercept for all local models. Hence, in this case the LOESS model is really just a global linear model. You can confirm this observation by plotting the fit obtained as follows:

```

symbol1 color=blue value=dot ;
symbol2 color=blue interpol=join value=none;
axis1 label=(angle=90 rotate=0);
%let opts = vaxis=axis1 frame cframe=ligr hm=3 vm=3 overlay;

proc gplot data=out2;

```

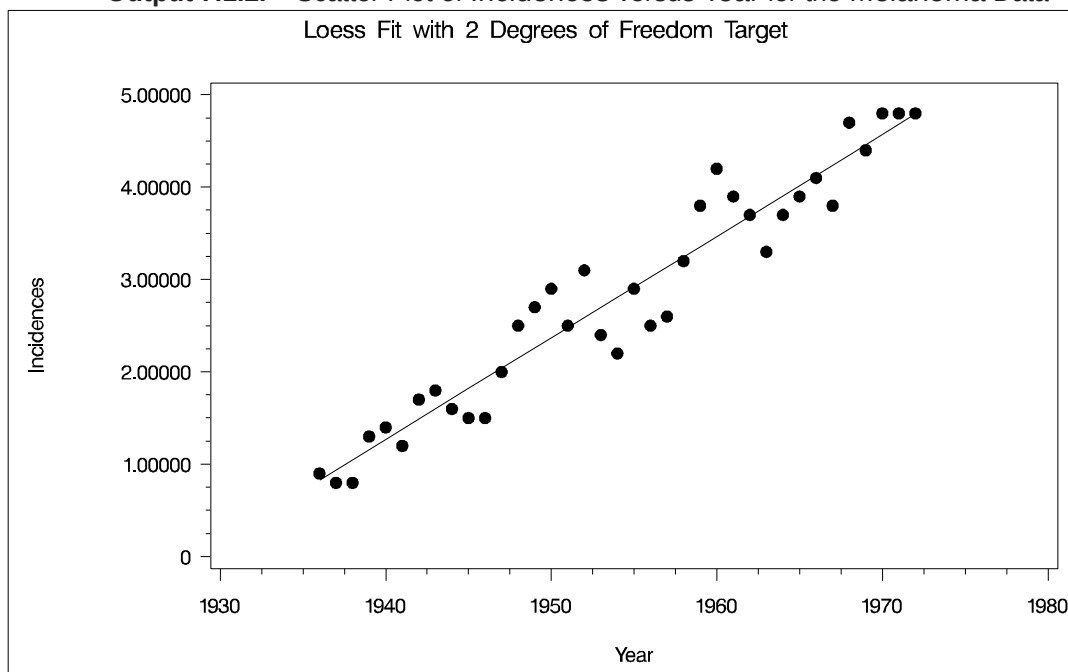
```

title1 'Melanoma Data with Default Loess Fit';
plot DepVar*Year Pred*Year/ &opts;
run;

```

The graph produced is shown in Output 7.1.2.

**Output 7.1.2.** Scatter Plot of Incidences versus Year for the Melanoma Data



The loess fit in the previous plot captures the increasing trend in the data. In order to capture the oscillations in the data that are caused by the eleven-year sunspot cycle, it is necessary to use a loess model with more degrees of freedom. The following statements fit such a model with a target value of 7 degrees of freedom. Loosely speaking, this corresponds to a global fit with a polynomial of degree 6.

```

proc loess data=Melanoma;
  model Incidences=Year / select=DF2(7);
  ods output outputStatistics=out7;
run;

```

The optimal smoothing parameter selection table is shown in Output 7.1.3.

**Output 7.1.3.** Optimal Smoothing Criterion Table from PROC LOESS

| The LOESS Procedure            |                        |
|--------------------------------|------------------------|
| Dependent Variable: Incidences |                        |
| Optimal Smoothing<br>Criterion |                        |
| DF2                            | Smoothing<br>Parameter |
| 7.31083                        | 0.25676                |

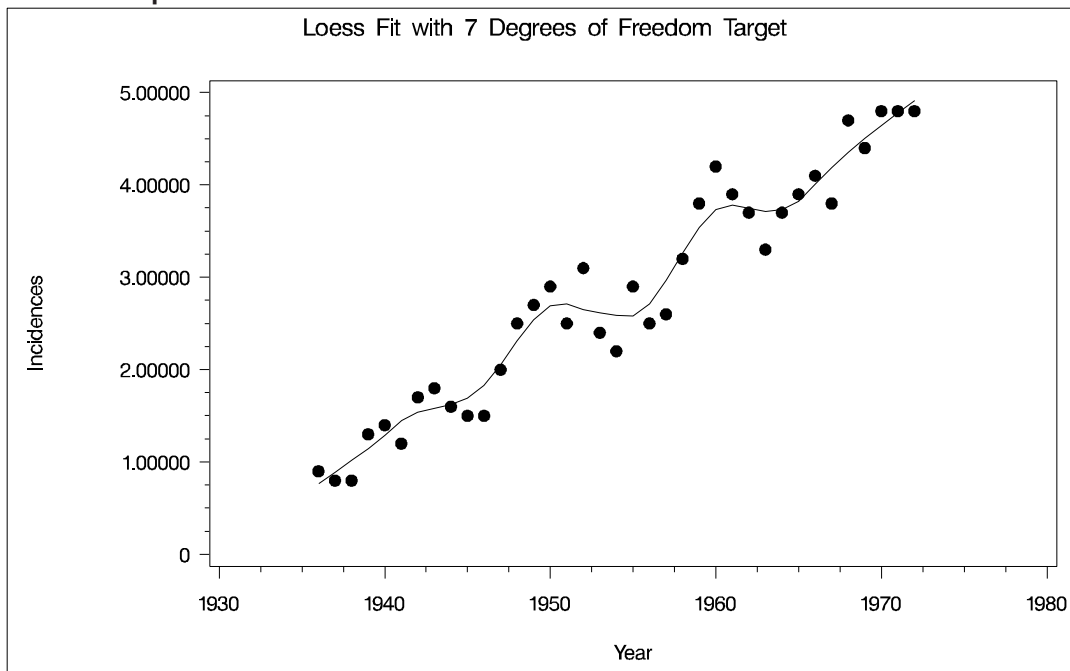
You can plot the fit with the following statements:

```
symbol1 color=blue value=dot ;
symbol2 color=blue interpol=join value=none;
axis1 label=(angle=90 rotate=0);
%let opts = vaxis=axis1 frame cframe=ligr hm=3 vm=3 overlay;

proc gplot data=out7;
  title1 'Melanoma Data with Default Loess Fit';
  plot DepVar*Year Pred*Year/ &opts;
run;
```

The result is shown in Output 7.1.4

**Output 7.1.4.** Scatter Plot of Incidences versus Year for the Melanoma Data



## References

- Akaike, H. (1973), "Information Theory and an Extension of the Maximum Likelihood Principle," in *Proceedings of the Second International Symposium on Information Theory*, eds. Petrov and Csaki, 267-281.
- Craven, P. and Wahba, G. (1979), "Smoothing Noisy Data with Spline Functions," *Numerical Mathematics*, 31, 377-403.
- Hastie, T.J. and Tibshirani, R.J. (1990), *Generalized Additive Models*, Monographs on Statistics and Applied Probability 43, London: Chapman and Hall.
- Houghton, A.N., Flannery, J., and Viola, M.V. (1980), "Malignant Melanoma in Connecticut and Denmark," *International Journal of Cancer*, 25, 95-104.

- Hurvich, C.M., Simonoff, J.S., and Tsai, C.L. (1998), “Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion,” *Journal of the Royal Statistical Society B*, 60, 271–293.





# Chapter 8

## The LOGISTIC Procedure

### Chapter Table of Contents

---

|                                                        |     |
|--------------------------------------------------------|-----|
| <b>OVERVIEW</b>                                        | 117 |
| <b>SYNTAX</b>                                          | 117 |
| MODEL Statement                                        | 117 |
| <b>DETAILS</b>                                         | 120 |
| Generalized Logit Model                                | 120 |
| Intercept Names for Ordinal Models in OUTEST= Data set | 123 |
| Exact Conditional Analysis: Computational Resources    | 123 |
| <b>EXAMPLE</b>                                         | 124 |
| Example 8.1 Generalized Logit Model                    | 124 |
| <b>REFERENCES</b>                                      | 128 |



## Chapter 8

# The LOGISTIC Procedure

---

### Overview

You can now fit a generalized logit model when you have nominal response data. In this multiple-logit model, each nonreference response category is contrasted against the reference category. This model reduces to the binary logit model when there are only two response categories. Unlike the cumulative logit model (for ordinal response data) that contains one set of slope parameters for the entire model, the generalized logit model contains a set of slope parameters for each response category other than the reference category. The generalized logit model is specified by the MODEL statement option LINK=GLOGIT. You can choose the response category to use as the reference in forming the various logit functions. The default is to use the last response category as the reference.

Options specific to the response variable can be specified immediately after the response variable in the MODEL statement by enclosing them in a pair of parentheses. These response variable options include

- the EVENT= option for specifying the event category for binary response models
- the REFERENCE= option for specifying the reference category
- the ORDER= option for specifying the sorting order of the response categories
- the DESCENDING option for reversing the order of the response categories

For ordinal models, names of the intercept parameters in the OUTEST= data are changed to reflect the corresponding response levels.

---

### Syntax

---

#### MODEL Statement

There are two sets of options in the MODEL statement. Options specific to the response variable are referred to as “Response Variable options.” They are specified after the response variable in a pair of parentheses. The other set of options, specified after a slash (/) following the model effects, are referred to as “MODEL options.”

**MODEL** *variable* < (*variable\_options*) > = < *effects* > < /*options* >;

The Response Variable options include the DESCENDING, EVENT=, ORDER=, and REF= options. The MODEL option LINK= has an additional value for specifying the generalized logit model.

**DESCENDING | DESC**

reverses the order of the response categories (levels). If both the DESCENDING and ORDER= options are specified, PROC LOGISTIC orders the response categories according to the ORDER= option and then reverses that order.

**EVENT='category' | keyword**

specifies the event category (level) for the binary response model. PROC LOGISTIC models the probability of the event category. The EVENT= option has no effect when there are more than two response categories. You can specify the event category (formatted if a format is applied) in quotes or you can specify one of the following keywords. The default is EVENT=FIRST.

FIRST    designates the first ordered category as the event

LAST     designates the last ordered category as the event

Consider the example where the response variable *Y* takes the values 1 and 0 for event and nonevent, respectively, and *Exposure* is the explanatory variable. To model the event probability, you simply specify

```
proc logistic;
  model Y(event='1') = Exposure;
run;
```

**ORDER= DATA | FORMATTED | FREQ | INTERNAL**

specifies the sorting order of the categories (levels) of the response variable. The following table shows the interpretation of the ORDER= values.

| Value of ORDER= | Levels Sorted By                                                                                                                       |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------|
| DATA            | order of appearance in the input data set                                                                                              |
| FORMATTED       | external formatted value, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) value |
| FREQ            | descending frequency count; levels with the most observations come first in the order                                                  |
| INTERNAL        | unformatted value                                                                                                                      |

By default, ORDER=FORMATTED. For the FORMATTED and INTERNAL specifications, the sort order may be machine dependent. For numeric response variables with no explicit format (that is, no corresponding FORMAT statement in the current PROC LOGISTIC run or in the DATA step that created the data set), ORDER=FORMATTED orders the response levels by their numeric values.\*

\*This represents a change from earlier releases for how response levels are ordered. In releases previous to Version 8, numeric levels with no explicit format were ordered by their BEST12. formatted values. If you want to continue to use this ordering, you would specify this format explicitly. The change was implemented because the former default behavior for ORDER=FORMATTED often resulted in levels not being ordered numerically and usually required the user to intervene with an explicit format or ORDER=INTERNAL to obtain the more natural ordering.

For more information on sorting order, see the chapter on the SORT procedure in the *SAS Procedures Guide*.

**REFERENCE**=*'category' | keyword*

**REF**=*'category' | keyword*

specifies the reference category for the generalized logit model and the binary response model. For the generalized logit model, each nonreference category is contrasted with the reference category. For the binary response model, specifying one response category as the reference is equivalent to specifying the other response category as the event category. You can specify the reference category (formatted if a format is applied) in quotes or you can specify one of the following keywords. The default is REF=LAST.

|       |                                                        |
|-------|--------------------------------------------------------|
| FIRST | designates the first ordered category as the reference |
| LAST  | designates the last ordered category as the reference  |

**LINK**=*keyword*

**L**=*keyword*

specifies the function linking the response probabilities to the linear predictors. You can specify one of the following keywords. The default is LINK=LOGIT.

|         |                                                                                                                                                                                                                                             |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CLOGLOG | the complementary log-log function. PROC LOGISTIC fits the binary CLOGLOG model when there are two response categories and fits the cumulative CLOGLOG model when there are more than two response categories.                              |
| GLOGIT  | the generalized logit function. PROC LOGISTIC fits the generalized logit model where each nonreference category is contrasted with the reference category. You can use the response variable option REF= to specify the reference category. |
| LOGIT   | the log odds function. PROC LOGISTIC fits the binary logit model when there are two response categories and fits the cumulative logit model when there are more than two response categories.                                               |
| PROBIT  | the inverse standard normal distribution function. PROC LOGISTIC fits the binary probit model when there are two response categories and fits the cumulative probit model when there are more than two response categories.                 |

## Details

### Generalized Logit Model

Formulation of the generalized logit models for nominal response variables can be found in Agresti (1990). Let  $Y$  be the response variable with categories  $1, \dots, r$ . Let  $\mathbf{x} = (x_0, x_1, \dots, x_p)'$  be a  $(p+1)$  vector of covariates, with  $x_0 \equiv 1$ . By choosing  $k$  as the reference category, the  $j$ th logit is given by

$$\log \left[ \frac{\Pr(Y = j|\mathbf{x})}{\Pr(Y = k|\mathbf{x})} \right] = \mathbf{x}'\boldsymbol{\beta}_j \quad 0 \leq j \leq r, j \neq k$$

where  $\boldsymbol{\beta}_j$  is a  $(p+1)$  vector of the regression coefficients for the  $j$ th logit.

For a sample of  $n$  subjects  $\{(y_i, \mathbf{x}_i), 1 \leq i \leq n\}$ , the log likelihood for the generalized logit model is

$$l = l(\boldsymbol{\beta}_j, 1 \leq j \leq r, j \neq k) = \sum_{i=1}^n \log(\Pr(Y = y_i|\mathbf{x}_i))$$

and the log likelihood is maximized with respect to  $\boldsymbol{\beta}_j, 1 \leq j \leq r, j \neq k$ .

#### Newton's Method of Parameter Estimation

For the convenience of notation, consider the last response level to be the reference level. The response probabilities  $\pi_1, \dots, \pi_r$  are given by

$$\begin{aligned} \pi_r \equiv \Pr(Y = r|\mathbf{x}) &= \frac{1}{1 + \sum_{l=1}^{r-1} e^{\mathbf{x}'\boldsymbol{\beta}_l}} \\ \pi_j \equiv \Pr(Y = j|\mathbf{x}) &= \pi_r e^{\mathbf{x}'\boldsymbol{\beta}_j} \quad 1 \leq j \leq r-1 \end{aligned}$$

For a single response vector  $y \equiv (y_1, \dots, y_r)'$  with  $y_j = 1$  if  $Y = j$  and 0 otherwise, the log likelihood is

$$l(\pi_1, \dots, \pi_r; \mathbf{y}) = \sum_{j=1}^r y_j \log(\pi_j)$$

Note that  $\sum_{j=1}^r y_j = 1$  and  $\sum_{j=1}^r \pi_j = 1$ , so that

$$\frac{\partial l(\pi_1, \dots, \pi_r; \mathbf{y})}{\partial \pi_j} = \frac{y_j}{\pi_j} - \frac{y_r}{\pi_r}, \quad 1 \leq j \leq r$$

The first partial derivatives of  $\pi_1, \dots, \pi_r$  with respect to  $\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_{r-1}$  are

$$\begin{aligned} \frac{\partial \pi_j}{\partial \boldsymbol{\beta}_l} &= \begin{cases} -\pi_j \pi_l \mathbf{x} & 1 \leq j \neq l \leq r-1 \\ (\pi_j - \pi_j^2) \mathbf{x} & 1 \leq j = l \leq r-1 \end{cases} \\ \frac{\partial \pi_r}{\partial \boldsymbol{\beta}_l} &= -\pi_r \pi_l \mathbf{x} \quad 1 \leq l \leq r-1 \end{aligned}$$

Denote  $\boldsymbol{\pi} = (\pi_1, \dots, \pi_{r-1})'$  and  $\boldsymbol{\pi}^* = (\pi_1, \dots, \pi_r)'$ . Then,

$$\mathbf{D} \equiv \frac{\partial(\pi_1, \dots, \pi_r)}{\partial(\beta_1, \dots, \beta_{r-1})} \equiv \begin{pmatrix} \frac{\partial' \pi_1}{\partial \beta_1} & \cdots & \frac{\partial' \pi_1}{\partial \beta_{r-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial' \pi_r}{\partial \beta_1} & \cdots & \frac{\partial' \pi_r}{\partial \beta_{r-1}} \end{pmatrix} = \begin{pmatrix} \text{diag}(\boldsymbol{\pi}) - \boldsymbol{\pi} \boldsymbol{\pi}' \\ -\pi_r \boldsymbol{\pi}' \end{pmatrix} \otimes \mathbf{x}'$$

The estimating equations become

$$\mathbf{D}' \mathbf{W} (\mathbf{y} - \boldsymbol{\pi}^*) = 0 \quad \text{where} \quad \mathbf{W} = \text{diag}(\pi_1^{-1}, \dots, \pi_r^{-1})$$

Let

$$\mathbf{H} = \mathbf{D}' \mathbf{W} \mathbf{D} = (\text{diag}(\boldsymbol{\pi}) - \boldsymbol{\pi} \boldsymbol{\pi}') \otimes \mathbf{x} \mathbf{x}'$$

$$\mathbf{g} = \mathbf{D}' \mathbf{W} (\mathbf{y} - \boldsymbol{\pi}^*) = (y_1 - \pi_1, \dots, y_{r-1} - \pi_{r-1})' \otimes \mathbf{x}$$

Let  $\boldsymbol{\beta} = (\beta_1', \dots, \beta_{r-1}')'$ . The estimate of  $\boldsymbol{\beta}$  is obtained iteratively as follows:

$$\boldsymbol{\beta}^{l+1} = \boldsymbol{\beta}^l + \left( \sum_i w_i \mathbf{H}_i \right)^{-1} \sum_i w_i \mathbf{g}_i$$

where  $i$  indexes the observations and  $w_i$  is the product of the corresponding weight and frequency.

### **Confidence Limits for the Predicted Probabilities**

By the delta method,

$$\sigma^2(\hat{\pi}_j) = \left( \frac{\partial \pi_j}{\partial \boldsymbol{\beta}} \right)' \mathbf{V}(\hat{\boldsymbol{\beta}}) \frac{\partial \pi_j}{\partial \boldsymbol{\beta}}$$

where

$$\left( \frac{\partial \pi_j}{\partial \boldsymbol{\beta}} \right)' = \left[ \left( \frac{\partial \pi_j}{\partial \beta_1} \right)', \dots, \left( \frac{\partial \pi_j}{\partial \beta_{r-1}} \right)' \right]$$

A  $100(1-\alpha)\%$  confidence level for  $\pi_j$  is given by

$$\hat{\pi}_j \pm z_{1-\alpha/2} \hat{\sigma}(\hat{\pi}_j)$$

where  $\hat{\sigma}(\hat{\pi}_j)$  is obtained by replacing  $\boldsymbol{\beta}$  by  $\hat{\boldsymbol{\beta}}$  in  $\sigma(\hat{\pi}_j)$ .

### **No Intercept Model**

When the NOINT option is specified with the LINK=GLOGIT option, all intercepts are suppressed. This differs from the cumulative model where only the first intercept is suppressed when the NOINT option is specified.

**Fit Statistics**

Suppose there are  $r$  response categories and  $s$  covariates (each dummy variable being counted as a separate covariate). The number of parameters estimated is  $p = (r - 1)(1 + s)$ . Let  $L$  be the likelihood.

Akaike Information Criterion:

$$AIC = -2 \log(L) + 2p$$

Schwartz Criterion:

$$SC = -2 \log(L) + p \log\left(\sum_j f_j\right)$$

where  $f_j$  is the frequency of the  $j$ th observation.

**Exact Conditional Analysis**

When an EXACT statement and the LINK=GLOGIT option is specified, the generalized logit model is fit as described in Hirji (1992). If there are only two response levels, the binary logit model is fit instead. Hypothesis tests for each effect are computed across logit functions, but individual parameters are estimated for each logit function.

**Association of Observed Values and Predicted Probabilities**

When the LINK=GLOGIT option is specified, the “Association of Observed Values and Predicted Probabilities” table is suppressed unless there are only two response levels; in that case, the generalized logit model is reduced to the binary logit model.

**Printing and Outputting Parameter Estimates**

Each logit function has a set of parameters for the intercept and covariates. Instead of printing and outputting the parameter estimates by logit function, PROC LOGISTIC presents all parameter estimates for the intercept first, followed by all estimates of the first covariate, etc.

Since each logit function contrasts a nonreference response category with the reference category, the “Analysis of Maximum Likelihood Estimates” table includes the response variable column whose values are used to identify the corresponding logit function.

For the OUTEST= data set, names of parameters corresponding to the nonreference category ‘xxx’ contain \_xxx as the suffix. For example, suppose the variable Net3 represents the television network viewed at a certain time, with values ‘ABC’, ‘CBS’, and ‘NBC’. The following code fits a generalized logit model with Age and Gender (a CLASS variable with values Female and Male) as explanatory variables.

```
proc logistic;
  class Gender;
  model Net3 = Age Gender / link=glogit;
run;
```

Since ‘NBC’ is the last value in the sorted order of the response categories, it corresponds to the default reference category. There are two logit functions, one contrasting ‘ABC’ with ‘NBC’ and the other contrasting ‘CBS’ with ‘NBC’. For each



logit, there are three parameters: an intercept parameter, a slope parameter for Age, and a slope parameter for Gender (since there are only two gender levels and the EFFECT parameterization is used by default). The names of the parameters and their descriptions are as follows.

| Parameter        | Description                                                        |
|------------------|--------------------------------------------------------------------|
| Intercept_ABC    | intercept parameter for the logit contrasting 'ABC' with 'NBC'     |
| Intercept_CBS    | intercept parameter for the logit contrasting 'CBS' with 'NBC'     |
| Age_ABC          | Age parameter for the logit contrasting 'ABC' with 'NBC'           |
| Age_CBS          | Age parameter for the logit contrasting 'CBS' with 'NBC'           |
| GenderFemale_ABC | Gender=Female parameter for the logit contrasting 'ABC' with 'NBC' |
| GenderFemale_CBS | Gender=Female parameter for the logit contrasting 'CBS' with 'NBC' |

### **Out= Output Data Set**

If any of the XBETA=, STDXBETA=, PREDICTED=, LOWER=, and UPPER= options are specified in the OUTPUT statement when there are more than two response categories, each input observation generates as many output observations as the number of response categories. The predicted probabilities and their confidence limits correspond to the probabilities of individual response categories rather than the cumulative probabilities as in the case of fitting a cumulative model. Regression diagnostics are suppressed when there are more than two response categories. You can specify PREDPROB=(I C) to obtain the predicted probabilities of individual response categories as well as the predicted cumulative probabilities.

---

### **Intercept Names for Ordinal Models in OUTEST= Data set**

If there are only two response categories in the entire data set, the intercept parameter is named Intercept; otherwise, the intercept parameters are named Intercept\_XXX, where 'XXX' is the value (formatted if a format is applied) of the corresponding response category.

---

## **Exact Conditional Analysis: Computational Resources**

PROC LOGISTIC uses a relatively fast and efficient algorithm for the exact analyses (Hirji, Mehta, and Patel 1987; Hirji 1992). This recently developed algorithm, together with improvements in computer power, makes it feasible to perform exact computations for data sets where previously only asymptotic methods could be applied, and for data sets where maximum likelihood methods fail to converge. Nevertheless, many problems require a prohibitive amount of time and memory for exact computations, depending on the speed and memory available on your computer. For such problems, consider whether exact methods are really necessary. Stokes, Davis,

and Koch (2000) suggest looking at exact  $p$ -values when the sample size is small and the approximate  $p$ -values from the unconditional analysis are less than 0.10.

A formula does not exist that can predict the amount of time and memory necessary to generate the exact conditional distributions for a particular problem. The time and memory required depends on several factors, including the total sample size, the number of parameters of interest, the number of nuisance parameters, and the order in which the parameters are processed. At any time while PROC LOGISTIC is deriving the distributions, you can terminate the computations by pressing the system interrupt key sequence (refer to the SAS Companion for your system) and choosing to stop computations. If you run out of memory, refer to the SAS Companion for your system to see how to allocate more.

You can use the MAXTIME= option in the EXACTOPTIONS option to limit the total amount of time PROC LOGISTIC uses to derive all of the exact distributions. If PROC LOGISTIC does not finish within that time, the procedure terminates. If you need to derive several distributions, it may be more feasible to request one distribution at a time.

You can monitor the progress of the procedure by submitting your program with the STATUSTIME= option in the EXACTOPTIONS option. If the procedure is too slow, you can try reordering the variables in the MODEL statement or reparameterizing your classification variables (Hirji, Mehta, and Patel 1987). This may help since processing large covariate values first enables the infeasibility criteria to reject paths earlier in the algorithm; for example, a continuous variable clustered around a few values may reject paths earlier than a more uniformly distributed covariate (refer to Derr (2000) for an illustration of how the algorithm works for a simple data set). PROC LOGISTIC attempts to speed up computations by first combining observations with the same covariate values, then sorting the dummy variables for the class variables you are conditioning on followed by the continuous variables being conditioned on, in order of their appearance in the MODEL statement. Each variable  $X_i$  is sorted based on the weights  $x_{ij}w_j$ , where  $i$  indexes the variable and  $j$  indexes the observation; for example, if  $\sum_j 1_{\{x_{ij}>0\}}x_{ij}w_j > \sum_j 1_{\{x_{ij}<0\}}|x_{ij}|w_j$ , then the positive values of  $x_{ij}$  are sorted in descending order followed by the negative values in ascending order followed by the zeros.

---

## Example

---

### Example 8.1. Generalized Logit Model

Halloween trick-or-treaters are given a choice of candy in three bowls: one bowl contains small chocolate candy bars, one contains lollipops, and the last contains sugar candies. The children are classified by gender and by apparent age (child and teenager), and the following candy preferences are observed:

| Gender | Age      | Candy     |          |       |
|--------|----------|-----------|----------|-------|
|        |          | chocolate | lollipop | sugar |
| boy    | child    | 2         | 13       | 3     |
| boy    | teenager | 10        | 9        | 3     |
| girl   | child    | 3         | 9        | 1     |
| girl   | teenager | 8         | 0        | 1     |

Interest centers on whether age or gender affects the choice of type of candy. A generalized logit model can be fit to relate these three factors. Since the data set may be too small for the asymptotic analysis to be valid, an exact analysis is also performed. The following statements perform this analysis.

```
data halloween;
  format Candy $9.;
  input Gender $ Age $ Candy $ count @@;
  datalines;
boy  child chocolate 2   boy  teenager chocolate 10
boy  child lollipop 13  boy  teenager lollipop 9
boy  child sugar 3     boy  teenager sugar 3
girl child chocolate 3  girl teenager chocolate 8
girl child lollipop 9   girl teenager lollipop 0
girl child sugar 1     girl teenager sugar 1
;
proc logistic data=halloween;
  freq count;
  class Gender(ref='girl') Age(ref='child') / param=ref;
  model Candy(ref='chocolate') = Gender Age / link=glogit;
  exact Gender Age / joint estimate=both;
run;
```

Reference levels for both Gender and Age are declared in the CLASS statement, while the reference level for Candy is specified in the MODEL statement. Since the response is nominal, a generalized logit model is fit by specifying the LINK=GLOGIT. For the exact analysis, a joint test for the parameters Gender and Age is requested, conditional on the intercepts. Output 8.1.1 through Output 8.1.6 display the results of the analyses.

#### Output 8.1.1.

| The LOGISTIC Procedure                                                                                         |                   |
|----------------------------------------------------------------------------------------------------------------|-------------------|
| Model Information                                                                                              |                   |
| Data Set                                                                                                       | WORK.HALLOWEEN    |
| Response Variable                                                                                              | Candy             |
| Number of Response Levels                                                                                      | 3                 |
| Number of Observations                                                                                         | 11                |
| Frequency Variable                                                                                             | count             |
| Sum of Frequencies                                                                                             | 62                |
| Model                                                                                                          | generalized logit |
| Optimization Technique                                                                                         | Fisher's scoring  |
| NOTE: 1 observation having zero frequency or weight was excluded since it does not contribute to the analysis. |                   |

**Output 8.1.2.**

| Response Profile                                                |           |                 |
|-----------------------------------------------------------------|-----------|-----------------|
| Ordered Value                                                   | Candy     | Total Frequency |
| 1                                                               | chocolate | 23              |
| 2                                                               | lollipop  | 31              |
| 3                                                               | sugar     | 8               |
| Logits modeled use Candy='chocolate' as the reference category. |           |                 |

The “Response Profile” table (Output 8.1.2) indicates that ‘chocolate’ is the reference category for the Candy variable, so the logits being modeled are

$$\log \left( \frac{\Pr(\text{Candy} = \text{lollipop})}{\Pr(\text{Candy} = \text{chocolate})} \right) \quad \text{and} \quad \log \left( \frac{\Pr(\text{Candy} = \text{sugar})}{\Pr(\text{Candy} = \text{chocolate})} \right)$$

**Output 8.1.3.**

| Class Level Information |          |                  |
|-------------------------|----------|------------------|
|                         |          | Design Variables |
| Class                   | Value    | 1                |
| Gender                  | boy      | 1                |
|                         | girl     | 0                |
| Age                     | child    | 0                |
|                         | teenager | 1                |

The “Class Level Information” table (Output 8.1.3) shows that ‘girl’ and ‘child’ are the reference levels for Gender and Age, respectively.

**Output 8.1.4.** Asymptotic Results

| Model Convergence Status                      |                |                          |
|-----------------------------------------------|----------------|--------------------------|
| Convergence criterion (GCONV=1E-8) satisfied. |                |                          |
| Model Fit Statistics                          |                |                          |
| Criterion                                     | Intercept Only | Intercept and Covariates |
| AIC                                           | 125.354        | 114.448                  |
| SC                                            | 129.608        | 127.210                  |
| -2 Log L                                      | 121.354        | 102.448                  |

**Output 8.1.4.** (continued)

| Testing Global Null Hypothesis: BETA=0 |            |    |            |
|----------------------------------------|------------|----|------------|
| Test                                   | Chi-Square | DF | Pr > ChiSq |
| Likelihood Ratio                       | 18.9061    | 4  | 0.0008     |
| Score                                  | 16.9631    | 4  | 0.0020     |
| Wald                                   | 12.8115    | 4  | 0.0122     |

| Type III Analysis of Effects |    |                    |            |
|------------------------------|----|--------------------|------------|
| Effect                       | DF | Wald<br>Chi-Square | Pr > ChiSq |
| Gender                       | 2  | 4.7168             | 0.0946     |
| Age                          | 2  | 12.2325            | 0.0022     |

All of the hypothesis tests in Output 8.1.4 show that the model fits, although the Type III tests indicate that Gender has marginal influence.

**Output 8.1.5.** Asymptotic Results (continued)

| Analysis of Maximum Likelihood Estimates |          |    |          |                   |                    |            |
|------------------------------------------|----------|----|----------|-------------------|--------------------|------------|
| Parameter                                | Candy    | DF | Estimate | Standard<br>Error | Wald<br>Chi-Square | Pr > ChiSq |
| Intercept                                | lollipop | 1  | 0.7698   | 0.5782            | 1.7722             | 0.1831     |
| Intercept                                | sugar    | 1  | -0.9033  | 0.8664            | 1.0869             | 0.2972     |
| Gender boy                               | lollipop | 1  | 1.5758   | 0.7569            | 4.3347             | 0.0373     |
| Gender boy                               | sugar    | 1  | 1.5261   | 1.0158            | 2.2570             | 0.1330     |
| Age teenager                             | lollipop | 1  | -2.6472  | 0.7572            | 12.2212            | 0.0005     |
| Age teenager                             | sugar    | 1  | -1.7416  | 0.9623            | 3.2754             | 0.0703     |

| Odds Ratio Estimates  |          |                   |                               |        |
|-----------------------|----------|-------------------|-------------------------------|--------|
| Effect                | Candy    | Point<br>Estimate | 95% Wald<br>Confidence Limits |        |
| Gender boy vs girl    | lollipop | 4.835             | 1.097                         | 21.313 |
| Gender boy vs girl    | sugar    | 4.600             | 0.628                         | 33.686 |
| Age teenager vs child | lollipop | 0.071             | 0.016                         | 0.313  |
| Age teenager vs child | sugar    | 0.175             | 0.027                         | 1.155  |

The parameter estimates and odds ratios for the asymptotic analysis are displayed in Output 8.1.5, and show that the odds of choosing a lollipop over a chocolate bar are five ( $4.835 \approx 5$ ) times higher for boys versus girls, and a child is 14 ( $1/0.071 \approx 14$ ) times more likely than a teenager to choose a lollipop over a chocolate bar.

Note in the “Analysis of Maximum Likelihood Estimates” table that the dummy parameters for the class variables are labeled by their nonreference level, and that the “Candy” column indicates the nonreference response category for the logit.

**Output 8.1.6.** Exact Results

| Exact Conditional Analysis |          |             |           |                       |         |         |
|----------------------------|----------|-------------|-----------|-----------------------|---------|---------|
| Conditional Exact Tests    |          |             |           |                       |         |         |
|                            |          |             |           | --- p-Value ---       |         |         |
|                            | Effect   | Test        | Statistic | Exact                 | Mid     |         |
|                            | Joint    | Score       | 16.6895   | 0.0013                | 0.0013  |         |
|                            |          | Probability | 7.115E-7  | 0.0009                | 0.0009  |         |
|                            | Gender   | Score       | 5.0830    | 0.0870                | 0.0835  |         |
|                            |          | Probability | 0.00697   | 0.0988                | 0.0953  |         |
|                            | Age      | Score       | 14.5093   | 0.0003                | 0.0003  |         |
|                            |          | Probability | 0.000032  | 0.0003                | 0.0003  |         |
| Exact Parameter Estimates  |          |             |           |                       |         |         |
| Parameter                  |          | Candy       | Estimate  | 95% Confidence Limits |         | p-Value |
| Gender                     | boy      | lollipop    | 1.5017    | -0.0692               | 3.4081  | 0.0641  |
| Gender                     | boy      | sugar       | 1.4114    | -0.7079               | 4.0869  | 0.2715  |
| Age                        | teenager | lollipop    | -2.5231   | -4.4303               | -0.9979 | 0.0002  |
| Age                        | teenager | sugar       | -1.6244   | -3.9734               | 0.5146  | 0.1673  |
| Exact Odds Ratios          |          |             |           |                       |         |         |
| Parameter                  |          | Candy       | Estimate  | 95% Confidence Limits |         | p-Value |
| Gender                     | boy      | lollipop    | 4.489     | 0.933                 | 30.209  | 0.0641  |
| Gender                     | boy      | sugar       | 4.102     | 0.493                 | 59.553  | 0.2715  |
| Age                        | teenager | lollipop    | 0.080     | 0.012                 | 0.369   | 0.0002  |
| Age                        | teenager | sugar       | 0.197     | 0.019                 | 1.673   | 0.1673  |

The exact analysis (Output 8.1.6) produces results similar to the asymptotic analysis. The exact score statistic for the joint test is very close to the asymptotic global test, and the parameter estimates and odds ratios are quite similar. However, the contrast between boys and girls is only marginally significant.

## References

- Agresti, A. (1990), *Categorical Data Analysis*, New York: John Wiley & Sons, Inc.
- Derr, R.E. (2000), "Performing Exact Logistic Regression with the SAS System," *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc.
- Hirji, K.F. (1992), "Computing Exact Distributions for Polytomous Response Data," *JASA*, 487–492.
- Hirji, K.F., Mehta, C.R., and Patel, N.R. (1987), "Computing Distributions for Exact Logistic Regression," *JASA*, 82, 1110–1117.
- Stokes, M.E., Davis, C.S., and Koch, G.G. (2000), *Categorical Data Analysis Using the SAS System*, Second Edition, Cary, NC: SAS Institute Inc.

# Chapter 9

## The MI Procedure

### Chapter Table of Contents

---

|                                                      |     |
|------------------------------------------------------|-----|
| <b>OVERVIEW</b>                                      | 131 |
| <b>GETTING STARTED</b>                               | 133 |
| <b>SYNTAX</b>                                        | 137 |
| PROC MI Statement                                    | 138 |
| BY Statement                                         | 141 |
| EM Statement                                         | 141 |
| FREQ Statement                                       | 142 |
| MCMC Statement                                       | 143 |
| MONOTONE Statement                                   | 149 |
| TRANSFORM Statement                                  | 150 |
| VAR Statement                                        | 151 |
| <b>DETAILS</b>                                       | 152 |
| Descriptive Statistics                               | 152 |
| EM Algorithm for Data with Missing Values            | 153 |
| Statistical Assumptions for Multiple Imputation      | 154 |
| Missing Data Patterns                                | 155 |
| Imputation Mechanisms                                | 156 |
| Regression Method for Monotone Missing Data          | 157 |
| Propensity Score Method for Monotone Missing Data    | 158 |
| MCMC Method for Arbitrary Missing Data               | 159 |
| Producing Monotone Missingness with the MCMC Method  | 164 |
| MCMC Method Specifications                           | 166 |
| Convergence in MCMC                                  | 167 |
| Input Data Sets                                      | 170 |
| Output Data Sets                                     | 171 |
| Combining Inferences from Multiply Imputed Data Sets | 173 |
| Multiple Imputation Efficiency                       | 174 |
| Imputer's Model Versus Analyst's Model               | 174 |
| Parameter Simulation Versus Multiple Imputation      | 175 |
| ODS Table Names                                      | 176 |
| <b>EXAMPLES</b>                                      | 177 |
| Example 9.1 EM Algorithm for MLE                     | 177 |

|                                                                |            |
|----------------------------------------------------------------|------------|
| Example 9.2 Propensity Score Method . . . . .                  | 181        |
| Example 9.3 Regression Method . . . . .                        | 184        |
| Example 9.4 MCMC Method . . . . .                              | 185        |
| Example 9.5 Producing Monotone Missingness with MCMC . . . . . | 188        |
| Example 9.6 Checking Convergence in MCMC . . . . .             | 191        |
| Example 9.7 Transformation to Normality . . . . .              | 194        |
| Example 9.8 Saving and Using Parameters for MCMC . . . . .     | 198        |
| <b>REFERENCES . . . . .</b>                                    | <b>199</b> |



## Chapter 9

# The MI Procedure

---

### Overview

The experimental MI procedure performs multiple imputation of missing data. Missing values are an issue in a substantial number of statistical analyses. Most SAS statistical procedures exclude observations with any missing variable values from the analysis. These observations are called incomplete cases. While analyzing only complete cases has its simplicity, the information contained in the incomplete cases is lost. This approach also ignores possible systematic differences between the complete cases and the incomplete cases, and the resulting inference may not be applicable to the population of all cases, especially with a smaller number of complete cases.

Some SAS procedures use all the available cases in an analysis, that is, cases with available information. For example, the CORR procedure estimates a variable mean by using all cases with nonmissing values for this variable, ignoring the possible missing values in other variables. PROC CORR also estimates a correlation by using all cases with nonmissing values for this pair of variables. This makes better use of the available data, but the resulting correlation matrix may not be positive definite.

Another strategy for handling missing data is simple imputation, which substitutes a value for each missing value. Standard statistical procedures for complete data analysis can then be used with the filled-in data set. For example, each missing value can be imputed with the variable mean of the complete cases, or it can be imputed with the mean conditional on observed values of other variables. This approach treats missing values as if they were known in the complete-data analysis. However, single imputation does not reflect the uncertainty about the predictions of the unknown missing values, and the resulting estimated variances of the parameter estimates will be biased toward zero (Rubin 1987, p. 13).

Instead of filling in a single value for each missing value, multiple imputation (Rubin 1976; 1987) replaces each missing value with a set of plausible values that represent the uncertainty about the right value to impute. The multiply imputed data sets are then analyzed by using standard procedures for complete data and combining the results from these analyses. No matter which complete-data analysis is used, the process of combining results from different data sets is essentially the same.

Multiple imputation does not attempt to estimate each missing value through simulated values but rather to represent a random sample of the missing values. This process results in valid statistical inferences that properly reflect the uncertainty due to missing values; for example, confidence intervals with the correct probability coverage.

Multiple imputation inference involves three distinct phases:

1. The missing data are filled in  $m$  times to generate  $m$  complete data sets.
2. The  $m$  complete data sets are analyzed using standard statistical analyses.
3. The results from the  $m$  complete data sets are combined to produce inferential results.

The new MI procedure creates multiply imputed data sets for incomplete multivariate data. It uses methods that incorporate appropriate variability across the  $m$  imputations. The method of choice depends on the patterns of missingness. A data set with variables  $Y_1, Y_2, \dots, Y_p$  (in that order) is said to have a monotone missing pattern when the event that a variable  $Y_j$  is missing for a particular individual implies that all subsequent variables  $Y_k, k > j$ , are missing for that individual.

For data sets with monotone missing patterns, either a parametric regression method (Rubin 1987) that assumes multivariate normality or a nonparametric method that uses propensity scores (Rubin 1987; Lavori, Dawson, and Shera 1995) is appropriate. For data sets with arbitrary missing patterns, a Markov Chain Monte Carlo (MCMC) method (Schafer 1997) that assumes multivariate normality is used to impute all missing values or just enough missing values to make the imputed data sets have monotone missing patterns.

Once the  $m$  complete data sets are analyzed using standard SAS procedures, the new MIANALYZE procedure can be used to generate valid statistical inferences about these parameters by combining results from the  $m$  analyses. These two procedures are available in experimental form in Release 8.2 of the SAS System.

Often, as few as three to five imputations are adequate in multiple imputation (Rubin 1996, p. 480). The relative efficiency of the small  $m$  imputation estimator is high for cases with little missing information (Rubin 1987, p. 114). Also see the “Multiple Imputation Efficiency” section on page 174.

Multiple imputation inference assumes that the model (variables) you used to analyze the multiply imputed data (the analyst’s model) is the same as the model used to impute missing values in multiple imputation (the imputer’s model). But in practice, the two models may not be the same. The consequence for different scenarios (Schafer 1997, pp. 139–143) is discussed in the “Imputer’s Model Versus Analyst’s Model” section on page 174.

In addition to the multiple imputation method, a simulation-based method of parameter simulation can also be used to analyze the data for many incomplete-data problems. Although the MI procedure does not offer a simulation-based method of parameter simulation, the choice between the two methods (Schafer 1997, pp. 89–90, 135–136) is examined in the “Parameter Simulation Versus Multiple Imputation” section on page 175.

## Getting Started

Consider the following Fitness data set that has been altered to contain an arbitrary pattern of missingness:

```
*----- Data on Physical Fitness -----*
| These measurements were made on men involved in a physical |
| fitness course at N.C. State University.                   |
| Only selected variables of                                |
| Oxygen (oxygen intake, ml per kg body weight per minute), |
| Runtime (time to run 1.5 miles in minutes), and           |
| RunPulse (heart rate while running) are used.             |
| Certain values were changed to missing for the analysis.  |
*-----*
```

```
data FitMiss;
  input Oxygen RunTime RunPulse @@;
  datalines;
44.609 11.37 178      45.313 10.07 185
54.297 8.65 156      59.571 . .
49.874 9.22 .        44.811 11.63 176
.      11.95 176      .      10.85 .
39.442 13.08 174     60.055 8.63 170
50.541 . .          37.388 14.03 186
44.754 11.12 176     47.273 . .
51.855 10.33 166     49.156 8.95 180
40.836 10.95 168     46.672 10.00 .
46.774 10.25 .       50.388 10.08 168
39.407 12.63 174     46.080 11.17 156
45.441 9.63 164      .      8.92 .
45.118 11.08 .       39.203 12.88 168
45.790 10.47 186     50.545 9.93 148
48.673 9.40 186      47.920 11.50 170
47.467 10.50 170
;
```

Suppose that the data are multivariate normally distributed and the missing data are missing at random (MAR). That is, the probability that an observation is missing can depend on the observed variable values of the individual, but not on the missing variable values of the individual. See the “Statistical Assumptions for Multiple Imputation” section on page 154 for a detailed description of the MAR assumption.

The following statements invoke the MI procedure and impute missing values for the FitMiss data set.

```
proc mi data=FitMiss seed=37851 mu0=50 10 180 out=outmi;
  var Oxygen RunTime RunPulse;
run;
```

| The MI Procedure                 |                   |
|----------------------------------|-------------------|
| Model Information                |                   |
| Data Set                         | WORK.FITMISS      |
| Method                           | MCMC              |
| Multiple Imputation Chain        | Single Chain      |
| Initial Estimates for MCMC       | EM Posterior Mode |
| Start                            | Starting Value    |
| Prior                            | Jeffreys          |
| Number of Imputations            | 5                 |
| Number of Burn-in Iterations     | 200               |
| Number of Iterations             | 100               |
| Seed for random number generator | 37851             |

**Figure 9.1.** Model Information

The “Model Information” table describes the method used in the multiple imputation process. By default, the procedure uses the Markov Chain Monte Carlo (MCMC) method with a single chain to create five imputations. The posterior mode, the highest observed-data posterior density, with a noninformative prior, is computed from the EM algorithm and is used as the starting value for the chain.

The MI procedure takes 200 burn-in iterations before the first imputation and 100 iterations between imputations. In a Markov chain, the information in the current iteration has influence on the state of the next iteration. The burn-in iterations are iterations in the beginning of each chain that are used both to eliminate the series of dependence on the starting value of the chain and to achieve the stationary distribution. The between-imputation iterations in a single chain are used to eliminate the series of dependence between the two imputations.

| The MI Procedure      |        |          |           |      |         |
|-----------------------|--------|----------|-----------|------|---------|
| Missing Data Patterns |        |          |           |      |         |
| Group                 | Oxygen | Run Time | Run Pulse | Freq | Percent |
| 1                     | X      | X        | X         | 21   | 67.74   |
| 2                     | X      | X        | .         | 4    | 12.90   |
| 3                     | X      | .        | .         | 3    | 9.68    |
| 4                     | .      | X        | X         | 1    | 3.23    |
| 5                     | .      | X        | .         | 2    | 6.45    |

| Missing Data Patterns |                       |           |            |
|-----------------------|-----------------------|-----------|------------|
| Group                 | -----Group Means----- |           |            |
|                       | Oxygen                | RunTime   | RunPulse   |
| 1                     | 46.353810             | 10.809524 | 171.666667 |
| 2                     | 47.109500             | 10.137500 | .          |
| 3                     | 52.461667             | .         | .          |
| 4                     | .                     | 11.950000 | 176.000000 |
| 5                     | .                     | 9.885000  | .          |

**Figure 9.2.** Missing Data Patterns

The “Missing Data Patterns” table lists distinct missing data patterns with corresponding frequencies and percents. Here, an “X” means that the variable is observed in the corresponding group and a “.” means that the variable is missing. The table also displays group-specific variable means. The MI procedure sorts the data into groups based on whether an individual’s value is observed or missing for each variable to be analyzed. For a detailed description of missing data patterns, see the “Missing Data Patterns” section on page 155.

| The MI Procedure                         |                               |                              |          |        |
|------------------------------------------|-------------------------------|------------------------------|----------|--------|
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | -----Variance-----            |                              |          | DF     |
|                                          | Between                       | Within                       | Total    |        |
| Oxygen                                   | 0.045321                      | 0.937239                     | 0.991624 | 26.113 |
| RunTime                                  | 0.005853                      | 0.072217                     | 0.079241 | 24.45  |
| RunPulse                                 | 0.611864                      | 3.247163                     | 3.981400 | 19.227 |
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | Relative Increase in Variance | Fraction Missing Information |          |        |
| Oxygen                                   | 0.058027                      | 0.056263                     |          |        |
| RunTime                                  | 0.097265                      | 0.092202                     |          |        |
| RunPulse                                 | 0.226116                      | 0.197941                     |          |        |

**Figure 9.3.** Variance Information

After the completion of  $m$  imputations, the “Multiple Imputation Variance Information” table displays the between-imputation variance, within-imputation variance, and total variance for combining complete-data inferences. It also displays the degrees of freedom for the total variance. The relative increase in variance due to missing values and the fraction of missing information for each variable are also displayed. A detailed description of these statistics is provided in the “Combining Inferences from Multiply Imputed Data Sets” section on page 173.

The following “Multiple Imputation Parameter Estimates” table displays the estimated mean and standard error of the mean for each variable. The inferences are based on the  $t$  distribution. The table also displays a 95% confidence interval for the mean and a  $t$  statistic with the associated  $p$ -value for the hypothesis that the population mean is equal to the value specified with the MU0= option. A detailed description of these statistics is provided in the “Combining Inferences from Multiply Imputed Data Sets” section on page 173.

| The MI Procedure                        |            |            |                       |                       |         |
|-----------------------------------------|------------|------------|-----------------------|-----------------------|---------|
| Multiple Imputation Parameter Estimates |            |            |                       |                       |         |
| Variable                                | Mean       | Std Error  | 95% Confidence Limits |                       | DF      |
| Oxygen                                  | 47.126919  | 0.995803   | 45.0804               | 49.1734               | 26.113  |
| RunTime                                 | 10.546494  | 0.281498   | 9.9661                | 11.1269               | 24.45   |
| RunPulse                                | 171.621676 | 1.995344   | 167.4487              | 175.7946              | 19.227  |
| Multiple Imputation Parameter Estimates |            |            |                       |                       |         |
| Variable                                | Minimum    | Maximum    | Mu0                   | t for H0:<br>Mean=Mu0 | Pr >  t |
| Oxygen                                  | 46.849494  | 47.318758  | 50.000000             | -2.89                 | 0.0077  |
| RunTime                                 | 10.464123  | 10.669193  | 10.000000             | 1.94                  | 0.0638  |
| RunPulse                                | 170.623678 | 172.680679 | 180.000000            | -4.20                 | 0.0005  |

**Figure 9.4.** Parameter Estimates

In addition to the output tables, the procedure also creates a data set with imputed values. The imputed data sets are stored in the `outmi` data set, with the index variable `_Imputation_` indicating the imputation numbers. The data set can now be analyzed using standard statistical procedures with `_Imputation_` as a BY variable.

The following statements list the first ten observations of data set `outmi`.

```
proc print data=outmi (obs=10);
  title 'First 10 Observations of the Imputed Data Set';
run;
```

| First 10 Observations of the Imputed Data Set |              |         |         |              |
|-----------------------------------------------|--------------|---------|---------|--------------|
| Obs                                           | _Imputation_ | Oxygen  | RunTime | Run<br>Pulse |
| 1                                             | 1            | 44.6090 | 11.3700 | 178.000      |
| 2                                             | 1            | 45.3130 | 10.0700 | 185.000      |
| 3                                             | 1            | 54.2970 | 8.6500  | 156.000      |
| 4                                             | 1            | 59.5710 | 6.1569  | 138.583      |
| 5                                             | 1            | 49.8740 | 9.2200  | 164.163      |
| 6                                             | 1            | 44.8110 | 11.6300 | 176.000      |
| 7                                             | 1            | 46.0264 | 11.9500 | 176.000      |
| 8                                             | 1            | 42.3040 | 10.8500 | 182.486      |
| 9                                             | 1            | 39.4420 | 13.0800 | 174.000      |
| 10                                            | 1            | 60.0550 | 8.6300  | 170.000      |

**Figure 9.5.** Imputed Data Set

The table shows that the precision of the imputed values differs from the precision of the observed values. You can use the `ROUND=` option to make the imputed values consistent with the observed values.

## Syntax

---

The following statements are available in PROC MI.

```
PROC MI < options > ;
    BY variables ;
    EM < options > ;
    FREQ variable ;
    MCMC < options > ;
    MONOTONE < options > ;
    TRANSFORM transform ( variables < / options > )
        < ... transform ( variables < / options > ) > ;
    VAR variables ;
```

The BY statement specifies groups in which separate multiple imputation analyses are performed.

The EM statement uses the EM algorithm to compute the maximum likelihood estimate (MLE) of the data with missing values, assuming a multivariate normal distribution for the data.

The FREQ statement specifies the variable that represents the frequency of occurrence for other values in the observation.

The MCMC statement uses a Markov chain Monte Carlo method to impute values for a data set with an arbitrary missing pattern. The MONOTONE statement uses either a parametric regression method or a nonparametric method based on propensity scores to impute values for a data set with a monotone missing pattern. Note that you can use either an MCMC statement or a MONOTONE statement, but not both. When neither of these two statements is specified, the MCMC method with its default options is used.

The TRANSFORM statement lists the variables to be transformed before the imputation process. The imputed values of these transformed variables will be reverse-transformed to the original forms before the imputation.

The VAR statement lists the numeric variables to be analyzed. If you omit the VAR statement, all numeric variables not listed in other statements are used.

The PROC MI statement is the only required statement for the MI procedure. The rest of this section provides detailed syntax information for each of these statements, beginning with the PROC MI statement. The remaining statements are in alphabetical order.

## PROC MI Statement

**PROC MI** < options > ;

The following table summarizes the options available in the PROC MI statement.

**Table 9.1.** Summary of PROC MI Options

| Tasks                                             | Options   |
|---------------------------------------------------|-----------|
| <b>Specify data sets</b>                          |           |
| input data set                                    | DATA=     |
| output data set with imputed values               | OUT=      |
| <b>Specify imputation details</b>                 |           |
| number of imputations                             | NIMPUTE=  |
| seed to begin random number generator             | SEED=     |
| units to round imputed variable values            | ROUND=    |
| maximum values for imputed variable values        | MAXIMUM=  |
| minimum values for imputed variable values        | MINIMUM=  |
| singularity tolerance                             | SINGULAR= |
| <b>Specify statistical analysis</b>               |           |
| level for the confidence interval, $(1 - \alpha)$ | ALPHA=    |
| means under the null hypothesis                   | MU0=      |
| <b>Control printed output</b>                     |           |
| suppress all displayed output                     | NOPRINT   |
| displays univariate statistics and correlations   | SIMPLE    |

The following options can be used in the PROC MI statement (in alphabetical order):

**ALPHA=** $\alpha$

specifies that confidence limits be constructed for the mean estimates with confidence level  $100(1 - \alpha)\%$ , where  $0 < \alpha < 1$ . The default is ALPHA=0.05.

**DATA=***SAS-data-set*

names the SAS data set to be analyzed by PROC MI. By default, the procedure uses the most recently created SAS data set.

**MAXIMUM=***numbers*

specifies maximum values for imputed variables. When an intended imputed value is greater than the maximum, PROC MI redraws another value for imputation. If only one number is specified, that number is used for all variables. If more than one number is specified, you must use a VAR statement, and the specified numbers must correspond to variables in the VAR statement. A missing value indicates no restriction on the maximum for the corresponding variable. The default is MAXIMUM=., no restriction on the maximum.



The **MAXIMUM=** option is related to the **MINIMUM=** and **ROUND=** options, which are used to make the imputed values more consistent with the observed variable values. These options are not applicable if you specify the **METHOD=PROPENSITY** option in the **MONOTONE** statement.

When specifying a maximum for the first variable only, you must also specify a missing value after the maximum. Otherwise, the maximum is used for all variables. For example, the **MAXIMUM= 100 .** option sets a maximum of 100 for the first analysis variable only and no maximum for the remaining variables. The **MAXIMUM= . 100** option sets a maximum of 100 for the second analysis variable only and no maximum for the other variables.

**MINIMUM=numbers**

specifies the minimum values for imputed variables. When an intended imputed value is less than the minimum, PROC MI redraws another value for imputation. If only one number is specified, that number is used for all variables. If more than one number is specified, you must use a **VAR** statement, and the specified numbers must correspond to variables in the **VAR** statement. A missing value indicates no restriction on the minimum for the corresponding variable. The default is **MINIMUM=.**, no restriction on the minimum.

**MU0=numbers**

**THETA0=numbers**

specifies the parameter values  $\mu_0$  under the null hypothesis  $\mu = \mu_0$  for the population means corresponding to the analysis variables. Each hypothesis is tested with a  $t$  test. If only one number is specified, that number is used for all variables. If more than one number is specified, you must use a **VAR** statement, and the specified numbers must correspond to variables in the **VAR** statement. The default is **MU0=0**.

If a variable is transformed as specified in a **TRANSFORM** statement, then the same transformation for that variable is also applied to its corresponding specified **MU0=** value in the  $t$  test. If the parameter values  $\mu_0$  for a transformed variable is not specified, then  $\mu_0 = 0$  is used for that transformed variable.

**NIMPUTE=number**

specifies the number of imputations. The default is **NIMPUTE=5**. You can specify **NIMPUTE=0** to skip the imputation. In this case, only tables of model information, missing data patterns, descriptive statistics (**SIMPLE** option), and MLE from the EM algorithm (**EM** statement) are displayed.

**NOPRINT**

suppresses the display of all output. Note that this option temporarily disables the Output Delivery System (ODS). For more information, refer to the chapter “Using the Output Delivery System” in the *SAS/STAT User’s Guide, Version 8*.

**OUT=SAS-data-set**

creates an output SAS data set containing imputation results. The data set includes an index variable, `_Imputation_`, to identify the imputation number. For each imputation, the data set contains all variables in the input data set with missing values replaced by the imputed values. See the “Output Data Sets” section on page 171 for a description of this data set.

If you want to create a permanent SAS data set, you must specify a two-level name. For more information on permanent SAS data sets, refer to the section “SAS Files” in *SAS Language Reference: Concepts, Version 8*.

**ROUND=numbers**

specifies the units to round variables in the imputation. If only one number is specified, that number is used for all variables. If more than one number is specified, you must use a VAR statement, and the specified numbers must correspond to variables in the VAR statement. The default number is a missing value, which indicates no rounding for imputed variables.

When specifying a roundoff unit for the first variable only, you must also specify a missing value after the roundoff unit. Otherwise, the roundoff unit is used for all variables. For example, the option “ROUND= 10 .” sets a roundoff unit of 10 for the first analysis variable only and no rounding for the remaining variables. The option “ROUND= . 10” sets a roundoff unit of 10 for the second analysis variable only and no rounding for other variables.

You can use the ROUND= option to set the precision of imputed values. For example, with a roundoff unit of 0.001, each value is rounded to the nearest multiple of 0.001. That is, each value has three significant digits after the decimal point. See Example 9.3 for a usage of this option.

**SEED=number**

specifies a positive integer. PROC MI uses the value of the SEED= option to start the pseudo-random number generator. The default is a value generated from reading the time of day from the computer’s clock. However, in order to duplicate the results under identical situations, you must control the value of the seed explicitly rather than rely on the clock reading.

The seed information is displayed in the “Model Information” table so that the results can be reproduced by specifying this seed with the SEED= option. You need to specify the same seed number in the future to reproduce the results.

**SIMPLE**

displays simple descriptive univariate statistics and pairwise correlations from available cases. For a detailed description of these statistics, see the “Descriptive Statistics” section on page 152.

**SINGULAR=p**

specifies the criterion for determining the singularity of a covariance matrix, where  $0 < p < 1$ . The default is SINGULAR=1E-8.

Suppose that  $\mathbf{S}$  is a covariance matrix and  $v$  is the number of variables in  $\mathbf{S}$ . Based on the spectral decomposition  $\mathbf{S} = \mathbf{\Gamma}\mathbf{\Lambda}\mathbf{\Gamma}'$ , where  $\mathbf{\Lambda}$  is a diagonal matrix of eigenvalues  $\lambda_j$ ,  $j = 1, \dots, v$ , where  $\lambda_i \geq \lambda_j$  when  $i < j$ , and  $\mathbf{\Gamma}$  is a matrix with the corresponding orthonormal eigenvectors of  $\mathbf{S}$  as columns,  $\mathbf{S}$  is considered singular when an eigenvalue  $\lambda_j$  is less than  $p\bar{\lambda}$ , where the average  $\bar{\lambda} = \sum_{k=1}^v \lambda_k / v$ .

---

## BY Statement

**BY** *variables* ;

You can specify a BY statement with PROC MI to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the MI procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables using the DATASETS procedure.

For more information on the BY statement, refer to the discussion in *SAS Language Reference: Concepts, Version 8*. For more information on the DATASETS procedure, refer to the discussion in the *SAS Procedures Guide, Version 8*.

---

## EM Statement

**EM** *< options >* ;

The expectation-maximization (EM) algorithm is a technique for maximum likelihood estimation in parametric models for incomplete data. The EM statement uses the EM algorithm to compute the MLE for  $(\mu, \Sigma)$ , the means and covariance matrix, of a multivariate normal distribution from the input data set with missing values. PROC MI uses the means and standard deviations from available cases as the initial estimates for the EM algorithm. The correlations are set to zero.

You can also use the EM statement with the NIMPUTE=0 option in the PROC statement to compute the EM estimates without multiple imputation, as shown in Example 9.1 in the “Examples” section on page 177.

The following five options are available with the EM statement.

**CONVERGE=*p***

sets the convergence criterion. The value must be between 0 and 1. The iterations are considered to have converged when the maximum change in the parameter estimates between iteration steps is less than the value specified. The change is a relative change if the parameter is greater than 0.01 in absolute value; otherwise, it is an absolute change. By default, CONVERGE=1E-4.

**ITPRINT**

prints the iteration history in the EM algorithm.

**MAXITER=*number***

specifies the maximum number of iterations used in the EM algorithm. The default is MAXITER=200.

**OUTEM=*SAS-data-set***

creates an output SAS data set of TYPE=COV containing the MLE of the parameter vector ( $\mu$ ,  $\Sigma$ ). These estimates are computed with the EM algorithm. See the “Output Data Sets” section on page 171 for a description of this output data set.

**OUTITER < ( options ) > =*SAS-data-set***

creates an output SAS data set of TYPE=COV containing parameters for each iteration. The data set includes a variable named `_iteration_` to identify the iteration number.

The parameters in the output data set depend on the options specified. You can specify the MEAN and COV options to output the mean and covariance parameters. When no options are specified, the output data set contains the mean parameters for each iteration. See the “Output Data Sets” section on page 171 for a description of this data set.

---

## FREQ Statement

**FREQ *variable* ;**

If one variable in your input data set represents the frequency of occurrence for other values in the observation, specify the variable name in a FREQ statement. PROC MI then treats the data set as if each observation appears  $n$  times, where  $n$  is the value of the FREQ variable for the observation. If the value of the FREQ variable is less than one, the observation is not used in the analysis. Only the integer portion of the value is used. The total number of observations is considered to be equal to the sum of the FREQ variable when PROC MI calculates significance probabilities.

## MCMC Statement

**MCMC** < *options* > ;

The MCMC statement specifies the details of the MCMC method for imputation. The following table summarizes the options available for the MCMC statement.

**Table 9.2.** Summary of Options in MCMC

| Tasks                                               | Options     |
|-----------------------------------------------------|-------------|
| <b>Specify data sets</b>                            |             |
| input parameter estimates for imputations           | INEST=      |
| output parameter estimates used in imputations      | OUTEST=     |
| output parameter estimates used in iterations       | OUTITER=    |
| <b>Specify imputation details</b>                   |             |
| monotone/full imputation                            | IMPUTE=     |
| single/multiple chain                               | CHAIN=      |
| number of burn-in iterations for each chain         | NBITER=     |
| number of iterations between imputations in a chain | NITER=      |
| initial parameter estimates for MCMC                | INITIAL=    |
| prior parameter information                         | PRIOR=      |
| starting parameters                                 | START=      |
| <b>Specify output graphics</b>                      |             |
| displays time-series plots                          | TIMEPLOT=   |
| displays autocorrelation plots                      | ACFPLOT=    |
| graphics catalog name for saving graphics output    | GOUT=       |
| <b>Control printed output</b>                       |             |
| displays worst linear function                      | WLF         |
| displays initial parameter values for MCMC          | DISPLAYINIT |

The following are the options available for the MCMC statement (in alphabetical order):

**ACFPLOT** < ( *options* < / *display-options* > ) >

displays the autocorrelation function plots of parameters from iterations.

The available options are:

**COV** < ( < *variables* > < *variable1\*variable2* > < ... *variable1\*variable2* > ) >

displays plots of variances for variables in the list and covariances for pairs of variables in the list. When the option COV is specified without variables, variances for all variables and covariances for all pairs of variables are used.

**MEAN** < (*variables*) >

displays plots of means for variables in the list. When the option MEAN is specified without variables, all variables are used.

**WLF**

displays the plot for the worst linear function.

When the ACFPLOT is specified without the preceding options, the procedure displays plots of means for all variables that are used.

The display-options provide additional information for the autocorrelation function plots. The available display-options are:

**CCONF**=*color*

specifies the color of the displayed confidence limits. The default is CCONF=BLACK.

**CFRAME**=*color*

specifies the color for filling the area enclosed by the axes and the frame. By default, this area is not filled.

**CNEEDLES**=*color*

specifies the color of the vertical line segments (needles) that connect autocorrelations to the reference line. The default is CNEEDLES=BLACK.

**CREF**=*color*

specifies the color of the displayed reference line. The default is CREF=BLACK.

**CSYMBOL**=*color*

specifies the color of the displayed data points. The default is CSYMBOL=BLACK.

**HSYMBOL**=*number*

specifies the height for data points in percentage screen units. The default is HSYMBOL=1.

**LCONF**=*linetype*

specifies the line type for the displayed confidence limits. The default is LREF=1, a solid line.

**LOG**

requests that the logarithmic transformations of parameters be used to compute the autocorrelations. It's generally used for the variances of variables. When a parameter has values less than or equal to zero, the corresponding plot is not created.

**LREF**=*linetype*

specifies the line type for the displayed reference line. The default is LREF=3, a dashed line.

**NLAG**=*number*

specifies the maximum lag of the series. The default is NLAG=20. The autocorrelations at each lag are displayed in the graph.

**SYMBOL=***value*

specifies the symbol for data points in percentage screen units. The default is SYMBOL=STAR.

**TITLE=***'string'*

specifies the title to be displayed in the autocorrelation function plots. The default is TITLE='Autocorrelation Plot'.

**WCONF=***number*

specifies the width for the displayed confidence limits in percentage screen units. If you specify the WCONF=0 option, the confidence limits are not displayed. The default is WCONF=1.

**WNEEDLES=***number*

specifies the width for the displayed needles that connect autocorrelations to the reference line in percentage screen units. If you specify the WNEEDLES=0 option, the needles are not displayed. The default is WNEEDLES=1.

**WREF=***number*

specifies the width for the displayed reference line in percentage screen units. If you specify the WREF=0 option, the reference line is not displayed. The default is WREF=1.

For example, the statement

```
acfplot( mean( y1) cov(y1) /log);
```

requests autocorrelation function plots for the means and variances of the variable *y1*, respectively. Logarithmic transformations of both the means and variances are used in the plots. For a detailed description of the autocorrelation function plot, see the “Autocorrelation Function Plot” section on page 169; refer also to Schafer (1997, pp. 120-126) and the *SAS/ETS User's Guide, Version 8*.

**CHAIN=**SINGLE | MULTIPLE

specifies whether a single chain is used for all imputations or a separate chain is used for each imputation. The default is CHAIN=SINGLE.

**DISPLAYINIT**

displays initial parameter values in the MCMC process for each imputation.

**GOUT=***graphics-catalog*

specifies the graphics catalog for saving graphics output from PROC MI. The default is WORK.GSEG. For more information, refer to the chapter “The GREPLAY Procedure” in *SAS/GRAPH Software: Reference, Version 8*.

**IMPUTE=**FULL | MONOTONE

specifies whether a full-data imputation is used for all missing values or a monotone-data imputation is used for a subset of missing values to make the imputed data sets have a monotone missing pattern. The default is IMPUTE=FULL. When IMPUTE=MONOTONE is specified, the order in the VAR statement is used to complete the monotone pattern.

**INEST=SAS-data-set**

names a SAS data set of TYPE=EST containing parameter estimates for imputations. These estimates are used to impute values for observations in the DATA= data set. A detailed description of the data set is provided in the “Input Data Sets” section on page 170.

**INITIAL=EM** < ( options ) >**INITIAL=INPUT=SAS-data-set**

specifies the initial mean and covariance estimates for the MCMC process. The default is INITIAL=EM.

You can specify INITIAL=INPUT=SAS-data-set to read the initial estimates of the mean and covariance matrix for each imputation from a SAS data set. See the “Input Data Sets” section on page 170 for a description of this data set.

With INITIAL=EM, PROC MI derives parameter estimates for a posterior mode, the highest observed-data posterior density, from the EM algorithm. The MLE from EM is used to start the EM algorithm for the posterior mode, and the resulting EM estimates are used to begin the MCMC process.

The following four options are available with INITIAL=EM.

**BOOTSTRAP** < =number >

requests bootstrap resampling, which uses a simple random sample with replacement from the input data set for the initial estimate. You can explicitly specify the number of observations in the random sample. Alternatively, you can implicitly specify the number of observations in the random sample by specifying the proportion  $p$ ,  $0 < p \leq 1$ , to request  $[np]$  observations in the random sample, where  $n$  is the number of observations in the data set and  $[np]$  is the integer part of  $np$ . This produces an overdispersed initial estimate that provides different starting values for the MCMC process. If you specify the BOOTSTRAP option without the number,  $p=0.75$  is used by default.

**CONVERGE=p**

sets the convergence criterion. The value must be between 0 and 1. The iterations are considered to have converged when the maximum change in the parameter estimates between iteration steps is less than the value specified. The change is a relative change if the parameter is greater than 0.01 in absolute value; otherwise, it is an absolute change. By default, CONVERGE=1E-4.

**ITPRINT**

prints the iteration history in the EM algorithm for the posterior mode.

**MAXITER=number**

specifies the maximum number of iterations used in the EM algorithm. The default is MAXITER=200.



**NBITER=number**

specifies the number of burn-in iterations before the first imputation in each chain. The default is NBITER=200.

**NITER=number**

specifies the number of iterations between imputations in a single chain. The default is NITER=100.

**OUTEST=SAS-data-set**

creates an output SAS data set of TYPE=EST. The data set contains parameter estimates used in each imputation. The data set also includes a variable named `_Imputation_` to identify the imputation number. See the “Output Data Sets” section on page 171 for a description of this data set.

**OUTITER < ( options ) > =SAS-data-set**

creates an output SAS data set of TYPE=COV containing parameters used in the imputation step for each iteration. The data set includes variables named `_Imputation_` and `_Iteration_` to identify the imputation number and iteration number.

The parameters in the output data set depend on the options specified. You can specify options MEAN, STD, COV, LR, LR\_POST, and WLF to output parameters of means, standard deviations, covariances, -2 log LR statistic, -2 log LR statistic of the posterior mode, and the worst linear function. When no options are specified, the output data set contains the mean parameters used in the imputation step for each iteration. See the “Output Data Sets” section on page 171 for a description of this data set.

**PRIOR=name**

specifies the prior information for the means and covariances. Valid values for *name* are as follows:

|                    |                                                    |
|--------------------|----------------------------------------------------|
| JEFFREYS           | specifies a noninformative prior.                  |
| RIDGE=number       | specifies a ridge prior.                           |
| INPUT=SAS-data-set | specifies a data set containing prior information. |

For a detailed description of the prior information, see the “Bayesian Estimation of the Mean Vector and Covariance Matrix” section on page 161 and the “Posterior Step” section on page 162. If you do not specify the PRIOR= option, the default is PRIOR=JEFFREYS.

The PRIOR=INPUT= option specifies a TYPE=COV data set from which the prior information of the mean vector and the covariance matrix is read. See the “Input Data Sets” section on page 170 for a description of this data set.

**START=VALUE | DIST**

specifies that the initial parameter estimates are used as either the starting value (START=VALUE) or as the starting distribution (START=DIST) in the first imputation step of each chain. The default is START=VALUE.

**TIMEPLOT** < ( *options* < / *display-options* > ) >

displays the time-series plots of parameters from iterations.

The available options are:

**COV** < ( < *variables* > < *variable1\*variable2* > < ... *variable1\*variable2* > ) >

displays plots of variances for variables in the list and covariances for pairs of variables in the list. When the option COV is specified without variables, variances for all variables and covariances for all pairs of variables are used.

**MEAN** < ( *variables* ) >

displays plots of means for variables in the list. When the option MEAN is specified without variables, all variables are used.

**WLF**

displays the plot for the worst linear function.

When the TIMEPLOT is specified without the preceding options, the procedure displays plots of means for all variables are used.

The display-options provide additional information for the time-series plots. The available display-options are:

**CFRAME**=*color*

specifies the color for filling the area enclosed by the axes and the frame. By default, this area is not filled.

**CSYMBOL**=*color*

specifies the color of the data points to be displayed in the time-series plots. The default is CSYMBOL=BLACK.

**HSYMBOL**=*number*

specifies the height for data points in percentage screen units. The default is HSYMBOL=1.

**LOG**

requests that the logarithmic transformations of parameters be used. It's generally used for the variances of variables. When a parameter value is less than or equal to zero, the value is not displayed in the corresponding plot.

**SYMBOL**=*value*

specifies the symbol for data points in percentage screen units. The default is SYMBOL=PLUS.

**TITLE**=*'string'*

specifies the title to be displayed in the time-series plots. The default is TITLE='Time-series Plot for Iterations'.

For a detailed description of the time-series plot, see the "Time-Series Plot" section on page 168 and Schafer (1997, pp. 120–126).

**WLF**

displays the worst linear function of parameters. This scalar function of parameters  $\mu$  and  $\Sigma$  is “worst” in the sense that its values from iterations converge most slowly among parameters. For a detailed description of this statistic, see the “Worst Linear Function of Parameters” section on page 168.

---

**MONOTONE Statement**

**MONOTONE** < *options* > ;

The MONOTONE statement specifies an imputation method for data sets with monotone missingness. You must also specify a VAR statement and the data set must have a monotone missing pattern with variables ordered in the VAR list. When both MONOTONE and MCMC statements are specified, the MONOTONE statement is not used.. You can specify the following options in a MONOTONE statement.

**METHOD=REG | REGRESSION**

**METHOD=PROPENSITY** < / **NGROUPS** = *number*>

specifies the imputation method for a data set with a monotone missing pattern. You can specify either METHOD=REG, a parametric regression method, or METHOD=PROPENSITY, a nonparametric method based on propensity scores. The default is METHOD=REG.

When METHOD=PROPENSITY is specified, the MAXIMUM=, MINIMUM=, and ROUND= options, which make the imputed values more consistent with the observed variable values, are not applicable.

**NGROUPS=number**

specifies the number of groups based on propensity scores for METHOD=PROPENSITY. The default is NGROUPS=5.

See the “Regression Method for Monotone Missing Data” section on page 157 for a detailed description of the regression method, and the “Propensity Score Method for Monotone Missing Data” section on page 158 for the propensity score method.

---

## TRANSFORM Statement

**TRANSFORM** *transform* ( *variables* < / *options* > )  
 < ... *transform* ( *variables* < / *options* > ) > ;

The TRANSFORM statement lists the transformations and their associated variables to be transformed. The options are transformation options that provide additional information for the transformation.

The MI procedure assumes that the data are from a multivariate normal distribution when either the regression method or the MCMC method is used. When some variables in a data set are clearly non-normal, it is useful to transform these variables to conform to the multivariate normality assumption. With a TRANSFORM statement, variables are transformed before the imputation process and these transformed variable values are displayed in all of the results. When you specify an OUT= option, the variable values are reverse-transformed to create the imputed data set.

The following transformations can be used as the *transform* in the TRANSFORM statement.

### BOXCOX

specifies the Box-Cox transformation of variables. The variable *Y* is transformed to  $\frac{(Y+c)^\lambda - 1}{\lambda}$ , where *c* is a constant such that each value of *Y* + *c* must be positive and the constant  $\lambda > 0$ .

### EXP

specifies the exponential transformation of variables. The variable *Y* is transformed to  $e^{(Y+c)}$ , where *c* is a constant.

### LOG

specifies the logarithmic transformation of variables. The variable *Y* is transformed to  $\log(Y + c)$ , where *c* is a constant such that each value of *Y*+*c* must be positive.

### LOGIT

specifies the logit transformation of variables. The variable *Y* is transformed to  $\log(\frac{Y/c}{1-Y/c})$ , where the constant  $c > 0$  and the values of *Y*/*c* must be between 0 and 1.

### POWER

specifies the power transformation of variables. The variable *Y* is transformed to  $(Y + c)^\lambda$ , where *c* is a constant such that each value of *Y* + *c* must be positive and the constant  $\lambda \neq 0$ .

The following options provide the constant  $c$  and  $\lambda$  values in the transformations.

**C=number**

specifies the  $c$  value in the transformation. The default is  $c = 1$  for logit transformation and  $c = 0$  for other transformations.

**LAMBDA=number**

specifies the  $\lambda$  value in the power and Box-Cox transformations. You must specify the  $\lambda$  value for these two transformations.

For example, the statement

```
transform log(y1) power(y2/c=1 lambda=.5);
```

requests that variables  $\log(y1)$ , a logarithmic transformation for the variable  $y1$ , and  $\sqrt{y2 + 1}$ , a power transformation for the variable  $y2$ , be used in the imputation.

If the MU0= option is used to specify a parameter value  $\mu_0$  for a transformed variable, the same transformation for the variable is also applied to its corresponding MU0= value in the  $t$  test. Otherwise,  $\mu_0 = 0$  is used for the transformed variable. See Example 9.7 for a usage of the TRANSFORM statement.

---

## VAR Statement

**VAR variables ;**

The VAR statement lists the variables to be analyzed. The variables must be numeric. If you omit the VAR statement, all numeric variables not mentioned in other statements are used. The VAR statement is required if you specify a MONOTONE statement, an IMPUTE=MONOTONE option in the MCMC statement, or more than one number in the MU0=, MAXIMUM=, MINIMUM=, or ROUND= option.

---

## Details

---

### Descriptive Statistics

---

Suppose  $\mathbf{Y}$  is the  $n \times p$  matrix of complete data, which may not be fully observed,  $n_0$  is the number of observations fully observed, and  $n_j$  is the number of observations with observed values for variable  $Y_j$ .

With complete cases, the sample mean vector is

$$\bar{\mathbf{y}} = \frac{1}{n_0} \sum \mathbf{y}_i$$

and the CSSCP matrix is

$$\sum (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})'$$

where each summation is over the fully observed observations.

The sample covariance matrix is

$$\mathbf{S} = \frac{1}{n_0 - 1} \sum (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})'$$

and is an unbiased estimate of the covariance matrix.

The correlation matrix  $\mathbf{R}$  containing the Pearson product-moment correlations of the variables is derived by scaling the corresponding covariance matrix:

$$\mathbf{R} = \mathbf{D}^{-1} \mathbf{S} \mathbf{D}^{-1}$$

where  $\mathbf{D}$  is a diagonal matrix whose diagonal elements are the square roots of the diagonal elements of  $\mathbf{S}$ .

With available cases, the corrected sum of squares for variable  $Y_j$  is

$$\sum (y_{ji} - \bar{y}_j)^2$$

where  $\bar{y}_j = \frac{1}{n_j} \sum y_{ji}$  is the sample mean and each summation is over observations with observed values for variable  $Y_j$ .

The variance is

$$s_{jj}^2 = \frac{1}{n_j - 1} \sum (y_{ji} - \bar{y}_j)^2$$

The correlations for available cases contain pairwise correlations for each pair of variables. Each correlation is computed from all observations that have nonmissing values for the corresponding pair of variables.

## EM Algorithm for Data with Missing Values

The EM algorithm (Dempster, Laird, and Rubin 1977) is a technique that finds maximum likelihood estimates in parametric models for incomplete data. The books by Little and Rubin (1987), Schafer (1997), and McLachlan and Krishnan (1997) provide detailed description and applications of the EM algorithm.

The EM algorithm is an iterative procedure that finds the MLE of the parameter vector by repeating the following steps:

### 1. The expectation E-step:

Given a set of parameter estimates, such as a mean vector and covariance matrix for a multivariate normal distribution, the E-step calculates the conditional expectation of the complete-data log likelihood given the observed data and the parameter estimates.

### 2. The maximization M-step:

Given a complete-data log likelihood, the M-step finds the parameter estimates to maximize the complete-data log likelihood from the E-step.

The two steps are iterated until the iterations converge.

In the EM process, the observed-data log likelihood is non-decreasing at each iteration. For multivariate normal data, suppose there are  $G$  groups with distinct missing patterns. Then the observed-data log likelihood being maximized can be expressed as

$$\ln L(\boldsymbol{\theta}|Y_{obs}) = \sum_{g=1}^G \ln L_g(\boldsymbol{\theta}|Y_{obs})$$

where  $\ln L_g(\boldsymbol{\theta}|Y_{obs})$  is the observed-data log likelihood from the  $g_{th}$  group, and

$$\ln L_g(\boldsymbol{\theta}|Y_{obs}) = -\frac{n_g}{2} \ln |\boldsymbol{\Sigma}_g| - \frac{1}{2} \sum_{ig} (\mathbf{y}_{ig} - \boldsymbol{\mu}_g)' \boldsymbol{\Sigma}_g^{-1} (\mathbf{y}_{ig} - \boldsymbol{\mu}_g)$$

where  $n_g$  is the number of observations in the  $g_{th}$  group, the summation is over observations in the  $g_{th}$  group,  $\mathbf{y}_{ig}$  is a vector of observed values corresponding to observed variables,  $\boldsymbol{\mu}_g$  is the corresponding mean vector, and  $\boldsymbol{\Sigma}_g$  is the associated covariance matrix.

Refer to Schafer (1997, pp. 163–181) for a detailed description of the EM algorithm for multivariate normal data.

PROC MI uses the means and standard deviations from available cases as the initial estimates for the EM algorithm. The correlations are set to zero. For a discussion of suggested starting values for the algorithm, see Schafer (1997, p. 169).

You can specify the convergence criterion with the CONVERGE= option in the EM statement. The iterations are considered to have converged when the maximum change in the parameter estimates between iteration steps is less than the value specified. You can also specify the maximum number of iterations used in the EM algorithm with the MAXITER= option.

The MI procedure displays tables of the initial parameter estimates used to begin the EM process and the MLE parameter estimates derived from EM. You can also display the EM iteration history with the option `ITPRINT`. `PROC MI` lists the iteration number, the likelihood  $-2 \log L$ , and parameter values  $\mu$  at each iteration. You can also save the MLE derived from the EM algorithm in a SAS data set specified with the `OUTEM=` option.

---

## Statistical Assumptions for Multiple Imputation

The MI procedure assumes that the data are from a continuous multivariate distribution and contain missing values that can occur on any of the variables. It also assumes that the data are from a multivariate normal distribution when either the regression method or the MCMC method is used.

Suppose  $\mathbf{Y}$  is the  $n \times p$  matrix of complete data, which is not fully observed, and denote the observed part of  $\mathbf{Y}$  by  $\mathbf{Y}_{obs}$  and the missing part by  $\mathbf{Y}_{mis}$ . The SAS MI and MIANALYZE procedures assume that the missing data are missing at random (MAR), that is, the probability that an observation is missing can depend on  $\mathbf{Y}_{obs}$ , but not on  $\mathbf{Y}_{mis}$  (Rubin 1976; 1987, p. 53).

To be more precise, suppose that  $\mathbf{R}$  is the  $n \times p$  matrix of response indicators whose elements are zero or one depending on whether the corresponding elements of  $\mathbf{Y}$  are missing or observed. Then the MAR assumption is that the distribution of  $\mathbf{R}$  can depend on  $\mathbf{Y}_{obs}$  but not on  $\mathbf{Y}_{mis}$ .

$$p(\mathbf{R} | \mathbf{Y}_{obs}, \mathbf{Y}_{mis}) = p(\mathbf{R} | \mathbf{Y}_{obs})$$

For example, consider a trivariate data set with variables  $Y_1$  and  $Y_2$  fully observed, and a variable  $Y_3$  that has missing values. MAR assumes that the probability that  $Y_3$  is missing for an individual can be related to the individual's values of variables  $Y_1$  and  $Y_2$ , but not to its value of  $Y_3$ . On the other hand, if a complete case and an incomplete case for  $Y_3$  with exactly the same values for variables  $Y_1$  and  $Y_2$  have systematically different values, then there exists a response bias for  $Y_3$ , and MAR is violated.

The MAR assumption is not the same as missing completely at random (MCAR), which is a special case of MAR. Under the MCAR assumption, the missing data values are a simple random sample of all data values; the missingness does not depend on the values of any variables in the data set.

Furthermore, the MI and MIANALYZE procedures assume that the parameters  $\theta$  of the data model and the parameters  $\phi$  of the model for the missing data indicators are distinct. That is, knowing the values of  $\theta$  does not provide any additional information about  $\phi$ , and vice versa. If both the MAR and distinctness assumptions are satisfied, the missing-data mechanism is said to be ignorable (Rubin 1987, pp. 50–54; Schafer 1997, pp. 10–11).



## Missing Data Patterns

The MI procedure sorts the data into groups based on whether an individual's value is observed or missing for each variable to be analyzed. The input data set does not need to be sorted in any order.

For example, with variables  $Y_1$ ,  $Y_2$ , and  $Y_3$  (in that order) in a data set, up to eight groups of observations can be formed from the data set. The following figure displays the eight groups of observations and an unique missing pattern for each group:

| Missing Data Patterns |    |    |    |  |
|-----------------------|----|----|----|--|
| Group                 | Y1 | Y2 | Y3 |  |
| 1                     | X  | X  | X  |  |
| 2                     | X  | X  | .  |  |
| 3                     | X  | .  | X  |  |
| 4                     | X  | .  | .  |  |
| 5                     | .  | X  | X  |  |
| 6                     | .  | X  | .  |  |
| 7                     | .  | .  | X  |  |
| 8                     | .  | .  | .  |  |

**Figure 9.6.** Missing Data Patterns

Here, an “X” means that the variable is observed in the corresponding group and a “.” means that the variable is missing.

The variable order is used to derive the order of the groups from the data set, and thus determines the order of missing values in the data to be imputed. If you specify a different order of variables in the VAR statement, then the results are different even if the other specifications remain the same.

A data set with variables  $Y_1, Y_2, \dots, Y_p$  (in that order) is said to have a monotone missing pattern when the event that a variable  $Y_j$  is missing for a particular individual implies that all subsequent variables  $Y_k, k > j$ , are missing for that individual. Alternatively, when a variable  $Y_j$  is observed for a particular individual, it is assumed that all previous variables  $Y_k, k < j$ , are also observed for that individual.

For example, the following figure displays a data set of three variables with a monotone missing pattern. Note that this data set does not have any observations with missing patterns such as in Groups 3, 5, 6, 7, or 8 in the previous example.

| Monotone Missing Data Patterns |    |    |    |  |
|--------------------------------|----|----|----|--|
| Group                          | Y1 | Y2 | Y3 |  |
| 1                              | X  | X  | X  |  |
| 2                              | X  | X  | .  |  |
| 3                              | X  | .  | .  |  |

**Figure 9.7.** Monotone Missing Patterns

---

## Imputation Mechanisms

This section describes the three methods for multiple imputation that are available in the MI procedure. The method of choice depends on the patterns of missingness in the data.

- For data sets with monotone missing patterns, either a parametric regression method (Rubin 1987) that assumes multivariate normality or a nonparametric method that uses propensity scores (Rubin 1987; Lavori, Dawson, and Shera 1995) is appropriate.
- For data sets with arbitrary missing patterns, a Markov Chain Monte Carlo (MCMC) method (Schafer 1997) that assumes multivariate normality is used to impute either all missing values or just enough missing values to make the imputed data sets have monotone missing patterns.

With a monotone missing data pattern, you have greater flexibility in your choice of strategies. For example, in addition to the MCMC method, you can also implement other methods, such as a regression method, that do not use Markov chains.

With an arbitrary missing data pattern, you can often use the MCMC method, which creates multiple imputations by drawing simulations from a Bayesian predictive distribution for normal data. Another way to handle a data set with an arbitrary missing data pattern is to use the MCMC approach to impute enough values to make the missing data pattern monotone. Then, you can use a more flexible imputation method. This approach is described in the “Producing Monotone Missingness with the MCMC Method” section on page 164.

Although the regression and MCMC methods assume multivariate normality, inferences based on multiple imputation can be robust to departures from the multivariate normality if the amount of missing information is not large. It often makes sense to use a normal model to create multiple imputations even when the observed data are somewhat non-normal, as supported by simulation studies described in Schafer (1997) and the original references therein.

You can also use a TRANSFORM statement to transform variables to conform to the multivariate normality assumption. With a TRANSFORM statement, variables are transformed before the imputation process and then are reverse-transformed to create the imputed data set.

Li (1988) presented an argument for convergence of the MCMC method in the continuous case in theory and used it to create imputations for incomplete multivariate continuous data. But in practice, it is not easy to check the convergence of a Markov chain, especially for parameters from a large number of variables. PROC MI generates statistics and plots that you can use to check for convergence of the MCMC process. The details are described in the “Convergence in MCMC” section on page 167.

## Regression Method for Monotone Missing Data

A data set with variables  $Y_1, Y_2, \dots, Y_p$  (in that order) is said to have a monotone missing pattern when the event that a variable  $Y_j$  is observed for a particular individual implies that all previous variables  $Y_k, k < j$ , are also observed for that individual.

In the regression method, a regression model is fitted for each variable with missing values, with the previous variables as covariates. Based on the fitted regression coefficients, a new regression model is simulated from the posterior predictive distribution of the parameters and is used to impute the missing values for each variable (Rubin 1987, pp. 166–167). The process is repeated sequentially for variables with missing values. That is, for a variable  $Y_j$  with missing values, a model

$$Y_j = \beta_0 + \beta_1 Y_1 + \beta_2 Y_2 + \dots + \beta_{j-1} Y_{j-1}$$

is fitted using observations with observed values for variables  $Y_1, Y_2, \dots, Y_j$ .

The fitted model includes the regression parameter estimates  $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_{j-1})$  and the associated covariance matrix  $\hat{\sigma}_j^2 \mathbf{V}_j$ , where  $\mathbf{V}_j$  is the usual  $\mathbf{X}'\mathbf{X}$  inverse matrix derived from the intercept and variables  $Y_1, Y_2, \dots, Y_{j-1}$ .

For each imputation, new parameters  $\beta_* = (\beta_{*0}, \beta_{*1}, \dots, \beta_{*(j-1)})$  and  $\sigma_{*j}^2$  are drawn from the posterior predictive distribution of the parameters. That is, they are simulated from  $(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_{j-1}), \sigma_j^2$ , and  $\mathbf{V}_j$ . The variance is drawn as

$$\sigma_{*j}^2 = \hat{\sigma}_j^2 (n_j - j) / g$$

where  $g$  is a  $\chi_{n_j-j}^2$  random variate and  $n_j$  is the number of nonmissing observations for  $Y_j$ . The regression coefficients are drawn as

$$\beta_* = \hat{\beta} + \sigma_{*j} \mathbf{V}_{hj}' \mathbf{Z}$$

where  $\mathbf{V}_{hj}'$  is the upper triangular matrix in the Cholesky decomposition,  $\mathbf{V}_j = \mathbf{V}_{hj}' \mathbf{V}_{hj}$ , and  $\mathbf{Z}$  is a vector of  $j$  independent random normal variates.

The missing values are then replaced by

$$\beta_{*0} + \beta_{*1} y_1 + \beta_{*2} y_2 + \dots + \beta_{*(j-1)} y_{j-1} + z_i \sigma_{*j}$$

where  $y_1, y_2, \dots, y_{j-1}$  are the covariate values of the first  $j - 1$  variables and  $z_i$  is a simulated normal deviate.

## Propensity Score Method for Monotone Missing Data

A propensity score is generally defined as the conditional probability of assignment to a particular treatment given a vector of observed covariates (Rosenbaum and Rubin 1983). In the propensity score method, for each variable with missing values, a propensity score is generated for each observation to estimate the probability that the observation is missing. The observations are then grouped based on these propensity scores, and an approximate Bayesian bootstrap imputation (Rubin 1987, p. 124) is applied to each group (Lavori, Dawson, and Shera 1995).

A data set with variables  $Y_1, Y_2, \dots, Y_p$  (in that order) is said to have a monotone missing pattern when the event that a variable  $Y_j$  is observed for a particular individual implies that all previous variables  $Y_k, k < j$ , are also observed for that individual. The propensity score method uses the following steps to impute values for each variable  $Y_j$  with missing values:

1. Create an indicator variable  $R_j$  with the value 0 for observations with missing  $Y_j$  and 1 otherwise.
2. Fit a logistic regression model

$$\text{logit}(p_j) = \beta_0 + \beta_1 Y_1 + \beta_2 Y_2 + \dots + \beta_{j-1} Y_{j-1}$$

where  $p_j = \Pr(R_j = 0 | Y_1, Y_2, \dots, Y_{j-1})$  and  $\text{logit}(p) = \log(p/(1-p))$ .

3. Create a propensity score for each observation to estimate the probability that it is missing.
4. Divide the observations into a fixed number of groups (typically assumed to be five) based on these propensity scores.
5. Apply an approximate Bayesian bootstrap imputation to each group. In group  $k$ , suppose that  $Y_{obs}$  denotes the  $n_1$  observations with nonmissing  $Y_j$  values and  $Y_{mis}$  denotes the  $n_0$  observations with missing  $Y_j$ . The approximate Bayesian bootstrap imputation first draws  $n_1$  observations randomly with replacement from  $Y_{obs}$  to create a new data set  $Y_{obs}^*$ . This is a nonparametric analogue of drawing parameters from the posterior predictive distribution of the parameters. The process then draws the  $n_0$  values for  $Y_{mis}$  randomly with replacement from  $Y_{obs}^*$ .

Steps 1 through 5 are repeated sequentially for each variable with missing values.

Note that the propensity score method was originally designed for a randomized experiment with repeated measures on the response variables. The goal was to impute the missing values on the response variables. The method uses only the covariate information that is associated with whether the imputed variable values are missing. It does not use correlations among variables. It is effective for inferences about the distributions of individual imputed variables, such as an univariate analysis, but it is not appropriate for analyses involving relationship among variables, such as a regression analysis. It can also produce badly biased estimates of regression coefficients when data on predictor variables are missing (Allison 2000).

## MCMC Method for Arbitrary Missing Data

The Markov Chain Monte Carlo (MCMC) method originated in physics as a tool for exploring equilibrium distributions of interacting molecules. In statistical applications, it is used to generate pseudo-random draws from multidimensional and otherwise intractable probability distributions via Markov chains. A Markov chain is a sequence of random variables in which the distribution of each element depends only on the value of the previous one.

In MCMC simulation, one constructs a Markov chain long enough for the distribution of the elements to stabilize to a stationary distribution, which is the distribution of interest. By repeatedly simulating steps of the chain, the method simulates draws from the distribution of interest. Refer to Schafer (1997) for a detailed discussion of this method.

In Bayesian inference, information about unknown parameters is expressed in the form of a posterior probability distribution. This posterior distribution is computed using Bayes' theorem

$$p(\boldsymbol{\theta}|y) = \frac{p(y|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(y|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}$$

MCMC has been applied as a method for exploring posterior distributions in Bayesian inference. That is, through MCMC, one can simulate the entire joint posterior distribution of the unknown quantities and obtain simulation-based estimates of posterior parameters that are of interest.

In many incomplete data problems, the observed-data posterior  $p(\boldsymbol{\theta}|Y_{obs})$  is intractable and cannot easily be simulated. However, when  $Y_{obs}$  is augmented by an estimated/simulated value of the missing data  $Y_{mis}$ , the complete-data posterior  $p(\boldsymbol{\theta}|Y_{obs}, Y_{mis})$  is much easier to simulate. Assuming that the data are from a multivariate normal distribution, data augmentation can be applied to Bayesian inference with missing data by repeating the following steps:

### 1. The imputation I-step:

Given an estimated mean vector and covariance matrix, the I-step simulates the missing values for each observation independently. That is, if you denote the variables with missing values for observation  $i$  by  $Y_{i(mis)}$  and the variables with observed values by  $Y_{i(obs)}$ , then the I-step draws values for  $Y_{i(mis)}$  from a conditional distribution for  $Y_{i(mis)}$  given  $Y_{i(obs)}$ .

### 2. The posterior P-step:

Given a complete sample, the P-step simulates the posterior population mean vector and covariance matrix. These new estimates are then used in the next I-step. Without prior information about the parameters, a noninformative prior is used. You can also use other informative priors. For example, a prior information about the covariance matrix can be helpful to stabilize the inference about the mean vector for a near singular covariance matrix.

The two steps are iterated long enough for the results to be reliable for a multiply imputed data set (Schafer 1997, p. 72). That is, with a current parameter estimate  $\theta^{(t)}$  at the  $t$ th iteration, the I-step draws  $Y_{mis}^{(t+1)}$  from  $p(Y_{mis}|Y_{obs}, \theta^{(t)})$  and the P-step draws  $\theta^{(t+1)}$  from  $p(\theta|Y_{obs}, Y_{mis}^{(t+1)})$ .

This creates a Markov chain

$$(Y_{mis}^{(1)}, \theta^{(1)}), (Y_{mis}^{(2)}, \theta^{(2)}), \dots,$$

which converges in distribution to  $p(Y_{mis}, \theta|Y_{obs})$ . Assuming the iterates converge to a stationary distribution, the goal is to simulate an approximately independent draw of the missing values from this distribution.

To validate the imputation results, you should repeat the process with different random number generators and starting values based on different initial parameter estimates.

The next three sections provide details for the imputation step, Bayesian estimation of the mean vector and covariance matrix, and the posterior step.

### Imputation Step

In each iteration, starting with a given mean vector  $\mu$  and covariance matrix  $\Sigma$ , the imputation step draws values for the missing data from the conditional distribution  $Y_{mis}$  given  $Y_{obs}$ .

Suppose  $\mu = [\mu'_1, \mu'_2]'$  is the partitioned mean vector of two sets of variables,  $Y_{obs}$  and  $Y_{mis}$ , where  $\mu_1$  is the mean vector for variables  $Y_{obs}$  and  $\mu_2$  is the mean vector for variables  $Y_{mis}$ .

Also suppose

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma'_{12} & \Sigma_{22} \end{bmatrix}$$

is the partitioned covariance matrix for these variables, where  $\Sigma_{11}$  is the covariance matrix for variables  $Y_{obs}$ ,  $\Sigma_{22}$  is the covariance matrix for variables  $Y_{mis}$ , and  $\Sigma_{12}$  is the covariance matrix between variables  $Y_{obs}$  and variables  $Y_{mis}$ .

By using the sweep operator (Goodnight 1979) on the pivots of the  $\Sigma_{11}$  submatrix, the matrix becomes

$$\begin{bmatrix} \Sigma_{11}^{-1} & \Sigma_{11}^{-1}\Sigma_{12} \\ -\Sigma'_{12}\Sigma_{11}^{-1} & \Sigma_{22.1} \end{bmatrix}$$

where  $\Sigma_{22.1} = \Sigma_{22} - \Sigma'_{12}\Sigma_{11}^{-1}\Sigma_{12}$  can be used to compute the conditional covariance matrix of  $Y_{mis}$  after controlling for  $Y_{obs}$ .

For an observation with the preceding missing pattern, the conditional distribution of  $Y_{mis}$  given  $Y_{obs} = \mathbf{y}_1$  is a multivariate normal distribution with the mean vector

$$\boldsymbol{\mu}_{2.1} = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}'_{12} \boldsymbol{\Sigma}_{11}^{-1} (\mathbf{y}_1 - \boldsymbol{\mu}_1)$$

and the conditional covariance matrix

$$\boldsymbol{\Sigma}_{22.1} = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}'_{12} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12}$$

### **Bayesian Estimation of the Mean Vector and Covariance Matrix**

Suppose that  $\mathbf{Y} = (\mathbf{y}'_1, \mathbf{y}'_2, \dots, \mathbf{y}'_n)'$  is an  $(n \times p)$  matrix made up of  $n$   $(p \times 1)$  independent vectors  $\mathbf{y}_i$ , each of which has a multivariate normal distribution with mean zero and covariance matrix  $\mathbf{\Lambda}$ . Then the SSCP matrix

$$\mathbf{A} = \mathbf{Y}'\mathbf{Y} = \sum_i \mathbf{y}_i \mathbf{y}_i'$$

has a Wishart distribution  $W(n, \mathbf{\Lambda})$ .

When each observation  $\mathbf{y}_i$  is distributed with a multivariate normal distribution with an unknown mean  $\boldsymbol{\mu}$ , then the CSSCP matrix

$$\mathbf{A} = \sum_i (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})'$$

has a Wishart distribution  $W(n - 1, \mathbf{\Lambda})$ .

If  $\mathbf{A}$  has a Wishart distribution  $W(n, \mathbf{\Lambda})$ , then  $\mathbf{B} = \mathbf{A}^{-1}$  has an inverted Wishart distribution  $W^{-1}(n, \boldsymbol{\Psi})$ , where  $n$  is the degrees of freedom and  $\boldsymbol{\Psi} = \mathbf{\Lambda}^{-1}$  is the precision matrix (Anderson 1984).

Note that, instead of using the parameter  $\boldsymbol{\Psi} = \mathbf{\Lambda}^{-1}$  for the inverted Wishart distribution, Schafer (1997) uses the parameter  $\mathbf{\Lambda}$ .

Suppose that each observation in the data matrix  $\mathbf{Y}$  has a multivariate normal distribution with mean  $\boldsymbol{\mu}$  and covariance matrix  $\boldsymbol{\Sigma}$ . Then with a prior inverted Wishart distribution for  $\boldsymbol{\Sigma}$  and a prior normal distribution for  $\boldsymbol{\mu}$

$$\begin{aligned} \boldsymbol{\Sigma} &\sim W^{-1}(m, \boldsymbol{\Psi}) \\ \boldsymbol{\mu} | \boldsymbol{\Sigma} &\sim N\left(\boldsymbol{\mu}_0, \frac{1}{\tau} \boldsymbol{\Sigma}\right) \end{aligned}$$

where  $\tau > 0$  is a fixed number. The posterior distribution (Anderson 1984, p. 270; Schafer 1997, p. 152) is

$$\begin{aligned} \boldsymbol{\Sigma} | \mathbf{Y} &\sim W^{-1}\left(n + m, (n - 1)\mathbf{S} + \boldsymbol{\Psi} + \frac{n\tau}{n + \tau}(\bar{\mathbf{y}} - \boldsymbol{\mu}_0)(\bar{\mathbf{y}} - \boldsymbol{\mu}_0)'\right) \\ \boldsymbol{\mu} | (\boldsymbol{\Sigma}, \mathbf{Y}) &\sim N\left(\frac{1}{n + \tau}(n\bar{\mathbf{y}} + \tau\boldsymbol{\mu}_0), \frac{1}{n + \tau}\boldsymbol{\Sigma}\right) \end{aligned}$$

where  $(n - 1)\mathbf{S}$  is the CSSCP matrix.

**Posterior Step**

In each iteration, the posterior step simulates the posterior population mean vector  $\mu$  and covariance matrix  $\Sigma$  from prior information for  $\mu$  and  $\Sigma$ , and the complete sample estimates.

You can specify the prior parameter information using one of the following methods:

- PRIOR=JEFFREYS, which uses a noninformative prior.
- PRIOR=INPUT=, which provides a prior information for  $\Sigma$  in the data set. Optionally, it also provides a prior information for  $\mu$  in the data set.
- PRIOR=RIDGE=, which uses a ridge prior.

The next four subsections provide details of the posterior step for different prior distributions.

**1. A Noninformative Prior**

Without prior information about the mean and covariance estimates, a noninformative prior can be used by specifying the PRIOR=JEFFREYS option. The posterior distributions (Schafer 1997, p. 154) are

$$\begin{aligned}\Sigma^{(t+1)}|\mathbf{Y} &\sim W^{-1}(n-1, (n-1)\mathbf{S}) \\ \mu^{(t+1)}|(\Sigma^{(t+1)}, \mathbf{Y}) &\sim N\left(\bar{\mathbf{y}}, \frac{1}{n}\Sigma^{(t+1)}\right)\end{aligned}$$

**2. An Informative Prior for  $\mu$  and  $\Sigma$** 

When prior information is available for the parameters  $\mu$  and  $\Sigma$ , you can provide it with a SAS data set that you specify with the PRIOR=INPUT= option.

$$\begin{aligned}\Sigma &\sim W^{-1}(d^*, d^*\mathbf{S}^*) \\ \mu|\Sigma &\sim N\left(\mu_0, \frac{1}{n_0}\Sigma\right)\end{aligned}$$

To obtain the prior distribution for  $\Sigma$ , PROC MI reads the matrix  $\mathbf{S}^*$  from observations in the data set with \_TYPE\_='COV', and it reads  $n^* = d^* + 1$  from observations with \_TYPE\_='N'.

To obtain the prior distribution for  $\mu$ , PROC MI reads the mean vector  $\mu_0$  from observations with \_TYPE\_='MEAN', and it reads  $n_0$  from observations with \_TYPE\_='N\_MEAN'. When there are no observations with \_TYPE\_='N\_MEAN', PROC MI reads  $n_0$  from observations with \_TYPE\_='N'.



The resulting posterior distribution, as described in the “Bayesian Estimation of the Mean Vector and Covariance Matrix” section on page 161, is given by

$$\begin{aligned}\Sigma^{(t+1)} | \mathbf{Y} &\sim W^{-1} (n + d^*, (n - 1)\mathbf{S} + d^*\mathbf{S}^* + \mathbf{S}_m) \\ \mu^{(t+1)} | (\Sigma^{(t+1)}, \mathbf{Y}) &\sim N \left( \frac{1}{n + n_0} (n\bar{\mathbf{y}} + n_0\mu_0), \frac{1}{n + n_0} \Sigma^{(t+1)} \right)\end{aligned}$$

where

$$\mathbf{S}_m = \frac{nn_0}{n + n_0} (\bar{\mathbf{y}} - \mu_0)(\bar{\mathbf{y}} - \mu_0)'$$

### 3. An Informative Prior for $\Sigma$

When the sample covariance matrix  $\mathbf{S}$  is singular or near singular, prior information about  $\Sigma$  can also be used without prior information about  $\mu$  to stabilize the inference about  $\mu$ . You can provide it with a SAS data set that you specify with the PRIOR=INPUT= option.

To obtain the prior distribution for  $\Sigma$ , PROC MI reads the matrix  $\mathbf{S}^*$  from observations in the data set with \_TYPE\_='COV', and it reads  $n^*$  from observations with \_TYPE\_='N'.

Note that if the PRIOR=INPUT= data set also contains observations with \_TYPE\_='MEAN', then a complete informative prior for both  $\mu$  and  $\Sigma$  will be used.

Corresponding to the prior for  $\Sigma$

$$\Sigma \sim W^{-1} (d^*, d^*\mathbf{S}^*)$$

the posterior distribution for  $\Sigma$  (Anderson 1984, p. 269) is

$$\Sigma^{(t+1)} | \mathbf{Y} \sim W^{-1} ((n - 1) + d^*, (n - 1)\mathbf{S} + d^*\mathbf{S}^*)$$

Thus, an estimate of  $\Sigma$  is given by the weighted average

$$\frac{1}{(n - 1) + d^*} ((n - 1)\mathbf{S} + d^*\mathbf{S}^*)$$

and the posterior distribution for  $(\mu, \Sigma)$  becomes

$$\begin{aligned}\Sigma^{(t+1)} | \mathbf{Y} &\sim W^{-1} ((n - 1) + d^*, (n - 1)\mathbf{S} + d^*\mathbf{S}^*) \\ \mu^{(t+1)} | (\Sigma^{(t+1)}, \mathbf{Y}) &\sim N \left( \bar{\mathbf{y}}, \frac{1}{n} \Sigma^{(t+1)} \right)\end{aligned}$$

#### 4. A Ridge Prior

A special case of the preceding adjustment is a ridge prior with  $\mathbf{S}^* = \text{Diag } \mathbf{S}$  (Schafer 1997, p. 156). That is,  $\mathbf{S}^*$  is a diagonal matrix with diagonal elements equal to the corresponding elements in  $\mathbf{S}$ .

You can request a ridge prior by using the `PRIOR=RIDGE=` option. You can explicitly specify the number  $d^* \geq 1$  in the `PRIOR=RIDGE= $d^*$`  option. Or you can implicitly specify the number by specifying the proportion  $p$  in the `PRIOR=RIDGE= $p$`  option to request  $d^* = (n - 1)p$ .

The posterior is then given by

$$\begin{aligned} \boldsymbol{\Sigma}^{(t+1)} | \mathbf{Y} &\sim W^{-1}((n - 1) + d^*, (n - 1)\mathbf{S} + d^*\mathbf{S}^*) \\ \boldsymbol{\mu}^{(t+1)} | (\boldsymbol{\Sigma}^{(t+1)}, \mathbf{Y}) &\sim N\left(\bar{y}, \frac{1}{n} \boldsymbol{\Sigma}^{(t+1)}\right) \end{aligned}$$

---

### Producing Monotone Missingness with the MCMC Method

The monotone data MCMC method was first proposed by Li (1988), and Liu (1993) described the algorithm. The method is useful especially when a data set is close to having a monotone missing pattern. In this case, the method only needs to impute a few missing values to the data set to have a monotone missing pattern in the imputed data set. Compared to a full data imputation that imputes all missing values, the monotone data MCMC method imputes fewer missing values in each iteration and achieves approximate stationarity in fewer iterations (Schafer 1997, p. 227).

You can request the monotone MCMC method by specifying the option `IMPUTE=MONOTONE` in the MCMC statement. The “Missing Data Patterns” table now denotes the variables with missing values by “.” or “O”. A “.” means that the variable is missing and will be imputed and an “O” means that the variable is missing and will not be imputed. The tables of “Multiple Imputation Variance Information” and “Multiple Imputation Parameter Estimates” are not created.

You must specify the variables in the VAR statement. The variable order in the list determines the monotone missing pattern in the imputed data set. With a different order in the VAR list, the results will be different because the monotone missing pattern to be constructed will be different.

Assuming that the data are from a multivariate normal distribution, then similar to the MCMC method, the monotone MCMC method repeats the following steps:

#### 1. The imputation I-step:

Given an estimated mean vector and covariance matrix, the I-step simulates the missing values for each observation independently. Only a subset of missing values are simulated to achieve a monotone pattern of missingness.

#### 2. The posterior P-step:

Given a new sample with a monotone pattern of missingness, the P-step simulates the posterior population mean vector and covariance matrix with a noninformative Jeffreys prior. These new estimates are then used in the next I-step.

### Imputation Step

The I-step is almost identical to the I-step described in the “MCMC Method for Arbitrary Missing Data” section on page 159 except that here only a subset of missing values need to be simulated. To state this precisely, denote the variables with observed values for observation  $i$  by  $Y_{i(obs)}$  and the variables with missing values by  $Y_{i(mis)} = (Y_{i(m1)} Y_{i(m2)})$ , where  $Y_{i(m1)}$  is a subset of the the missing variables that will result a monotone missingness when their values are imputed. Then the I-step draws values for  $Y_{i(m1)}$  from a conditional distribution for  $Y_{i(m1)}$  given  $Y_{i(obs)}$ .

### Posterior Step

The P-step is different from the P-step described in the “MCMC Method for Arbitrary Missing Data” section on page 159. Instead of simulating the  $\mu$  and  $\Sigma$  parameters from the full imputed data set, the P-step here simulates the  $\mu$  and  $\Sigma$  parameters through simulated regression coefficients from regression models based on the imputed data set with a monotone pattern of missingness. The step is similar to the process described in the “Regression Method for Monotone Missing Data” section on page 157.

That is, for the variable  $Y_j$ , a model

$$Y_j = \beta_0 + \beta_1 Y_1 + \beta_2 Y_2 + \dots + \beta_{j-1} Y_{j-1}$$

is fitted using nonmissing observations.

The fitted model consists of the regression parameter estimates  $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_{j-1})$  and the associated covariance matrix  $\hat{\sigma}_j^2 \mathbf{V}_j$ , where  $\mathbf{V}_j$  is the usual  $\mathbf{X}'\mathbf{X}$  inverse matrix from the intercept and variables  $Y_1, Y_2, \dots, Y_{j-1}$ .

For each imputation, new parameters  $\beta_* = (\beta_{*0}, \beta_{*1}, \dots, \beta_{*(j-1)})$  and  $\sigma_{*j}^2$  are drawn from the posterior predictive distribution of the parameters. That is, they are simulated from  $(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_{j-1})$ ,  $\sigma_j^2$ , and  $\mathbf{V}_j$ . The variance is drawn as

$$\sigma_{*j}^2 = \hat{\sigma}_j^2 (n_j - j) / g$$

where  $g$  is a  $\chi_{n_j - p + j - 1}^2$  random variate and  $n_j$  is the number of nonmissing observations for  $Y_j$ . The regression coefficients are drawn as

$$\beta_* = \hat{\beta} + \sigma_{*j} \mathbf{V}_{hj}' \mathbf{Z}$$

where  $\mathbf{V}_{hj}'$  is the upper triangular matrix in the Cholesky decomposition  $\mathbf{V}_j = \mathbf{V}_{hj}' \mathbf{V}_{hj}$  and  $\mathbf{Z}$  is a vector of  $j$  independent random normal variates.

These simulated values of  $\beta_*$  and  $\sigma_{*j}^2$  are then used to re-create the parameters  $\mu$  and  $\Sigma$ . For a detailed description of how to produce monotone-missingness with the MCMC method for a multivariate normal data, refer to Schafer (1997, pp. 226–235).

## MCMC Method Specifications

---

With MCMC, you can impute either all missing values (`IMPUTE=FULL`) or just enough missing values to make the imputed data set have a monotone missing pattern (`IMPUTE=MONOTONE`). In the process, either a single chain for all imputations (`CHAIN=SINGLE`) or a separate chain for each imputation (`CHAIN=MULTIPLE`) is used. Refer to Schafer (1997, pp. 137–138) for a discussion of single versus multiple chains.

You can specify the number of initial burn-in iterations before the first imputation with the `NBITER=` option. This number is also used for subsequent chains for multiple chains. For a single chain, you can also specify the number of iterations between imputations with the `NITER=` option.

You can explicitly specify initial parameter values for the MCMC process with the `INITIAL=INPUT=` data set option. Alternatively, you can use the EM algorithm to derive a set of initial parameter values for MCMC with the option `INITIAL=EM`. These estimates are used as either the starting value (`START=VALUE`) or as the starting distribution (`START=DIST`) for the MCMC process. For multiple chains, these estimates are used again as either the starting value (`START=VALUE`) or as the starting distribution (`START=DIST`) for the subsequent chains.

You can specify the prior parameter information in the `PRIOR=` option. You can use a noninformative prior (`PRIOR=JEFFREYS`), a ridge prior (`PRIOR=RIDGE`), or an informative prior specified in a data set (`PRIOR=INPUT`).

The parameter estimates used to generate imputed values in each imputation can be saved in a data set with the `OUTEST=` option. Later, this data set can be read with the `INEST=` option to provide the reference distribution for imputing missing values for a new data set.

By default, the MCMC method uses a single chain to produce five imputations. It completes 200 burn-in iterations before the first imputation and 100 iterations between imputations. The posterior mode computed from the EM algorithm with a noninformative prior is used as the starting values for the MCMC process.

### ***INITIAL=EM Specifications***

The EM algorithm is used to find the maximum likelihood estimates for incomplete data in the EM statement. You can also use the EM algorithm to find a posterior mode, the parameter estimates that maximize the observed-data posterior density. The resulting posterior mode provides a good starting value for the MCMC process.

With `INITIAL=EM`, PROC MI uses the MLE of the parameter vector as the initial estimates in the EM algorithm for the posterior mode. You can use the `ITPRINT` option in `INITIAL=EM` to display the iteration history for the EM algorithm.

You can use the `CONVERGE=` option to specify the convergence criterion in deriving the EM posterior mode. The iterations are considered to have converged when the maximum change in the parameter estimates between iteration steps is less than the value specified. By default, `CONVERGE=1E-4`.

You can also use the MAXITER= option to specify the maximum number of iterations in the EM algorithm. By default, MAXITER=200.

With the BOOTSTRAP option, you can use overdispersed starting values for the MCMC process. In this case, PROC MI applies the EM algorithm to a bootstrap sample, a simple random sample with replacement from the input data set, to derive the initial estimates for each chain (Schafer 1997, p. 128).

---

## Convergence in MCMC

The theoretical convergence of the MCMC process has been explored under various conditions, as described in Schafer (1997, p. 70). However, in practice, verification of convergence is not a simple matter and cannot be easily implemented in the MI procedure.

The parameters used in the imputation step for each iteration can be saved in an output data set with the OUTITER= option. These include the means, standard deviations, covariances, the worst linear function, and observed-data LR statistics. You can then monitor the convergence in a single chain by displaying time-series plots and autocorrelations for those parameter values (Schafer 1997, p. 120). The time-series and autocorrelation function plots for parameters such as variable means, covariances, and the worst linear function can be displayed by specifying the TIMEPLOT and ACFPLOT option.

You can apply EM to a bootstrap sample to obtain overdispersed starting values for multiple chains (Gelman and Rubin 1992). This provides a conservative estimate of the number of iterations needed before each imputation.

The next four subsections provide useful statistics and plots that can be used to check the convergence of the MCMC process.

### LR Statistics

You can save the observed-data likelihood ratio (LR) statistic in each iteration with the LR option in the OUTITER= data set. The statistic is based on the observed-data likelihood with parameter values used in the iteration and the observed-data maximum likelihood derived from the EM algorithm.

In each iteration, the LR statistic is given by

$$-2 \log \left( \frac{f(\hat{\theta}_i)}{f(\hat{\theta})} \right)$$

where  $f(\hat{\theta})$  is the observed-data maximum likelihood derived from the EM algorithm and  $f(\hat{\theta}_i)$  is the observed-data likelihood for  $\hat{\theta}_i$  used in the iteration.

Similarly, you can also save the observed-data LR posterior mode statistic for each iteration with the LR\_POST option. This statistic is based on the observed-data posterior density with parameter values used in each iteration and the observed-data posterior mode derived from the EM algorithm for posterior mode.

For large samples, these LR statistics tends to be approximately  $\chi^2$  distributed with degrees of freedom equal to the dimension of  $\theta$  (Schafer 1997, p. 131). For example, with a large number of iterations, if the values of the LR statistic do not behave like a random sample from the described  $\chi^2$  distribution, then there is evidence that the MCMC process has not converged.

### **Worst Linear Function of Parameters**

The worst linear function (WLF) of parameters (Schafer 1997, pp. 129-131) is a scalar function of parameters  $\mu$  and  $\Sigma$  that is “worst” in the sense that its function values converge most slowly among parameters in the MCMC process. The convergence of this function is evidence that other parameters are likely to converge as well.

For linear functions of parameters  $\theta = (\mu, \Sigma)$ , a worst linear function of  $\theta$  has the highest asymptotic rate of missing information. The function can be derived from the iterative values of  $\theta$  near the posterior mode in the EM algorithm. That is, an estimated worst linear function of  $\theta$  is

$$w(\theta) = \mathbf{v}'(\theta - \hat{\theta})$$

where  $\hat{\theta}$  is the posterior mode and the coefficients  $\mathbf{v} = \hat{\theta}_{(-1)} - \hat{\theta}$  is the difference between the estimated value of  $\theta$  one step prior to convergence and the converged value  $\hat{\theta}$ .

You can display the coefficients of the worst linear function,  $\mathbf{v}$ , by specifying the WLF option in the MCMC statement. You can save the function value from each iteration in an OUTITER= data set by specifying the WLF option in the OUTITER option. You can also display the worst linear function values from iterations in an autocorrelation plot or a time-series plot by specifying WLF as an ACFPLOT or TIMEPLOT option, respectively.

Note that when the observed-data posterior is nearly normal, the WLF is one of the slowest functions to approach stationarity. When the posterior is not close to normal, other functions may take much longer than the WLF to converge, as described in Schafer (1997, p.130).

### **Time-Series Plot**

A time-series plot for a parameter  $\xi$  is a scatter plot of successive parameter estimates  $\xi_i$  against the iteration number  $i$ . The plot provides a simple way to examine the convergence behavior of the estimation algorithm for  $\xi$ . Long-term trends in the plot indicate that successive iterations are highly correlated and that the series of iterations has not converged.

You can display time-series plots for the worst linear function, the variable means, variable variances, and covariances of variables. You can also request logarithmic transformations for positive parameters in the plots with the LOG option. When a parameter value is less than or equal to zero, the value is not displayed in the corresponding plot.

By default, the MI procedure uses the plus sign (+) as the plot symbol to display the points with a height of one (percentage screen unit) in a time-series plot. You can use the SYMBOL=, CSYMBOL=, and HSYMBOL= options to change the shape, color, and height of the plot symbol.

By default, the plot title “Time-Series Plot” is displayed in a time-series plot. You can request another title by using the TITLE= option in TIMEPLOT. When another title is also specified in a TITLE statement, this title is displayed as the main title and the plot title is displayed as a subtitle in the plot.

You can use options in the GOPTIONS statement to change the color and height of the title. Refer to the chapter “The SAS/GRAPH Statements” in *SAS/GRAPH Software: Reference, Version 8* for a description of title options. See Example 9.6 for a usage of the time-series plot.

### **Autocorrelation Function Plot**

To examine relationships of successive parameter estimates  $\xi$ , the autocorrelation function (ACF) can be used. For a stationary series,  $\xi_i, i \geq 1$ , in time series data, the autocorrelation function at lag  $k$  is

$$\rho_k = \frac{\text{Cov}(\xi_i, \xi_{i+k})}{\text{Var}(\xi_i)}$$

The sample  $k_{th}$  order autocorrelation is computed as

$$r_k = \frac{\sum_{i=1}^{n-k} (\xi_i - \bar{\xi})(\xi_{i+k} - \bar{\xi})}{\sum_{i=1}^n (\xi_i - \bar{\xi})^2}$$

You can display autocorrelation function plots for the worst linear function, the variable means, variable variances, and covariances of variables. You can also request logarithmic transformations for parameters in the plots with the LOG option. When a parameter has values less than or equal to zero, the corresponding plot is not created.

You specify the maximum number of lags of the series with the NLAG= option. The autocorrelations at each lag less than or equal to the specified lag are displayed in the graph. In addition, the plot also displays approximate 95% confidence limits for the autocorrelations. At lag  $k$ , the confidence limits indicate a set of approximate 95% critical values for testing the hypothesis  $\rho_j = 0, j \geq k$ .

By default, the MI procedure uses the star sign (\*) as the plot symbol to display the points with a height of one (percentage screen unit) in the plot, a solid line to display the reference line of zero autocorrelation, vertical line segments to connect autocorrelations to the reference line, and a pair of dashed lines to display approximately 95% confidence limits for the autocorrelations.

You can use the SYMBOL=, CSYMBOL=, and HSYMBOL= options to change the shape, color, and height of the plot symbol, and the CNEEDLES= and WNEEDLES= options to change the color and width of the needles. You can also use the LREF=, CREF=, and WREF= options to change the line type, color, and width of the reference line. Similarly, you can use the LCONF=, CCONF=, and WCONF= options to change the line type, color, and width of the confidence limits.

By default, the plot title “Autocorrelation Plot” is displayed in a autocorrelation function plot. You can request another title by using the `TITLE=` option in `ACFPLOT`. When another title is also specified in a `TITLE` statement, this title is displayed as the main title and the plot title is displayed as a subtitle in the plot.

You can use options in the `GOPTIONS` statement to change the color and height of the title. Refer to the chapter “The SAS/GRAPH Statements” in *SAS/GRAPH Software: Reference, Version 8* for a description of title options. See Example 9.6 for a usage of the autocorrelation function plot.

---

## Input Data Sets

You can specify the input data set with missing values with the `DATA=` option in the `PROC MI` statement. When a MCMC method is used, you can specify the data set containing the reference distribution information for imputation with the `INEST=` option, the data set containing initial parameter estimates for the MCMC process with the `INITIAL=INPUT=` option, and the data set containing information for the prior distribution with the `PRIOR=INPUT=` option in the MCMC statement.

### **DATA=SAS-data-set**

The input `DATA=` data set is an ordinary SAS data set containing multivariate data with missing values.

### **INEST=SAS-data-set**

The input `INEST=` data set is a `TYPE=EST` data set and contains a variable `_Imputation_` to identify the imputation number. For each imputation, `PROC MI` reads the point estimate from the observations with `_TYPE_='PARM'` or `_TYPE_='PARMS'` and the associated covariances from the observations with `_TYPE_='COV'` or `_TYPE_='COVB'`. These estimates are used as the reference distribution to impute values for observations in the `DATA=` data set. When the input `INEST=` data set also contains observations with `_TYPE_='SEED'`, `PROC MI` reads the seed information for the random number generator from these observations. Otherwise, the `SEED=` option provides the seed information. See Example 9.8 for a usage of this option.

### **INITIAL=INPUT=SAS-data-set**

The input `INITIAL=INPUT=` data set is a `TYPE=COV` or `CORR` data set and provides initial parameter estimates for the MCMC process. The covariances derived from the `TYPE=COV/CORR` data set are divided by the number of observations to get the correct covariance matrix for the point estimate (sample mean).

If `TYPE=COV`, `PROC MI` reads the number of observations from the observations with `_TYPE_='N'`, the point estimate from the observations with `_TYPE_='MEAN'`, and the covariances from the observations with `_TYPE_='COV'`.

If `TYPE=CORR`, `PROC MI` reads the number of observations from the observations with `_TYPE_='N'`, the point estimate from the observations with `_TYPE_='MEAN'`, the correlations from the observations with `_TYPE_='CORR'`, and the standard deviations from the observations with `_TYPE_='STD'`.



**PRIOR=INPUT=SAS-data-set**

The input PRIOR=INPUT= data set is a TYPE=COV data set that provides information for the prior distribution. You can use the data set to specify a prior distribution for  $\Sigma$  of the form

$$\Sigma \sim W^{-1}(d^*, d^* \mathbf{S}^*)$$

where  $d^* = n^* - 1$  is the degrees of freedom. PROC MI reads the matrix  $\mathbf{S}^*$  from observations with \_TYPE\_='COV' and  $n^*$  from observations with \_TYPE\_='N'.

You can also use this data set to specify a prior distribution for  $\mu$  of the form

$$\mu \sim N\left(\mu_0, \frac{1}{n_0} \Sigma\right)$$

PROC MI reads the mean vector  $\mu_0$  from observations with \_TYPE\_='MEAN' and  $n_0$  from observations with \_TYPE\_='N\_MEAN'. When there are no observations with \_TYPE\_='N\_MEAN', PROC MI reads  $n_0$  from observations with \_TYPE\_='N'.

---

## Output Data Sets

You can specify the output data set of imputed values with the OUT= option in the PROC MI statement. When an EM statement is used, you can specify the data set containing MLE computed with the EM algorithm with the OUTEM= option in the EM statement. When a MCMC method is used, you can specify the data set containing parameter estimates used in each imputation with the OUTEST= option and the data set containing parameters used in the imputation step for each iteration with the OUTITER option in the MCMC statement.

**OUT=SAS-data-set**

The OUT= data set contains all the variables in the original data set and a new variable named \_Imputation\_ that identifies the imputation. For each imputation, the data set contains all variables in the input DATA= data set with missing values replaced by imputed values.

**OUTEM=SAS-data-set**

The OUTEM= data set is a TYPE=COV data set and contains the MLE computed with the EM algorithm. The observations with \_TYPE\_='MEAN' contain the estimated mean and the observations with \_TYPE\_='COV' contain the estimated covariances.

**OUTEST=SAS-data-set**

The OUTEST= data set is a TYPE=EST data set and contains parameter estimates used in each imputation in the MCMC method. It also includes an index variable named \_Imputation\_, which identifies the imputation.

The observations with `_TYPE_='SEED'` contain the seed information for the random number generator. The observations with `_TYPE_='PARM'` or `_TYPE_='PARMS'` contain the point estimate and the observations with `_TYPE_='COV'` or `_TYPE_='COVB'` contain the associated covariances. These estimates are used as the parameters of the reference distribution to impute values for observations in the `DATA=` dataset.

Note that these estimates are the values used in the I-step before each imputation. These are not the parameter values simulated from the P-step in the same iteration. See Example 9.8 for a usage of this option.

**OUTITER < ( options ) > =SAS-data-set in an EM statement**

The `OUTITER=` data set in an EM statement is a `TYPE=COV` data set and contains parameters for each iteration. It also includes a variable `_Iteration_` that provides the iteration number.

The parameters in the output data set depend on the options specified. You can specify the `MEAN` and `COV` options for `OUTITER`. With the `MEAN` option, the output data set contains the mean parameters in observations with the variable `_TYPE_='MEAN'`. Similarly, with the `COV` option, the output data set contains the covariance parameters in observations with the variable `_TYPE_='COV'`. When no options are specified, the output data set contains the mean parameters for each iteration.

**OUTITER < ( options ) > =SAS-data-set in a MCMC statement**

The `OUTITER=` data set in a MCMC statement is a `TYPE=COV` data set and contains parameters used in the imputation step for each iteration. It also includes variables named `_Imputation_` and `_Iteration_`, which provide the imputation number and iteration number.

The parameters in the output data set depend on the options specified. The following table summarizes the options available for `OUTITER` and the corresponding values for the output variable `_TYPE_`.

**Table 9.3.** Summary of Options for `OUTITER` in a MCMC statement

| Options | Output Parameters                         | _TYPE_   |
|---------|-------------------------------------------|----------|
| MEAN    | mean parameters                           | MEAN     |
| STD     | standard deviations                       | STD      |
| COV     | covariances                               | COV      |
| LR      | -2 log LR statistic                       | LOG_LR   |
| LR_POST | -2 log LR statistic of the posterior mode | LOG_POST |
| WLF     | worst linear function                     | WLF      |

When no options are specified, the output data set contains the mean parameters used in the imputation step for each iteration. For a detailed description of the worst linear function and LR statistics, see the “Convergence in MCMC” section on page 167.

## Combining Inferences from Multiply Imputed Data Sets

With  $m$  imputations,  $m$  different sets of the point and variance estimates for a parameter  $Q$  can be computed. Suppose  $\hat{Q}_i$  and  $\hat{U}_i$  are the point and variance estimates from the  $i$ th imputed data set,  $i=1, 2, \dots, m$ . Then the combined point estimate for  $Q$  from multiple imputation is the average of the  $m$  complete-data estimates:

$$\bar{Q} = \frac{1}{m} \sum_{i=1}^m \hat{Q}_i$$

Suppose  $\bar{U}$  is the within-imputation variance, which is the average of the  $m$  complete-data estimates:

$$\bar{U} = \frac{1}{m} \sum_{i=1}^m \hat{U}_i$$

and  $B$  is the between-imputation variance

$$B = \frac{1}{m-1} \sum_{i=1}^m (\hat{Q}_i - \bar{Q})^2$$

Then the variance estimate associated with  $\bar{Q}$  is the total variance (Rubin 1987)

$$T = \bar{U} + \left(1 + \frac{1}{m}\right)B$$

The statistic  $(Q - \bar{Q})T^{-(1/2)}$  is approximately distributed as  $t$  with  $v_m$  degrees of freedom (Rubin 1987), where

$$v_m = (m-1) \left[1 + \frac{\bar{U}}{(1 + m^{-1})B}\right]^2$$

When the complete-data degrees of freedom  $v_0$  is small, and there is only a modest proportion of missing data, the computed degrees of freedom,  $v_m$ , can be much larger than  $v_0$ , which is inappropriate. Barnard and Rubin (1999) recommend the use of an adjusted degrees of freedom

$$v_m^* = \left[ \frac{1}{v_m} + \frac{1}{\hat{v}_{obs}} \right]^{-1}$$

where  $\hat{v}_{obs} = (1 - \gamma) v_0 (v_0 + 1) / (v_0 + 3)$  and  $\gamma = (1 + m^{-1})B/T$ .

Note that the MI procedure uses the adjusted degrees of freedom,  $v_m^*$ , for inference.

The degrees of freedom  $v_m$  depends on  $m$  and the ratio

$$r = \frac{(1 + m^{-1})B}{\bar{U}}$$

The ratio  $r$  is called the relative increase in variance due to nonresponse (Rubin 1987). When there is no missing information about  $Q$ , the values of  $r$  and  $B$  are both zero. With a large value of  $m$  or a small value of  $r$ , the degrees of freedom  $v$  will be large and the distribution of  $(Q - \bar{Q})T^{-(1/2)}$  will be approximately normal.

Another useful statistic is the fraction of missing information about  $Q$ :

$$\hat{\lambda} = \frac{r + 2/(v + 3)}{r + 1}$$

Both statistics  $r$  and  $\lambda$  are helpful diagnostics for assessing how the missing data contribute to the uncertainty about  $Q$ .

---

## Multiple Imputation Efficiency

The relative efficiency (RE) of using the finite  $m$  imputation estimator, rather than using an infinite number for the fully efficient imputation, in units of variance, is approximately a function of  $m$  and  $\lambda$  (Rubin 1987, p. 114).

$$RE = \left(1 + \frac{\lambda}{m}\right)^{-1}$$

The following table shows relative efficiencies with different values of  $m$  and  $\lambda$ . For cases with little missing information, only a small number of imputations are necessary.

**Table 9.4.** Relative Efficiency

| $m$ | $\lambda$ |        |        |        |        |
|-----|-----------|--------|--------|--------|--------|
|     | 10%       | 20%    | 30%    | 50%    | 70%    |
| 3   | 0.9677    | 0.9375 | 0.9091 | 0.8571 | 0.8108 |
| 5   | 0.9804    | 0.9615 | 0.9434 | 0.9091 | 0.8772 |
| 10  | 0.9901    | 0.9804 | 0.9709 | 0.9524 | 0.9346 |
| 20  | 0.9950    | 0.9901 | 0.9852 | 0.9756 | 0.9662 |

---

## Imputer's Model Versus Analyst's Model

Schafer (1997, pp. 139-143) provides comprehensive coverage of this topic, and the following discussion is largely based on his work.

Multiple imputation inference assumes that the model you used to analyze the multiply imputed data (the analyst's model) is the same as the model used to impute missing values in multiple imputation (the imputer's model). But in practice, the two models may not be the same.

For example, consider the same trivariate data set with variables  $Y_1$  and  $Y_2$  fully observed, and a variable  $Y_3$  with missing values. An imputer creates multiple imputations with the model  $Y_3 = Y_1 Y_2$ . However, the analyst can later use the simpler model  $Y_3 = Y_1$ . In this case, the analyst assumes more than the imputer. That is, the analyst assumes there is no relationship between variables  $Y_3$  and  $Y_2$ .

The effect of the discrepancy between the models depends on whether the analyst's additional assumption is true. If the assumption is true, the imputer's model still applies. The inferences derived from multiple imputations will still be valid, although they may be somewhat conservative because they reflect the additional uncertainty of estimating the relationship between  $Y_3$  and  $Y_2$ .

On the other hand, suppose that the analyst models  $Y_3 = Y_1$ , and there is a relationship between variables  $Y_3$  and  $Y_2$ . Then the model  $Y_3 = Y_1$  will be biased and is inappropriate. Appropriate results can be generated only from appropriate analyst's models.

Another type of discrepancy occurs when the imputer assumes more than the analyst. For example, suppose that an imputer creates multiple imputations with the model  $Y_3 = Y_1$ , but the analyst later fits a model  $Y_3 = Y_1 Y_2$ . When the assumption is true, the imputer's model is a correct model and the inferences still hold.

On the other hand, suppose there is a relationship between  $Y_3$  and  $Y_2$ . Imputations created under the incorrect assumption that there is no relationship between  $Y_3$  and  $Y_2$  will make the analyst's estimate of the relationship biased toward zero. Multiple imputations created under an incorrect model can lead to incorrect conclusions.

Thus, generally you should include as many variables as you can when doing multiple imputation. The precision you lose when you include unimportant predictors is usually a relatively small price to pay for the general validity of analyses of the resultant multiply imputed data set (Rubin 1996).

Note that it is good practice to include a description of the imputer's model with the multiply imputed data set. That way, the analysts will have information about the variables involved in the imputation and which relationships among the variables have been implicitly set to zero.

---

## Parameter Simulation Versus Multiple Imputation

For many incomplete-data problems, simulation-based methods of parameter simulation and multiple imputation can be used to analyze the data. In parameter simulation, you simulate random values of parameters from the observed-data posterior distribution and make simple inferences about these parameters (Schafer 1997, p. 89).

When a set of well-defined population parameters  $\theta$  are of interest, parameter simulation can be used to directly examine and summarize simulated values of  $\theta$ . This usually requires a large number of iterations, and involves calculating appropriate summaries of the resulting dependent sample of the iterates of the  $\theta$ . If only a small set of parameters are involved, parameter simulation can be suitable (Schafer 1997).

In multiple imputation, the unknown missing data are replaced by multiple sets of simulated values. Each complete data set is then analyzed by standard complete-data methods. The variability among the results from these repeated analyses provides a measure of the uncertainty due to missing data. Combining this between-imputation variation with the ordinary within-imputation sample variation provides statistical inference for the parameters of interest. Multiple imputation is suitable for analyses that are more exploratory in nature.

Multiple imputation only requires a small number of imputations. Generating and storing a few imputations can be more efficient than generating and storing a large number of iterations for parameter simulation.

When fractions of missing information are low, methods that average over simulated values of the missing data, as in multiple imputation, can be much more efficient than methods that average over simulated values of  $\theta$  as in parameter simulation (Schafer 1997).

---

## ODS Table Names

PROC MI assigns a name to each table it creates. You must use these names to reference tables when using the Output Delivery System (ODS). These names are listed in the following table. For more information on ODS, refer to the chapter “Using the Output Delivery System” in the *SAS/STAT User’s Guide, Version 8*.

**Table 9.5.** ODS Tables Produced in PROC MI

| ODS Table Name | Description                                                | Option                  |
|----------------|------------------------------------------------------------|-------------------------|
| ModelInfo      | Model information                                          |                         |
| MissPattern    | Missing data patterns                                      |                         |
| Transform      | Variable Transformations                                   | TRANSFORM statement     |
| Univariate     | Univariate statistics for available cases                  | SIMPLE                  |
| Corr           | Pairwise correlations for available cases                  | SIMPLE                  |
| EMInitEst      | Initial parameter values for EM                            | EM statement            |
| EMEst          | MLE of the parameter vector from EM                        | EM statement            |
| EMIter         | EM iteration history for MLE                               | ITPRINT in EM statement |
| EMPIter        | EM iteration history for posterior mode                    | ITPRINT in INITIAL=EM   |
| EMPEst         | Posterior mode parameter values from EM                    | INITIAL=EM              |
| EMWlf          | Coefficients of the worst linear function                  | WLF                     |
| MCMCInitEst    | Initial parameter estimates for MCMC                       | DISPLAYINIT in MCMC     |
| VarianceInfo   | Between-imputation, within-imputation, and total variances |                         |
| ParmEst        | Parameter estimates                                        |                         |

## Examples

The following FitMono data set has a monotone missing data pattern and is used in Example 9.2 with the propensity score method and in Example 9.3 with the regression method. The FitMiss data set created in the “Getting Started” section is used in other examples. Note that the original data set has been altered for these examples.

```
*----- Data on Physical Fitness -----*
| These measurements were made on men involved in a physical |
| fitness course at N.C. State University.                  |
| Only selected variables of                                |
| Oxygen (oxygen intake, ml per kg body weight per minute), |
| Runtime (time to run 1.5 miles in minutes), and           |
| RunPulse (heart rate while running) are used.             |
| Certain values were changed to missing for the analysis.  |
*-----*;
```

| Oxygen | RunTime | RunPulse |
|--------|---------|----------|
| 44.609 | 11.37   | 178      |
| 54.297 | 8.65    | 156      |
| 49.874 | 9.22    | .        |
| 45.681 | 11.95   | 176      |
| 39.442 | 13.08   | 174      |
| 50.541 | .       | .        |
| 44.754 | 11.12   | 176      |
| 51.855 | 10.33   | 166      |
| 40.836 | 10.95   | 168      |
| 46.774 | 10.25   | .        |
| 39.407 | 12.63   | 174      |
| 45.441 | 9.63    | 164      |
| 45.118 | 11.08   | .        |
| 45.790 | 10.47   | 186      |
| 48.673 | 9.40    | 186      |
| 47.467 | 10.50   | 170      |

```
;
data FitMono;
  input Oxygen RunTime RunPulse @@;
  datalines;
44.609 11.37 178      45.313 10.07 185
54.297 8.65 156      59.571 . .
49.874 9.22 .        44.811 11.63 176
45.681 11.95 176     49.091 10.85 .
39.442 13.08 174     60.055 8.63 170
50.541 . .          37.388 14.03 186
44.754 11.12 176     47.273 . .
51.855 10.33 166     49.156 8.95 180
40.836 10.95 168     46.672 10.00 .
46.774 10.25 .       50.388 10.08 168
39.407 12.63 174     46.080 11.17 156
45.441 9.63 164      54.625 8.92 146
45.118 11.08 .       39.203 12.88 168
45.790 10.47 186     50.545 9.93 148
48.673 9.40 186      47.920 11.50 170
47.467 10.50 170
;
;
```

### Example 9.1. EM Algorithm for MLE

This example uses the EM algorithm to compute the maximum likelihood estimates for the parameters of a multivariate normal distribution using data with missing values. The following statements invoke the MI procedure and request the EM algorithm to compute the MLE for  $(\mu, \Sigma)$  of a multivariate normal distribution from the input data set FitMiss.

```
proc mi data=FitMiss seed=55417 simple nimpute=0;
  em itprint outem=outem;
  var Oxygen RunTime RunPulse;
run;
```

Note when you specify the option NIMPUTE=0, the missing values will not be imputed. The procedure generates the following output:

**Output 9.1.1.** Model Information

| The MI Procedure                 |                   |
|----------------------------------|-------------------|
| Model Information                |                   |
| Data Set                         | WORK.FITMISS      |
| Method                           | MCMC              |
| Multiple Imputation Chain        | Single Chain      |
| Initial Estimates for MCMC       | EM Posterior Mode |
| Start                            | Starting Value    |
| Prior                            | Jeffreys          |
| Number of Imputations            | 0                 |
| Number of Burn-in Iterations     | 200               |
| Number of Iterations             | 100               |
| Seed for random number generator | 55417             |

The “Model Information” table describes the method and options used in the procedure.

**Output 9.1.2.** Missing Data Patterns

| The MI Procedure      |                       |           |            |      |         |
|-----------------------|-----------------------|-----------|------------|------|---------|
| Missing Data Patterns |                       |           |            |      |         |
| Group                 | Oxygen                | Run Time  | Run Pulse  | Freq | Percent |
| 1                     | X                     | X         | X          | 21   | 67.74   |
| 2                     | X                     | X         | .          | 4    | 12.90   |
| 3                     | X                     | .         | .          | 3    | 9.68    |
| 4                     | .                     | X         | X          | 1    | 3.23    |
| 5                     | .                     | X         | .          | 2    | 6.45    |
| Missing Data Patterns |                       |           |            |      |         |
| Group                 | -----Group Means----- |           |            |      |         |
|                       | Oxygen                | RunTime   | RunPulse   |      |         |
| 1                     | 46.353810             | 10.809524 | 171.666667 |      |         |
| 2                     | 47.109500             | 10.137500 | .          |      |         |
| 3                     | 52.461667             | .         | .          |      |         |
| 4                     | .                     | 11.950000 | 176.000000 |      |         |
| 5                     | .                     | 9.885000  | .          |      |         |

The “Missing Data Patterns” table lists distinct missing data patterns with corresponding frequencies and percents. Here, “X” means that the variable is observed in the corresponding group and “.” means that the variable is missing. The table also displays group-specific variable means.



With the SIMPLE option, the procedure displays simple descriptive univariate statistics for available cases in the “Univariate Statistics” table and correlations from pairwise available cases in the “Pairwise Correlations” table.

**Output 9.1.3.** Univariate Statistics

| The MI Procedure      |    |           |          |           |           |
|-----------------------|----|-----------|----------|-----------|-----------|
| Univariate Statistics |    |           |          |           |           |
| Variable              | N  | Mean      | Std Dev  | Minimum   | Maximum   |
| Oxygen                | 28 | 47.11618  | 5.41305  | 37.38800  | 60.05500  |
| RunTime               | 28 | 10.68821  | 1.37988  | 8.63000   | 14.03000  |
| RunPulse              | 22 | 171.86364 | 10.14324 | 148.00000 | 186.00000 |

**Output 9.1.4.** Pairwise Correlations

| The MI Procedure      |              |              |              |  |
|-----------------------|--------------|--------------|--------------|--|
| Pairwise Correlations |              |              |              |  |
|                       | Oxygen       | RunTime      | RunPulse     |  |
| Oxygen                | 1.000000000  | -0.849118562 | -0.343961742 |  |
| RunTime               | -0.849118562 | 1.000000000  | 0.247258191  |  |
| RunPulse              | -0.343961742 | 0.247258191  | 1.000000000  |  |

With the EM statement, the procedure displays the initial parameter estimates for EM.

**Output 9.1.5.** Initial Parameter Estimates for EM

| The MI Procedure                   |          |           |           |            |
|------------------------------------|----------|-----------|-----------|------------|
| Initial Parameter Estimates for EM |          |           |           |            |
| _TYPE_                             | _NAME_   | Oxygen    | RunTime   | RunPulse   |
| MEAN                               |          | 47.116179 | 10.688214 | 171.863636 |
| COV                                | Oxygen   | 29.301078 | 0         | 0          |
| COV                                | RunTime  | 0         | 1.904067  | 0          |
| COV                                | RunPulse | 0         | 0         | 102.885281 |

With the ITPRINT option, the “EM (MLE) Iteration History” table displays the iteration history for the EM algorithm.

**Output 9.1.6.** EM (MLE) Iteration History

| The MI Procedure           |                 |               |                |                 |
|----------------------------|-----------------|---------------|----------------|-----------------|
| EM (MLE) Iteration History |                 |               |                |                 |
| <u>_Iteration_</u>         | <u>-2 Log L</u> | <u>Oxygen</u> | <u>RunTime</u> | <u>RunPulse</u> |
| 0                          | 289.544782      | 47.116179     | 10.688214      | 171.863636      |
| 1                          | 263.549489      | 47.116179     | 10.688214      | 171.863636      |
| 2                          | 255.851312      | 47.139089     | 10.603506      | 171.538203      |
| 3                          | 254.616428      | 47.122353     | 10.571685      | 171.426790      |
| 4                          | 254.494971      | 47.111080     | 10.560585      | 171.398296      |
| 5                          | 254.483973      | 47.106523     | 10.556768      | 171.389208      |
| 6                          | 254.482920      | 47.104899     | 10.555485      | 171.385257      |
| 7                          | 254.482813      | 47.104348     | 10.555062      | 171.383345      |
| 8                          | 254.482801      | 47.104165     | 10.554923      | 171.382424      |
| 9                          | 254.482800      | 47.104105     | 10.554878      | 171.381992      |
| 10                         | 254.482800      | 47.104086     | 10.554864      | 171.381796      |

The procedure then displays the EM (MLE) parameter estimates, the maximum likelihood estimates for  $\mu$  and  $\Sigma$  of a multivariate normal distribution from the data set FitMiss.

**Output 9.1.7.** EM (MLE) Parameter Estimates

| The MI Procedure             |               |               |                |                 |
|------------------------------|---------------|---------------|----------------|-----------------|
| EM (MLE) Parameter Estimates |               |               |                |                 |
| <u>_TYPE_</u>                | <u>_NAME_</u> | <u>Oxygen</u> | <u>RunTime</u> | <u>RunPulse</u> |
| MEAN                         |               | 47.104086     | 10.554864      | 171.381796      |
| COV                          | Oxygen        | 27.798014     | -6.457929      | -18.030790      |
| COV                          | RunTime       | -6.457929     | 2.015491       | 3.516092        |
| COV                          | RunPulse      | -18.030790    | 3.516092       | 97.766559       |

You can also output the EM (MLE) parameter estimates into an output data set with the OUTEM= option. The following statements list the observations in the output data set outem.

```
proc print data=outem;
  title 'EM Estimates';
run;
```

**Output 9.1.8.** EM Estimates

| EM Estimates |        |          |          |         |          |
|--------------|--------|----------|----------|---------|----------|
| Obs          | _TYPE_ | _NAME_   | Oxygen   | RunTime | RunPulse |
| 1            | MEAN   |          | 47.1041  | 10.5549 | 171.382  |
| 2            | COV    | Oxygen   | 27.7980  | -6.4579 | -18.031  |
| 3            | COV    | RunTime  | -6.4579  | 2.0155  | 3.516    |
| 4            | COV    | RunPulse | -18.0308 | 3.5161  | 97.767   |

The output data set outem is a TYPE=COV data set. The observation with \_TYPE\_='MEAN' contains the MLE for the parameter  $\mu$  and the observations with \_TYPE\_='COV' contain the MLE for the parameter  $\Sigma$  of a multivariate normal distribution from the data set FitMiss.

## Example 9.2. Propensity Score Method

This example uses the propensity score method to impute missing values in a data set with a monotone missing pattern. The following statements invoke the MI procedure and request the propensity score method. The resulting data set is named outpscore.

```
proc mi data=FitMono seed=55417 simple out=outpscore;
  monotone method=propensity;
  var Oxygen RunTime RunPulse;
run;
```

Note that the VAR statement is required and the data set must have a monotone missing pattern with variables as ordered in the VAR statement. The procedure generates the following output:

**Output 9.2.1.** Model Information

| The MI Procedure                 |              |
|----------------------------------|--------------|
| Model Information                |              |
| Data Set                         | WORK.FITMONO |
| Method                           | Propensity   |
| Number of Imputations            | 5            |
| Number of Groups on Propensity   | 5            |
| Seed for random number generator | 55417        |

The “Model Information” table describes the method and options used in the multiple imputation process. By default, the observations are sorted into five groups based on the propensity scores, and five imputations are created for the missing data.

**Output 9.2.2.** Missing Data Patterns

| The MI Procedure      |                       |           |            |      |         |
|-----------------------|-----------------------|-----------|------------|------|---------|
| Missing Data Patterns |                       |           |            |      |         |
| Group                 | Oxygen                | Run Time  | Run Pulse  | Freq | Percent |
| 1                     | X                     | X         | X          | 23   | 74.19   |
| 2                     | X                     | X         | .          | 5    | 16.13   |
| 3                     | X                     | .         | .          | 3    | 9.68    |
| Missing Data Patterns |                       |           |            |      |         |
| Group                 | -----Group Means----- |           |            |      |         |
|                       | Oxygen                | RunTime   | RunPulse   |      |         |
| 1                     | 46.684174             | 10.776957 | 170.739130 |      |         |
| 2                     | 47.505800             | 10.280000 | .          |      |         |
| 3                     | 52.461667             | .         | .          |      |         |

The “Missing Data Patterns” table lists distinct missing data patterns with corresponding frequencies and percents. Here, “X” means that the variable is observed in the corresponding group and “.” means that the variable is missing. The table also displays group-specific variable means.

**Output 9.2.3.** Variance Information

| The MI Procedure                         |                               |                              |          |        |
|------------------------------------------|-------------------------------|------------------------------|----------|--------|
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | -----Variance-----            |                              |          | DF     |
|                                          | Between                       | Within                       | Total    |        |
| RunTime                                  | 0.001068                      | 0.059100                     | 0.060382 | 27.498 |
| RunPulse                                 | 1.147555                      | 4.686646                     | 6.063711 | 17.006 |
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | Relative Increase in Variance | Fraction Missing Information |          |        |
| RunTime                                  | 0.021688                      | 0.021448                     |          |        |
| RunPulse                                 | 0.293828                      | 0.246288                     |          |        |

After the completion of  $m$  imputations, the “Multiple Imputation Variance Information” table displays the between-imputation variance, within-imputation variance, and total variance for combining complete-data inferences. It also displays the degrees of freedom for the total variance. The relative increase in variance due to miss-

ingness and the fraction of missing information for each variable are also displayed. A detailed description of these statistics is provided in the “Combining Inferences from Multiply Imputed Data Sets” section on page 173.

The “Multiple Imputation Parameter Estimates” table displays the estimated mean and standard error of the mean for each variable. The inferences are based on the  $t$ -distributions. For each variable, the table also displays a 95% mean confidence interval and a  $t$ -statistic with the associated  $p$ -value for the hypothesis that the population mean is equal to the value specified in the MU0= option, which is zero by default.

#### Output 9.2.4. Parameter Estimates

| The MI Procedure                        |            |            |                                   |          |        |
|-----------------------------------------|------------|------------|-----------------------------------|----------|--------|
| Multiple Imputation Parameter Estimates |            |            |                                   |          |        |
| Variable                                | Mean       | Std Error  | 95% Confidence Limits             |          | DF     |
| RunTime                                 | 10.603677  | 0.245727   | 10.0999                           | 11.1074  | 27.498 |
| RunPulse                                | 170.400000 | 2.462460   | 165.2048                          | 175.5952 | 17.006 |
| Multiple Imputation Parameter Estimates |            |            |                                   |          |        |
| Variable                                | Minimum    | Maximum    | t for H0:<br>Mu0 Mean=Mu0 Pr >  t |          |        |
| RunTime                                 | 10.558065  | 10.648387  | 0                                 | 43.15    | <.0001 |
| RunPulse                                | 168.967742 | 171.838710 | 0                                 | 69.20    | <.0001 |

The following statements list the first ten observations of the data set outpscore.

```
proc print data=outpscore(obs=10);
  title 'First 10 Observations of the Imputed Data Set';
run;
```

#### Output 9.2.5. Imputed Data Set

| First 10 Observations of the Imputed Data Set |              |        |          |           |
|-----------------------------------------------|--------------|--------|----------|-----------|
| Obs                                           | _Imputation_ | Oxygen | Run Time | Run Pulse |
| 1                                             | 1            | 44.609 | 11.37    | 178       |
| 2                                             | 1            | 45.313 | 10.07    | 185       |
| 3                                             | 1            | 54.297 | 8.65     | 156       |
| 4                                             | 1            | 59.571 | 8.63     | 146       |
| 5                                             | 1            | 49.874 | 9.22     | 156       |
| 6                                             | 1            | 44.811 | 11.63    | 176       |
| 7                                             | 1            | 45.681 | 11.95    | 176       |
| 8                                             | 1            | 49.091 | 10.85    | 156       |
| 9                                             | 1            | 39.442 | 13.08    | 174       |
| 10                                            | 1            | 60.055 | 8.63     | 170       |

### Example 9.3. Regression Method

This example uses the regression method to impute missing values in a data set with a monotone missing pattern. The following statements invoke the MI procedure and request the regression method. The resulting data set is named `outreg`.

```
proc mi data=FitMono round=.001 .01 1 mu0= 50 10 150
    seed=55417 out=outreg;
    monotone method=reg;
    var Oxygen RunTime RunPulse;
run;
```

The `ROUND=` option is used to round the imputed values to the same precision as observed values. The values specified with the `ROUND=` option are matched with the variables `Oxygen`, `RunTime`, and `RunPulse` in the order listed with the `VAR` statement. The `MU0=` option requests *t* tests for the hypotheses that the population means corresponding to the variables in the `VAR` statement are `Oxygen=50`, `RunTime=10`, and `RunPulse=150`.

The “Missing Data Patterns” table lists distinct missing data patterns with corresponding frequencies and percents. It is identical to the table in the previous example.

After the completion of five imputations by default, the “Multiple Imputation Variance Information” table displays the between-imputation variance, within-imputation variance, and total variance for combining complete-data inferences. The relative increase in variance due to missingness and the fraction of missing information for each variable are also displayed. These statistics are described in the “Combining Inferences from Multiply Imputed Data Sets” section on page 173.

**Output 9.3.1.** Variance Information

| The MI Procedure                         |                               |                              |          |        |
|------------------------------------------|-------------------------------|------------------------------|----------|--------|
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | -----Variance-----            |                              |          | DF     |
|                                          | Between                       | Within                       | Total    |        |
| RunTime                                  | 0.004443                      | 0.068684                     | 0.074016 | 25.294 |
| RunPulse                                 | 1.790531                      | 4.045134                     | 6.193770 | 11.846 |
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | Relative Increase in Variance | Fraction Missing Information |          |        |
| RunTime                                  | 0.077629                      | 0.074435                     |          |        |
| RunPulse                                 | 0.531166                      | 0.382947                     |          |        |

The “Multiple Imputation Parameter Estimates” table displays a 95% mean confidence interval and a *t*-statistic with its associated *p*-value for each of the hypotheses requested with the `MU0=` option.

**Output 9.3.2.** Parameter Estimates

| The MI Procedure                        |            |            |                       |                       |         |
|-----------------------------------------|------------|------------|-----------------------|-----------------------|---------|
| Multiple Imputation Parameter Estimates |            |            |                       |                       |         |
| Variable                                | Mean       | Std Error  | 95% Confidence Limits |                       | DF      |
| RunTime                                 | 10.575871  | 0.272059   | 10.0159               | 11.1359               | 25.294  |
| RunPulse                                | 170.425806 | 2.488729   | 164.9955              | 175.8561              | 11.846  |
| Multiple Imputation Parameter Estimates |            |            |                       |                       |         |
| Variable                                | Minimum    | Maximum    | Mu0                   | t for H0:<br>Mean=Mu0 | Pr >  t |
| RunTime                                 | 10.506452  | 10.680968  | 10.000000             | 2.12                  | 0.0443  |
| RunPulse                                | 169.290323 | 171.935484 | 150.000000            | 8.21                  | <.0001  |

The following statements list the first ten observations of the data set `outreg`. Note that the imputed values rounded to the same precision as the observed values.

```
proc print data=outreg(obs=10);
  title 'First 10 Observations of the Imputed Data Set';
run;
```

**Output 9.3.3.** Imputed Data Set

| First 10 Observations of the Imputed Data Set |              |        |          |           |
|-----------------------------------------------|--------------|--------|----------|-----------|
| Obs                                           | _Imputation_ | Oxygen | Run Time | Run Pulse |
| 1                                             | 1            | 44.609 | 11.37    | 178       |
| 2                                             | 1            | 45.313 | 10.07    | 185       |
| 3                                             | 1            | 54.297 | 8.65     | 156       |
| 4                                             | 1            | 59.571 | 7.18     | 156       |
| 5                                             | 1            | 49.874 | 9.22     | 192       |
| 6                                             | 1            | 44.811 | 11.63    | 176       |
| 7                                             | 1            | 45.681 | 11.95    | 176       |
| 8                                             | 1            | 49.091 | 10.85    | 174       |
| 9                                             | 1            | 39.442 | 13.08    | 174       |
| 10                                            | 1            | 60.055 | 8.63     | 170       |

**Example 9.4. MCMC Method**

This example uses the MCMC method to impute missing values for a data set with an arbitrary missing pattern. The following statements invoke the MI procedure and specify the MCMC method with three imputations.

```
proc mi data=FitMiss seed=55417 nimpute=3 mu0=50 10 180;
  mcmc chain=multiple displayinit initial=em(itprint);
  var Oxygen RunTime RunPulse;
run;
```

**Output 9.4.1.** Model Information

| The MI Procedure                 |                   |
|----------------------------------|-------------------|
| Model Information                |                   |
| Data Set                         | WORK.FITMISS      |
| Method                           | MCMC              |
| Multiple Imputation Chain        | Multiple Chains   |
| Initial Estimates for MCMC       | EM Posterior Mode |
| Start                            | Starting Value    |
| Prior                            | Jeffreys          |
| Number of Imputations            | 3                 |
| Number of Burn-in Iterations     | 200               |
| Seed for random number generator | 55417             |

With CHAIN=MULTIPLE, the procedure uses multiple chains and completes the default 200 burn-in iterations before each imputation. The 200 burn-in iterations are used to make the iterations converge to the stationary distribution before the imputation.

By default, the procedure uses a noninformative Jeffreys prior to derive the posterior mode from the EM algorithm as the starting values for the MCMC process.

The following “Missing Data Patterns” table lists distinct missing data patterns with corresponding statistics.

**Output 9.4.2.** Missing Data Patterns

| The MI Procedure      |        |          |           |      |         |
|-----------------------|--------|----------|-----------|------|---------|
| Missing Data Patterns |        |          |           |      |         |
| Group                 | Oxygen | Run Time | Run Pulse | Freq | Percent |
| 1                     | X      | X        | X         | 21   | 67.74   |
| 2                     | X      | X        | .         | 4    | 12.90   |
| 3                     | X      | .        | .         | 3    | 9.68    |
| 4                     | .      | X        | X         | 1    | 3.23    |
| 5                     | .      | X        | .         | 2    | 6.45    |

| Missing Data Patterns |           |           |            |
|-----------------------|-----------|-----------|------------|
| -----Group Means----- |           |           |            |
| Group                 | Oxygen    | RunTime   | RunPulse   |
| 1                     | 46.353810 | 10.809524 | 171.666667 |
| 2                     | 47.109500 | 10.137500 | .          |
| 3                     | 52.461667 | .         | .          |
| 4                     | .         | 11.950000 | 176.000000 |
| 5                     | .         | 9.885000  | .          |

With the ITPRINT option in INITIAL=EM, the procedure also displays the “EM (Posterior Mode) Iteration History” table.



**Output 9.4.3.** EM (Posterior Mode) Iteration History

| The MI Procedure                      |            |                  |           |           |
|---------------------------------------|------------|------------------|-----------|-----------|
| EM (Posterior Mode) Iteration History |            |                  |           |           |
| _Iteration_                           | -2 Log L   | -2 Log Posterior | Oxygen    | RunTime   |
| 0                                     | 254.482800 | 282.909590       | 47.104086 | 10.554864 |
| 1                                     | 255.081159 | 282.051588       | 47.104079 | 10.554859 |
| 2                                     | 255.271405 | 282.017488       | 47.104077 | 10.554858 |
| 3                                     | 255.318621 | 282.015372       | 47.104002 | 10.554524 |
| 4                                     | 255.330259 | 282.015232       | 47.103861 | 10.554388 |
| 5                                     | 255.333160 | 282.015222       | 47.103797 | 10.554341 |
| 6                                     | 255.333896 | 282.015222       | 47.103774 | 10.554325 |
| 7                                     | 255.334085 | 282.015222       | 47.103766 | 10.554320 |
| EM (Posterior Mode) Iteration History |            |                  |           |           |
| _Iteration_                           | RunPulse   |                  |           |           |
| 0                                     | 171.381796 |                  |           |           |
| 1                                     | 171.381708 |                  |           |           |
| 2                                     | 171.381669 |                  |           |           |
| 3                                     | 171.381853 |                  |           |           |
| 4                                     | 171.382058 |                  |           |           |
| 5                                     | 171.382152 |                  |           |           |
| 6                                     | 171.382186 |                  |           |           |
| 7                                     | 171.382197 |                  |           |           |

With the DISPLAYINIT option in the MCMC statement, the following “Initial Parameter Estimates for MCMC” table displays the starting mean and covariance estimates used in MCMC. The same starting estimates are used for the MCMC process for multiple chains because the EM algorithm is applied to the same data set in each chain. You can explicitly specify different initial estimates for different imputations, or you can use the bootstrap to generate different parameter estimates from the EM algorithm for the MCMC process.

**Output 9.4.4.** Initial Parameter Estimates

| The MI Procedure                     |          |            |           |            |
|--------------------------------------|----------|------------|-----------|------------|
| Initial Parameter Estimates for MCMC |          |            |           |            |
| _TYPE_                               | _NAME_   | Oxygen     | RunTime   | RunPulse   |
| MEAN                                 |          | 47.103766  | 10.554320 | 171.382197 |
| COV                                  | Oxygen   | 24.549968  | -5.726112 | -15.926034 |
| COV                                  | RunTime  | -5.726112  | 1.781407  | 3.124798   |
| COV                                  | RunPulse | -15.926034 | 3.124798  | 83.164044  |

The following two tables display variance information and parameter estimates from the multiple imputation.

**Output 9.4.5.** Variance Information

| The MI Procedure                         |                               |                              |          |        |
|------------------------------------------|-------------------------------|------------------------------|----------|--------|
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | -----Variance-----            |                              |          | DF     |
|                                          | Between                       | Within                       | Total    |        |
| Oxygen                                   | 0.009200                      | 0.987880                     | 1.000148 | 27.778 |
| RunTime                                  | 0.002255                      | 0.069112                     | 0.072119 | 26.388 |
| RunPulse                                 | 0.043126                      | 3.650388                     | 3.707889 | 27.653 |
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | Relative Increase in Variance | Fraction Missing Information |          |        |
| Oxygen                                   | 0.012418                      | 0.012414                     |          |        |
| RunTime                                  | 0.043503                      | 0.043351                     |          |        |
| RunPulse                                 | 0.015752                      | 0.015744                     |          |        |

**Output 9.4.6.** Parameter Estimates

| The MI Procedure                        |            |            |                       |                    |         |
|-----------------------------------------|------------|------------|-----------------------|--------------------|---------|
| Multiple Imputation Parameter Estimates |            |            |                       |                    |         |
| Variable                                | Mean       | Std Error  | 95% Confidence Limits |                    | DF      |
| Oxygen                                  | 47.198228  | 1.000074   | 45.1489               | 49.2475            | 27.778  |
| RunTime                                 | 10.510911  | 0.268549   | 9.9593                | 11.0625            | 26.388  |
| RunPulse                                | 172.113649 | 1.925588   | 168.1670              | 176.0603           | 27.653  |
| Multiple Imputation Parameter Estimates |            |            |                       |                    |         |
| Variable                                | Minimum    | Maximum    | Mu0                   | t for H0: Mean=Mu0 | Pr >  t |
| Oxygen                                  | 47.132351  | 47.308274  | 50.000000             | -2.80              | 0.0092  |
| RunTime                                 | 10.456079  | 10.538446  | 10.000000             | 1.90               | 0.0681  |
| RunPulse                                | 171.943144 | 172.344920 | 180.000000            | -4.10              | 0.0003  |

## Example 9.5. Producing Monotone Missingness with MCMC

This example uses the MCMC method to impute just enough missing values for a data set with an arbitrary missing pattern so that each imputed data set has a monotone missing pattern based on the order of variables in the VAR statement.

The following statements invoke the MI procedure and specify the the IMPUTE=MONOTONE option to create the imputed data set with a monotone missing pattern. You must specify a VAR list to provide the order of variables for the imputed data to achieve a monotone missing pattern.

```
proc mi data=FitMiss seed=55417 out=outmono;
  mcmc impute=monotone;
  var Oxygen RunTime RunPulse;
run;
```

**Output 9.5.1.** Model Information

| The MI Procedure                 |                    |
|----------------------------------|--------------------|
| Model Information                |                    |
| Data Set                         | WORK.FITMISS       |
| Method                           | Monotone-data MCMC |
| Multiple Imputation Chain        | Single Chain       |
| Initial Estimates for MCMC       | EM Posterior Mode  |
| Start                            | Starting Value     |
| Prior                            | Jeffreys           |
| Number of Imputations            | 5                  |
| Number of Burn-in Iterations     | 200                |
| Number of Iterations             | 100                |
| Seed for random number generator | 55417              |

The following “Missing Data Patterns” table lists distinct missing data patterns with corresponding statistics. Here, an “X” means that the variable is observed in the corresponding group, a “.” means that the variable is missing and will be imputed to achieve the monotone missingness for the imputed data set, and an “O” means that the variable is missing and will not be imputed. The table also displays group-specific variable means.

**Output 9.5.2.** Missing Data Pattern

| The MI Procedure      |           |           |            |      |         |
|-----------------------|-----------|-----------|------------|------|---------|
| Missing Data Patterns |           |           |            |      |         |
| Group                 | Oxygen    | Run Time  | Run Pulse  | Freq | Percent |
| 1                     | X         | X         | X          | 21   | 67.74   |
| 2                     | X         | X         | O          | 4    | 12.90   |
| 3                     | X         | O         | O          | 3    | 9.68    |
| 4                     | .         | X         | X          | 1    | 3.23    |
| 5                     | .         | X         | O          | 2    | 6.45    |
| Missing Data Patterns |           |           |            |      |         |
| -----Group Means----- |           |           |            |      |         |
| Group                 | Oxygen    | RunTime   | RunPulse   |      |         |
| 1                     | 46.353810 | 10.809524 | 171.666667 |      |         |
| 2                     | 47.109500 | 10.137500 | .          |      |         |
| 3                     | 52.461667 | .         | .          |      |         |
| 4                     | .         | 11.950000 | 176.000000 |      |         |
| 5                     | .         | 9.885000  | .          |      |         |

As shown in the table, the MI procedure only needs to impute three missing values from Group 4 and Group 5 to achieve a monotone missing pattern for the imputed data set.

When using the MCMC method to produce an imputed data set with a monotone missing pattern, tables of variance information and parameter estimates are not created.

The following statements are used just to show the monotone missingness of the output data set `outmono`.

```
proc mi data=outmono ( where= (_Imputation_=1) )
  nimpute=0;
  var Oxygen RunTime RunPulse;
run;
```

**Output 9.5.3.** Monotone Missing Data Pattern

| The MI Procedure      |                       |           |            |      |         |
|-----------------------|-----------------------|-----------|------------|------|---------|
| Missing Data Patterns |                       |           |            |      |         |
| Group                 | Oxygen                | Run Time  | Run Pulse  | Freq | Percent |
| 1                     | X                     | X         | X          | 22   | 70.97   |
| 2                     | X                     | X         | .          | 6    | 19.35   |
| 3                     | X                     | .         | .          | 3    | 9.68    |
| Missing Data Patterns |                       |           |            |      |         |
| Group                 | -----Group Means----- |           |            |      |         |
|                       | Oxygen                | RunTime   | RunPulse   |      |         |
| 1                     | 46.307744             | 10.861364 | 171.863636 |      |         |
| 2                     | 46.372151             | 10.053333 | .          |      |         |
| 3                     | 52.461667             | .         | .          |      |         |

The following statements impute one value for each missing value in the monotone missingness data set `outmono`. The variable `_Imputation_` is renamed to `Impute` so that it will not be overwritten by the the new variable `_Imputation_` being created in the MI procedure.

```
proc mi data=outmono( rename=(_Imputation_=Impute))
  nimpute=1 seed=43672
  out=outds( rename=(Impute=_Imputation_) drop=_Imputation_);
  monotone method=reg;
  var Oxygen RunTime RunPulse;
  by Impute;
run;
```

The variable `Impute` is renamed to `_Imputation_` in the output data `outsds`. This makes the output data set have the same structure as output data sets generated from other imputation methods. You can then analyze these data sets by using other SAS procedures and combine these results by using the procedure `MIANALYZE`. Note that the `VAR` statement is required with a `MONOTONE` statement to provide the variable order for the monotone missing pattern.

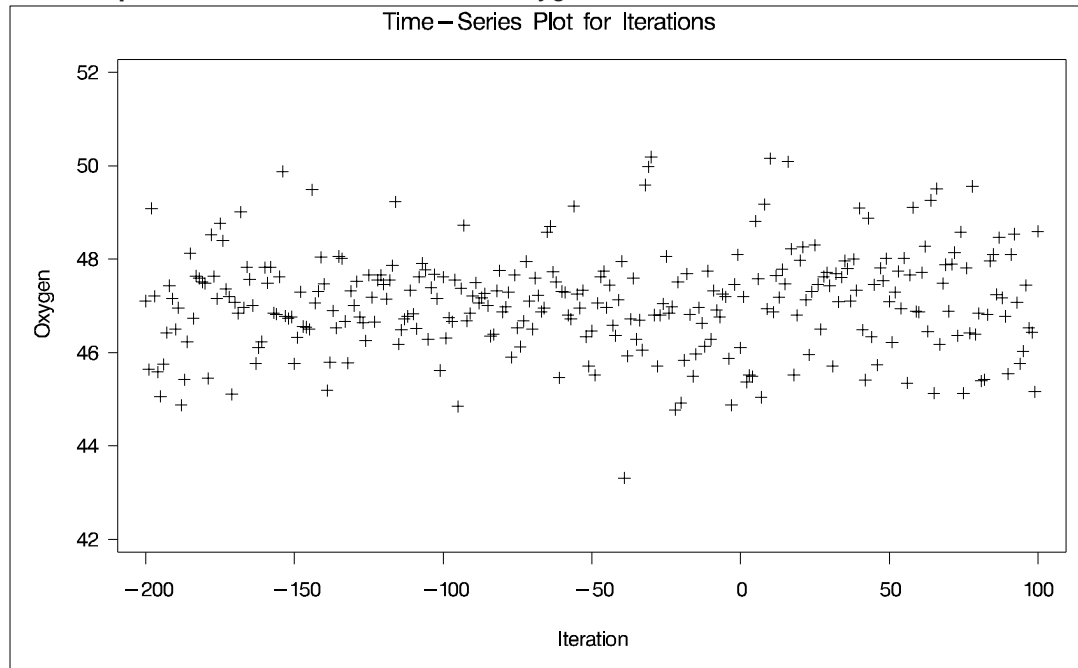
## Example 9.6. Checking Convergence in MCMC

This example uses the MCMC method with a single chain. It also displays time-series and autocorrelation plots to check convergence for the single chain.

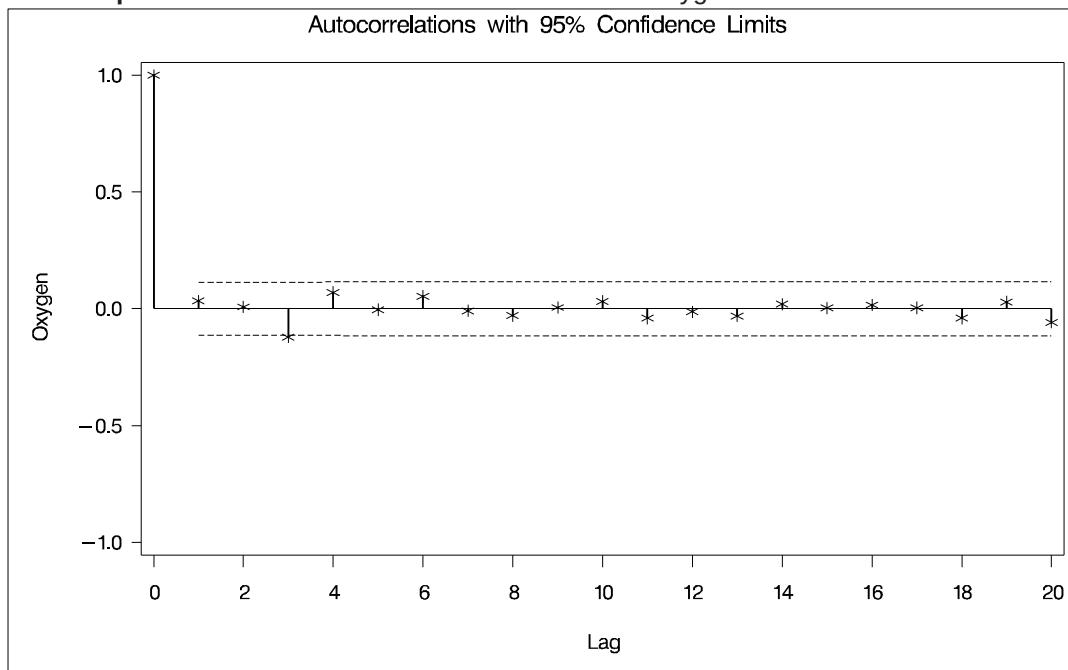
The following statements use the MCMC method to create an iteration plot for the successive estimates of the mean of `Oxygen`. Note that iterations during the burn-in period are indicated with negative iteration numbers. These statements also create an autocorrelation function plot for the variable `Oxygen`.

```
proc mi data=FitMiss seed=37921 noprint nimpute=2;
  mcmc timeplot(mean(Oxygen)) acfplot(mean(Oxygen));
  var Oxygen RunTime RunPulse;
run;
```

Output 9.6.1. Time-Series Plot for Oxygen



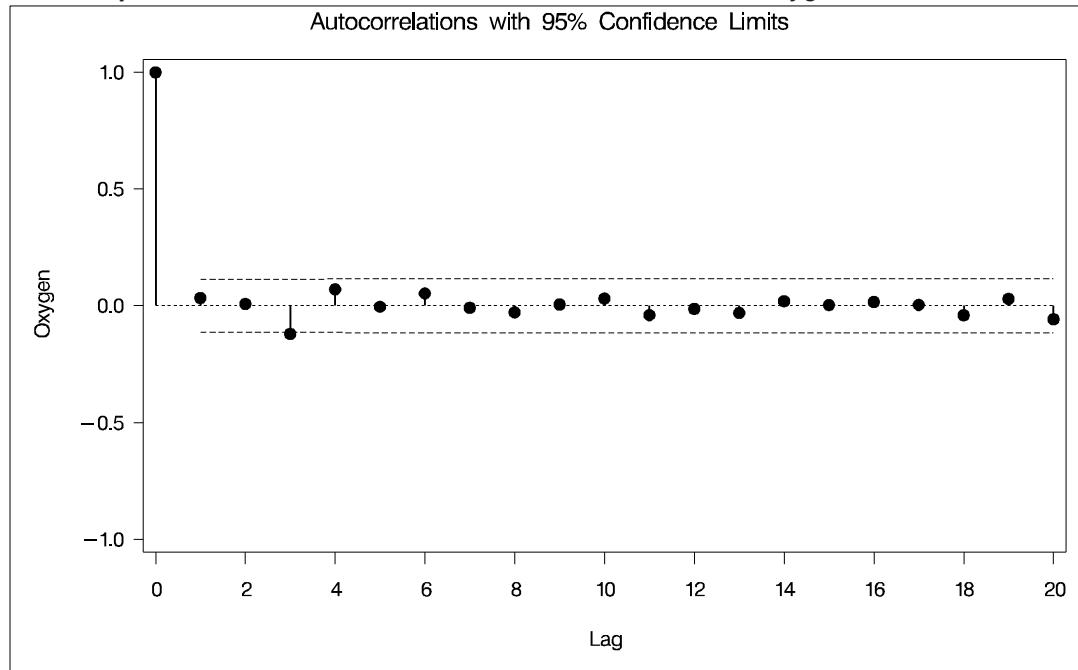
By default, the MI procedure uses the plus sign (+) as the plot symbol to display the points in the plot. The time-series plot shows no apparent trends for the variable `Oxygen`.

**Output 9.6.2.** Autocorrelation Function Plot for Oxygen

By default, the MI procedure uses the star sign (\*) as the plot symbol to display the points in the plot, a solid line to display the reference line of zero autocorrelation, and a pair of dashed lines to display approximately 95% confidence limits for the autocorrelations. The autocorrelation function plot shows no significant positive or negative autocorrelation.

The following statements use display options to modify the autocorrelation function plot for Oxygen.

```
proc mi data=FitMiss seed=37921 noprint nimpute=2;
  mcmc acfplot(mean(Oxygen) / symbol=dot lref=2);
  var Oxygen RunTime RunPulse;
run;
```

**Output 9.6.3.** Modified Autocorrelation Function Plot for Oxygen

You can also create plots for the worst linear function, the means of other variables, the variances of variables, and covariances between variables. Alternatively, you can use the OUTITER option to save statistics such as the means, standard deviations, covariances,  $-2 \log LR$  statistic,  $-2 \log LR$  statistic of the posterior mode, and worst linear function from each iteration in an output data set. Then you can do a more in-depth time-series analysis of the iterations with other procedures, such as PROC AUTOREG and PROC ARIMA in the *SAS/ETS User's Guide, Version 8*.

## Example 9.7. Transformation to Normality

This example applies the MCMC method to the FitMiss data set in which the variable Oxygen is transformed. Assume that Oxygen is skewed and can be transformed to normality with a logarithmic transformation. The following statements invoke the MI procedure and specify the transformation. The TRANSFORM statement specifies the log transformation for Oxygen. Note that the values displayed for Oxygen in all of the results correspond to transformed values.

```
proc mi data=FitMiss seed=37921 mu0=50 10 180 out=outmi;
  transform log(Oxygen);
  mcmc chain=multiple displayinit;
  var Oxygen RunTime RunPulse;
run;
```

The following “Missing Data Patterns” table lists distinct missing data patterns with corresponding statistics for the FitMiss data. Note that the values of Oxygen shown in the tables are transformed values.

**Output 9.7.1.** Missing Data Pattern

| The MI Procedure              |          |           |            |      |         |
|-------------------------------|----------|-----------|------------|------|---------|
| Missing Data Patterns         |          |           |            |      |         |
| Group                         | Oxygen   | Run Time  | Run Pulse  | Freq | Percent |
| 1                             | X        | X         | X          | 21   | 67.74   |
| 2                             | X        | X         | .          | 4    | 12.90   |
| 3                             | X        | .         | .          | 3    | 9.68    |
| 4                             | .        | X         | X          | 1    | 3.23    |
| 5                             | .        | X         | .          | 2    | 6.45    |
| Transformed Variables: Oxygen |          |           |            |      |         |
| Missing Data Patterns         |          |           |            |      |         |
| -----Group Means-----         |          |           |            |      |         |
| Group                         | Oxygen   | RunTime   | RunPulse   |      |         |
| 1                             | 3.829760 | 10.809524 | 171.666667 |      |         |
| 2                             | 3.851813 | 10.137500 | .          |      |         |
| 3                             | 3.955298 | .         | .          |      |         |
| 4                             | .        | 11.950000 | 176.000000 |      |         |
| 5                             | .        | 9.885000  | .          |      |         |
| Transformed Variables: Oxygen |          |           |            |      |         |



The following “Variable Transformations” table lists the variables that have been transformed.

**Output 9.7.2.** Missing Data Pattern

| The MI Procedure         |             |
|--------------------------|-------------|
| Variable Transformations |             |
| Variable                 | _Transform_ |
| Oxygen                   | LOG         |

The following “Initial Parameter Estimates for MCMC” table displays the starting mean and covariance estimates used in the MCMC process.

**Output 9.7.3.** Initial Parameter Estimates

| The MI Procedure                     |          |           |           |            |
|--------------------------------------|----------|-----------|-----------|------------|
| Initial Parameter Estimates for MCMC |          |           |           |            |
| _TYPE_                               | _NAME_   | Oxygen    | RunTime   | RunPulse   |
| MEAN                                 |          | 3.846122  | 10.557605 | 171.382949 |
| COV                                  | Oxygen   | 0.010827  | -0.120891 | -0.328772  |
| COV                                  | RunTime  | -0.120891 | 1.744580  | 3.011179   |
| COV                                  | RunPulse | -0.328772 | 3.011179  | 82.747608  |
| Transformed Variables: Oxygen        |          |           |           |            |

The following table displays variance information from the multiple imputation.

**Output 9.7.4.** Variance Information

| The MI Procedure                         |                               |                              |          |        |
|------------------------------------------|-------------------------------|------------------------------|----------|--------|
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | -----Variance-----            |                              |          | DF     |
|                                          | Between                       | Within                       | Total    |        |
| * Oxygen                                 | 0.000004541                   | 0.000398                     | 0.000404 | 27.766 |
| RunTime                                  | 0.000814                      | 0.063128                     | 0.064105 | 27.708 |
| RunPulse                                 | 0.182700                      | 3.498974                     | 3.718214 | 25.923 |
| * Transformed Variables                  |                               |                              |          |        |
| Multiple Imputation Variance Information |                               |                              |          |        |
| Variable                                 | Relative Increase in Variance | Fraction Missing Information |          |        |
| * Oxygen                                 | 0.013685                      | 0.013590                     |          |        |
| RunTime                                  | 0.015478                      | 0.015356                     |          |        |
| RunPulse                                 | 0.062658                      | 0.060595                     |          |        |
| * Transformed Variables                  |                               |                              |          |        |

The following table displays parameter estimates from the multiple imputation. Note that the parameter value of Mu0 has also been transformed using the logarithmic transformation.

**Output 9.7.5.** Parameter Estimates

| The MI Procedure                        |            |            |                       |                    |         |
|-----------------------------------------|------------|------------|-----------------------|--------------------|---------|
| Multiple Imputation Parameter Estimates |            |            |                       |                    |         |
| Variable                                | Mean       | Std Error  | 95% Confidence Limits |                    | DF      |
| * Oxygen                                | 3.845991   | 0.020091   | 3.8048                | 3.8872             | 27.766  |
| RunTime                                 | 10.586242  | 0.253190   | 10.0674               | 11.1051            | 27.708  |
| RunPulse                                | 170.849654 | 1.928267   | 166.8855              | 174.8138           | 25.923  |
| * Transformed Variables                 |            |            |                       |                    |         |
| Multiple Imputation Parameter Estimates |            |            |                       |                    |         |
| Variable                                | Minimum    | Maximum    | Mu0                   | t for H0: Mean=Muo | Pr >  t |
| * Oxygen                                | 3.843860   | 3.848775   | 3.912023              | -3.29              | 0.0028  |
| RunTime                                 | 10.547440  | 10.616746  | 10.000000             | 2.32               | 0.0282  |
| RunPulse                                | 170.315955 | 171.324638 | 180.000000            | -4.75              | <.0001  |
| * Transformed Variables                 |            |            |                       |                    |         |

The following statements list the first ten observations of the data set `outmi`. Note that the values for `Oxygen` are in the original scale.

```
proc print data=outmi(obs=10);
  title 'First 10 Observations of the Imputed Data Set';
run;
```

**Output 9.7.6.** Imputed Data Set in Original Scale

| First 10 Observations of the Imputed Data Set |              |         |         |              |
|-----------------------------------------------|--------------|---------|---------|--------------|
| Obs                                           | _Imputation_ | Oxygen  | RunTime | Run<br>Pulse |
| 1                                             | 1            | 44.6090 | 11.3700 | 178.000      |
| 2                                             | 1            | 45.3130 | 10.0700 | 185.000      |
| 3                                             | 1            | 54.2970 | 8.6500  | 156.000      |
| 4                                             | 1            | 59.5710 | 8.4840  | 155.503      |
| 5                                             | 1            | 49.8740 | 9.2200  | 166.031      |
| 6                                             | 1            | 44.8110 | 11.6300 | 176.000      |
| 7                                             | 1            | 43.4130 | 11.9500 | 176.000      |
| 8                                             | 1            | 44.6435 | 10.8500 | 173.761      |
| 9                                             | 1            | 39.4420 | 13.0800 | 174.000      |
| 10                                            | 1            | 60.0550 | 8.6300  | 170.000      |

The preceding results can also be produced from the following statements without using a `TRANSFORM` statement.

```
data temp;
  set FitMiss;
  LogOxygen= log(Oxygen);
run;

proc mi data=temp seed=37921 mu0=3.91202 10 180 out=outtemp;
  mcmc chain=multiple displayinit;
  var LogOxygen RunTime RunPulse;
run;

data outmi;
  set outtemp;
  Oxygen= exp(LogOxygen);
run;
```

Note that a transformed value of  $\log(50)=3.91202$  is used in the `MU0=` option.

## Example 9.8. Saving and Using Parameters for MCMC

This example uses the MCMC method with multiple chains as specified in Example 9.4. It saves the parameter values used for each imputation in an output data set of type EST. This output data set can then be used to impute missing values in other similar input data sets. The following statements invoke the MI procedure and specify the MCMC method with multiple chains to create three imputations.

```
proc mi data=FitMiss seed=55417 nimpute=3 mu0=50 10 180 noprint;
  mcmc chain=multiple outest=miest;
  var Oxygen RunTime RunPulse;
run;
```

The following statements list the parameters used for the imputations. Note that the data set includes observations with `_TYPE_='SEED'` containing the seed to start the next random number generator.

```
proc print data=miest;
  title 'Parameters for the Imputations';
run;
```

**Output 9.8.1.** OUTEST Data Set

| Parameters for the Imputations |              |        |          |               |               |               |
|--------------------------------|--------------|--------|----------|---------------|---------------|---------------|
| Obs                            | _Imputation_ | _TYPE_ | _NAME_   | Oxygen        | RunTime       | RunPulse      |
| 1                              | 1            | SEED   |          | 2099769086.00 | 2099769086.00 | 2099769086.00 |
| 2                              | 1            | PARM   |          | 49.31         | 10.00         | 172.19        |
| 3                              | 1            | COV    | Oxygen   | 32.05         | -7.47         | -28.32        |
| 4                              | 1            | COV    | RunTime  | -7.47         | 2.41          | 6.75          |
| 5                              | 1            | COV    | RunPulse | -28.32        | 6.75          | 128.61        |
| 6                              | 2            | SEED   |          | 419117425.00  | 419117425.00  | 419117425.00  |
| 7                              | 2            | PARM   |          | 47.49         | 10.43         | 171.58        |
| 8                              | 2            | COV    | Oxygen   | 41.02         | -8.60         | -34.29        |
| 9                              | 2            | COV    | RunTime  | -8.60         | 2.25          | 7.61          |
| 10                             | 2            | COV    | RunPulse | -34.29        | 7.61          | 142.94        |
| 11                             | 3            | SEED   |          | 535522494.00  | 535522494.00  | 535522494.00  |
| 12                             | 3            | PARM   |          | 45.98         | 10.82         | 172.45        |
| 13                             | 3            | COV    | Oxygen   | 43.24         | -9.90         | 8.14          |
| 14                             | 3            | COV    | RunTime  | -9.90         | 2.75          | -2.72         |
| 15                             | 3            | COV    | RunPulse | 8.14          | -2.72         | 218.32        |

The following statements invoke the MI procedure and use the `INEST=` option in the MCMC statement.

```
proc mi data=FitMiss;
  mcmc inest=miest;
  var Oxygen RunTime RunPulse;
run;
```

**Output 9.8.2. Model Information**

| The MI Procedure      |              |
|-----------------------|--------------|
| Model Information     |              |
| Data Set              | WORK.FITMISS |
| Method                | MCMC         |
| INEST Data Set        | WORK.MIEST   |
| Number of Imputations | 3            |

The remaining tables for the example are identical to the tables in Example 9.4.

---

## References

- Anderson, T.W. (1984), *An Introduction to Multivariate Statistical Analysis*, Second Edition, New York: John Wiley & Sons, Inc.
- Allison, P.D. (2000), "Multiple Imputation for Missing Data: A Cautionary Tale," *Sociological Methods and Research*, 28, 301–309.
- Barnard, J. and Rubin, D.B. (1999), "Small-Sample Degrees of Freedom with Multiple Imputation," *Biometrika*, 86, 948–955.
- Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977), "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, Ser. B.*, 39, 1–38.
- Gelman, A. and Rubin, D.B. (1992), "Inference from Iterative Simulation Using Multiple Sequences," *Statistical Science*, 7, 457–472.
- Goodnight, J.H. (1979), "A Tutorial on the Sweep Operator," *American Statistician*, 33, 149–158.
- Lavori, P.W., Dawson, R., and Shera, D. (1995), "A Multiple Imputation Strategy for Clinical Trials with Truncation of Patient Data," *Statistics in Medicine*, 14, 1913–1925.
- Li, K.H. (1988), "Imputation Using Markov Chains," *Journal of Statistical Computation and Simulation*, 30, 57–79.
- Li, K.H., Raghunathan, T.E., and Rubin, D.B. (1991), "Large-Sample Significance Levels from Multiply Imputed Data Using Moment-Based Statistics and an F Reference Distribution," *Journal of the American Statistical Association*, 86, 1065–1073.
- Little, R.J.A. and Rubin, D.B. (1987), *Statistical Analysis with Missing Data*, New York: John Wiley & Sons, Inc.
- Liu, C. (1993), "Bartlett's Decomposition of the Posterior Distribution of the Covariance for Normal Monotone Ignorable Missing Data," *Journal of Multivariate Analysis*, 46, 198–206.
- McLachlan, G.J. and Krishnan, T. (1997), *The EM Algorithm and Extensions*, New York: John Wiley & Sons, Inc.

- Rosenbaum, P.R. and Rubin, D.B. (1983), “The Central Role of the Propensity Score in Observational Studies for Causal Effects,” *Biometrika*, 70, 41–55.
- Rubin, D.B. (1976), “Inference and Missing Data,” *Biometrika*, 63, 581–592.
- Rubin, D.B. (1987), *Multiple Imputation for Nonresponse in Surveys*, New York: John Wiley & Sons, Inc.
- Rubin, D.B. (1996), “Multiple Imputation After 18+ Years,” *Journal of the American Statistical Association*, 91, 473–489.
- SAS Institute Inc. (1999), *SAS/ETS User’s Guide, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS/GRAPH Software: Reference, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS Language Reference: Concepts, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS Procedures Guide, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS/STAT User’s Guide, Version 8*, Cary, NC: SAS Institute Inc.
- Schafer, J.L. (1997), *Analysis of Incomplete Multivariate Data*, New York: Chapman and Hall.
- Tanner, M.A. and Wong, W.H. (1987), “The Calculation of Posterior Distributions by Data Augmentation,” *Journal of the American Statistical Association*, 82, 528–540.

# Chapter 10

## The MIANALYZE Procedure

### Chapter Table of Contents

---

|                                                                                          |     |
|------------------------------------------------------------------------------------------|-----|
| <b>OVERVIEW</b>                                                                          | 203 |
| <b>GETTING STARTED</b>                                                                   | 203 |
| <b>SYNTAX</b>                                                                            | 206 |
| PROC MIANALYZE Statement                                                                 | 207 |
| BY Statement                                                                             | 208 |
| VAR Statement                                                                            | 209 |
| <b>DETAILS</b>                                                                           | 209 |
| Input Data Sets                                                                          | 209 |
| Combining Inferences from Imputed Data Sets                                              | 211 |
| Multiple Imputation Efficiency                                                           | 212 |
| Multivariate Inferences                                                                  | 213 |
| Examples of the Complete-Data Inferences                                                 | 214 |
| ODS Table Names                                                                          | 216 |
| <b>EXAMPLES</b>                                                                          | 217 |
| Example 10.1 Reading Means and Covariance Matrices from a DATA= COV<br>Data Set          | 217 |
| Example 10.2 Reading Regression Results from a DATA= EST Data Set                        | 221 |
| Example 10.3 Reading Mixed Model Results from PARMS= and COVB=<br>Data Sets              | 223 |
| Example 10.4 Reading Generalized Linear Model Results from PARMS=<br>and COVB= Data Sets | 224 |
| Example 10.5 Reading GLM Results from PARMS= and XPXI= Data Sets                         | 226 |
| Example 10.6 Combining Correlation Coefficients                                          | 227 |
| Example 10.7 Combining Ratios of Variable Means                                          | 230 |
| <b>REFERENCES</b>                                                                        | 233 |





# Chapter 10

## The MIANALYZE Procedure

---

### Overview

The experimental MIANALYZE procedure combines the results of the analyses of imputations and generates valid statistical inferences. Multiple imputation provides a useful strategy for analyzing data sets with missing values. Instead of filling in a single value for each missing value, Rubin's (1976; 1987) multiple imputation strategy replaces each missing value with a set of plausible values that represent the uncertainty about the right value to impute. You can implement the strategy with two SAS procedures: PROC MI, which generates imputed data sets, and PROC MIANALYZE, which combines the results of analyses carried out on the data sets. These two procedures are available in experimental form in Release 8.2 of the SAS System.

These analyses of imputations are obtained by using standard SAS procedures (such as PROC REG) for complete data. No matter which complete-data analysis is used, the process of combining results from different imputed data sets is essentially the same. This results in valid statistical inferences that properly reflect the uncertainty due to missing values.

The MIANALYZE procedure reads the parameter estimates and associated covariance matrix that are computed by the standard statistical procedure for each imputed data set. The MIANALYZE procedure then derives valid univariate and multivariate inferences for these parameters.

For some parameters of interest, it is not straightforward to compute estimates and associated covariance matrices with standard statistical SAS procedures. Examples include correlation coefficients between two variables and ratios of variable means. Special cases such as these are described in the "Examples of the Complete-Data Inferences" section on page 214.

---

### Getting Started

The Fitness data set has been altered to contain an arbitrary missing pattern:

```
*----- Data on Physical Fitness -----*
| These measurements were made on men involved in a physical |
| fitness course at N.C. State University.                   |
| Only selected variables of                                  |
| Oxygen (oxygen intake, ml per kg body weight per minute), |
| Runtime (time to run 1.5 miles in minutes), and            |
| RunPulse (heart rate while running) are used.              |
| Certain values were changed to missing for the analysis.   |
*-----*
```

```

data FitMiss;
  input Oxygen RunTime RunPulse @@;
  datalines;
44.609 11.37 178 45.313 10.07 185
54.297 8.65 156 59.571 . .
49.874 9.22 . 44.811 11.63 176
. 11.95 176 49.091 10.85 .
39.442 13.08 174 60.055 8.63 170
50.541 . . 37.388 14.03 186
44.754 11.12 176 47.273 . .
51.855 10.33 166 49.156 8.95 180
40.836 10.95 168 46.672 10.00 .
. 10.25 . 50.388 10.08 168
39.407 12.63 174 46.080 11.17 156
45.441 9.63 164 . 8.92 146
45.118 11.08 . 39.203 12.88 168
45.790 10.47 186 50.545 9.93 148
48.673 9.40 186 47.920 11.50 170
47.467 10.50 170
;

```

Assume that the data are multivariate normally distributed and that the missing data are missing at random (see the “Statistical Assumptions for Multiple Imputation” section in “The MI Procedure” chapter for a description of these assumptions). The following statements use the MI procedure to impute missing values for the FitMiss data set.

```

proc mi data=FitMiss noprint out=outmi seed=37851;
  var Oxygen RunTime RunPulse;
run;

```

The MI procedure creates imputed data sets, which are stored in the outmi data set. A variable named `_Imputation_` indicates the imputation numbers. Based on  $m$  imputations,  $m$  different sets of the point and variance estimates for a parameter can be computed. In this example,  $m = 5$  is the default.

The following statements generate regression coefficients for each of the five imputed data sets:

```

proc reg data=outmi outest=outreg covout noprint;
  model Oxygen= RunTime RunPulse;
  by _Imputation_;
run;

proc print data=outreg(obs=8);
  var _Imputation_ _Type_ _Name_
      Intercept RunTime RunPulse;
  title 'Parameter Estimates from Imputed Data Sets';
run;

```

| Parameter Estimates from Imputed Data Sets |              |        |           |           |          |          |
|--------------------------------------------|--------------|--------|-----------|-----------|----------|----------|
| Obs                                        | _Imputation_ | _TYPE_ | _NAME_    | Intercept | RunTime  | RunPulse |
| 1                                          | 1            | PARMS  |           | 97.2874   | -2.98892 | -0.10684 |
| 2                                          | 1            | COV    | Intercept | 55.7516   | -0.73348 | -0.27870 |
| 3                                          | 1            | COV    | RunTime   | -0.7335   | 0.15167  | -0.00509 |
| 4                                          | 1            | COV    | RunPulse  | -0.2787   | -0.00509 | 0.00194  |
| 5                                          | 2            | PARMS  |           | 90.9324   | -2.93338 | -0.07391 |
| 6                                          | 2            | COV    | Intercept | 37.5576   | -0.25970 | -0.20442 |
| 7                                          | 2            | COV    | RunTime   | -0.2597   | 0.13978  | -0.00722 |
| 8                                          | 2            | COV    | RunPulse  | -0.2044   | -0.00722 | 0.00166  |

Figure 10.1. Parameter Estimates

The following statements combine the five sets of regression coefficients:

```
proc mianalyze data=outreg;
  var Intercept RunTime RunPulse;
run;
```

| The MIANALYZE Procedure                  |                               |                              |            |        |
|------------------------------------------|-------------------------------|------------------------------|------------|--------|
| Model Information                        |                               |                              |            |        |
| Data Set                                 |                               | WORK.OUTREG                  |            |        |
| Number of Imputations                    |                               | 5                            |            |        |
| Multiple Imputation Variance Information |                               |                              |            |        |
| Parameter                                | -----Variance-----            |                              |            | DF     |
|                                          | Between                       | Within                       | Total      |        |
| Intercept                                | 74.179857                     | 57.287519                    | 146.303348 | 10.805 |
| RunTime                                  | 0.034202                      | 0.142151                     | 0.183193   | 79.694 |
| RunPulse                                 | 0.001533                      | 0.002304                     | 0.004144   | 20.292 |
| Multiple Imputation Variance Information |                               |                              |            |        |
| Parameter                                | Relative Increase in Variance | Fraction Missing Information |            |        |
| Intercept                                | 1.553843                      | 0.665161                     |            |        |
| RunTime                                  | 0.288719                      | 0.242803                     |            |        |
| RunPulse                                 | 0.798522                      | 0.491731                     |            |        |

Figure 10.2. Model Information and Variance Information Tables

The “Model Information” table lists the input data set(s) and the number of imputations. The “Multiple Imputation Variance Information” table displays the between-imputation, within-imputation, and total variances for combining complete-data inferences. It also displays the degrees of freedom for the total variance, the relative increase in variance due to missing values, and the fraction of missing information for each parameter estimate.

| The MIANALYZE Procedure                 |           |                  |                       |          |        |
|-----------------------------------------|-----------|------------------|-----------------------|----------|--------|
| Multiple Imputation Parameter Estimates |           |                  |                       |          |        |
| Parameter                               | Estimate  | Std Error        | 95% Confidence Limits |          | DF     |
| Intercept                               | 92.156840 | 12.095592        | 65.47596              | 118.8377 | 10.805 |
| RunTime                                 | -2.955317 | 0.428011         | -3.80714              | -2.1035  | 79.694 |
| RunPulse                                | -0.079851 | 0.064376         | -0.21401              | 0.0543   | 20.292 |
| Multiple Imputation Parameter Estimates |           |                  |                       |          |        |
| Parameter                               | Minimum   |                  | Maximum               |          |        |
| Intercept                               | 77.939497 |                  | 99.920480             |          |        |
| RunTime                                 | -3.159663 |                  | -2.660085             |          |        |
| RunPulse                                | -0.112277 |                  | -0.015111             |          |        |
| Multiple Imputation Parameter Estimates |           |                  |                       |          |        |
| Parameter                               | Theta0    | t for H0:        |                       | Pr >  t  |        |
|                                         |           | Parameter=Theta0 |                       |          |        |
| Intercept                               | 0         | 7.62             | <.0001                |          |        |
| RunTime                                 | 0         | -6.90            | <.0001                |          |        |
| RunPulse                                | 0         | -1.24            | 0.2290                |          |        |

**Figure 10.3.** Multiple Imputation Parameter Estimates

The “Multiple Imputation Parameter Estimates” table displays a combined estimate and standard error for each regression coefficient (parameter). Inferences are based on  $t$  distributions. The table displays a 95% confidence interval and a  $t$ -test with the associated  $p$ -value for the hypothesis that the parameter is equal to the value specified with the THETA0= option (in this case, zero by default). The minimum and maximum parameter estimates from the imputed data sets are also displayed.

## Syntax

The following statements are available in PROC MIANALYZE.

**PROC MIANALYZE** < options > ;

**BY** variables ;

**VAR** variables ;

The BY statement specifies groups in which separate analyses are performed.

The VAR statement lists the numeric variables to be analyzed. The statement is required.

The rest of this section gives detailed syntax information for each of these statements. The PROC MIANALYZE statement and the VAR statement are required for the MIANALYZE procedure.

## PROC MIANALYZE Statement

**PROC MIANALYZE** < options > ;

The following table summarizes the options in the PROC MIANALYZE statement.

**Table 10.1.** Summary of PROC MIANALYZE Options

| Tasks                                         | Options       |
|-----------------------------------------------|---------------|
| <b>Specify input data sets</b>                |               |
| a COV, CORR, or EST type data set             | DATA=         |
| parameter estimates and covariance matrices   | PARMS=, COVB= |
| parameter estimates and $(X'X)^{-1}$ matrices | PARMS=, XPXI= |
| <b>Specify statistical analysis</b>           |               |
| parameters under the null hypothesis          | THETA0=       |
| level for the confidence interval             | ALPHA=        |
| complete-data degrees of freedom              | EDF=          |
| multivariate inferences                       | MULT          |

The following are explanations of the options that can be used in the PROC MIANALYZE statement (in alphabetical order):

**ALPHA=***p*

specifies that confidence limits are to be constructed for the parameter estimates with confidence level  $100(1 - p)\%$ , where  $0 < p < 1$ . The default is  $p=0.05$ .

**COVB=***SAS-data-set*

names an input SAS data set that contains covariance matrices of the parameter estimates from imputed data sets. If you provide a COVB= data set, you must also provide a PARMS= data set.

**DATA=***SAS-data-set*

names a specially structured input SAS data set that contains estimates from imputed data sets. This data set must have a TYPE of EST, COV, or CORR:

- if TYPE=EST, the data set contains the parameter estimates and associated covariance matrices.
- if TYPE=COV, the data set contains the sample means, sample sizes, and covariance matrices. Each covariance matrix for variables is divided by the sample size  $n$  to create the covariance matrix for parameter estimates.
- if TYPE=CORR, the data set contains the sample means, sample sizes, standard errors, and correlation matrices. The covariance matrices are computed from the correlation matrices and associated standard errors. Each covariance matrix for variables is divided by the sample size  $n$  to create the covariance matrix for parameter estimates.

If you do not specify an input data set with the DATA=, COVB=, or XPXI= option, then the most recently created SAS data set is used as an input DATA= data set.

**EDF=number**

specifies the complete-data degrees of freedom for the parameter estimates. This is used to compute an adjusted degrees of freedom for each parameter estimate. By default, EDF= $\infty$  and the degrees of freedom for each parameter estimate is not adjusted.

**MULT****MULTIVARIATE**

requests multivariate inference for the variables.

**PARMS=SAS-data-set**

names an input SAS data set that contains parameter estimates computed from imputed data sets. If you provide a PARMS= data set, you must also provide a COVB= or XPXI= data set.

**THETA0=numbers****MU0=numbers**

specifies the parameter values  $\theta_0$  under the null hypothesis  $\theta = \theta_0$  in the  $t$  tests for location for the variables. If only one number  $\theta_0$  is specified, that number is used for all variables. If more than one number is specified, the specified numbers correspond to variables in the VAR statement in the order in which they appear in the VAR statement.

**XPXI=SAS-data-set**

names an input SAS data set that contains the  $(X'X)^{-1}$  matrices associated with the parameter estimates computed from imputed data sets. If you provide an XPXI= data set, you must also provide a PARMS= data set. In this case, PROC MIANALYZE reads the standard errors of the estimates from the PARMS= data. The standard errors and  $(X'X)^{-1}$  matrices are used to derive the covariance matrices.

---

## BY Statement

**BY variables ;**

You can specify a BY statement with PROC MIANALYZE to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the MI procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables using the DATASETS procedure.

For more information on the BY statement, refer to the discussion in *SAS Language Reference: Concepts, Version 8*. For more information on the DATASETS procedure, refer to the discussion in the *SAS Procedures Guide, Version 8*.

---

## VAR Statement

**VAR** *variables* ;

The VAR statement lists the variables to be analyzed. The statement is required, and the variables must be numeric.

---

## Details

---

### Input Data Sets

You can specify input data sets using one of the following option combinations:

- DATA=, which provides both parameter estimates and the associated covariance matrix in a single input data set.
- PARMS= and COVB=, which provide parameter estimates and the associated covariance matrix in separate data sets, respectively.
- PARMS= and XPXI=, which provide parameter estimates and the associated standard errors in a PARMS= data set and the associated  $(X'X)^{-1}$  matrix in an XPXI= data set.

The appropriate combination depends on the SAS procedure you used to create the parameter estimates and associated covariance matrix. For instance, if you used PROC REG to create an OUTEST= data set containing the parameter estimates and covariance matrix, you would use the DATA= option to read the OUTEST= data set. Each input data set contains the variable `_Imputation_` to identify the imputation by number.

If you do not specify an input data set with the DATA=, COVB=, or XPXI= option, then the most recently created SAS data set is used as an input DATA= data set.

**DATA= data set**

The input DATA= data set is a specially structured SAS data set created by statistical procedures available with SAS software. The data set must have a TYPE of EST, COV, or CORR.

With TYPE=EST, the MIANALYZE procedure reads parameter estimates from observations with \_TYPE\_='PARM' or \_TYPE\_='PARMS', and covariance matrices for parameter estimates from observations with \_TYPE\_='COV' or \_TYPE\_='COVB'.

With TYPE=COV, the procedure reads sample means from observations with \_TYPE\_='MEAN', sample size  $n$  from observations with \_TYPE\_='N', and covariance matrices for variables from observations with \_TYPE\_='COV'.

With TYPE=CORR, the procedure reads sample means from observations with \_TYPE\_='MEAN', sample size  $n$  from observations with \_TYPE\_='N', correlation matrices for variables from observations with \_TYPE\_='CORR', and standard errors for variables from observations with \_TYPE\_='STD'. The standard errors and correlation matrix are used to generate a covariance matrix for the variables.

Note that with TYPE=COV or CORR, each covariance matrix for the variables is divided by  $n$  to create the covariance matrix for the sample means.

**PARMS= and COVB= data sets**

The input PARMS= data set contains parameter estimates, and the input COVB= data set contains associated covariance matrices computed from imputed data sets. Such data sets are typically created with an ODS OUTPUT statement using procedures such as PROC MIXED and PROC GENMOD.

The MIANALYZE procedure uses a PARMS= data set to read parameter names from the variable *Parameter* or *Effect*, and it reads parameter estimates from the variable *Estimate*.

The MIANALYZE procedure uses a COVB= data set to read parameter names from the variable *Parameter*, *Effect*, or *RowName*, and it reads covariance matrices from the subsequent variables *Col1*, *Col2*, ... or *Prm1*, *Prm2*, ... in the data set.

**PARMS= and XPXI= data sets**

The input PARMS= data set contains parameter estimates, and the input XPXI= data set contains associated  $(X'X)^{-1}$  matrices computed from imputed data sets. Such data sets are typically created with an ODS OUTPUT statement using a procedure such as PROC GLM.

The MIANALYZE procedure uses a PARMS= data set to read parameter names from the variable *Parameter*; it reads parameter estimates from the variable *Estimate*, and it reads standard errors for parameter estimates from the variable *StdErr*.

The MIANALYZE procedure uses an XPXI= data set to read parameter names from the variable *Parameter*, and it reads  $(X'X)^{-1}$  matrices from the subsequent variables in the data set.



## Combining Inferences from Imputed Data Sets

With  $m$  imputations,  $m$  different sets of the point and variance estimates for a parameter  $Q$  can be computed. Let  $\hat{Q}_i$  and  $\hat{U}_i$  be the point and variance estimates from the  $i$ th imputed data set,  $i=1, 2, \dots, m$ . Then the combined point estimate for  $Q$  from multiple imputation is the average of the  $m$  complete-data estimates:

$$\bar{Q} = \frac{1}{m} \sum_{i=1}^m \hat{Q}_i$$

Let  $\bar{U}$  be the within-imputation variance, which is the average of the  $m$  complete-data estimates:

$$\bar{U} = \frac{1}{m} \sum_{i=1}^m \hat{U}_i$$

and  $B$  be the between-imputation variance

$$B = \frac{1}{m-1} \sum_{i=1}^m (\hat{Q}_i - \bar{Q})^2$$

Then the variance estimate associated with  $\bar{Q}$  is the total variance (Rubin 1987)

$$T = \bar{U} + \left(1 + \frac{1}{m}\right)B$$

The statistic  $(Q - \bar{Q})T^{-(1/2)}$  is approximately distributed as  $t$  with  $v_m$  degrees of freedom (Rubin 1987), where

$$v_m = (m-1) \left[1 + \frac{\bar{U}}{(1 + m^{-1})B}\right]^2$$

When the complete-data degrees of freedom  $v_0$  is small, and there is only a modest proportion of missing data, the computed degrees of freedom,  $v_m$ , can be much larger than  $v_0$ , which is inappropriate. Barnard and Rubin (1999) recommend the use of an adjusted degrees of freedom

$$v_m^* = \left[ \frac{1}{v_m} + \frac{1}{\hat{v}_{obs}} \right]^{-1}$$

where  $\hat{v}_{obs} = (1 - \gamma) v_0 (v_0 + 1) / (v_0 + 3)$  and  $\gamma = (1 + m^{-1})B/T$ .

If you specify the complete-data degrees of freedom  $v_0$  with the EDF= option, the MIANALYZE procedure uses the adjusted degrees of freedom,  $v_m^*$ , for inference. Otherwise, the degrees of freedom  $v_m$  is used.

The degrees of freedom  $v_m$  depends on  $m$  and the ratio

$$r = \frac{(1 + m^{-1})B}{\bar{U}}$$

The ratio  $r$  is called the relative increase in variance due to nonresponse (Rubin 1987). When there is no missing information about  $Q$ , the values of  $r$  and  $B$  are both zero. With a large value of  $m$  or a small value of  $r$ , the degrees of freedom  $v$  will be large and the distribution of  $(Q - \bar{Q})T^{-(1/2)}$  will be approximately normal.

Another useful statistic is the fraction of missing information about  $Q$ :

$$\hat{\lambda} = \frac{r + 2/(v + 3)}{r + 1}$$

Both statistics  $r$  and  $\lambda$  are helpful diagnostics for assessing how the missing data contribute to the uncertainty about  $Q$ .

---

## Multiple Imputation Efficiency

The relative efficiency (RE) of using the finite  $m$  imputation estimator, rather than using an infinite number for the fully efficient imputation, in units of variance, is approximately a function of  $m$  and  $\lambda$  (Rubin 1987, p. 114).

$$RE = (1 + \frac{\lambda}{m})^{-1}$$

The following table shows relative efficiencies with different values of  $m$  and  $\lambda$ . For cases with little missing information, only a small number of imputations are necessary.

**Table 10.2.** Relative Efficiency

| $m$ | $\lambda$ |        |        |        |        |
|-----|-----------|--------|--------|--------|--------|
|     | 10%       | 20%    | 30%    | 50%    | 70%    |
| 3   | 0.9677    | 0.9375 | 0.9091 | 0.8571 | 0.8108 |
| 5   | 0.9804    | 0.9615 | 0.9434 | 0.9091 | 0.8772 |
| 10  | 0.9901    | 0.9804 | 0.9709 | 0.9524 | 0.9346 |
| 20  | 0.9950    | 0.9901 | 0.9852 | 0.9756 | 0.9662 |

## Multivariate Inferences

Multivariate inference based on Wald tests can be done with  $m$  imputed data sets. The approach is a generalization of the approach taken in the univariate case (Rubin 1987, p. 137; Schafer 1997, p. 113). Suppose that  $\hat{\mathbf{Q}}_i$  and  $\hat{\mathbf{U}}_i$  are the point and covariance matrix estimates for a vector-valued parameter  $\mathbf{Q}$  (such as a multivariate mean) from the  $i$ th imputed data set,  $i=1, 2, \dots, m$ . Then the combined point estimate for  $\mathbf{Q}$  from the multiple imputation is the average of the  $m$  complete-data estimates:

$$\bar{\mathbf{Q}} = \frac{1}{m} \sum_{i=1}^m \hat{\mathbf{Q}}_i$$

Suppose that  $\bar{\mathbf{U}}$  is the within-imputation covariance matrix, which is the average of the  $m$  complete-data estimates

$$\bar{\mathbf{U}} = \frac{1}{m} \sum_{i=1}^m \hat{\mathbf{U}}_i$$

and suppose that  $\mathbf{B}$  is the between-imputation covariance matrix

$$\mathbf{B} = \frac{1}{m-1} \sum_{i=1}^m (\hat{\mathbf{Q}}_i - \bar{\mathbf{Q}})(\hat{\mathbf{Q}}_i - \bar{\mathbf{Q}})'$$

Then the covariance matrix associated with  $\bar{\mathbf{Q}}$  is the total covariance matrix

$$\mathbf{T}_0 = \bar{\mathbf{U}} + (1 + \frac{1}{m})\mathbf{B}$$

The natural multivariate extension of the  $t$  statistic used in the univariate case is the  $F$  statistic

$$F_0 = (\mathbf{Q} - \bar{\mathbf{Q}})' \mathbf{T}_0^{-1} (\mathbf{Q} - \bar{\mathbf{Q}})$$

with degrees of freedom  $p$  and

$$v = (m-1)(1 + 1/r)^2$$

where

$$r = (1 + \frac{1}{m}) \text{trace}(\mathbf{B}\bar{\mathbf{U}}^{-1})/p$$

is an average relative increase in variance due to nonresponse (Rubin 1987, p. 137; Schafer 1997, p. 114).

However, the reference distribution of the statistic  $F_0$  is not easily derived. Especially for small  $m$ , the between-imputation covariance matrix  $\mathbf{B}$  is unstable and does not have full rank for  $m \leq p$  (Schafer 1997, p. 113).

One solution is to make an additional assumption that the population between-imputation and within-imputation covariance matrices are proportional to each other (Schafer 1997, p. 113). This assumption implies that the fractions of missing information for all components of  $\mathbf{Q}$  are equal. Under this assumption, a more stable estimate of the total covariance matrix is

$$\mathbf{T} = (1 + r)\overline{\mathbf{U}}$$

With the total covariance matrix  $\mathbf{T}$ , the  $F$  statistic (Rubin 1987, p. 137)

$$F = (\mathbf{Q} - \overline{\mathbf{Q}})' \mathbf{T}^{-1} (\mathbf{Q} - \overline{\mathbf{Q}}) / p$$

has an  $F$  distribution with degrees of freedom  $p$  and  $v_1$ , where

$$v_1 = \frac{1}{2}(p + 1)(m - 1)\left(1 + \frac{1}{r}\right)^2$$

For  $t = p(m - 1) \leq 4$ , PROC MIANALYZE uses the degrees of freedom  $v_1$  in the analysis. For  $t = p(m - 1) > 4$ , PROC MIANALYZE uses  $v_2$ , a better approximation of the degrees of freedom given by Li, Raghunathan, and Rubin (1991).

$$v_2 = 4 + (t - 4)\left(1 + \frac{1}{r}\left(1 - \frac{2}{t}\right)\right)^2$$

---

## Examples of the Complete-Data Inferences

For a given parameter of interest, it is not always possible to compute the estimate and associated covariance matrix directly from a SAS procedure. This section gives examples of parameters with their estimates and associated covariance matrices, which provide the input to the MIANALYZE procedure. Some are straightforward, and others require special techniques.

### Means

For a population mean vector  $\boldsymbol{\mu}$ , the usual estimate is the sample mean vector

$$\bar{\mathbf{y}} = \frac{1}{n} \sum \mathbf{y}_i$$

A variance estimate for  $\bar{\mathbf{y}}$  is  $\frac{1}{n}\mathbf{S}$ , where  $\mathbf{S}$  is the sample covariance matrix

$$\mathbf{S} = \frac{1}{n - 1} \sum (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})'$$

These statistics can be computed from a procedure such as CORR. This approach is illustrated in Example 10.1.

### Regression Coefficients

Many SAS procedures are available for regression analysis. Among them, REG provides the most general analysis capabilities, and others like LOGISTIC and MIXED provide more specialized analyses.

Some regression procedures, such as REG and LOGISTIC, create an EST type data set that contains both the parameter estimates for the regression coefficients and their associated covariance matrix. You can read an EST type data set in the MIANALYZE procedure with the DATA= option. This approach is illustrated in Example 10.2.

Other procedures, such as GLM, MIXED, and GENMOD, do not generate EST type data sets for regression coefficients. For MIXED and GENMOD, you can use ODS OUTPUT statement to save parameter estimates in a data set and the associated covariance matrix in a separate data set. These data sets are then read in the MIANALYZE procedure with the PARMS= and COVB= options, respectively. This approach is illustrated in Example 10.3 for PROC MIXED and in Example 10.4 for PROC GENMOD.

PROC GLM does not display tables for covariance matrices. However, you can use the ODS OUTPUT statement to save parameter estimates and associated standard errors in a data set and the associated  $(X'X)^{-1}$  matrix in a separate data set. These data sets are then read in the MIANALYZE procedure with the PARMS= and XPXI= options, respectively. This approach is illustrated in Example 10.5.

### Correlation Coefficients

For the population correlation coefficient  $\rho$ , a point estimate is the sample correlation coefficient  $r$ . However, for nonzero  $\rho$ , the distribution of  $r$  is skewed.

The distribution of  $r$  can be normalized through Fisher's  $Z$  transformation

$$z(r) = \frac{1}{2} \log \left( \frac{1+r}{1-r} \right)$$

$z(r)$  is approximately normally distributed with mean  $z(\rho)$  and variance  $1/(n-3)$ .

With a point estimate  $\hat{z}$  and an approximate 95% confidence interval  $(z_1, z_2)$  for  $z(\rho)$ , a point estimate  $\hat{r}$  and a 95% confidence interval  $(r_1, r_2)$  for  $\rho$  can be obtained by applying the inverse transformation

$$r = \frac{e^{2z} - 1}{e^{2z} + 1}$$

to  $z = \hat{z}, z_1$ , and  $z_2$ .

This approach is illustrated in Example 10.3.

**Ratios of Variable Means**

For the ratio  $\mu_1/\mu_2$  of means for variables  $Y_1$  and  $Y_2$ , the point estimate is  $\bar{y}_1/\bar{y}_2$ , the ratio of the sample means. The Taylor expansion and delta method can be applied to the function  $y_1/y_2$  to obtain the variance estimate (Schafer 1997, p. 196)

$$\frac{1}{n} \left[ \left( \frac{\bar{y}_1}{\bar{y}_2} \right)^2 s_{22} - 2 \left( \frac{\bar{y}_1}{\bar{y}_2} \right) \left( \frac{1}{\bar{y}_2} \right) s_{12} + \left( \frac{1}{\bar{y}_2} \right)^2 s_{11} \right]$$

where  $s_{11}$  and  $s_{22}$  are the sample variances of  $Y_1$  and  $Y_2$ , respectively, and  $s_{12}$  is the sample covariance between  $Y_1$  and  $Y_2$ .

A ratio of sample means will be approximately unbiased and normally distributed if the coefficient of variation of the denominator (the standard error for the mean divided by the estimated mean) is 10% or less (Cochran 1977, p. 166; Schafer 1997, p. 196). This approach is illustrated in Example 10.7.

---

**ODS Table Names**

PROC MIANALYZE assigns a name to each table it creates. You must use these names to reference tables when using the Output Delivery System (ODS). These names are listed in the following table. For more information on ODS, refer to the chapter “Using the Output Delivery System” in the *SAS/STAT User’s Guide, Version 8*.

**Table 10.3.** ODS Tables Produced in PROC MIANALYZE

| ODS Table Name | Description                          | Option |
|----------------|--------------------------------------|--------|
| ModelInfo      | Model information                    |        |
| VarianceInfo   | Variance information                 |        |
| ParmEst        | Parameter estimates                  |        |
| BetweenCov     | Between-imputation covariance matrix | MULT   |
| WithinCov      | Within-imputation covariance matrix  | MULT   |
| TotalCov       | Total covariance matrix              | MULT   |
| MultiInf       | Multivariate inference               | MULT   |

## Examples

The following statements generate five imputed data sets to be used in this section. The data set FitMiss was created in the section “Getting Started” on page 203. See “The MI Procedure” chapter for details concerning the MI procedure.

```
proc mi data=FitMiss seed=37851 noprint out=outmi;
  var Oxygen RunTime RunPulse;
run;
```

### Example 10.1. Reading Means and Covariance Matrices from a DATA= COV Data Set

This example creates a COV type data set that contains sample means and covariance matrices computed from imputed data sets. These estimates are then combined to generate valid statistical inferences about the population means.

The following statements use the CORR procedure to generate sample means and a covariance matrix for the variables in each imputed data set.

```
proc corr data=outmi cov out=outcov(type=cov) nocorr noprint;
  var Oxygen RunTime RunPulse;
  by _Imputation_;
run;

proc print data=outcov(obs=12);
  title 'CORR Means and Covariance Matrices'
        ' (First Two Imputations)';
run;
```

Output 10.1.1. COV Data Set

| CORR Means and Covariance Matrices (First Two Imputations) |              |        |          |          |         |          |
|------------------------------------------------------------|--------------|--------|----------|----------|---------|----------|
| Obs                                                        | _Imputation_ | _TYPE_ | _NAME_   | Oxygen   | RunTime | RunPulse |
| 1                                                          | 1            | COV    | Oxygen   | 28.2139  | -5.9997 | -29.700  |
| 2                                                          | 1            | COV    | RunTime  | -5.9997  | 1.8353  | 4.812    |
| 3                                                          | 1            | COV    | RunPulse | -29.7002 | 4.8121  | 143.368  |
| 4                                                          | 1            | MEAN   |          | 47.3458  | 10.5859 | 171.301  |
| 5                                                          | 1            | STD    |          | 5.3117   | 1.3547  | 11.974   |
| 6                                                          | 1            | N      |          | 31.0000  | 31.0000 | 31.000   |
| 7                                                          | 2            | COV    | Oxygen   | 28.7419  | -6.6895 | -39.006  |
| 8                                                          | 2            | COV    | RunTime  | -6.6895  | 2.0553  | 8.938    |
| 9                                                          | 2            | COV    | RunPulse | -39.0060 | 8.9385  | 172.998  |
| 10                                                         | 2            | MEAN   |          | 47.3316  | 10.6004 | 169.204  |
| 11                                                         | 2            | STD    |          | 5.3612   | 1.4336  | 13.153   |
| 12                                                         | 2            | N      |          | 31.0000  | 31.0000 | 31.000   |

Note that the covariance matrices in the data set `OUTCOV` are estimated covariance matrices of variables,  $V(\mathbf{y})$ . The estimated covariance matrix of the sample means is  $V(\bar{\mathbf{y}}) = V(\mathbf{y})/n$ , where  $n$  is the sample size, and is not the same as an estimated covariance matrix for variables.

The following statements combine the results for the imputed data sets, and derive both univariate and multivariate inferences about the means. The `EDF=` option is specified to request that the adjusted degrees of freedom be used in the analysis. For sample means based on 31 observations, the complete-data error degrees of freedom is 30.

```
proc mianalyze data=outcov edf=30 mult;
    var Oxygen RunTime RunPulse;
run;
```

**Output 10.1.2.** Multiple Imputation Variance Information

| The MIANALYZE Procedure                  |                                     |                                    |          |        |
|------------------------------------------|-------------------------------------|------------------------------------|----------|--------|
| Multiple Imputation Variance Information |                                     |                                    |          |        |
| Parameter                                | -----Variance-----                  |                                    |          | DF     |
|                                          | Between                             | Within                             | Total    |        |
| Oxygen                                   | 0.007552                            | 0.969527                           | 0.978590 | 27.904 |
| RunTime                                  | 0.001577                            | 0.070505                           | 0.072397 | 27.317 |
| RunPulse                                 | 1.363982                            | 4.469865                           | 6.106642 | 15.052 |
| Multiple Imputation Variance Information |                                     |                                    |          |        |
| Parameter                                | Relative<br>Increase<br>in Variance | Fraction<br>Missing<br>Information |          |        |
| Oxygen                                   | 0.009348                            | 0.009304                           |          |        |
| RunTime                                  | 0.026834                            | 0.026466                           |          |        |
| RunPulse                                 | 0.366181                            | 0.292981                           |          |        |

The “Multiple Imputation Variance Information” table displays the between-imputation variance, within-imputation variance, and total variance for each univariate inference. It also displays the degrees of freedom for the total variance. The relative increase in variance due to missing values and the fraction of missing information for each variable are also displayed. A detailed description of these statistics is provided in the “Combining Inferences from Imputed Data Sets” section on page 211 and the “Multiple Imputation Efficiency” section on page 212.



**Output 10.1.3.** Multiple Imputation Parameter Estimates

| The MIANALYZE Procedure                 |            |                  |                       |          |        |
|-----------------------------------------|------------|------------------|-----------------------|----------|--------|
| Multiple Imputation Parameter Estimates |            |                  |                       |          |        |
| Parameter                               | Estimate   | Std Error        | 95% Confidence Limits |          | DF     |
| Oxygen                                  | 47.360514  | 0.989237         | 45.3338               | 49.3872  | 27.904 |
| RunTime                                 | 10.557687  | 0.269067         | 10.0059               | 11.1095  | 27.317 |
| RunPulse                                | 169.970152 | 2.471162         | 164.7046              | 175.2357 | 15.052 |
| Multiple Imputation Parameter Estimates |            |                  |                       |          |        |
| Parameter                               | Minimum    |                  | Maximum               |          |        |
| Oxygen                                  | 47.273244  |                  | 47.506505             |          |        |
| RunTime                                 | 10.505186  |                  | 10.600445             |          |        |
| RunPulse                                | 169.053679 |                  | 171.301061            |          |        |
| Multiple Imputation Parameter Estimates |            |                  |                       |          |        |
| Parameter                               | Theta0     | t for H0:        |                       |          |        |
|                                         |            | Parameter=Theta0 | Pr >  t               |          |        |
| Oxygen                                  | 0          | 47.88            | <.0001                |          |        |
| RunTime                                 | 0          | 39.24            | <.0001                |          |        |
| RunPulse                                | 0          | 68.78            | <.0001                |          |        |

The “Multiple Imputation Parameter Estimates” table displays the estimated mean and corresponding standard error for each variable. The table also displays a 95% confidence interval for the mean and a  $t$  statistic with the associated  $p$ -value for testing the hypothesis that the mean is equal to the value specified. You can use the THETA0= option to specify the value for the null hypothesis, which is zero by default. The table also displays the minimum and maximum parameter estimates from the imputed data sets.

With the MULT option, the procedure displays the between-imputation covariance matrix, within-imputation covariance matrix, and total covariance matrix for multivariate inference.

**Output 10.1.4.** Within-Imputation Covariance Matrices

| The MIANALYZE Procedure              |              |              |              |
|--------------------------------------|--------------|--------------|--------------|
| Within-Imputation Covariance Matrix  |              |              |              |
|                                      | Oxygen       | RunTime      | RunPulse     |
| Oxygen                               | 0.969526960  | -0.222851446 | -0.997345212 |
| RunTime                              | -0.222851446 | 0.070505098  | 0.222928579  |
| RunPulse                             | -0.997345212 | 0.222928579  | 4.469864543  |
| Between-Imputation Covariance Matrix |              |              |              |
|                                      | Oxygen       | RunTime      | RunPulse     |
| Oxygen                               | 0.007552454  | -0.002444417 | 0.068595335  |
| RunTime                              | -0.002444417 | 0.001576638  | -0.010973127 |
| RunPulse                             | 0.068595335  | -0.010973127 | 1.363981626  |
| Total Covariance Matrix              |              |              |              |
|                                      | Oxygen       | RunTime      | RunPulse     |
| Oxygen                               | 1.160483133  | -0.266743840 | -1.193780414 |
| RunTime                              | -0.266743840 | 0.084391647  | 0.266836165  |
| RunPulse                             | -1.193780414 | 0.266836165  | 5.350240500  |

Assuming that the between-imputation covariance matrix is proportional to the within-imputation covariance matrix, the procedure also displays a multivariate inference for all the parameters taken jointly.

**Output 10.1.5.** Multiple Imputation Multivariate Inference

| The MIANALYZE Procedure                                        |        |        |                            |        |  |
|----------------------------------------------------------------|--------|--------|----------------------------|--------|--|
| Multiple Imputation Multivariate Inference                     |        |        |                            |        |  |
| Assuming Proportionality of Between/Within Covariance Matrices |        |        |                            |        |  |
| Avg Relative Increase in Variance                              | Num DF | Den DF | F for H0: Parameter=Theta0 | Pr > F |  |
| 0.196958                                                       | 3      | 222.91 | 11408.0                    | <.0001 |  |

Assuming that the within-imputation covariance matrix is proportional to the between-imputation covariance matrix, the table shows a significant  $p$ -value for the null hypothesis that the population means are all equal to zero.

With the exception of the multivariate inference, the preceding results could also have been obtained with the MI procedure.

## Example 10.2. Reading Regression Results from a DATA= EST Data Set

This example creates an EST type data set that contains regression coefficients and their corresponding covariance matrices computed from imputed data sets. These estimates are then combined to generate valid statistical inferences about the regression model.

The following statements use the REG procedure to generate regression coefficients:

```
proc reg data=outmi outest=outreg covout noprint;
    model Oxygen= RunTime RunPulse;
    by _Imputation_;
run;

proc print data=outreg(obs=8);
    var _Imputation_ _Type_ _Name_
        Intercept RunTime RunPulse;
    title 'REG Model Coefficients and Covariance matrices'
        ' (First Two Imputations)';
run;
```

Output 10.2.1. EST Type Data Set

| REG Model Coefficients and Covariance matrices (First Two Imputations) |              |        |           |           |          |          |
|------------------------------------------------------------------------|--------------|--------|-----------|-----------|----------|----------|
| Obs                                                                    | _Imputation_ | _TYPE_ | _NAME_    | Intercept | RunTime  | RunPulse |
| 1                                                                      | 1            | PARMS  |           | 97.2874   | -2.98892 | -0.10684 |
| 2                                                                      | 1            | COV    | Intercept | 55.7516   | -0.73348 | -0.27870 |
| 3                                                                      | 1            | COV    | RunTime   | -0.7335   | 0.15167  | -0.00509 |
| 4                                                                      | 1            | COV    | RunPulse  | -0.2787   | -0.00509 | 0.00194  |
| 5                                                                      | 2            | PARMS  |           | 90.9324   | -2.93338 | -0.07391 |
| 6                                                                      | 2            | COV    | Intercept | 37.5576   | -0.25970 | -0.20442 |
| 7                                                                      | 2            | COV    | RunTime   | -0.2597   | 0.13978  | -0.00722 |
| 8                                                                      | 2            | COV    | RunPulse  | -0.2044   | -0.00722 | 0.00166  |

The following statements combine the results for the imputed data sets. The EDF= option is specified to request that the adjusted degrees of freedom be used in the analysis. For a regression model with three independent variables (including the intercept) and 31 observations, the complete-data error degrees of freedom is 28.

```
proc mianalyze data=outreg edf=28;
    var Intercept RunTime RunPulse;
run;
```

**Output 10.2.2.** Multiple Imputation Variance Information

| The MIANALYZE Procedure                  |                               |                              |            |        |
|------------------------------------------|-------------------------------|------------------------------|------------|--------|
| Multiple Imputation Variance Information |                               |                              |            |        |
| Parameter                                | -----Variance-----            |                              |            | DF     |
|                                          | Between                       | Within                       | Total      |        |
| Intercept                                | 74.179857                     | 57.287519                    | 146.303348 | 5.2619 |
| RunTime                                  | 0.034202                      | 0.142151                     | 0.183193   | 16.195 |
| RunPulse                                 | 0.001533                      | 0.002304                     | 0.004144   | 8.4786 |
| Multiple Imputation Variance Information |                               |                              |            |        |
| Parameter                                | Relative Increase in Variance | Fraction Missing Information |            |        |
| Intercept                                | 1.553843                      | 0.665161                     |            |        |
| RunTime                                  | 0.288719                      | 0.242803                     |            |        |
| RunPulse                                 | 0.798522                      | 0.491731                     |            |        |

The “Multiple Imputation Variance Information” table displays the between-imputation, within-imputation, and total variances for combining complete-data inferences.

**Output 10.2.3.** Multiple Imputation Parameter Estimates

| The MIANALYZE Procedure                 |           |                  |                       |          |        |
|-----------------------------------------|-----------|------------------|-----------------------|----------|--------|
| Multiple Imputation Parameter Estimates |           |                  |                       |          |        |
| Parameter                               | Estimate  | Std Error        | 95% Confidence Limits |          | DF     |
| Intercept                               | 92.156840 | 12.095592        | 61.52383              | 122.7898 | 5.2619 |
| RunTime                                 | -2.955317 | 0.428011         | -3.86177              | -2.0489  | 16.195 |
| RunPulse                                | -0.079851 | 0.064376         | -0.22686              | 0.0672   | 8.4786 |
| Multiple Imputation Parameter Estimates |           |                  |                       |          |        |
| Parameter                               | Minimum   | Maximum          |                       |          |        |
| Intercept                               | 77.939497 | 99.920480        |                       |          |        |
| RunTime                                 | -3.159663 | -2.660085        |                       |          |        |
| RunPulse                                | -0.112277 | -0.015111        |                       |          |        |
| Multiple Imputation Parameter Estimates |           |                  |                       |          |        |
| Parameter                               | Theta0    | t for H0:        |                       |          |        |
|                                         |           | Parameter=Theta0 | Pr >  t               |          |        |
| Intercept                               | 0         | 7.62             | 0.0005                |          |        |
| RunTime                                 | 0         | -6.90            | <.0001                |          |        |
| RunPulse                                | 0         | -1.24            | 0.2481                |          |        |

The “Multiple Imputation Parameter Estimates” table displays the estimated mean and standard error of the regression coefficients. The inferences are based on the  $t$  distribution. The table also displays a 95% mean confidence interval and a  $t$  test with the associated  $p$ -value for the hypothesis that the regression coefficient is equal to zero. Since the  $p$ -value for RunPulse is 0.1597, this variable can be removed from the regression model.

### Example 10.3. Reading Mixed Model Results from PARMS= and COVB= Data Sets

This example creates data sets containing parameter estimates and covariance matrices computed by a mixed model analysis for a set of imputed data sets. These estimates are then combined to generate valid statistical inferences about the parameters.

The following PROC MIXED statements generate the fixed-effect parameter estimates and covariance matrix for each imputed data set:

```
proc mixed data=outmi;
  model Oxygen= RunTime RunPulse/solution covb;
  by _Imputation_;
  ods output SolutionF=mixparms CovB=mixcovb;
run;

proc print data=mixparms (obs=6);
  var _Imputation_ Effect Estimate;
  title 'MIXED Model Coefficients (First Two Imputations)';
run;

proc print data=mixcovb (obs=6);
  var _Imputation_ Effect Col1 Col2 Col3;
  title 'MIXED Covariance Matrices (First Two Imputations)';
run;
```

Output 10.3.1. PROC MIXED Model Coefficients

| MIXED Model Coefficients (First Two Imputations) |              |           |          |  |
|--------------------------------------------------|--------------|-----------|----------|--|
| Obs                                              | _Imputation_ | Effect    | Estimate |  |
| 1                                                | 1            | Intercept | 97.2874  |  |
| 2                                                | 1            | RunTime   | -2.9889  |  |
| 3                                                | 1            | RunPulse  | -0.1068  |  |
| 4                                                | 2            | Intercept | 90.9324  |  |
| 5                                                | 2            | RunTime   | -2.9334  |  |
| 6                                                | 2            | RunPulse  | -0.07391 |  |

**Output 10.3.2.** PROC MIXED Covariance Matrices

| Covariance Matrices (First Two Imputations) |              |           |         |          |          |
|---------------------------------------------|--------------|-----------|---------|----------|----------|
| Obs                                         | _Imputation_ | Effect    | Col1    | Col2     | Col3     |
| 1                                           | 1            | Intercept | 55.7516 | -0.7335  | -0.2787  |
| 2                                           | 1            | RunTime   | -0.7335 | 0.1517   | -0.00509 |
| 3                                           | 1            | RunPulse  | -0.2787 | -0.00509 | 0.001942 |
| 4                                           | 2            | Intercept | 37.5576 | -0.2597  | -0.2044  |
| 5                                           | 2            | RunTime   | -0.2597 | 0.1398   | -0.00722 |
| 6                                           | 2            | RunPulse  | -0.2044 | -0.00722 | 0.001661 |

The following statements use the MIANALYZE procedure with PARMS= and COVB= input data sets to produce the same results as in Example 10.2:

```
proc mianalyze parms=mixparms covb=mixcovb edf=28;
  var Intercept RunTime RunPulse;
run;
```

### Example 10.4. Reading Generalized Linear Model Results from PARMS= and COVB= Data Sets

This example creates data sets containing parameter estimates and corresponding covariance matrices computed by a generalized linear model analysis for a set of imputed data sets. These estimates are then combined to generate valid statistical inferences about the model parameters.

The following statements use PROC GENMOD to generate the parameter estimates and covariance matrix for each imputed data set:

```
proc genmod data=outmi;
  model Oxygen= RunTime RunPulse/covb;
  by _Imputation_;
  ods output ParameterEstimates=gmparms
             CovB=gmcovb;
run;

proc print data=gmparms (obs=8);
  var _Imputation_ Parameter Estimate;
  title 'GENMOD Model Coefficients (First Two Imputations)';
run;

proc print data=gmcovb (obs=8);
  var _Imputation_ RowName Prm1 Prm2 Prm3;
  title 'GENMOD Covariance Matrices (First Two Imputations)';
run;
```

**Output 10.4.1.** PROC GENMOD Model Coefficients

| GENMOD Model Coefficients (First Two Imputations) |              |           |          |  |
|---------------------------------------------------|--------------|-----------|----------|--|
| Obs                                               | _Imputation_ | Parameter | Estimate |  |
| 1                                                 | 1            | Intercept | 97.2874  |  |
| 2                                                 | 1            | RunTime   | -2.9889  |  |
| 3                                                 | 1            | RunPulse  | -0.1068  |  |
| 4                                                 | 1            | Scale     | 2.6227   |  |
| 5                                                 | 2            | Intercept | 90.9324  |  |
| 6                                                 | 2            | RunTime   | -2.9334  |  |
| 7                                                 | 2            | RunPulse  | -0.0739  |  |
| 8                                                 | 2            | Scale     | 2.4567   |  |

The following table displays the covariance matrices for the first two imputed data sets. Note that the GENMOD procedure computes maximum likelihood estimates for the covariance matrix.

**Output 10.4.2.** PROC GENMOD Covariance Matrices

| GENMOD Covariance Matrices (First Two Imputations) |              |          |           |           |           |
|----------------------------------------------------|--------------|----------|-----------|-----------|-----------|
| Obs                                                | _Imputation_ | Row Name | Prm1      | Prm2      | Prm3      |
| 1                                                  | 1            | Prm1     | 50.356306 | -0.662498 | -0.251728 |
| 2                                                  | 1            | Prm2     | -0.662498 | 0.1369885 | -0.004598 |
| 3                                                  | 1            | Prm3     | -0.251728 | -0.004598 | 0.0017536 |
| 4                                                  | 1            | Scale    | -4.14E-16 | 3.635E-16 | -1.53E-17 |
| 5                                                  | 2            | Prm1     | 33.923008 | -0.234572 | -0.184639 |
| 6                                                  | 2            | Prm2     | -0.234572 | 0.1262507 | -0.006523 |
| 7                                                  | 2            | Prm3     | -0.184639 | -0.006523 | 0.0014999 |
| 8                                                  | 2            | Scale    | 1.858E-14 | 8.886E-16 | -1.58E-16 |

The following statements use the MIANALYZE procedure with PARMS= and COVB= input data sets:

```
proc mianalyze parms=gmparms covb=gmcovb;
  var Intercept RunTime RunPulse;
run;
```

Since the GENMOD procedure computes maximum likelihood estimates for the covariance matrix, the EDF= option is not used. The resulting model coefficients are identical to the estimates from Example 10.2, but the standard errors are slightly different because in this example, maximum likelihood estimates for the standard errors are combined without the EDF= option, whereas in Example 10.2, unbiased estimates for the standard errors are combined with the EDF= option.

### Example 10.5. Reading GLM Results from PARMS= and XPXI= Data Sets

This example creates data sets containing parameter estimates and corresponding  $(X'X)^{-1}$  matrices computed by a general linear model analysis for a set of imputed data sets. These estimates are then combined to generate valid statistical inferences about the model parameters.

The following statements use PROC GLM to generate the parameter estimates and  $(X'X)^{-1}$  matrix for each imputed data set:

```
proc glm data=outmi;
  model Oxygen= RunTime RunPulse/inverse;
  by _Imputation_;
  ods output ParameterEstimates=glmparms
             InvXPX=glmxpxi;
run;

proc print data=glmparms (obs=6);
  var _Imputation_ Parameter Estimate;
  title 'GLM Model Coefficients (First Two Imputations)';
run;

proc print data=glmxpxi (obs=8);
  var _Imputation_ Parameter Intercept RunTime RunPulse;
  title 'GLM X'X Inverse Matrices (First Two Imputations)';
run;
```

Output 10.5.1. PROC GLM Model Coefficients

| GLM Model Coefficients (First Two Imputations) |              |           |             |
|------------------------------------------------|--------------|-----------|-------------|
| Obs                                            | _Imputation_ | Parameter | Estimate    |
| 1                                              | 1            | Intercept | 97.28741708 |
| 2                                              | 1            | RunTime   | -2.98892274 |
| 3                                              | 1            | RunPulse  | -0.10683710 |
| 4                                              | 2            | Intercept | 90.93235575 |
| 5                                              | 2            | RunTime   | -2.93337517 |
| 6                                              | 2            | RunPulse  | -0.07390872 |

Output 10.5.2. PROC GLM  $(X'X)^{-1}$  Matrices

| GLM X'X Inverse Matrices (First Two Imputations) |              |           |              |              |              |
|--------------------------------------------------|--------------|-----------|--------------|--------------|--------------|
| Obs                                              | _Imputation_ | Parameter | Intercept    | RunTime      | RunPulse     |
| 1                                                | 1            | Intercept | 7.3205589063 | -0.096310799 | -0.036595038 |
| 2                                                | 1            | RunTime   | -0.096310799 | 0.0199147383 | -0.000668435 |
| 3                                                | 1            | RunPulse  | -0.036595038 | -0.000668435 | 0.0002549371 |
| 4                                                | 1            | Oxygen    | 97.28741708  | -2.988922744 | -0.106837096 |
| 5                                                | 2            | Intercept | 5.620924071  | -0.038867743 | -0.030594091 |
| 6                                                | 2            | RunTime   | -0.038867743 | 0.0209193054 | -0.00108086  |
| 7                                                | 2            | RunPulse  | -0.030594091 | -0.00108086  | 0.0002485262 |
| 8                                                | 2            | Oxygen    | 90.932355745 | -2.933375175 | -0.073908724 |



The following statements use the MIANALYZE procedure with PARMS= and XPXI= input data sets to produce the same results as in Example 10.2:

```
proc mianalyze parms=glmparms xpxi=glmxpxi edf=28;
  var Intercept RunTime RunPulse;
run;
```

## Example 10.6. Combining Correlation Coefficients

This example combines sample correlation coefficients and associated variances computed from a set of imputed data sets.

The following statements use the CORR procedure to compute the correlation  $r$  between variables Oxygen and RunTime for each imputed data set:

```
proc corr data=outmi out=outcorr;
  var Oxygen RunTime;
  by _Imputation_;
run;

proc print data=outcorr (obs=10);
  title 'Correlations (First Two Imputations)';
run;
```

Output 10.6.1. CORR Type Data Set

| Correlations (First Two Imputations) |              |        |         |         |         |
|--------------------------------------|--------------|--------|---------|---------|---------|
| Obs                                  | _Imputation_ | _TYPE_ | _NAME_  | Oxygen  | RunTime |
| 1                                    | 1            | MEAN   |         | 47.3458 | 10.5859 |
| 2                                    | 1            | STD    |         | 5.3117  | 1.3547  |
| 3                                    | 1            | N      |         | 31.0000 | 31.0000 |
| 4                                    | 1            | CORR   | Oxygen  | 1.0000  | -0.8338 |
| 5                                    | 1            | CORR   | RunTime | -0.8338 | 1.0000  |
| 6                                    | 2            | MEAN   |         | 47.3316 | 10.6004 |
| 7                                    | 2            | STD    |         | 5.3612  | 1.4336  |
| 8                                    | 2            | N      |         | 31.0000 | 31.0000 |
| 9                                    | 2            | CORR   | Oxygen  | 1.0000  | -0.8704 |
| 10                                   | 2            | CORR   | RunTime | -0.8704 | 1.0000  |

The following statements compute Fisher's  $Z$  transformation of  $r$

$$z = \frac{1}{2} \log \left( \frac{1+r}{1-r} \right)$$

and the variance estimate corresponding to  $z$ ,  $1/(n-3)$ .

```

data ztrans(type=EST);
  set outcorr (drop= RunTime rename= Oxygen= Z);
  if (_type_ = 'N' or _name_ = 'RunTime');

  if (_type_ = 'CORR') then do;
    _type_ = 'PARMS';
    _name_ = '';
    Z= 0.5 * log((1+Z)/(1-Z));
  end;

  else if (_type_ = 'N') then do;
    _type_ = 'COVB';
    _name_ = 'Z';
    Z= 1. / (Z-3);
  end;
run;

proc print data=ztrans;
  title 'EST Type Data Set with Fisher''s Z Transformation';
run;

```

**Output 10.6.2.** Fisher's Z Transformation

| EST Type Data Set with Fisher's Z Transformation |              |        |        |          |
|--------------------------------------------------|--------------|--------|--------|----------|
| Obs                                              | _Imputation_ | _TYPE_ | _NAME_ | Z        |
| 1                                                | 1            | COVB   | Z      | 0.03571  |
| 2                                                | 1            | PARMS  |        | -1.20037 |
| 3                                                | 2            | COVB   | Z      | 0.03571  |
| 4                                                | 2            | PARMS  |        | -1.33458 |
| 5                                                | 3            | COVB   | Z      | 0.03571  |
| 6                                                | 3            | PARMS  |        | -1.34657 |
| 7                                                | 4            | COVB   | Z      | 0.03571  |
| 8                                                | 4            | PARMS  |        | -1.20194 |
| 9                                                | 5            | COVB   | Z      | 0.03571  |
| 10                                               | 5            | PARMS  |        | -1.29004 |

The following statements use the MIANALYZE procedure to generate a combined parameter estimate  $\hat{z}$  and variance for Fisher's  $z$ . The ODS statement is used to save the parameter estimates in an output data set.

```

proc mianalyze data=ztrans;
  ods output ParmEst=parms;
  var z;
run;

```

**Output 10.6.3.** Inferences Based on Fisher's Z

| The MIANALYZE Procedure                 |           |                               |                       |          |        |
|-----------------------------------------|-----------|-------------------------------|-----------------------|----------|--------|
| Multiple Imputation Parameter Estimates |           |                               |                       |          |        |
| Parameter                               | Estimate  | Std Error                     | 95% Confidence Limits |          | DF     |
| z                                       | -1.274701 | 0.204097                      | -1.67720              | -0.87220 | 196.63 |
| Multiple Imputation Parameter Estimates |           |                               |                       |          |        |
| Parameter                               | Minimum   |                               | Maximum               |          |        |
| z                                       | -1.346574 |                               | -1.200373             |          |        |
| Multiple Imputation Parameter Estimates |           |                               |                       |          |        |
| Parameter                               | Theta0    | t for H0:<br>Parameter=Theta0 |                       | Pr >  t  |        |
| z                                       | 0         | -6.25                         |                       | <.0001   |        |

In addition to the estimate for  $z$ , PROC MIANALYZE also generates 95% confidence limits for  $z$ ,  $\hat{z}_{.025}$  and  $\hat{z}_{.975}$ . An estimate of the correlation coefficient and 95% confidence limits can then be generated from the inverse transformation

$$r = \frac{e^{2z} - 1}{e^{2z} + 1}$$

for  $z = \hat{z}$ ,  $\hat{z}_{.025}$ , and  $\hat{z}_{.975}$ .

The following statements print the estimate for  $z$  and 95% confidence limits for  $z$ :

```
proc print data=parms;
  title 'Parameter Estimates with 95% Confidence Limits';
  var Estimate LCLMean UCLMean;
run;
```

**Output 10.6.4.** Parameter Estimates with 95% Confidence Limits

| Parameter Estimates with 95% Confidence Limits |           |          |          |
|------------------------------------------------|-----------|----------|----------|
| Obs                                            | Estimate  | LCLMean  | UCLMean  |
| 1                                              | -1.274701 | -1.67720 | -0.87220 |

The following statements generate an estimate of the correlation coefficient and 95% confidence limits:

```
data corr_ci;
  set parms;
  r=      (exp(2*Estimate)-1)/(exp(2*Estimate)+1);
  r_low= (exp(2*LCLMean)-1)/(exp(2*LCLMean)+1);
  r_upp= (exp(2*UCLMean)-1)/(exp(2*UCLMean)+1);
;

proc print data=corr_ci;
  title 'Estimated Correlation Coefficient'
        ' with 95% Confidence Limits';
  var r r_low r_upp;
run;
```

**Output 10.6.5.** Estimated Correlation Coefficient

| Estimated Correlation Coefficient with 95% Confidence Limits |          |          |          |  |
|--------------------------------------------------------------|----------|----------|----------|--|
| Obs                                                          | r        | r_low    | r_upp    |  |
| 1                                                            | -0.85507 | -0.93250 | -0.70249 |  |

## Example 10.7. Combining Ratios of Variable Means

This example combines ratios of variable means and associated variances computed from imputed data sets.

The following statements use the CORR procedure to compute the means and covariance matrix of variables Oxygen and RunTime for each imputed data set. Within each imputation, the data set is sorted so that observations with `_TYPE_='MEAN'` and `_TYPE_='N'` are read before observations with `_TYPE_='COV'`.

```
proc corr data=outmi cov nocorr noprint out=outcov;
  var RunTime RunPulse;
  by _Imputation_;
run;

proc sort data=outcov;
  by _Imputation_ _name_;
run;

proc print data=outcov (obs=10);
  title 'Means and Covariance Matrices (First Two Imputations)';
run;
```

**Output 10.7.1.** Means and Covariance Matrices

| Means and Covariance Matrices (First Two Imputations) |              |        |          |         |          |
|-------------------------------------------------------|--------------|--------|----------|---------|----------|
| Obs                                                   | _Imputation_ | _TYPE_ | _NAME_   | RunTime | RunPulse |
| 1                                                     | 1            | MEAN   |          | 10.5859 | 171.301  |
| 2                                                     | 1            | STD    |          | 1.3547  | 11.974   |
| 3                                                     | 1            | N      |          | 31.0000 | 31.000   |
| 4                                                     | 1            | COV    | RunPulse | 4.8121  | 143.368  |
| 5                                                     | 1            | COV    | RunTime  | 1.8353  | 4.812    |
| 6                                                     | 2            | MEAN   |          | 10.6004 | 169.204  |
| 7                                                     | 2            | STD    |          | 1.4336  | 13.153   |
| 8                                                     | 2            | N      |          | 31.0000 | 31.000   |
| 9                                                     | 2            | COV    | RunPulse | 8.9385  | 172.998  |
| 10                                                    | 2            | COV    | RunTime  | 2.0553  | 8.938    |

For each imputation, the following statements compute a ratio estimate of the means for the variables RunPulse and RunTime, and a corresponding variance estimate:

```

data vratio (type=EST);
  set outcov;
  keep _Imputation_ _type_ _name_ ratio;
  retain TMean PMean N VarP;

  if (_type_ = 'N') then N= RunTime;

  if (_type_ = 'MEAN') then do;
    TMean= RunTime;
    PMean= RunPulse;
    Ratio= RunPulse / RunTime;
    _type_ = 'PARMS';
  end;

  if (_type_ = 'COV' and _name_ = 'RunPulse')
    then VarP= RunPulse;

  if (_type_ = 'COV' and _name_ = 'RunTime') then do;
    Ratio= ( PMean**2/TMean**4 * RunTime
             - 2 * PMean/TMean**3 * RunPulse
             + (1./ TMean**2) * VarP ) / N;
    _name_ = 'Ratio';
  end;

  if ( _type_ = 'STD' or _type_ = 'N' or _name_ = 'RunPulse')
    then delete;
run;

proc print data=vratio;
  title 'EST Type Data Set with Ratios of Variable Means';
run;

```

**Output 10.7.2.** Ratio of Variable Means

| EST Type Data Set with Ratios of Variable Means |              |        |        |         |
|-------------------------------------------------|--------------|--------|--------|---------|
| Obs                                             | _Imputation_ | _TYPE_ | _NAME_ | Ratio   |
| 1                                               | 1            | PARMS  |        | 16.1821 |
| 2                                               | 1            | COV    | Ratio  | 0.1348  |
| 3                                               | 2            | PARMS  |        | 15.9620 |
| 4                                               | 2            | COV    | Ratio  | 0.1181  |
| 5                                               | 3            | PARMS  |        | 16.0602 |
| 6                                               | 3            | COV    | Ratio  | 0.1376  |
| 7                                               | 4            | PARMS  |        | 15.9968 |
| 8                                               | 4            | COV    | Ratio  | 0.1409  |
| 9                                               | 5            | PARMS  |        | 16.2961 |
| 10                                              | 5            | COV    | Ratio  | 0.1678  |

The following statements use the MIANALYZE procedure to generate a combined point estimate for the ratio of variable means and a variance estimate:

```
proc mianalyze data=vratio theta0=1;
  var ratio;
run;
```

**Output 10.7.3.** Inferences for a Ratio of Variable Means

| The MIANALYZE Procedure                 |           |                               |                       |          |        |
|-----------------------------------------|-----------|-------------------------------|-----------------------|----------|--------|
| Multiple Imputation Parameter Estimates |           |                               |                       |          |        |
| Parameter                               | Estimate  | Std Error                     | 95% Confidence Limits |          | DF     |
| ratio                                   | 16.099445 | 0.403440                      | 15.30394              | 16.89495 | 201.33 |
| Multiple Imputation Parameter Estimates |           |                               |                       |          |        |
| Parameter                               | Minimum   |                               | Maximum               |          |        |
| ratio                                   | 15.961997 |                               | 16.296124             |          |        |
| Multiple Imputation Parameter Estimates |           |                               |                       |          |        |
| Parameter                               | Theta0    | t for H0:<br>Parameter=Theta0 |                       | Pr >  t  |        |
| ratio                                   | 1.000000  | 37.43                         |                       | <.0001   |        |

The variable RunTime has an estimated mean of about 10.6 and a standard error of about 1.4 in each imputed data set. An estimated coefficient of variation is  $(1.4/\sqrt{31})/10.6 = 2.37\%$ . Since the coefficient of variation is less than 10%, the ratio of sample means is approximately unbiased and normally distributed.

---

## References

- Barnard, J. and Rubin, D.B. (1999), “Small-Sample Degrees of Freedom with Multiple Imputation,” *Biometrika*, 86, 948–955.
- Cochran, W.J. (1977), *Sampling Techniques*, Second Edition, New York: John Wiley & Sons, Inc.
- Li, K.H., Raghunathan, T.E., and Rubin, D.B. (1991), “Large-Sample Significance Levels from Multiply Imputed Data Using Moment-Based Statistics and an F Reference Distribution,” *Journal of the American Statistical Association*, 86, 1065–1073.
- Rubin, D.B. (1976), “Inference and Missing Data,” *Biometrika*, 63, 581–592.
- Rubin, D.B. (1987), *Multiple Imputation for Nonresponse in Surveys*, New York: John Wiley & Sons, Inc.
- Rubin, D.B. (1996), “Multiple Imputation After 18+ Years,” *Journal of the American Statistical Association*, 91, 473–489.
- SAS Institute Inc. (1999), *SAS Language Reference: Concepts, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS Procedures Guide, Version 8*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1999), *SAS/STAT User’s Guide, Version 8*, Cary, NC: SAS Institute Inc.
- Schafer, J.L. (1997), *Analysis of Incomplete Multivariate Data*, New York: Chapman and Hall.





# Chapter 11

## The NPAR1WAY Procedure

### Chapter Table of Contents

---

|                 |     |
|-----------------|-----|
| <b>OVERVIEW</b> | 237 |
| <b>SYNTAX</b>   | 237 |
| EXACT Statement | 237 |



# Chapter 11

## The NPAR1WAY Procedure

---

### Overview

The new KS option in the EXACT statement provides the exact Kolmogorov-Smirnov test for two-sample data. The new POINT option in the EXACT statement provides exact point probabilities for the test statistics.

---

### Syntax

---

#### EXACT Statement

**EXACT** *statistic-options* < / *computation-options* > ;

The EXACT statement requests exact tests for the specified statistics. Optionally, PROC NPAR1WAY computes Monte Carlo estimates of the exact  $p$ -values. The *statistic-options* specify the statistics for which to provide exact tests. The *computation-options* specify options for the computation of exact statistics.

The following new *statistic-option* is now available in the EXACT statement:

#### KS

requests the exact Kolmogorov-Smirnov test for two-sample data.

The following new *computation-option* is now available in the EXACT statement after the slash (/):

#### POINT

requests exact point probabilities for the test statistics.

The POINT option is available for all EXACT statement *statistic-options*. The POINT option is not available with the MC option, which provides Monte Carlo estimation of exact  $p$ -values instead of direct exact  $p$ -value computation.



# Chapter 12

## The PROBIT Procedure

### Chapter Table of Contents

---

|                                    |     |
|------------------------------------|-----|
| <b>OVERVIEW</b>                    | 241 |
| <b>SYNTAX</b>                      | 241 |
| PROC PROBIT Statement              | 241 |
| CDFPLOT Statement                  | 242 |
| INSET Statement                    | 250 |
| IPPPLOT Statement                  | 252 |
| LPREDPLOT Statement                | 260 |
| MODEL Statement                    | 268 |
| PREDPPLOT Statement                | 269 |
| <b>DETAILS</b>                     | 277 |
| INEST= <i>SAS-data-set</i>         | 277 |
| Inverse Confidence Limits          | 278 |
| Lack of Fit Tests                  | 279 |
| Rescaling the Covariance Matrix    | 280 |
| XDATA= <i>SAS-data-set</i>         | 280 |
| <b>EXAMPLES</b>                    | 281 |
| Example 12.1 Multilevel Response   | 281 |
| Example 12.2 An Epidemiology Study | 286 |
| <b>REFERENCES</b>                  | 298 |



## Chapter 12

# The PROBIT Procedure

---

### Overview

You can specify interaction terms among any of the specified explanatory variables in the MODEL statement in the same way as in the GLM procedure. Because of the more complicated models that you can specify, only a single MODEL statement is now allowed for each PROC PROBIT statement. If more than one MODEL statement is specified, only the last one is used.

You can now graphically display results of model parameter estimation. For binary data analysis, you can construct plots of predicted probabilities, inverse predicted probabilities, and cumulative probabilities. For ordinal multinomial models, you can display plots of predicted probabilities, linear predictors, and cumulative probabilities. In addition, you can control graphical features such as plot layout, colors, plotting symbols, line styles, and the text fonts used in the plots.

You can use the new XDATA= data set to provide the values for effects other than the single continuous dosage effect when the predicted dosage and their fiducial limits are computed. These specified values can also be used for generating plots.

You can use the new INEST= data set to provide the initial values for parameters in the model. The structure of this data set is similar to that of the OUTEST= data set. This data set will overwrite the original INITIAL= option in the MODEL statement.

The OUTEST= data set is extended to both binomial and multinomial models. It also generates the OUTEST data set whether there is a CLASS statement or not. The OUTEST= data set does not include a fixed threshold.

You can use the new AGGREGATE= option in the MODEL statement to define the subpopulations for calculating the Pearson chi-square statistic and the deviance.

---

### Syntax

---

#### PROC PROBIT Statement

```
PROC PROBIT < options > ;
```

**GOUT=***graphics-catalog*

specifies a graphics catalog in which to save graphics output.

**INEST=** *SAS-data-set*

specifies an input SAS data set that contains initial estimates for all the parameters

in the model. See the section “INEST= *SAS-data-set*” on page 277 for a detailed description of the contents of the INEST= data set.

**NAMELEN=*n***

specifies the length of effect names in tables and output data sets to be *n* characters, where *n* is a value between 20 and 200. The default length is 20 characters.

**XDATA= *SAS-data-set***

specifies an input SAS data set that contains values for all the independent variables in the MODEL statement and variables in the CLASS statement. If there are covariates specified in a MODEL statement, you specify fixed values for the effects in the MODEL statement by the XDATA= data set when predicted values and/or fiducial limits for a single continuous variable (dose variable) are required. These specified values for the effects in the MODEL statement are also used for generating plots. See the section “XDATA= *SAS-data-set*” on page 280 for a detailed description of the contents of the XDATA= data set.

---

## CDFPLOT Statement

**CDFPLOT** <*var = variable*> <*options*>;

The CDFPLOT statement plots the predicted cumulative distribution function (CDF) of the multinomial response variable as a function of a single continuous independent variable (dose variable). You can only use this statement after a multinomial model statement.

**VAR= (*variable*)**

specifies a single continuous variable (dose variable) in the independent variable list of the MODEL statement. If a VAR= variable is not specified, the first single continuous variable in the independent variable list of the MODEL statement is used. If such a variable does not exist in the independent variable list of the MODEL statement, an error is reported.

The predicted cumulative distribution function is defined as

$$\hat{F}_j(\mathbf{x}) = C + (1 - C)F(\hat{a}_j + \mathbf{x}'\hat{\mathbf{b}})$$

where  $j = 1, \dots, k$  are the indexes of the  $k$  levels of the multinomial response variable,  $F$  is the CDF of the distribution used to model the cumulative probabilities,  $\hat{\mathbf{b}}$  is the vector of estimated parameters,  $\mathbf{x}$  is the covariate vector,  $\hat{a}_j$  are estimated ordinal intercepts with  $\hat{a}_1 = 0$ , and  $C$  is the threshold parameter, either known or estimated from the model. Let  $x_1$  be the covariate corresponding to the dose variable and  $\mathbf{x}_{-1}$  be the vector of the rest of the covariates. Let the corresponding estimated parameters be  $\hat{b}_1$  and  $\hat{\mathbf{b}}_{-1}$ . Then

$$\hat{F}_j(\mathbf{x}) = C + (1 - C)F(\hat{a}_j + x_1\hat{b}_1 + \mathbf{x}_{-1}'\hat{\mathbf{b}}_{-1})$$



To plot  $\hat{F}_j$  as a function of  $x_1$ ,  $\mathbf{x}_{-1}$  must be specified. You can use the XDATA= option to provide the values of  $\mathbf{x}_{-1}$  (see the XDATA= option in the PROC PROBIT statement for details), or use the default values that follow the rules:

- If the effect contains a continuous variable (or variables), the overall mean of this effect is used.
- If the effect is a single classification variable, the highest level of the variable is used.

### options

specify the levels of the multinomial response variable for which the cdf curves are requested, and add features to the plot. There are  $k - 1$  curves for a  $k$ -level multinomial response variable (for the highest level, it is the constant line 1). You can specify any of them to be plotted by the LEVEL= option in the CDFPLOT statement. See the LEVEL= option for how to specify the levels.

An attached box on the right side of the plot is used to label these curves with the names of their levels. You can specify the color of this box using the CLABBOX= option.

You can use options in the CDFPLOT statement to

- superimpose specification limits
- specify the levels for which the cdf curves are requested
- specify graphical enhancements (such as color or text height)

### Summary of Options

The following tables list all *options* by function. The “Dictionary of Options” on page 246 describes each option in detail.

#### CDF Options

**Table 12.1.** Options for CDFPLOT

|                              |                                                                          |
|------------------------------|--------------------------------------------------------------------------|
| LEVEL= <i>character-list</i> | specifies the names of the levels for which the cdf curves are requested |
| NOTHRESH                     | suppresses the threshold line                                            |
| THRESHLABPOS= <i>value</i>   | specifies the position for the label of the threshold line               |

**General Options****Table 12.2.** Color Options

|                       |                                   |
|-----------------------|-----------------------------------|
| CAXIS= <i>color</i>   | specifies color for axis          |
| CFIT= <i>color</i>    | specifies color for fitted curves |
| CFRAME= <i>color</i>  | specifies color for frame         |
| CGRID= <i>color</i>   | specifies color for grid lines    |
| CHREF= <i>color</i>   | specifies color for HREF= lines   |
| CLABBOX= <i>color</i> | specifies color for label box     |
| CTEXT= <i>color</i>   | specifies color for text          |
| CVREF= <i>color</i>   | specifies color for VREF= lines   |

**Table 12.3.** Options to Enhance Plots Produced on Graphics Devices

|                           |                                                     |
|---------------------------|-----------------------------------------------------|
| ANNOTATE=<br>SAS-data-set | specifies an ANNOTATE data set                      |
| INBORDER                  | requests a border around plot                       |
| LFIT= <i>linetype</i>     | specifies line style for fitted curves              |
| LGRID= <i>linetype</i>    | specifies line style for grid lines                 |
| NOFRAME                   | suppresses the frame around plotting areas          |
| NOGRID                    | suppresses grid lines                               |
| NOFIT                     | suppresses cdf curves                               |
| NOHLABEL                  | suppresses horizontal labels                        |
| NOHTICK                   | suppresses horizontal ticks                         |
| NOVTICK                   | suppresses vertical ticks                           |
| TURNVLABELS               | vertically string out characters in vertical labels |
| WFIT= <i>n</i>            | specifies thickness for fitted curves               |
| WGRID= <i>n</i>           | specifies thickness for grids                       |
| WREFL= <i>n</i>           | specifies thickness for reference lines             |

**Table 12.4.** Axis Options

|                                                        |                                                |
|--------------------------------------------------------|------------------------------------------------|
| HAXIS= <i>value1 to value2</i><br>< <i>by value3</i> > | specifies tick mark values for horizontal axis |
| HOFFSET= <i>value</i>                                  | specifies offset for horizontal axis           |
| HLOWER= <i>value</i>                                   | specifies lower limit on horizontal axis scale |
| HUPPER= <i>value</i>                                   | specifies upper limit on horizontal axis scale |
| NHTICK= <i>n</i>                                       | specifies number of ticks for horizontal axis  |
| NVTICK= <i>n</i>                                       | specifies number of ticks for vertical axis    |
| VAXIS= <i>value1 to value2</i><br>< <i>by value3</i> > | specifies tick mark values for vertical axis   |
| VAXISLABEL='label'                                     | specifies label for vertical axis              |
| VOFFSET= <i>value</i>                                  | specifies offset for vertical axis             |
| VLOWER= <i>value</i>                                   | specifies lower limit on vertical axis scale   |
| VUPPER= <i>value</i>                                   | specifies upper limit on vertical axis scale   |
| WAXIS= <i>n</i>                                        | specifies thickness for axis                   |

**Table 12.5.** Graphics Catalog Options

|                      |                                                   |
|----------------------|---------------------------------------------------|
| DESCRIPTION='string' | specifies description for graphics catalog member |
| NAME='string'        | specifies name for plot in graphics catalog       |

**Table 12.6.** Options for Text Enhancement

|                        |                                                      |
|------------------------|------------------------------------------------------|
| FONT= <i>font</i>      | specifies software font for text                     |
| HEIGHT= <i>value</i>   | specifies height of text used outside framed areas   |
| INFONT= <i>font</i>    | specifies software font for text inside framed areas |
| INHEIGHT= <i>value</i> | specifies height of text inside framed areas         |

**Table 12.7.** Options for Reference Lines

|                                               |                                                         |
|-----------------------------------------------|---------------------------------------------------------|
| HREF < (INTERSECT) ><br>=value-list           | requests horizontal reference line                      |
| HREFLABELS=<br>( 'label1', . . . , 'labeln' ) | specifies labels for HREF= lines                        |
| HREFLABPOS= <i>n</i>                          | specifies vertical position of labels for HREF= lines   |
| LHREF= <i>linetype</i>                        | specifies line style for HREF= lines                    |
| LVREF= <i>linetype</i>                        | specifies line style for VREF= lines                    |
| VREF < (INTERSECT) ><br>=value-list           | requests vertical reference line                        |
| VREFLABELS=<br>( 'label1', . . . , 'labeln' ) | specifies labels for VREF= lines                        |
| VREFLABPOS= <i>n</i>                          | specifies horizontal position of labels for VREF= lines |

**Dictionary of Options**

The following entries provide detailed descriptions of the *options* in the CDFPLOT statement.

**ANNOTATE**=*SAS-data-set*

**ANNO**=*SAS-data-set*

specifies an ANNOTATE data set, as described in *SAS/GRAPH Software: Reference*, that enables you to add features to the cdf plot. The ANNOTATE= data set you specify in the CDFPLOT statement is used for all plots created by the statement.

**CAXIS**=*color*

**CAXES**=*color*

specifies the color used for the axes and tick marks. This option overrides any COLOR= specifications in an AXIS statement. The default is the first color in the device color list.

**CFIT**=*color*

specifies the color for the fitted cdf curves. The default is the first color in the device color list.

**CFRAME**=*color*

**CFR**=*color*

specifies the color for the area enclosed by the axes and frame. This area is not shaded by default.

**CGRID**=*color*

specifies the color for grid lines. The default is the first color in the device color list.

**CLABBOX**=*color*

specifies the color for the area enclosed by the label box for cdf curves. This area is not shaded by default.

**CHREF**=*color*

**CH**=*color*

specifies the color for lines requested by the HREF= option. The default is the first color in the device color list.

**CTEXT**=*color*

specifies the color for tick mark values and axis labels. The default is the color specified for the CTEXT= option in the most recent GOPTIONS statement.

**CVREF**=*color*

**CV**=*color*

specifies the color for lines requested by the VREF= option. The default is the first color in the device color list.

**DESCRIPTION**=*'string'*

**DES**=*'string'*

specifies a description, up to 40 characters, that appears in the PROC GREPLAY master menu. The default is the variable name.

**FONT=***font*

specifies a software font for reference line and axis labels. You can also specify fonts for axis labels in an **AXIS** statement. The **FONT=** font takes precedence over the **FTEXT=** font specified in the most recent **GOPTIONS** statement. Hardware characters are used by default.

**HAXIS=***value1 to value2*<*by value3*>

specifies tick mark values for the horizontal axis. *value1*, *value2*, and *value3* must be numeric, and *value1* must be less than *value2*. The lower tick mark is *value1*. Tick marks are drawn at increments of *value3*. The last tick mark is the greatest value that does not exceed *value2*. If *value3* is omitted, a value of 1 is used.

Examples of **HAXIS=** lists are:

```
haxis = 0 to 10
haxis = 2 to 10 by 2
haxis = 0 to 200 by 10
```

**HEIGHT=***value*

specifies the height of text used outside framed areas. The default value is 3.846 (in percentage).

**HLOWER=***value*

specifies the lower limit on the horizontal axis scale. The **HLOWER=** option specifies *value* as the lower horizontal axis tick mark. The tick mark interval and the upper axis limit are determined automatically. This option has no effect if the **HAXIS=** option is used.

**HOFFSET=***value*

specifies offset for horizontal axis. The default value is 1.

**HUPPER=***value*

specifies *value* as the upper horizontal axis tick mark. The tick mark interval and the lower axis limit are determined automatically. This option has no effect if the **HAXIS=** option is used.

**HREF <(INTERSECT)> =***value-list*

requests reference lines perpendicular to the horizontal axis. If **(INTERSECT)** is specified, a second reference line perpendicular to the vertical axis is drawn that intersects the fit line at the same point as the horizontal axis reference line. If a horizontal axis reference line label is specified, the intersecting vertical axis reference line is labeled with the vertical axis value. See also the **CHREF=**, **HREFLABELS=**, and **LHREF=** options.

**HREFLABELS=**'*label1*', ..., '*labeln*'**HREFLABEL=**'*label1*', ..., '*labeln*'**HREFLAB=**'*label1*', ..., '*labeln*'

specifies labels for the lines requested by the **HREF=** option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**HREFLABPOS=*n***

specifies the vertical position of labels for HREF= lines. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement         |
|----------|-------------------------|
| 1        | top                     |
| 2        | staggered from top      |
| 3        | bottom                  |
| 4        | staggered from bottom   |
| 5        | alternating from top    |
| 6        | alternating from bottom |

**INBORDER**

requests a border around cdf plots.

**LEVEL=** (*character-list*)**ORDINAL=** (*character-list*)

specifies the names of the levels for which cdf curves are requested. Names should be quoted and separated by space. If there is no correct name provided, no cdf curve is plotted.

**LFIT=***linetype*

specifies a line style for fitted curves. By default, fitted curves are drawn by connecting solid lines (*linetype* = 1).

**LGRID=***linetype*

specifies a line style for all grid lines. *linetype* is between 1 and 46. The default is 35.

**LHREF=***linetype***LH=***linetype*

specifies the line type for lines requested by the HREF= option. The default is 2, which produces a dashed line.

**LVREF=***linetype***LV=***linetype*

specifies the line type for lines requested by the VREF= option. The default is 2, which produces a dashed line.

**NAME=**'*string*'

specifies a name for the plot, up to eight characters, that appears in the PROC GREPLAY master menu. The default is 'PROBIT'.

**NOFIT**

suppresses the fitted cdf curves.

**NOFRAME**

suppresses the frame around plotting areas.

**NOGRID**

suppresses grid lines.

**NOHLABEL**

suppresses horizontal labels.

**NOHTICK**

suppresses horizontal tick marks.

**NOTHRESH**

suppresses the threshold line.

**NOVLABEL**

suppresses vertical labels.

**NOVTICK**

suppresses vertical tick marks.

**THRESHLABPOS=*n***

specifies the horizontal position of labels for the threshold line. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | left            |
| 2        | right           |

**VAXIS=*value1 to value2*<*by value3*>**

specifies tick mark values for the vertical axis. *value1*, *value2*, and *value3* must be numeric, and *value1* must be less than *value2*. The lower tick mark is *value1*. Tick marks are drawn at increments of *value3*. The last tick mark is the greatest value that does not exceed *value2*. This method of specification of tick marks is not valid for logarithmic axes. If *value3* is omitted, a value of 1 is used.

Examples of VAXIS= lists are:

```
vaxis = 0 to 10
vaxis = 0 to 2 by .1
```

**VAXISLABEL='string'**

specifies a label for the vertical axis.

**VLOWER=*value***

specifies the lower limit on the vertical axis scale. The VLOWER= option specifies *value* as the lower vertical axis tick mark. The tick mark interval and the upper axis limit are determined automatically. This option has no effect if the VAXIS= option is used.

**VREF=*value-list***

requests reference lines perpendicular to the vertical axis. If (INTERSECT) is specified, a second reference line perpendicular to the horizontal axis is drawn that intersects the fit line at the same point as the vertical axis reference line. If a vertical axis reference line label is specified, the intersecting horizontal axis reference line is labeled with the horizontal axis value. See also the CVREF=, LVREF=, and VREFLABELS= options.

**VREFLABELS='label1',...,'labeln'****VREFLABEL='label1',...,'labeln'****VREFLAB='label1',...,'labeln'**

specifies labels for the lines requested by the VREF= option. The number of labels

must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**VREFLABPOS=*n***

specifies the horizontal position of labels for VREF= lines. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | left            |
| 2        | right           |

**VUPPER=*value***

specifies the upper limit on the vertical axis scale. The VUPPER= option specifies *value* as the upper vertical axis tick mark. The tick mark interval and the lower axis limit are determined automatically. This option has no effect if the VAXIS= option is used.

**WAXIS=*n***

specifies line thickness for axes and frame. The default value is 1.

**WFIT=*n***

specifies line thickness for fitted curves. The default value is 1.

**WGRID=*n***

specifies line thickness for grids. The default value is 1.

**WREFL=*n***

specifies line thickness for reference lines. The default value is 1.

---

## INSET Statement

**INSET** <keyword-list> </ options>;

The box or table of summary information produced on plots made with the CDFPLOT, IPPPLOT, LPREDPLOT, and PREDPPLOT statement is called an *inset*. You can use the INSET statement to customize both the information that is printed in the inset box and the appearance of the inset box. To supply the information that is displayed in the inset box, you specify *keywords* corresponding to the information you want shown. For example, the following statements produce a predicted probability plot with the number of trials, the number of events, the name of the distribution, and the estimated optimum natural threshold in the inset.

```
proc probit data=epidemic;
  model r/n = dose;
  predpplot ;
  inset nobs ntrials nevents dist optc;
run;
```

By default, inset entries are identified with appropriate labels. However, you can provide a customized label by specifying the *keyword* for that entry followed by the



equal sign (=) and the label in quotes. For example, the following INSET statement produces an inset containing the number of observations and the name of the distribution, labeled “Sample Size” and “Distribution” in the inset.

```
inset nobs='Sample Size' dist='Distribution';
```

If you specify a keyword that does not apply to the plot you are creating, then the keyword is ignored.

The *options* control the appearance of the box.

If you specify more than one INSET statement, only the first one is used.

### **Keywords Used in the INSET Statement**

The following tables list keywords available in the INSET statement to display summary statistics, distribution parameters, and distribution fitting information.

**Table 12.8.** Summary Statistics

|          |                                           |
|----------|-------------------------------------------|
| NOBS     | number of observations                    |
| NTRIALS  | number of trials                          |
| NEVENTS  | number of events                          |
| C        | the user inputted threshold               |
| OPTC     | the estimated natural threshold           |
| NRESPLEV | number of levels of the response variable |

**Table 12.9.** General Information

|            |                                                                             |
|------------|-----------------------------------------------------------------------------|
| CONFIDENCE | confidence coefficient for all confidence intervals or for the Weibayes fit |
| DIST       | name of the distribution                                                    |

### **Options Used in the INSET Statement**

The following tables list the options available in the INSET statement.

**Table 12.10.** General Appearance Options

|                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FONT= <i>font</i>                     | specifies software font for text                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| HEIGHT= <i>value</i>                  | specifies height of text                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| HEADER= <i>'quoted string'</i>        | specifies text for header or box title                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| NOFRAME                               | omits frame around box                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| POS= <i>value</i><br><DATA   PERCENT> | determines the position of the inset. The <i>value</i> can be a compass point (N, NE, E, SE, S, SW, W, NW) or a pair of coordinates (x, y) enclosed in parentheses. The coordinates can be specified in axis percent units or axis data units.                                                                                                                                                                                                                                                                             |
| REFPOINT= <i>name</i>                 | specifies the reference point for an inset that is positioned by a pair of coordinates with the POS= option. You use the REFPOINT= option in conjunction with the POS= coordinates. The REFPOINT= option specifies which corner of the inset frame you have specified with coordinates (x, y) and it can take the value of BR (bottom right), BL (bottom left), TR (top right), or TL (top left). The default is REFPOINT=BL. If the inset position is specified as a compass point, then the REFPOINT= option is ignored. |

**Table 12.11.** Color and Pattern Options

|                       |                              |
|-----------------------|------------------------------|
| CFILL= <i>color</i>   | color for filling box        |
| CFILLH= <i>color</i>  | color for filling box header |
| CFRAME= <i>color</i>  | color for frame              |
| CHEADER= <i>color</i> | color for text in header     |
| CTEXT= <i>color</i>   | color for text               |

---

## IPPPLOT Statement

**IPPPLOT** <*var = variable*> <*options*>;

The IPPPLOT statement plots the inverse of the predicted probability against a single continuous variable (dose variable) in the MODEL statement for the binomial model. You can only use this statement after a binomial model statement. The confidence limits for the predicted values of the dose variable are the computed fiducial limits, not the inverse of the confidence limits of the predicted probabilities. Refer to the section “Inverse Confidence Limits” on page 278 for more detail.

### **VAR=** (*variable*)

specifies a single continuous variable (dose variable) in the independent variable list of the MODEL statement. If a VAR= variable is not specified, the first single continuous variable in the independent variable list of the MODEL statement is used. If such a variable does not exist in the independent variable list of the MODEL statement, an error is reported.

For the binomial model, the response variable is a probability. An estimate of the dose level  $\hat{x}_1$  needed for a response of  $p$  is given by

$$\hat{x}_1 = (F^{-1}(p) - \mathbf{x}_{-1}'\hat{\mathbf{b}}_{-1})/\hat{b}_1$$

where  $F$  is the cumulative distribution function used to model the probability,  $\mathbf{x}_{-1}$  is the vector of the rest of the covariates,  $\hat{\mathbf{b}}_{-1}$  is the vector of the estimated parameters corresponding to  $\mathbf{x}_{-1}$ , and  $\hat{b}_1$  is the estimated parameter for the dose variable of interest.

To plot  $\hat{x}_1$  as a function of  $p$ ,  $\mathbf{x}_{-1}$  must be specified. You can use the XDATA= option to provide the values of  $\mathbf{x}_{-1}$  (see the XDATA= option in the PROC PROBIT statement for details), or use the default values that follow the rules:

- If the effect contains a continuous variable (or variables), the overall mean of this effect is used.
- If the effect is a single classification variable, the highest level of the variable is used.

### options

add features to the plot.

You can use options in the IPPPLOT statement to

- superimpose specification limits
- suppress or add the observed data points on the plot
- suppress or add the fiducial limits on the plot
- specify graphical enhancements (such as color or text height)

### Summary of Options

The following tables list all *options* by function. The “Dictionary of Options” on page 256 describes each option in detail.

#### IPP Options

**Table 12.12.** Plot Layout Options for IPPPLOT

|                            |                                                            |
|----------------------------|------------------------------------------------------------|
| NOCONF                     | suppresses fiducial limits                                 |
| NODATA                     | suppresses observed data points on the plot                |
| NOTHRESH                   | suppresses the threshold line                              |
| THRESHLABPOS= <i>value</i> | specifies the position for the label of the threshold line |

**General Options****Table 12.13.** Color Options

|                      |                                   |
|----------------------|-----------------------------------|
| CAXIS= <i>color</i>  | specifies color for axis          |
| CFIT= <i>color</i>   | specifies color for fitted curves |
| CFRAME= <i>color</i> | specifies color for frame         |
| CGRID= <i>color</i>  | specifies color for grid lines    |
| CHREF= <i>color</i>  | specifies color for HREF= lines   |
| CTEXT= <i>color</i>  | specifies color for text          |
| CVREF= <i>color</i>  | specifies color for VREF= lines   |

**Table 12.14.** Options to Enhance Plots Produced on Graphics Devices

|                           |                                                              |
|---------------------------|--------------------------------------------------------------|
| ANNOTATE=<br>SAS-data-set | specifies an ANNOTATE data set                               |
| INBORDER                  | requests a border around plot                                |
| LFIT= <i>linetype</i>     | specifies line style for fitted curves and confidence limits |
| LGRID= <i>linetype</i>    | specifies line style for grid lines                          |
| NOFRAME                   | suppresses the frame around plotting areas                   |
| NOGRID                    | suppresses grid lines                                        |
| NOFIT                     | suppresses fitted curves                                     |
| NOHLABEL                  | suppresses horizontal labels                                 |
| NOHTICK                   | suppresses horizontal ticks                                  |
| NOVTICK                   | suppresses vertical ticks                                    |
| TURNVLABELS               | vertically string out characters in vertical labels          |
| WFIT= <i>n</i>            | specifies thickness for fitted curves                        |
| WGRID= <i>n</i>           | specifies thickness for grids                                |
| WREFL= <i>n</i>           | specifies thickness for reference lines                      |

**Table 12.15.** Axis Options

|                                                       |                                                |
|-------------------------------------------------------|------------------------------------------------|
| HAXIS= <i>value1 to value2</i><br><by <i>value3</i> > | specifies tick mark values for horizontal axis |
| HOFFSET= <i>value</i>                                 | specifies offset for horizontal axis           |
| HLOWER= <i>value</i>                                  | specifies lower limit on horizontal axis scale |
| HUPPER= <i>value</i>                                  | specifies upper limit on horizontal axis scale |
| NHTICK= <i>n</i>                                      | specifies number of ticks for horizontal axis  |
| NVTICK= <i>n</i>                                      | specifies number of ticks for vertical axis    |
| VAXIS= <i>value1 to value2</i><br><by <i>value3</i> > | specifies tick mark values for vertical axis   |
| VAXISLABEL='label'                                    | specifies label for vertical axis              |
| VOFFSET= <i>value</i>                                 | specifies offset for vertical axis             |
| VLOWER= <i>value</i>                                  | specifies lower limit on vertical axis scale   |
| VUPPER= <i>value</i>                                  | specifies upper limit on vertical axis scale   |
| WAXIS= <i>n</i>                                       | specifies thickness for axis                   |

**Table 12.16.** Options for Reference Lines

|                                                      |                                                         |
|------------------------------------------------------|---------------------------------------------------------|
| HREF<(INTERSECT)><br>=value-list                     | requests horizontal reference line                      |
| HREFLABELS=<br>( <i>'label1', . . . , 'labeln'</i> ) | specifies labels for HREF= lines                        |
| HREFLABPOS= <i>n</i>                                 | specifies vertical position of labels for HREF= lines   |
| LHREF= <i>linetype</i>                               | specifies line style for HREF= lines                    |
| LVREF= <i>linetype</i>                               | specifies line style for VREF= lines                    |
| VREF<(INTERSECT)><br>=value-list                     | requests vertical reference line                        |
| VREFLABELS=<br>( <i>'label1', . . . , 'labeln'</i> ) | specifies labels for VREF= lines                        |
| VREFLABPOS= <i>n</i>                                 | specifies horizontal position of labels for VREF= lines |

**Table 12.17.** Graphics Catalog Options

|                      |                                                   |
|----------------------|---------------------------------------------------|
| DESCRIPTION='string' | specifies description for graphics catalog member |
| NAME='string'        | specifies name for plot in graphics catalog       |

**Table 12.18.** Options for Text Enhancement

|                        |                                                      |
|------------------------|------------------------------------------------------|
| FONT= <i>font</i>      | specifies software font for text                     |
| HEIGHT= <i>value</i>   | specifies height of text used outside framed areas   |
| INFONT= <i>font</i>    | specifies software font for text inside framed areas |
| INHEIGHT= <i>value</i> | specifies height of text inside framed areas         |

**Dictionary of Options**

The following entries provide detailed descriptions of the *options* in the IPPLOT statement.

**ANNOTATE=SAS-data-set**

**ANNO=SAS-data-set**

specifies an ANNOTATE data set, as described in *SAS/GRAPH Software: Reference*, that enables you to add features to the ipp plot. The ANNOTATE= data set you specify in the IPPLOT statement is used for all plots created by the statement.

**CAXIS=color**

**CAXES=color**

specifies the color used for the axes and tick marks. This option overrides any COLOR= specifications in an AXIS statement. The default is the first color in the device color list.

**CFIT=color**

specifies the color for the fitted ipp curves. The default is the first color in the device color list.

**CFRAME=color**

**CFR=color**

specifies the color for the area enclosed by the axes and frame. This area is not shaded by default.

**CGRID=color**

specifies the color for grid lines. The default is the first color in the device color list.

**CHREF=color**

**CH=color**

specifies the color for lines requested by the HREF= option. The default is the first color in the device color list.

**CTEXT=color**

specifies the color for tick mark values and axis labels. The default is the color specified for the CTEXT= option in the most recent GOPTIONS statement.

**CVREF=color**

**CV=color**

specifies the color for lines requested by the VREF= option. The default is the first color in the device color list.

**DESCRIPTION='string'**

**DES='string'**

specifies a description, up to 40 characters, that appears in the PROC GREPLAY master menu. The default is the variable name.

**FONT=font**

specifies a software font for reference line and axis labels. You can also specify fonts for axis labels in an AXIS statement. The FONT= font takes precedence over the FTEXT= font specified in the most recent GOPTIONS statement. Hardware characters are used by default.

**HAXIS=***value1 to value2*<*by value3*>

specifies tick mark values for the horizontal axis. *value1*, *value2*, and *value3* must be numeric, and *value1* must be less than *value2*. The lower tick mark is *value1*. Tick marks are drawn at increments of *value3*. The last tick mark is the greatest value that does not exceed *value2*. If *value3* is omitted, a value of 1 is used.

Examples of HAXIS= lists are:

```
haxis = 0 to 10
haxis = 2 to 10 by 2
haxis = 0 to 200 by 10
```

**HEIGHT=***value*

specifies the height of text used outside framed areas. The default value is 3.846 (in percentage).

**HLOWER=***value*

specifies the lower limit on the horizontal axis scale. The HLOWER= option specifies *value* as the lower horizontal axis tick mark. The tick mark interval and the upper axis limit are determined automatically. This option has no effect if the HAXIS= option is used.

**HOFFSET=***value*

specifies offset for horizontal axis. The default value is 1.

**HUPPER=***value*

specifies *value* as the upper horizontal axis tick mark. The tick mark interval and the lower axis limit are determined automatically. This option has no effect if the HAXIS= option is used.

**HREF < (INTERSECT) > =***value-list*

requests reference lines perpendicular to the horizontal axis. If (INTERSECT) is specified, a second reference line perpendicular to the vertical axis is drawn that intersects the fit line at the same point as the horizontal axis reference line. If a horizontal axis reference line label is specified, the intersecting vertical axis reference line is labeled with the vertical axis value. See also the CHREF=, HREFLABELS=, and LHREF= options.

**HREFLABELS=***'label1', ..., 'labeln'***HREFLABEL=***'label1', ..., 'labeln'***HREFLAB=***'label1', ..., 'labeln'*

specifies labels for the lines requested by the HREF= option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**HREFLABPOS=***n*

specifies the vertical position of labels for HREF= lines. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement         |
|----------|-------------------------|
| 1        | top                     |
| 2        | staggered from top      |
| 3        | bottom                  |
| 4        | staggered from bottom   |
| 5        | alternating from top    |
| 6        | alternating from bottom |

**INBORDER**

requests a border around ipp plots.

**LFIT=linetype**

specifies a line style for fitted curves and confidence limits. By default, fitted curves are drawn by connecting solid lines (*linetype* = 1) and confidence limits are drawn by connecting dashed lines (*linetype* = 3).

**LGRID=linetype**

specifies a line style for all grid lines. *linetype* is between 1 and 46. The default is 35.

**LHREF=linetype****LH=linetype**

specifies the line type for lines requested by the HREF= option. The default is 2, which produces a dashed line.

**LVREF=linetype****LV=linetype**

specifies the line type for lines requested by the VREF= option. The default is 2, which produces a dashed line.

**NAME='string'**

specifies a name for the plot, up to eight characters, that appears in the PROC GREPLAY master menu. The default is 'PROBIT'.

**NOCONF**

suppresses fiducial limits from the plot.

**NODATA**

suppresses observed data points from the plot.

**NOFIT**

suppresses the fitted ipp curves.

**NOFRAME**

suppresses the frame around plotting areas.

**NOGRID**

suppresses grid lines.

**NOHLABEL**

suppresses horizontal labels.

**NOHTICK**

suppresses horizontal tick marks.



**NOTHRESH**

suppresses the threshold line.

**NOVLABEL**

suppresses vertical labels.

**NOVTICK**

suppresses vertical tick marks.

**THREHLABPOS=*n***

specifies the vertical position of labels for the threshold line. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | top             |
| 2        | bottom          |

**VAXIS=*value1 to value2<by value3>***

specifies tick mark values for the vertical axis. *value1*, *value2*, and *value3* must be numeric, and *value1* must be less than *value2*. The lower tick mark is *value1*. Tick marks are drawn at increments of *value3*. The last tick mark is the greatest value that does not exceed *value2*. This method of specification of tick marks is not valid for logarithmic axes. If *value3* is omitted, a value of 1 is used.

Examples of VAXIS= lists are:

```
vaxis = 0 to 10
vaxis = 0 to 2 by .1
```

**VAXISLABEL=*'string'***

specifies a label for the vertical axis.

**VLOWER=*value***

specifies the lower limit on the vertical axis scale. The VLOWER= option specifies *value* as the lower vertical axis tick mark. The tick mark interval and the upper axis limit are determined automatically. This option has no effect if the VAXIS= option is used.

**VREF=*value-list***

requests reference lines perpendicular to the vertical axis. If (INTERSECT) is specified, a second reference line perpendicular to the horizontal axis is drawn that intersects the fit line at the same point as the vertical axis reference line. If a vertical axis reference line label is specified, the intersecting horizontal axis reference line is labeled with the horizontal axis value. See also the CVREF=, LVREF=, and VREFLABELS= options.

**VREFLABELS=*'label1',...,'labeln'*****VREFLABEL=*'label1',...,'labeln'*****VREFLAB=*'label1',...,'labeln'***

specifies labels for the lines requested by the VREF= option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**VREFLABPOS=*n***

specifies the horizontal position of labels for VREF= lines. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | left            |
| 2        | right           |

**VUPPER=*value***

specifies the upper limit on the vertical axis scale. The VUPPER= option specifies *value* as the upper vertical axis tick mark. The tick mark interval and the lower axis limit are determined automatically. This option has no effect if the VAXIS= option is used.

**WAXIS=*n***

specifies line thickness for axes and frame. The default value is 1.

**WFIT=*n***

specifies line thickness for fitted curves. The default value is 1.

**WGRID=*n***

specifies line thickness for grids. The default value is 1.

**WREFL=*n***

specifies line thickness for reference lines. The default value is 1.

---

## LPREDPLOT Statement

**LPREDPLOT** <*var = variable*> <*options*>;

The LPREDPLOT statement plots the linear predictor  $\mathbf{x}'\mathbf{b}$  against a single continuous variable (dose variable) in the MODEL statement for either the binomial model or the multinomial model. The confidence limits for the predicted values are only available for the binomial model.

**VAR=** (*variable*)

specifies a single continuous variable (dose variable) in the independent variable list of the MODEL statement for which the linear predictor plot is plotted. If a VAR= variable is not specified, the first single continuous variable in the independent variable list of the MODEL statement is used. If such a variable does not exist in the independent variable list of the MODEL statement, an error is reported.

Let  $x_1$  be the covariate of the dose variable,  $\mathbf{x}_{-1}$  be the vector of the rest of the covariates,  $\hat{\mathbf{b}}_{-1}$  be the vector of estimated parameters corresponding to  $\mathbf{x}_{-1}$ , and  $\hat{b}_1$  be the estimated parameter for the dose variable of interest.

To plot  $\hat{\mathbf{x}}'\mathbf{b}$  as a function of  $x_1$ ,  $\mathbf{x}_{-1}$  must be specified. You can use the XDATA= option to provide the values of  $\mathbf{x}_{-1}$  (see the XDATA= option in the PROC PROBIT statement for details), or use the default values that follow the rules:

- If the effect contains a continuous variable (or variables), the overall mean of this effect is used.

- If the effect is a single classification variable, the highest level of the variable is used.

### options

add features to the plot.

For the multinomial model, you can use the LEVEL= option to specify the levels for which the linear predictor lines are plotted. The lines are labeled by the names of their levels in the middle.

You can use options in the LPREDPLOT statement to

- superimpose specification limits
- suppress or add the observed data points on the plot for the binomial model
- suppress or add the confidence limits for the binomial model
- specify the levels for which the linear predictor lines are requested for the multinomial model
- specify graphical enhancements (such as color or text height)

### Summary of Options

The following tables list all *options* by function. The “Dictionary of Options” on page 263 describes each option in detail.

#### LPRED Options

**Table 12.19.** Plot Layout Options for LPREDPLOT

|                              |                                                                                                                        |
|------------------------------|------------------------------------------------------------------------------------------------------------------------|
| LEVEL= <i>character-list</i> | specifies the names of the levels for which the linear predictor lines are requested (only for the multinomial model ) |
| NOCONF                       | suppresses fiducial limits (only for the binomial model)                                                               |
| NODATA                       | suppresses observed data points on the plot (only for the binomial model)                                              |
| NOTHRESH                     | suppresses the threshold line                                                                                          |
| THRESLABPOS= <i>value</i>    | specifies the position for the label of the threshold line                                                             |

#### General Options

**Table 12.20.** Color Options

|                      |                                   |
|----------------------|-----------------------------------|
| CAXIS= <i>color</i>  | specifies color for axis          |
| CFIT= <i>color</i>   | specifies color for fitted curves |
| CFRAME= <i>color</i> | specifies color for frame         |
| CGRID= <i>color</i>  | specifies color for grid lines    |
| CHREF= <i>color</i>  | specifies color for HREF= lines   |
| CTEXT= <i>color</i>  | specifies color for text          |
| CVREF= <i>color</i>  | specifies color for VREF= lines   |

**Table 12.21.** Options to Enhance Plots Produced on Graphics Devices

|                           |                                                              |
|---------------------------|--------------------------------------------------------------|
| ANNOTATE=<br>SAS-data-set | specifies an ANNOTATE data set                               |
| INBORDER                  | requests a border around plot                                |
| LFIT=linetype             | specifies line style for fitted curves and confidence limits |
| LGRID=linetype            | specifies line style for grid lines                          |
| NOFRAME                   | suppresses the frame around plotting areas                   |
| NOGRID                    | suppresses grid lines                                        |
| NOFIT                     | suppresses fitted curves                                     |
| NOHLABEL                  | suppresses horizontal labels                                 |
| NOHTICK                   | suppresses horizontal ticks                                  |
| NOVTICK                   | suppresses vertical ticks                                    |
| TURNVLABELS               | vertically strings out characters in vertical labels         |
| WFIT=n                    | specifies thickness for fitted curves                        |
| WGRID=n                   | specifies thickness for grids                                |
| WREFL=n                   | specifies thickness for reference lines                      |

**Table 12.22.** Axis Options

|                                       |                                                |
|---------------------------------------|------------------------------------------------|
| HAXIS=value1 to value2<br><by value3> | specifies tick mark values for horizontal axis |
| HOFFSET=value                         | specifies offset for horizontal axis           |
| HLOWER=value                          | specifies lower limit on horizontal axis scale |
| HUPPER=value                          | specifies upper limit on horizontal axis scale |
| NHTICK=n                              | specifies number of ticks for horizontal axis  |
| NVTICK=n                              | specifies number of ticks for vertical axis    |
| VAXIS=value1 to value2<br><by value3> | specifies tick mark values for vertical axis   |
| VAXISLABEL='label'                    | specifies label for vertical axis              |
| VOFFSET=value                         | specifies offset for vertical axis             |
| VLOWER=value                          | specifies lower limit on vertical axis scale   |
| VUPPER=value                          | specifies upper limit on vertical axis scale   |
| WAXIS=n                               | specifies thickness for axis                   |

**Table 12.23.** Graphics Catalog Options

|                      |                                                   |
|----------------------|---------------------------------------------------|
| DESCRIPTION='string' | specifies description for graphics catalog member |
| NAME='string'        | specifies name for plot in graphics catalog       |

**Table 12.24.** Options for Text Enhancement

|                        |                                                      |
|------------------------|------------------------------------------------------|
| FONT= <i>font</i>      | specifies software font for text                     |
| HEIGHT= <i>value</i>   | specifies height of text used outside framed areas   |
| INFONT= <i>font</i>    | specifies software font for text inside framed areas |
| INHEIGHT= <i>value</i> | specifies height of text inside framed areas         |

**Table 12.25.** Options for Reference Lines

|                                             |                                                         |
|---------------------------------------------|---------------------------------------------------------|
| HREF<(INTERSECT)><br>=value-list            | requests horizontal reference line                      |
| HREFLABELS=<br>(‘label1’, . . . , ‘labeln’) | specifies labels for HREF= lines                        |
| HREFLABPOS= <i>n</i>                        | specifies vertical position of labels for HREF= lines   |
| LHREF= <i>linetype</i>                      | specifies line style for HREF= lines                    |
| LVREF= <i>linetype</i>                      | specifies line style for VREF= lines                    |
| VREF<(INTERSECT)><br>=value-list            | requests vertical reference line                        |
| VREFLABELS=<br>(‘label1’, . . . , ‘labeln’) | specifies labels for VREF= lines                        |
| VREFLABPOS= <i>n</i>                        | specifies horizontal position of labels for VREF= lines |

**Dictionary of Options**

The following entries provide detailed descriptions of the *options* in the LPREDPLOT statement.

**ANNOTATE=SAS-data-set**

**ANNO=SAS-data-set**

specifies an ANNOTATE data set, as described in *SAS/GRAPH Software: Reference*, that enables you to add features to the lpred plot. The ANNOTATE= data set you specify in the LPREDPLOT statement is used for all plots created by the statement.

**CAXIS=color**

**CAXES=color**

specifies the color used for the axes and tick marks. This option overrides any COLOR= specifications in an AXIS statement. The default is the first color in the device color list.

**CFIT=color**

specifies the color for the fitted lpred lines. The default is the first color in the device color list.

**CFRAME=color**

**CFR=color**

specifies the color for the area enclosed by the axes and frame. This area is not shaded by default.

**CGRID=***color*

specifies the color for grid lines. The default is the first color in the device color list.

**CHREF=***color***CH=***color*

specifies the color for lines requested by the HREF= option. The default is the first color in the device color list.

**CTEXT=***color*

specifies the color for tick mark values and axis labels. The default is the color specified for the CTEXT= option in the most recent GOPTIONS statement.

**CVREF=***color***CV=***color*

specifies the color for lines requested by the VREF= option. The default is the first color in the device color list.

**DESCRIPTION=**'*string*'**DES=**'*string*'

specifies a description, up to 40 characters, that appears in the PROC GREPLAY master menu. The default is the variable name.

**FONT=***font*

specifies a software font for reference line and axis labels. You can also specify fonts for axis labels in an AXIS statement. The FONT= font takes precedence over the FTEXT= font specified in the most recent GOPTIONS statement. Hardware characters are used by default.

**HAXIS=***value1 to value2*<*by value3*>

specifies tick mark values for the horizontal axis. *value1*, *value2*, and *value3* must be numeric, and *value1* must be less than *value2*. The lower tick mark is *value1*. Tick marks are drawn at increments of *value3*. The last tick mark is the greatest value that does not exceed *value2*. If *value3* is omitted, a value of 1 is used.

Examples of HAXIS= lists are:

```
haxis = 0 to 10
```

```
haxis = 2 to 10 by 2
```

```
haxis = 0 to 200 by 10
```

**HEIGHT=***value*

specifies the height of text used outside framed areas. The default value is 3.846 (in percentage).

**HLOWER=***value*

specifies the lower limit on the horizontal axis scale. The HLOWER= option specifies *value* as the lower horizontal axis tick mark. The tick mark interval and the upper axis limit are determined automatically. This option has no effect if the HAXIS= option is used.

**HOFFSET=***value*

specifies offset for horizontal axis. The default value is 1.

**HUPPER=*value***

specifies *value* as the upper horizontal axis tick mark. The tick mark interval and the lower axis limit are determined automatically. This option has no effect if the HAXIS= option is used.

**HREF < (INTERSECT) > =*value-list***

requests reference lines perpendicular to the horizontal axis. If (INTERSECT) is specified, a second reference line perpendicular to the vertical axis is drawn that intersects the fit line at the same point as the horizontal axis reference line. If a horizontal axis reference line label is specified, the intersecting vertical axis reference line is labeled with the vertical axis value. See also the CHREF=, HREFLABELS=, and LHREF= options.

**HREFLABELS='label1',...,'labeln'****HREFLABEL='label1',...,'labeln'****HREFLAB='label1',...,'labeln'**

specifies labels for the lines requested by the HREF= option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**HREFLABPOS=*n***

specifies the vertical position of labels for HREF= lines. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement         |
|----------|-------------------------|
| 1        | top                     |
| 2        | staggered from top      |
| 3        | bottom                  |
| 4        | staggered from bottom   |
| 5        | alternating from top    |
| 6        | alternating from bottom |

**INBORDER**

requests a border around lpred plots.

**LEVEL= (*character-list*)****ORDINAL= (*character-list*)**

specifies the names of the levels for which linear predictor lines are requested. Names should be quoted and separated by space. If there is no correct name provided, no lpred line is plotted.

**LFIT=*linetype***

specifies a line style for fitted curves and confidence limits. By default, fitted curves are drawn by connecting solid lines (*linetype* = 1) and confidence limits are drawn by connecting dashed lines (*linetype* = 3).

**LGRID=*linetype***

specifies a line style for all grid lines. *linetype* is between 1 and 46. The default is 35.

**LHREF=linetype****LH=linetype**

specifies the line type for lines requested by the HREF= option. The default is 2, which produces a dashed line.

**LVREF=linetype****LV=linetype**

specifies the line type for lines requested by the VREF= option. The default is 2, which produces a dashed line.

**NAME='string'**

specifies a name for the plot, up to eight characters, that appears in the PROC GREPLAY master menu. The default is 'PROBIT'.

**NOCONF**

suppresses confidence limits from the plot. This only works for the binomial model. Confidence limits are not plotted for the multinomial model.

**NODATA**

suppresses observed data points from the plot. This only works for the binomial model. Data points are not plotted for the multinomial model.

**NOFIT**

suppresses the fitted lpred lines.

**NOFRAME**

suppresses the frame around plotting areas.

**NOGRID**

suppresses grid lines.

**NOHLABEL**

suppresses horizontal labels.

**NOHTICK**

suppresses horizontal tick marks.

**NOTHRESH**

suppresses the threshold line.

**NOVLABEL**

suppresses vertical labels.

**NOVTICK**

suppresses vertical tick marks.

**THRESHLABPOS=*n***

specifies the horizontal position of labels for the threshold line. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | left            |
| 2        | right           |



**VAXIS=***value1 to value2*<*by value3*>

specifies tick mark values for the vertical axis. *value1*, *value2*, and *value3* must be numeric, and *value1* must be less than *value2*. The lower tick mark is *value1*. Tick marks are drawn at increments of *value3*. The last tick mark is the greatest value that does not exceed *value2*. This method of specification of tick marks is not valid for logarithmic axes. If *value3* is omitted, a value of 1 is used.

Examples of VAXIS= lists are:

```
vaxis = 0 to 10
vaxis = 0 to 2 by .1
```

**VAXISLABEL=***'string'*

specifies a label for the vertical axis.

**VLOWER=***value*

specifies the lower limit on the vertical axis scale. The VLOWER= option specifies *value* as the lower vertical axis tick mark. The tick mark interval and the upper axis limit are determined automatically. This option has no effect if the VAXIS= option is used.

**VREF=***value-list*

requests reference lines perpendicular to the vertical axis. If (INTERSECT) is specified, a second reference line perpendicular to the horizontal axis is drawn that intersects the fit line at the same point as the vertical axis reference line. If a vertical axis reference line label is specified, the intersecting horizontal axis reference line is labeled with the horizontal axis value. See also the CVREF=, LVREF=, and VREFLABELS= options.

**VREFLABELS=***'label1',...,'labeln'*

**VREFLABEL=***'label1',...,'labeln'*

**VREFLAB=***'label1',...,'labeln'*

specifies labels for the lines requested by the VREF= option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**VREFLABPOS=***n*

specifies the horizontal position of labels for VREF= lines. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | left            |
| 2        | right           |

**VUPPER=***value*

specifies the upper limit on the vertical axis scale. The VUPPER= option specifies *value* as the upper vertical axis tick mark. The tick mark interval and the lower axis limit are determined automatically. This option has no effect if the VAXIS= option is used.

**WAXIS=***n*

specifies line thickness for axes and frame. The default value is 1.

**WFIT=*n***

specifies line thickness for fitted lines. The default value is 1.

**WGRID=*n***

specifies line thickness for grids. The default value is 1.

**WREFL=*n***

specifies line thickness for reference lines. The default value is 1.

---

## MODEL Statement

*<label>* **MODEL** *response=effects < / options >* ;*<label>* **MODEL** *events/trials=effects < / options >* ;

### AGGREGATE

**AGGREGATE=** (*variable-list*)

specifies the subpopulations on which the Pearson chi-square test statistic and the log-likelihood ratio chi-square test statistic (deviance) are calculated if the LACKFIT option is specified. See the section “Rescaling the Covariance Matrix” on page 280 for details of Pearson’s chi-square and deviance calculations.

Observations with common values in the given list of variables are regarded as coming from the same subpopulation. Variables in the list can be any variables in the input data set. Specifying the AGGREGATE option is equivalent to specifying the AGGREGATE= option with a variable list that includes all independent variables in the MODEL statement. The PROBIT procedure sorts the input data set according to the variables specified in this list. Information for the sorted data set is reported in the “Response-Covariate Profile” table.

The deviance and Pearson goodness-of-fit statistics are calculated if the LACKFIT option is specified in the MODEL statement. The calculated results are reported in the “Goodness-of-Fit” table. If the Pearson chi-square test is significant with the test level specified by the HPROB= option, the fiducial limits, if required with the INVERSECL option in the MODEL statement, are modified. Also, the covariance matrix is re-scaled by the dispersion parameter when the SCALE= option is specified.

**ALPHA=***value*

sets the significance level for the confidence intervals for regression parameters, fiducial limits for the predicted values, and confidence intervals for the predicted probabilities. The value must be between 0 and 1. The default value is ALPHA=0.05.

**CONVERGE=***value*

The default is changed to CONVERGE=1E–8.

**SCALE=** *scale*enables you to specify the method for estimating the dispersion parameter. To correct for overdispersion or underdispersion, the covariance matrix is multiplied by the estimate of the dispersion parameter. Valid values for *scale* are as follows:

|              |                                                                                                                                                         |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| D   DEVIANCE | specifies that the dispersion parameter be estimated by the deviance divided by its degrees of freedom.                                                 |
| P   PEARSON  | specifies that the dispersion parameter be estimated by the Pearson chi-square statistic divided by its degrees of freedom. This is set as the default. |

You can use the AGGREGATE= option to define the subpopulations for calculating the Pearson chi-square statistic and the deviance.

The “Goodness-of-Fit” table includes the Pearson chi-square statistic, the deviance, their degrees of freedom, the ratio of each statistic divided by its degrees of freedom, and the corresponding *p*-value.

---

## PREDPLOT Statement

**PREDPLOT** *<var = variable> <options>;*

The PREDPLOT statement plots the predicted probability against a single continuous variable (dose variable) in the MODEL statement for both the binomial model and the multinomial model. Confidence limits are only available for the binomial model. An attached box on the right side of the plot is used to label predicted probability curves with the names of their levels for the multinomial model. You can specify the color of this box using the CLABBOX= option.

### **VAR=** (*variable*)

specifies a single continuous variable (dose variable) in the independent variable list of the MODEL statement. If a VAR= variable is not specified, the first single continuous variable in the independent variable list of the MODEL statement is used. If such a variable does not exist in the independent variable list of the MODEL statement, an error is reported.

The predicted probability is

$$\hat{p} = C + (1 - C)F(\mathbf{x}'\hat{\mathbf{b}})$$

for the binomial model and

$$\begin{aligned}\hat{p}_1 &= C + (1 - C)F(\mathbf{x}'\hat{\mathbf{b}}) \\ \hat{p}_j &= (1 - C)(F(\hat{a}_j + \mathbf{x}'\hat{\mathbf{b}}) - F(\hat{a}_{j-1} + \mathbf{x}'\hat{\mathbf{b}})) \quad j = 2, \dots, k - 1 \\ \hat{p}_k &= (1 - C)(1 - F(\hat{a}_{k-1} + \mathbf{x}'\hat{\mathbf{b}}))\end{aligned}$$

for the multinomial model with *k* response levels, where *F* is the cumulative distribution function used to model the probability,  $\mathbf{x}'$  is the vector of the covariates,  $\hat{a}_j$  are the estimated ordinal intercepts with  $\hat{a}_1 = 0$ , *C* is the threshold parameter, either known or estimated from the model, and  $\hat{\mathbf{b}}'$  is the vector of estimated parameters.

To plot  $\hat{p}$  (or  $\hat{p}_j$ ) as a function of a continuous variable  $x_1$ , the remaining covariates  $\mathbf{x}_{-1}$  must be specified. You can use the XDATA= option to provide the values of  $\mathbf{x}_{-1}$  (see the XDATA= option in the PROC PROBIT statement for details), or use the default values that follow the rules:

- If the effect contains a continuous variable (or variables), the overall mean of this effect is used.
- If the effect is a single classification variable, the highest level of the variable is used.

### options

enable you to plot the observed data and add features to the plot.

You can use options in the PREDPLOT statement to

- superimpose specification limits
- suppress or add observed data points for the binomial model
- suppress or add confidence limits for the binomial model
- specify the levels for which predicted probability curves are requested for the multinomial model
- specify graphical enhancements (such as color or text height)

### Summary of Options

The following tables list all *options* by function. The “Dictionary of Options” on page 273 describes each option in detail.

#### PREDPLOT Options

**Table 12.26.** Plot Layout Options for PREDPLOT

|                              |                                                                                                                             |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| LEVEL= <i>character-list</i> | specifies the names of the levels for which the predicted probability curves are requested (only for the multinomial model) |
| NOCONF                       | suppresses confidence limits                                                                                                |
| NODATA                       | suppresses observed data points on the plot                                                                                 |
| NOTHRESH                     | suppresses the threshold line                                                                                               |
| THRESHLABPOS= <i>value</i>   | specifies the position for the label of the threshold line                                                                  |

## General Options

**Table 12.27.** Color Options

|                       |                                   |
|-----------------------|-----------------------------------|
| CAXIS= <i>color</i>   | specifies color for the axes      |
| CFIT= <i>color</i>    | specifies color for fitted curves |
| CFRAME= <i>color</i>  | specifies color for frame         |
| CGRID= <i>color</i>   | specifies color for grid lines    |
| CHREF= <i>color</i>   | specifies color for HREF= lines   |
| CLABBOX= <i>color</i> | specifies color for label box     |
| CTEXT= <i>color</i>   | specifies color for text          |
| CVREF= <i>color</i>   | specifies color for VREF= lines   |

**Table 12.28.** Options to Enhance Plots Produced on Graphics Devices

|                                   |                                                              |
|-----------------------------------|--------------------------------------------------------------|
| ANNOTATE=<br>SAS- <i>data-set</i> | specifies an ANNOTATE data set                               |
| INBORDER                          | requests a border around plot                                |
| LFIT= <i>linetype</i>             | specifies line style for fitted curves and confidence limits |
| LGRID= <i>linetype</i>            | specifies line style for grid lines                          |
| NOFRAME                           | suppresses the frame around plotting areas                   |
| NOGRID                            | suppresses grid lines                                        |
| NOFIT                             | suppresses fitted curves                                     |
| NOHLABEL                          | suppresses horizontal labels                                 |
| NOHTICK                           | suppresses horizontal ticks                                  |
| NOVTICK                           | suppresses vertical ticks                                    |
| TURNVLABELS                       | vertically string out characters in vertical labels          |
| WFIT= <i>n</i>                    | specifies thickness for fitted curves                        |
| WGRID= <i>n</i>                   | specifies thickness for grids                                |
| WREFL= <i>n</i>                   | specifies thickness for reference lines                      |

**Table 12.29.** Axis Options

|                                                       |                                                |
|-------------------------------------------------------|------------------------------------------------|
| HAXIS= <i>value1 to value2</i><br><by <i>value3</i> > | specifies tick mark values for horizontal axis |
| HOFFSET= <i>value</i>                                 | specifies offset for horizontal axis           |
| HLOWER= <i>value</i>                                  | specifies lower limit on horizontal axis scale |
| HUPPER= <i>value</i>                                  | specifies upper limit on horizontal axis scale |
| NHTICK= <i>n</i>                                      | specifies number of ticks for horizontal axis  |
| NVTICK= <i>n</i>                                      | specifies number of ticks for vertical axis    |
| VAXIS= <i>value1 to value2</i><br><by <i>value3</i> > | specifies tick mark values for vertical axis   |
| VAXISLABEL= <i>'label'</i>                            | specifies label for vertical axis              |
| VOFFSET= <i>value</i>                                 | specifies offset for vertical axis             |
| VLOWER= <i>value</i>                                  | specifies lower limit on vertical axis scale   |
| VUPPER= <i>value</i>                                  | specifies upper limit on vertical axis scale   |
| WAXIS= <i>n</i>                                       | specifies thickness for axis                   |

**Table 12.30.** Graphics Catalog Options

|                              |                                                   |
|------------------------------|---------------------------------------------------|
| DESCRIPTION= <i>'string'</i> | specifies description for graphics catalog member |
| NAME= <i>'string'</i>        | specifies name for plot in graphics catalog       |

**Table 12.31.** Options for Text Enhancement

|                        |                                                      |
|------------------------|------------------------------------------------------|
| FONT= <i>font</i>      | specifies software font for text                     |
| HEIGHT= <i>value</i>   | specifies height of text used outside framed areas   |
| INFONT= <i>font</i>    | specifies software font for text inside framed areas |
| INHEIGHT= <i>value</i> | specifies height of text inside framed areas         |

**Table 12.32.** Options for Reference Lines

|                                                 |                                                         |
|-------------------------------------------------|---------------------------------------------------------|
| HREF<(INTERSECT)><br>= <i>value-list</i>        | requests horizontal reference line                      |
| HREFLABELS=<br>( <i>'label1',...,'labeln'</i> ) | specifies labels for HREF= lines                        |
| HREFLABPOS= <i>n</i>                            | specifies vertical position of labels for HREF= lines   |
| LHREF= <i>linetype</i>                          | specifies line style for HREF= lines                    |
| LVREF= <i>linetype</i>                          | specifies line style for VREF= lines                    |
| VREF<(INTERSECT)><br>= <i>value-list</i>        | requests vertical reference line                        |
| VREFLABELS=<br>( <i>'label1',...,'labeln'</i> ) | specifies labels for VREF= lines                        |
| VREFLABPOS= <i>n</i>                            | specifies horizontal position of labels for VREF= lines |

## Dictionary of Options

The following entries provide detailed descriptions of the *options* in the PREDPPLOT statement.

**ANNOTATE=SAS-data-set**

**ANNO=SAS-data-set**

specifies an ANNOTATE data set, as described in *SAS/GRAPH Software: Reference*, that enables you to add features to the predicted probability plot. The ANNOTATE= data set you specify in the PREDPPLOT statement is used for all plots created by the statement.

**CAXIS=color**

**CAXES=color**

specifies the color used for the axes and tick marks. This option overrides any COLOR= specifications in an AXIS statement. The default is the first color in the device color list.

**CFIT=color**

specifies the color for the fitted predicted probability curves. The default is the first color in the device color list.

**CFRAME=color**

**CFR=color**

specifies the color for the area enclosed by the axes and frame. This area is not shaded by default.

**CGRID=color**

specifies the color for grid lines. The default is the first color in the device color list.

**CHREF=color**

**CH=color**

specifies the color for lines requested by the HREF= option. The default is the first color in the device color list.

**CTEXT=color**

specifies the color for tick mark values and axis labels. The default is the color specified for the CTEXT= option in the most recent GOPTIONS statement.

**CVREF=color**

**CV=color**

specifies the color for lines requested by the VREF= option. The default is the first color in the device color list.

**DESCRIPTION='string'**

**DES='string'**

specifies a description, up to 40 characters, that appears in the PROC GREPLAY master menu. The default is the variable name.

**FONT=font**

specifies a software font for reference line and axis labels. You can also specify fonts for axis labels in an **AXIS** statement. The **FONT= font** takes precedence over the **FTEXT= font** specified in the most recent **GOPTIONS** statement. Hardware characters are used by default.

**HAXIS=value1 to value2<by value3>**

specifies tick mark values for the horizontal axis. *value1*, *value2*, and *value3* must be numeric, and *value1* must be less than *value2*. The lower tick mark is *value1*. Tick marks are drawn at increments of *value3*. The last tick mark is the greatest value that does not exceed *value2*. If *value3* is omitted, a value of 1 is used.

Examples of **HAXIS=** lists are:

**haxis = 0 to 10**

**haxis = 2 to 10 by 2**

**haxis = 0 to 200 by 10**

**HEIGHT=value**

specifies the height of text used outside framed areas.

**HLOWER=value**

specifies the lower limit on the horizontal axis scale. The **HLOWER=** option specifies *value* as the lower horizontal axis tick mark. The tick mark interval and the upper axis limit are determined automatically. This option has no effect if the **HAXIS=** option is used.

**HOFFSET=value**

specifies the offset for the horizontal axis. The default value is 1.

**HUPPER=value**

specifies *value* as the upper horizontal axis tick mark. The tick mark interval and the lower axis limit are determined automatically. This option has no effect if the **HAXIS=** option is used.

**HREF < (INTERSECT) > =value-list**

requests reference lines perpendicular to the horizontal axis. If **(INTERSECT)** is specified, a second reference line perpendicular to the vertical axis is drawn that intersects the fit line at the same point as the horizontal axis reference line. If a horizontal axis reference line label is specified, the intersecting vertical axis reference line is labeled with the vertical axis value. See also the **CHREF=**, **HREFLABELS=**, and **LHREF=** options.

**HREFLABELS='label1', ..., 'labeln'****HREFLABEL='label1', ..., 'labeln'****HREFLAB='label1', ..., 'labeln'**

specifies labels for the lines requested by the **HREF=** option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**HREFLABPOS=n**

specifies the vertical position of labels for **HREF=** lines. The following table shows valid values for *n* and the corresponding label placements.



| <i>n</i> | <b>label placement</b>  |
|----------|-------------------------|
| 1        | top                     |
| 2        | staggered from top      |
| 3        | bottom                  |
| 4        | staggered from bottom   |
| 5        | alternating from top    |
| 6        | alternating from bottom |

**INBORDER**

requests a border around predicted probability plots.

**LEVEL=** (*character-list*)

**ORDINAL=** (*character-list*)

specifies the names of the levels for which predicted probability curves are requested. Names should be quoted and separated by space. If there is no correct name provided, no fitted probability curve is plotted.

**LFIT=***linetype*

specifies a line style for fitted curves and confidence limits. By default, fitted curves are drawn by connecting solid lines (*linetype* = 1) and confidence limits are drawn by connecting dashed lines (*linetype* = 3).

**LGRID=***linetype*

specifies a line style for all grid lines. *linetype* is between 1 and 46. The default is 35.

**LHREF=***linetype*

**LH=***linetype*

specifies the line type for lines requested by the HREF= option. The default is 2, which produces a dashed line.

**LVREF=***linetype*

**LV=***linetype*

specifies the line type for lines requested by the VREF= option. The default is 2, which produces a dashed line.

**NAME=**'*string*'

specifies a name for the plot, up to eight characters, that appears in the PROC GREPLAY master menu. The default is 'PROBIT'.

**NOCONF**

suppresses confidence limits from the plot. This only works for the binomial model. Confidence limits are not plotted for the multinomial model.

**NODATA**

suppresses observed data points from the plot. This only works for the binomial model. The data points are not plotted for the multinomial model.

**NOFIT**

suppresses the fitted predicted probability curves.

**NOFRAME**

suppresses the frame around plotting areas.

**NOGRID**

suppresses grid lines.

**NOHLABEL**

suppresses horizontal labels.

**NOHTICK**

suppresses horizontal tick marks.

**NOTHRESH**

suppresses the threshold line.

**NOVLABEL**

suppresses vertical labels.

**NOVTICK**

suppresses vertical tick marks.

**THRESLABPOS=*n***

specifies the horizontal position of labels for the threshold line. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | left            |
| 2        | right           |

**VAXIS=*value1 to value2<by value3>***

specifies tick mark values for the vertical axis. *value1*, *value2*, and *value3* must be numeric, and *value1* must be less than *value2*. The lower tick mark is *value1*. Tick marks are drawn at increments of *value3*. The last tick mark is the greatest value that does not exceed *value2*. This method of specification of tick marks is not valid for logarithmic axes. If *value3* is omitted, a value of 1 is used.

Examples of VAXIS= lists are:

```
vaxis = 0 to 10
vaxis = 0 to 2 by .1
```

**VAXISLABEL=*'string'***

specifies a label for the vertical axis.

**VLOWER=*value***

specifies the lower limit on the vertical axis scale. The VLOWER= option specifies *value* as the lower vertical axis tick mark. The tick mark interval and the upper axis limit are determined automatically. This option has no effect if the VAXIS= option is used.

**VREF=*value-list***

requests reference lines perpendicular to the vertical axis. If (INTERSECT) is specified, a second reference line perpendicular to the horizontal axis is drawn that intersects the fit line at the same point as the vertical axis reference line. If a vertical axis reference line label is specified, the intersecting horizontal axis reference line is labeled with the horizontal axis value. See also the CVREF=, LVREF=, and VREFLABELS= options.

**VREFLABELS**=*'label1',...,'labeln'*

**VREFLABEL**=*'label1',...,'labeln'*

**VREFLAB**=*'label1',...,'labeln'*

specifies labels for the lines requested by the VREF= option. The number of labels must equal the number of lines. Enclose each label in quotes. Labels can be up to 16 characters.

**VREFLABPOS**=*n*

specifies the horizontal position of labels for VREF= lines. The following table shows valid values for *n* and the corresponding label placements.

| <i>n</i> | label placement |
|----------|-----------------|
| 1        | left            |
| 2        | right           |

**VUPPER**=*value*

specifies the upper limit on the vertical axis scale. The VUPPER= option specifies *value* as the upper vertical axis tick mark. The tick mark interval and the lower axis limit are determined automatically. This option has no effect if the VAXIS= option is used.

**WAXIS**=*n*

specifies line thickness for axes and frame. The default value is 1.

**WFIT**=*n*

specifies line thickness for fitted curves. The default value is 1.

**WGRID**=*n*

specifies line thickness for grids. The default value is 1.

**WREFL**=*n*

specifies line thickness for reference lines. The default value is 1.

---

## Details

---

### INEST= SAS-data-set

The INEST= data set names a SAS data set that specifies initial estimates for all the parameters in the model.

The INEST= data set must contain the intercept variables (named Intercept for binary response model and Intercept, Intercept2, Intercept3, and so forth, for multinomial response models) and all independent variables in the MODEL statement.

If BY processing is used, the INEST= data set should also include the BY variables, and there must be at least one observation for each BY group. If there is more than one observation in one BY group, the first one read is used for that BY group.

If the INEST= data set also contains the `_TYPE_` variable, only observations with `_TYPE_` value 'PARMS' are used as starting values. Combining the INEST= data set and the option MAXIT= in the MODEL statement, partial scoring can be done, such as predicting on a validation data set by using the model built from a training data set.

You can specify starting values for the iterative algorithm in the INEST= data set. This data set overwrites the INITIAL= option in the MODEL statement, which is a little difficult to use for models with multilevel interaction effects. The INEST= data set has the same structure as the OUTEST= data set, but is not required to have all the variables or observations that appear in the OUTEST= data set. One simple use of the INEST= option is passing the previous OUTEST= data set directly to the next model as an INEST= data set, assuming that the two models have the same parameterization.

---

## Inverse Confidence Limits

In bioassay problems, estimates of the values of the independent variables that yield a desired response are often needed. For instance, the value yielding a 50% response rate (called the ED50 or LD50) is often used. The INVERSECL option requests that confidence limits be computed for the value of the independent variable that yields a specified response. These limits are computed only for the first continuous variable effect in the model. The other variables are set either at their mean values if they are continuous or at the reference (last) level if they are discrete variables. For a discussion of inverse confidence limits, refer to Hubert, Bohidar, and Peace (1988).

For the PROBIT procedure, the response variable is a probability. An estimate of the first continuous variable value needed to achieve a response of  $p$  is given by

$$\hat{x}_1 = \frac{1}{b_1} (F^{-1}(p) - \mathbf{x}^* \mathbf{b}^*)$$

where  $F$  is the cumulative distribution function used to model the probability,  $\mathbf{x}^*$  is the vector of independent variables excluding the first one, which can be specified by the XDATA= option described in the section “XDATA= SAS-data-set” on page 280,  $\mathbf{b}^*$  is the vector of parameter estimates excluding the first one, and  $b_1$  is the estimated regression coefficient for the independent variable of interest. Note that, for both binary and ordinal models, the INVERSECL option provides estimates of the value of  $x_1$  yielding  $\Pr(\text{first response level}) = p$ , for various values of  $p$ .

This estimator is given as a ratio of random variables, for example,  $r = a/b$ . Confidence limits for this ratio can be computed using Fieller’s theorem. A brief description of this theorem follows. Refer to Finney (1971) for a more complete description of Fieller’s theorem.

If the random variables  $a$  and  $b$  are thought to be distributed as jointly normal, then for any fixed value  $r$  the following probability statement holds if  $z$  is an  $\alpha/2$  quantile from the standard normal distribution and  $\mathbf{V}$  is the variance-covariance matrix of  $a$  and  $b$ .

$$\Pr((a - rb)^2 > z^2(V_{aa} - 2rV_{ab} + r^2V_{bb})) = \alpha$$

Usually the inequality can be solved for  $r$  to yield a confidence interval. The PROBIT procedure uses a value of 1.96 for  $z$ , corresponding to an  $\alpha$  value of 0.05, unless the goodness-of-fit  $p$ -value is less than the specified value of the HPROB= option. When this happens, the covariance matrix is scaled by the heterogeneity factor, and a  $t$  distribution quantile is used for  $z$ .

It is possible for the roots of the equation for  $r$  to be imaginary or for the confidence interval to be all points outside of an interval. In these cases, the limits are set to missing by the PROBIT procedure.

Although the normal and logistic distribution give comparable fitted values of  $p$  if the empirically observed proportions are not too extreme, they can give appreciably different values when extrapolated into the tails. Correspondingly, the estimates of the confidence limits and dose values can be different for the two distributions even when they agree quite well in the body of the data. Extrapolation outside of the range of the actual data are often sensitive to model assumptions, and caution is advised if extrapolation is necessary.

---

## Lack of Fit Tests

Two goodness-of-fit tests can be requested from the PROBIT procedure: a Pearson chi-square test and a log-likelihood ratio chi-square test.

To compute the test statistics, you can use the AGGREGATE or AGGREGATE= option grouping the observations into subpopulations. If neither AGGREGATE nor AGGREGATE= is specified, PROC PROBIT assumes that each observation is from a separate subpopulation and computes the goodness-of-fit test statistics only for the *events/trials* syntax.

If the Pearson goodness-of-fit chi-square test is requested and the  $p$ -value for the test is too small, variances and covariances are adjusted by a heterogeneity factor (the goodness-of-fit chi-square divided by its degrees of freedom) and a critical value from the  $t$  distribution is used to compute the fiducial limits. The Pearson chi-square test statistic is computed as

$$\chi_P^2 = \sum_{i=1}^m \sum_{j=1}^k \frac{(r_{ij} - n_i \hat{p}_{ij})^2}{n_i \hat{p}_{ij}}$$

where the sum on  $i$  is over grouping, the sum on  $j$  is over levels of response, the  $r_{ij}$  is the frequency of response level  $j$  for the  $i$ th grouping,  $n_i$  is the total frequency for the  $i$ th grouping, and  $\hat{p}_{ij}$  is the fitted probability for the  $j$ th level at the  $i$ th grouping.

The likelihood ratio chi-square test statistic is computed as

$$\chi_D^2 = 2 \sum_{i=1}^m \sum_{j=1}^k r_{ij} \ln \left( \frac{r_{ij}}{n_i \hat{p}_{ij}} \right)$$

This quantity is sometimes called the deviance. If the modeled probabilities fit the data, these statistics should be approximately distributed as chi-square with degrees of freedom equal to  $(k-1) \times m - q$ , where  $k$  is the number of levels of the multinomial or binomial response,  $m$  is the number of sets of independent variable values (covariate patterns), and  $q$  is the number of parameters fit in the model.

In order for the Pearson statistic and the deviance to be distributed as chi-square, there must be sufficient replication within the groupings. When this is not true, the data are

sparse, and the  $p$ -values for these statistics are not valid and should be ignored. Similarly, these statistics, divided by their degrees of freedom, cannot serve as indicators of overdispersion. A large difference between the Pearson statistic and the deviance provides some evidence that the data are too sparse to use either statistic.

---

## Rescaling the Covariance Matrix

One way of correcting overdispersion is to multiply the covariance matrix by a dispersion parameter. You can supply the value of the dispersion parameter directly, or you can estimate the dispersion parameter based on either the Pearson chi-square statistic or the deviance for the fitted model.

The Pearson chi-square statistic  $\chi_P^2$  and the deviance  $\chi_D^2$  are defined in the section “Lack of Fit Tests” on page 279. If the SCALE= option is specified in the MODEL statement, the dispersion parameter is estimated by

$$\hat{\sigma}^2 = \begin{cases} \chi_P^2 / (m(k-1) - q) & \text{SCALE=PEARSON} \\ \chi_D^2 / (m(k-1) - q) & \text{SCALE=DEVIANE} \\ (constant)^2 & \text{SCALE=constant} \end{cases}$$

In order for the Pearson statistic and the deviance to be distributed as chi-square, there must be sufficient replication within the subpopulations. When this is not true, the data are sparse, and the  $p$ -values for these statistics are not valid and should be ignored. Similarly, these statistics, divided by their degrees of freedom, cannot serve as indicators of overdispersion. A large difference between the Pearson statistic and the deviance provides some evidence that the data are too sparse to use either statistic.

You can use the AGGREGATE (or AGGREGATE=) option to define the subpopulation profiles. If you do not specify this option, each observation is regarded as coming from a separate subpopulation. For *events/trials* syntax, each observation represents  $n$  Bernoulli trials, where  $n$  is the value of the *trials* variable; for *single-trial* syntax, each observation represents a single trial. Without the AGGREGATE (or AGGREGATE=) option, the Pearson chi-square statistic and the deviance are calculated only for *events/trials* syntax.

Note that the parameter estimates are not changed by this method. However, their standard errors are adjusted for overdispersion, affecting their significance tests.

---

## XDATA= SAS-data-set

The XDATA= data set is used for specifying values for the effects in the MODEL statement when predicted values and/or fiducial limits for a single continuous variable (dose variable) are required. It is also used for plots specified by the CDFPLOT, IPPLOT, LPREDPLOT, and PREDPLOT statement.

The XDATA= data names a SAS data set that contains user input values for all the independent variables in the MODEL statement and the variables in the CLASS statement. The XDATA= data set has the same structure as the DATA= data set but is not required to have all the variables or observations that appear in the DATA= data set.

The XDATA= data set must contain all the independent variables in the MODEL statement and the variables in the CLASS statement. Even though variables in the CLASS statement may not be used in the MODEL statement, valid values are required for these variables in the XDATA= data set. Missing values are not allowed. For independent variables in the MODEL statement, although the dose variable's value is not used in the computing of predicted values and/or fiducial limits for the dose variable, missing values are not allowed in the XDATA= data set for any of the independent variables. Missing values are allowed for the dependent variables and other variables if they are included in the XDATA= data set and not listed in the CLASS statement.

If BY processing is used, the XDATA= data set should also include the BY variables, and there must be at least one valid observation for each BY group. If there is more than one valid observation in a BY group, the last one read is used for that BY group.

If there is no XDATA= data set in the PROC PROBIT statement, by default, the PROBIT procedure will use overall mean for effects containing a continuous variable (or variables) and the highest level of a single classification variable as the reference level. The rules are summarized as follows:

- If the effect contains a continuous variable (or variables), the overall mean of this effect is used.
- If the effect is a single classification variable, the highest level of the variable is used.

---

## Examples

---

### Example 12.1. Multilevel Response

In this example, two preparations, a standard preparation and a test preparation, are each given at several dose levels to groups of insects. The symptoms are recorded for each insect within each group, and two multilevel probit models are fit. Because the natural sort order of the three levels is not the same as the response order, the ORDER=DATA option is specified in the PROC PROBIT statement to get the desired order.

The following statements produce Output 12.1.1:

```
data multi;
  input Prep $ Dose Symptoms $ N;
  LDose=log10(Dose);
  if Prep='test' then PrepDose=LDose;
  else PrepDose=0;
  datalines;
stand    10    None    33
stand    10    Mild    7
stand    10    Severe   10
stand    20    None    17
```

|       |    |        |    |
|-------|----|--------|----|
| stand | 20 | Mild   | 13 |
| stand | 20 | Severe | 17 |
| stand | 30 | None   | 14 |
| stand | 30 | Mild   | 3  |
| stand | 30 | Severe | 28 |
| stand | 40 | None   | 9  |
| stand | 40 | Mild   | 8  |
| stand | 40 | Severe | 32 |
| test  | 10 | None   | 44 |
| test  | 10 | Mild   | 6  |
| test  | 10 | Severe | 0  |
| test  | 20 | None   | 32 |
| test  | 20 | Mild   | 10 |
| test  | 20 | Severe | 12 |
| test  | 30 | None   | 23 |
| test  | 30 | Mild   | 7  |
| test  | 30 | Severe | 21 |
| test  | 40 | None   | 16 |
| test  | 40 | Mild   | 6  |
| test  | 40 | Severe | 19 |

```

;

proc probit order=data;
  class Prep Symptoms;
  nonpara: model Symptoms=Prep LDose PrepDose / lackfit;
  weight N;
  title 'Probit Models for Symptom Severity';
run;

proc probit order=data;
  class Prep Symptoms;
  parallel: model Symptoms=Prep LDose / lackfit;
  weight N;
  title 'Probit Models for Symptom Severity';
run;

```

The first model allows for nonparallelism between the dose response curves for the two preparations by inclusion of an interaction between Prep and LDose. The interaction term is labeled PrepDose in the “Analysis of Parameter Estimates” table. The results of this first model indicate that the parameter for the interaction term is not significant, having a Wald chi-square of 0.73. Also, since the first model is a generalization of the second, a likelihood ratio test statistic for this same parameter can be obtained by multiplying the difference in log likelihoods between the two models by 2. The value obtained,  $2 \times (-345.94 - (-346.31))$ , is 0.73. This is in close agreement with the Wald chi-square from the first model. The lack-of-fit test statistics for the two models do not indicate a problem with either fit.



Output 12.1.1. Multilevel Response: PROC PROBIT

| Probit Models for Symptom Severity |        |                  |
|------------------------------------|--------|------------------|
| Probit Procedure                   |        |                  |
| Class Level Information            |        |                  |
| Name                               | Levels | Values           |
| Prep                               | 2      | stand test       |
| Symptoms                           | 3      | None Mild Severe |

| Probit Models for Symptom Severity |  |              |
|------------------------------------|--|--------------|
| Probit Procedure                   |  |              |
| Model Information                  |  |              |
| Data Set                           |  | WORK.MULTI   |
| Dependent Variable                 |  | Symptoms     |
| Weight Variable                    |  | N            |
| Number of Observations             |  | 23           |
| Missing Values                     |  | 1            |
| Name of Distribution               |  | Normal       |
| Log Likelihood                     |  | -345.9401767 |

| Probit Models for Symptom Severity |       |          |                |                       |         |            |            |        |
|------------------------------------|-------|----------|----------------|-----------------------|---------|------------|------------|--------|
| Probit Procedure                   |       |          |                |                       |         |            |            |        |
| Analysis of Parameter Estimates    |       |          |                |                       |         |            |            |        |
| Parameter                          | DF    | Estimate | Standard Error | 95% Confidence Limits |         | Chi-Square | Pr > ChiSq |        |
| Intercept                          | 1     | 3.8080   | 0.6252         | 2.5827                | 5.0333  | 37.10      | <.0001     |        |
| Intercept2                         | 1     | 0.4684   | 0.0559         | 0.3589                | 0.5780  | 70.19      | <.0001     |        |
| Prep                               | stand | 1        | -1.2573        | 0.8190                | -2.8624 | 0.3479     | 2.36       | 0.1247 |
| Prep                               | test  | 0        | 0.0000         | 0.0000                | 0.0000  | .          | .          |        |
| LDose                              | 1     | -2.1512  | 0.3909         | -2.9173               | -1.3851 | 30.29      | <.0001     |        |
| PrepDose                           | 1     | -0.5072  | 0.5945         | -1.6724               | 0.6580  | 0.73       | 0.3935     |        |

| Probit Models for Symptom Severity |        |                  |
|------------------------------------|--------|------------------|
| Probit Procedure                   |        |                  |
| Class Level Information            |        |                  |
| Name                               | Levels | Values           |
| Prep                               | 2      | stand test       |
| Symptoms                           | 3      | None Mild Severe |

| Probit Models for Symptom Severity |             |
|------------------------------------|-------------|
| Probit Procedure                   |             |
| Model Information                  |             |
| Data Set                           | WORK.MULTI  |
| Dependent Variable                 | Symptoms    |
| Weight Variable                    | N           |
| Number of Observations             | 23          |
| Missing Values                     | 1           |
| Name of Distribution               | Normal      |
| Log Likelihood                     | -346.306141 |

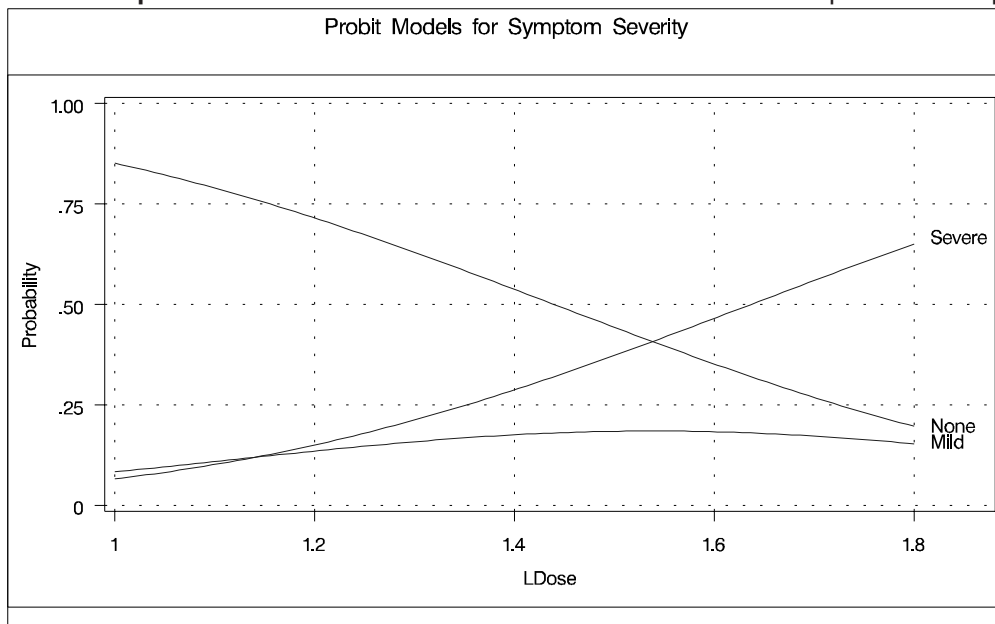
| Probit Models for Symptom Severity |    |          |                |                       |         |            |            |
|------------------------------------|----|----------|----------------|-----------------------|---------|------------|------------|
| Probit Procedure                   |    |          |                |                       |         |            |            |
| Analysis of Parameter Estimates    |    |          |                |                       |         |            |            |
| Parameter                          | DF | Estimate | Standard Error | 95% Confidence Limits |         | Chi-Square | Pr > ChiSq |
| Intercept                          | 1  | 3.4148   | 0.4126         | 2.6061                | 4.2235  | 68.50      | <.0001     |
| Intercept2                         | 1  | 0.4678   | 0.0558         | 0.3584                | 0.5772  | 70.19      | <.0001     |
| Prep stand                         | 1  | -0.5675  | 0.1259         | -0.8142               | -0.3208 | 20.33      | <.0001     |
| Prep test                          | 0  | 0.0000   | 0.0000         | 0.0000                | 0.0000  | .          | .          |
| LDose                              | 1  | -2.3721  | 0.2949         | -2.9502               | -1.7940 | 64.68      | <.0001     |

The negative coefficient associated with LDose indicates that the probability of having no symptoms (Symptoms='None') or no or mild symptoms (Symptoms='None' or Symptoms='Mild') decreases as LDose increases; that is, the probability of a severe symptom increases with LDose. This association is apparent for both treatment groups.

The negative coefficient associated with the standard treatment group (Prep = stand) indicates that the standard treatment is associated with more severe symptoms across all LDose values.

The following statements use the PREDPPLOT statement to create the plot shown in Output 12.1.2. It plots the probabilities of the response taking on individual levels as a function of LDose. Since there are two covariates, LDose and Prep, the value of the CLASS variable Prep is fixed at the highest level, test. Although not shown here, the CDFPLOT statement creates similar plots of the cumulative response probabilities, instead of individual response level probabilities.

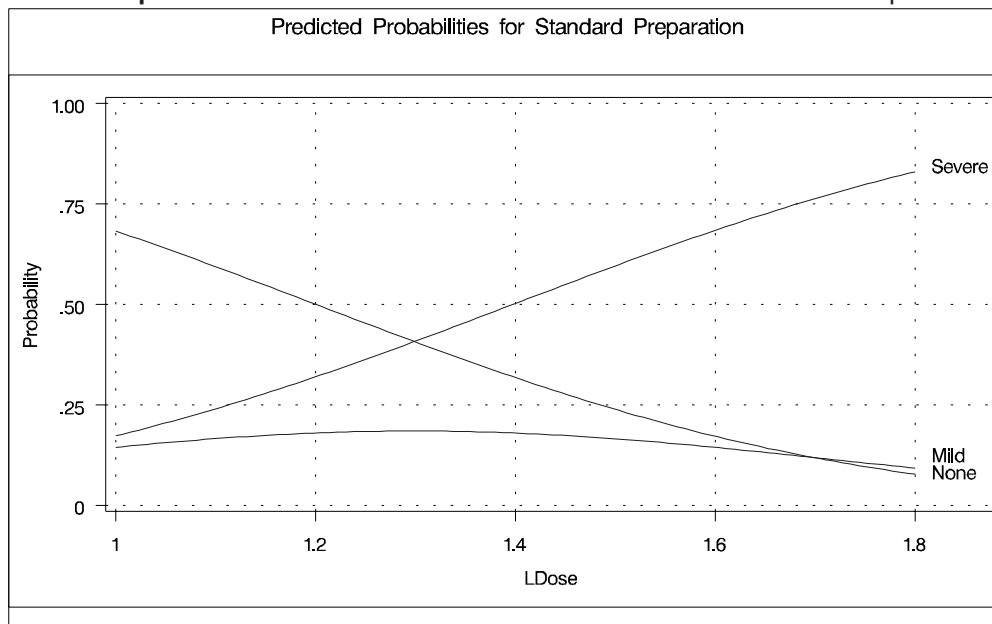
```
proc probit data=multi order=data;
  class Prep Symptoms;
  parallel: model Symptoms=Prep LDose / lackfit;
  predpplot var=ldose level=("None" "Mild" "Severe")
             cfit=blue cframe=ligr inborder noconf ;
  weight N;
  title 'Probit Models for Symptom Severity';
run;
```

**Output 12.1.2.** Plot of Predicted Probabilities for the Test Preparation Group

The following statements use the XDATA= data set to create a plot of the predicted probabilities with Prep set to the stand level. The resulting plot is shown in Output 12.1.3.

```
data xrow;
  input Prep $ Dose Symptoms $ N;
  LDose=log10(Dose);
  datalines;
stand      40      Severe      32
run;

proc probit data=multi order=data xdata=xrow;
  class Prep Symptoms;
  parallel: model Symptoms=Prep LDose / lackfit;
  predpplot var=ldose level=("None" "Mild" "Severe")
             cfit=blue cframe=ligr inborder noconf ;
  weight N;
  title 'Predicted Probabilities for Standard Preparation';
run;
```

**Output 12.1.3.** Plot of Predicted Probabilities for the Standard Preparation Group

## Example 12.2. An Epidemiology Study

The data, which is from an epidemiology study, consists of five variables: the number,  $r$ , of individuals surviving after an epidemic, out of  $n$  treated, for combinations of medicine dosage (`dose`), treatment (`treat = A, B`), and sex (`sex = 0(Female), 1(Male)`).

To see if the two treatments have different effects on male and female individual survival rate, the interaction term between the two variables `treat` and `sex` is included in the model.

The following invocation of PROC PROBIT fits the binary probit model to the grouped data:

```
data epidemic;
  input treat$ dose n r sex;
  label dose = Dose;
  datalines;
A 2.17 142 142 0
A .57 132 47 1
A 1.68 128 105 1
A 1.08 126 100 0
A 1.79 125 118 0
B 1.66 117 115 1
B 1.49 127 114 0
B 1.17 51 44 1
B 2.00 127 126 0
B .80 129 100 1
;
```

```

data xval;
  input treat $ dose sex ;
  datalines;
B 2. 1
;

title 'Epidemiology Study';
symbol v=dot c=white;
proc probit optc lackfit covout data = epidemic
  outest = out1 xdata = xval;

  class treat sex;
  model r/n = dose treat sex sex*treat/corrb covb inversec1;
  output out = out2 p =p;

  predpplot
    var = dose
    font = swiss
    vref(intersect) = .6667
    vreflab = 'two thirds'
    vreflabpos = 2
    cfit=blue
    cframe=ligr
    ;
  inset /
    cfill = white
    ctext = blue
    pos = se ;

  ippplot
    font = swiss
    href(intersect) = .75
    hreflab = 'three quarters'
    vreflabpos = 2
    threshlabpos = 2
    cfit=blue
    cframe=ligr
    ;
  inset /
    cfill = white
    ctext = blue;

  lpredplot
    font = swiss
    vref(intersect) = 1.
    vreflab = 'unit probit'
    vreflabpos = 2
    cfit=blue
    cframe=ligr
    ;
  inset /
    cfill = white
    ctext = blue;

run;

```

The results of this analysis are shown in the following tables and figures.

Beginning with SAS Release 8.2, the PROBIT procedure does not support multiple MODEL statements. Only the last one is used if there is more than one MODEL statement in one invocation of the PROBIT procedure.

**Output 12.2.1.** Class Level Information

| Epidemiology Study      |        |        |
|-------------------------|--------|--------|
| Probit Procedure        |        |        |
| Class Level Information |        |        |
| Name                    | Levels | Values |
| treat                   | 2      | A B    |
| sex                     | 2      | 0 1    |

Output 12.2.1 displays the table of level information for *all* classification variables in the CLASS statement.

**Output 12.2.2.** Parameter Information

| Epidemiology Study    |           |       |     |
|-----------------------|-----------|-------|-----|
| Probit Procedure      |           |       |     |
| Parameter Information |           |       |     |
| Parameter             | Effect    | treat | sex |
| Intercept             | Intercept |       |     |
| dose                  | dose      |       |     |
| treatA                | treat     | A     |     |
| treatB                | treat     | B     |     |
| sex0                  | sex       |       | 0   |
| sex1                  | sex       |       | 1   |
| treatAsex0            | treat*sex | A     | 0   |
| treatAsex1            | treat*sex | A     | 1   |
| treatBsex0            | treat*sex | B     | 0   |
| treatBsex1            | treat*sex | B     | 1   |

Output 12.2.2 displays the table of parameter information for the effects in the MODEL statement. The name of a parameter is generated from combining the variable names and level names in the effect. The maximum length of a parameter name is 32. The name of the effects are specified in the MODEL statement. The length of names of effects can be specified by the NAMELEN= option in the PROC PROBIT statement, with the default length 20.

**Output 12.2.3.** Model Information

| Epidemiology Study     |               |
|------------------------|---------------|
| Probit Procedure       |               |
| Model Information      |               |
| Data Set               | WORK.EPIDEMIC |
| Events Variable        | r             |
| Trials Variable        | n             |
| Number of Observations | 10            |
| Number of Events       | 1011          |
| Number of Trials       | 1204          |
| Name of Distribution   | Normal        |
| Log Likelihood         | -387.2467391  |

Algorithm converged.

Output 12.2.3 displays background information about the model fit. Included are the name of the input data set, the response variables used, and the number of observations, events, and trials. The table also includes the status of the convergence of the model fitting algorithm and the final value of log-likelihood function.

**Output 12.2.4.** Goodness-of-Fit Tests and Response-Covariate Profile

| Epidemiology Study    |        |    |            |
|-----------------------|--------|----|------------|
| Probit Procedure      |        |    |            |
| Goodness-of-Fit Tests |        |    |            |
| Statistic             | Value  | DF | Pr > ChiSq |
| Pearson Chi-Square    | 4.9317 | 4  | 0.2944     |
| L.R. Chi-Square       | 5.7079 | 4  | 0.2220     |

| Response-Covariate Profile |    |
|----------------------------|----|
| Response Levels            | 2  |
| Number of Covariate Values | 10 |

Output 12.2.4 displays the table of goodness-of-fit tests requested with the LACKFIT option in the PROC PROBIT statement. Two goodness-of-fit statistics, the Pearson chi-square statistic and the likelihood ratio chi-square statistic, are computed. The grouping method for computing these statistics can be specified by the AGGREGATE= option. The details can be found in the AGGREGATE= option and an example can be found in the second part of this example. By default, the PROBIT procedure uses the covariates in the MODEL statement to do grouping. Observations with the same values of the covariates in the MODEL statement are grouped into cells, and the two statistics are computed according to these cells. The total number of cells and the number of levels for the response variable are reported next in the “Response-Covariate Profile.”

In this example, neither the Pearson chi-square nor the log-likelihood ratio chi-square tests are significant at the 0.1 level, which is the default test level used by the PROBIT

procedure. That means that the model, which includes the interaction of `treat` and `sex`, is suitable for this epidemiology data set. (Further investigation shows that models without the interaction of `treat` and `sex` are not acceptable by either test.)

**Output 12.2.5.** Type III Tests

| Epidemiology Study           |    |                    |            |
|------------------------------|----|--------------------|------------|
| Probit Procedure             |    |                    |            |
| Type III Analysis of Effects |    |                    |            |
| Effect                       | DF | Wald<br>Chi-Square | Pr > ChiSq |
| dose                         | 1  | 42.1691            | <.0001     |
| treat                        | 1  | 16.1421            | <.0001     |
| sex                          | 1  | 1.7710             | 0.1833     |
| treat*sex                    | 1  | 13.9343            | 0.0002     |

Output 12.2.5 displays the Type III test results for all effects specified in the MODEL statement, which include the degrees of freedom for the effect, the Wald Chi-Square test statistic, and the *p*-value.

**Output 12.2.6.** Analysis of Parameter Estimates

| Epidemiology Study              |       |          |                   |                          |         |                |            |
|---------------------------------|-------|----------|-------------------|--------------------------|---------|----------------|------------|
| Probit Procedure                |       |          |                   |                          |         |                |            |
| Analysis of Parameter Estimates |       |          |                   |                          |         |                |            |
| Parameter                       | DF    | Estimate | Standard<br>Error | 95% Confidence<br>Limits |         | Chi-<br>Square | Pr > ChiSq |
| Intercept                       | 1     | -0.8871  | 0.3632            | -1.5991                  | -0.1752 | 5.96           | 0.0146     |
| dose                            | 1     | 1.6774   | 0.2583            | 1.1711                   | 2.1837  | 42.17          | <.0001     |
| treat                           | A 1   | -1.2537  | 0.2616            | -1.7664                  | -0.7410 | 22.97          | <.0001     |
| treat                           | B 0   | 0.0000   | 0.0000            | 0.0000                   | 0.0000  | .              | .          |
| sex                             | 0 1   | -0.4633  | 0.2289            | -0.9119                  | -0.0147 | 4.10           | 0.0429     |
| sex                             | 1 0   | 0.0000   | 0.0000            | 0.0000                   | 0.0000  | .              | .          |
| treat*sex                       | A 0 1 | 1.2899   | 0.3456            | 0.6126                   | 1.9672  | 13.93          | 0.0002     |
| treat*sex                       | A 1 0 | 0.0000   | 0.0000            | 0.0000                   | 0.0000  | .              | .          |
| treat*sex                       | B 0 0 | 0.0000   | 0.0000            | 0.0000                   | 0.0000  | .              | .          |
| treat*sex                       | B 1 0 | 0.0000   | 0.0000            | 0.0000                   | 0.0000  | .              | .          |
| _C_                             | 1     | 0.2735   | 0.0946            | 0.0881                   | 0.4589  |                |            |

Output 12.2.6 displays the table of parameter estimates for the model. The PROBIT procedure displays information for all the parameters of an effect. Degenerate parameters are indicated by 0 degree of freedom. Confidence intervals are computed for all parameters with non-zero degrees of freedom, including the natural threshold C if the OPTC option is specified in the PROC PROBIT statement. The confidence level can be specified by the ALPHA= option in the MODEL statement. The default confidence level is 95%.

From this table, you can see the following results:

- dose has significant positive effect on the survival rate.



- Individuals under treatment A have a lower survival rate.
- Male individuals have a higher survival rate.
- Female individuals under treatment A have a higher survival rate.

**Output 12.2.7.** Estimated Covariance Matrix

| Epidemiology Study          |           |           |           |           |            |
|-----------------------------|-----------|-----------|-----------|-----------|------------|
| Probit Procedure            |           |           |           |           |            |
| Estimated Covariance Matrix |           |           |           |           |            |
|                             | Intercept | dose      | treatA    | sex0      | treatAsex0 |
| Intercept                   | 0.131944  | -0.087353 | 0.053551  | 0.030285  | -0.067056  |
| dose                        | -0.087353 | 0.066723  | -0.047506 | -0.034081 | 0.058620   |
| treatA                      | 0.053551  | -0.047506 | 0.068425  | 0.036063  | -0.075323  |
| sex0                        | 0.030285  | -0.034081 | 0.036063  | 0.052383  | -0.063599  |
| treatAsex0                  | -0.067056 | 0.058620  | -0.075323 | -0.063599 | 0.119408   |
| _C_                         | -0.028073 | 0.018196  | -0.017084 | -0.008088 | 0.019134   |
| Estimated Covariance Matrix |           |           |           |           |            |
|                             | _C_       |           |           |           |            |
| Intercept                   | -0.028073 |           |           |           |            |
| dose                        | 0.018196  |           |           |           |            |
| treatA                      | -0.017084 |           |           |           |            |
| sex0                        | -0.008088 |           |           |           |            |
| treatAsex0                  | 0.019134  |           |           |           |            |
| _C_                         | 0.008948  |           |           |           |            |

**Output 12.2.8.** Estimated Correlation Matrix

| Epidemiology Study           |           |           |           |           |            |
|------------------------------|-----------|-----------|-----------|-----------|------------|
| Probit Procedure             |           |           |           |           |            |
| Estimated Correlation Matrix |           |           |           |           |            |
|                              | Intercept | dose      | treatA    | sex0      | treatAsex0 |
| Intercept                    | 1.000000  | -0.930998 | 0.563595  | 0.364284  | -0.534227  |
| dose                         | -0.930998 | 1.000000  | -0.703083 | -0.576477 | 0.656744   |
| treatA                       | 0.563595  | -0.703083 | 1.000000  | 0.602359  | -0.833299  |
| sex0                         | 0.364284  | -0.576477 | 0.602359  | 1.000000  | -0.804154  |
| treatAsex0                   | -0.534227 | 0.656744  | -0.833299 | -0.804154 | 1.000000   |
| _C_                          | -0.817027 | 0.744699  | -0.690420 | -0.373565 | 0.585364   |
| Estimated Correlation Matrix |           |           |           |           |            |
|                              | _C_       |           |           |           |            |
| Intercept                    | -0.817027 |           |           |           |            |
| dose                         | 0.744699  |           |           |           |            |
| treatA                       | -0.690420 |           |           |           |            |
| sex0                         | -0.373565 |           |           |           |            |
| treatAsex0                   | 0.585364  |           |           |           |            |
| _C_                          | 1.000000  |           |           |           |            |

Output 12.2.7 and Output 12.2.8 display tables of estimated covariance matrix and estimated correlation matrix for estimated parameters with a non-zero degree of freedom, respectively. They are computed by the inverse of the Hessian matrix of the estimated parameters.

**Output 12.2.9.** Probit Analysis on Dose

| Epidemiology Study      |          |                     |          |  |
|-------------------------|----------|---------------------|----------|--|
| Probit Procedure        |          |                     |          |  |
| Probit Analysis on dose |          |                     |          |  |
| Probability             | dose     | 95% Fiducial Limits |          |  |
| 0.01                    | -0.85801 | -1.81301            | -0.33743 |  |
| 0.02                    | -0.69549 | -1.58167            | -0.21116 |  |
| 0.03                    | -0.59238 | -1.43501            | -0.13093 |  |
| 0.04                    | -0.51482 | -1.32476            | -0.07050 |  |
| 0.05                    | -0.45172 | -1.23513            | -0.02130 |  |
| 0.06                    | -0.39802 | -1.15888            | 0.02063  |  |
| 0.07                    | -0.35093 | -1.09206            | 0.05742  |  |
| 0.08                    | -0.30877 | -1.03226            | 0.09039  |  |
| 0.09                    | -0.27043 | -0.97790            | 0.12040  |  |
| 0.10                    | -0.23513 | -0.92788            | 0.14805  |  |
| 0.15                    | -0.08900 | -0.72107            | 0.26278  |  |
| 0.20                    | 0.02714  | -0.55706            | 0.35434  |  |
| 0.25                    | 0.12678  | -0.41669            | 0.43322  |  |
| 0.30                    | 0.21625  | -0.29095            | 0.50437  |  |
| 0.35                    | 0.29917  | -0.17477            | 0.57064  |  |
| 0.40                    | 0.37785  | -0.06487            | 0.63387  |  |
| 0.45                    | 0.45397  | 0.04104             | 0.69546  |  |
| 0.50                    | 0.52888  | 0.14481             | 0.75654  |  |
| 0.55                    | 0.60380  | 0.24800             | 0.81819  |  |
| 0.60                    | 0.67992  | 0.35213             | 0.88157  |  |
| 0.65                    | 0.75860  | 0.45879             | 0.94803  |  |
| 0.70                    | 0.84151  | 0.56985             | 1.01942  |  |
| 0.75                    | 0.93099  | 0.68770             | 1.09847  |  |
| 0.80                    | 1.03063  | 0.81571             | 1.18970  |  |
| 0.85                    | 1.14677  | 0.95926             | 1.30171  |  |
| 0.90                    | 1.29290  | 1.12867             | 1.45386  |  |
| 0.91                    | 1.32819  | 1.16747             | 1.49273  |  |
| 0.92                    | 1.36654  | 1.20867             | 1.53590  |  |
| 0.93                    | 1.40870  | 1.25284             | 1.58450  |  |
| 0.94                    | 1.45579  | 1.30084             | 1.64012  |  |
| 0.95                    | 1.50949  | 1.35397             | 1.70515  |  |
| 0.96                    | 1.57258  | 1.41443             | 1.78353  |  |
| 0.97                    | 1.65015  | 1.48626             | 1.88238  |  |
| 0.98                    | 1.75326  | 1.57833             | 2.01720  |  |
| 0.99                    | 1.91577  | 1.71776             | 2.23537  |  |

Output 12.2.9 displays the computed values and fiducial limits for the first single continuous variable `dose` in the MODEL statement, given the probability levels, without the effect of the natural threshold, and when the option `INVERSECL` in the MODEL statement is specified. If there is no single continuous variable in the MODEL specification but the `INVERSECL` option is specified, an error is reported. If the `XDATA=` option is used to input a data set for the independent variables in the MODEL statement, the PROBIT procedure uses these values for the independent variables other than the single continuous variable. Missing values are not permitted in the `XDATA=` data set for the independent variables, although the value for the single continuous variable is not used in the computing of the fiducial limits. A suitable valid value

should be given. In the data set `xval` created by the SAS statements on page 286, `Dose = 2`.

See the section “`XDATA= SAS-data-set`” on page 280 for the default values for those effects other than the single continuous variable, for which the fiducial limits are computed.

In this example, there are two classification variables, `treat` and `sex`. Fiducial limits for the `dose` variable are computed for the highest level of the classification variables, `treat = B` and `sex = 1`, which is the default specification. Since these are the default values, you would get the same values and fiducial limits if you did not specify the `XDATA=` option in this example. The confidence level for the fiducial limits can be specified by the `ALPHA=` option in the `MODEL` statement. The default value is 0.05, resulting in 95% confidence limits.

If a `LOG10` or `LOG` option is used in the `PROC PROBIT` statement, the values and the fiducial limits are computed for both the single continuous variable and its logarithm.

**Output 12.2.10.** Outest Data Set for Epidemiology Study

| Obs | _MODEL_    | _NAME_ | _TYPE_ | _DIST_ | _STATUS_  | _LNLIKE_ | r        | Intercept |
|-----|------------|--------|--------|--------|-----------|----------|----------|-----------|
| 1   | r          | PARMS  | Normal | 0      | Converged | -387.247 | -1.00000 | -0.88714  |
| 2   | Intercept  | COV    | Normal | 0      | Converged | -387.247 | -0.88714 | 0.13194   |
| 3   | dose       | COV    | Normal | 0      | Converged | -387.247 | 1.67739  | -0.08735  |
| 4   | treatA     | COV    | Normal | 0      | Converged | -387.247 | -1.25367 | 0.05355   |
| 5   | treatB     | COV    | Normal | 0      | Converged | -387.247 | 0.00000  | 0.00000   |
| 6   | sex0       | COV    | Normal | 0      | Converged | -387.247 | -0.46329 | 0.03029   |
| 7   | sex1       | COV    | Normal | 0      | Converged | -387.247 | 0.00000  | 0.00000   |
| 8   | treatAsex0 | COV    | Normal | 0      | Converged | -387.247 | 1.28991  | -0.06706  |
| 9   | treatAsex1 | COV    | Normal | 0      | Converged | -387.247 | 0.00000  | 0.00000   |
| 10  | treatBsex0 | COV    | Normal | 0      | Converged | -387.247 | 0.00000  | 0.00000   |
| 11  | treatBsex1 | COV    | Normal | 0      | Converged | -387.247 | 0.00000  | 0.00000   |
| 12  | _C_        | COV    | Normal | 0      | Converged | -387.247 | 0.27347  | -0.02807  |

| Obs | dose     | treatA   | treatB | sex0     | sex1 | treatAsex0 | treatAsex1 | treatBsex0 | treatBsex1 | _C_      |
|-----|----------|----------|--------|----------|------|------------|------------|------------|------------|----------|
| 1   | 1.67739  | -1.25367 | 0      | -0.46329 | 0    | 1.28991    | 0          | 0          | 0          | 0.27347  |
| 2   | -0.08735 | 0.05355  | 0      | 0.03029  | 0    | -0.06706   | 0          | 0          | 0          | -0.02807 |
| 3   | 0.06672  | -0.04751 | 0      | -0.03408 | 0    | 0.05862    | 0          | 0          | 0          | 0.01820  |
| 4   | -0.04751 | 0.06843  | 0      | 0.03606  | 0    | -0.07532   | 0          | 0          | 0          | -0.01708 |
| 5   | 0.00000  | 0.00000  | 0      | 0.00000  | 0    | 0.00000    | 0          | 0          | 0          | 0.00000  |
| 6   | -0.03408 | 0.03606  | 0      | 0.05238  | 0    | -0.06360   | 0          | 0          | 0          | -0.00809 |
| 7   | 0.00000  | 0.00000  | 0      | 0.00000  | 0    | 0.00000    | 0          | 0          | 0          | 0.00000  |
| 8   | 0.05862  | -0.07532 | 0      | -0.06360 | 0    | 0.11941    | 0          | 0          | 0          | 0.01913  |
| 9   | 0.00000  | 0.00000  | 0      | 0.00000  | 0    | 0.00000    | 0          | 0          | 0          | 0.00000  |
| 10  | 0.00000  | 0.00000  | 0      | 0.00000  | 0    | 0.00000    | 0          | 0          | 0          | 0.00000  |
| 11  | 0.00000  | 0.00000  | 0      | 0.00000  | 0    | 0.00000    | 0          | 0          | 0          | 0.00000  |
| 12  | 0.01820  | -0.01708 | 0      | -0.00809 | 0    | 0.01913    | 0          | 0          | 0          | 0.00895  |

Output 12.2.10 displays the `OUTEST=` data set. All parameters for an effect are included.

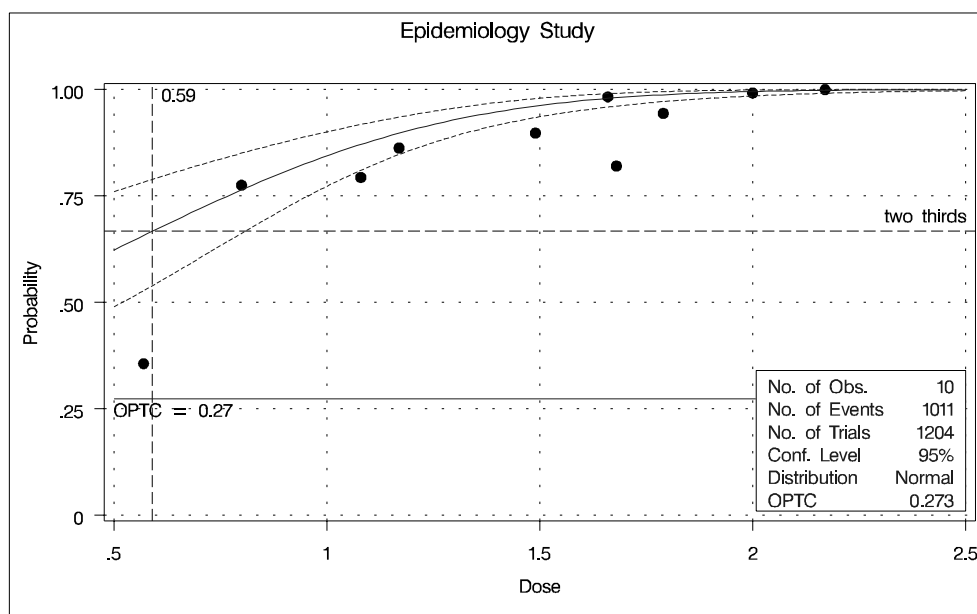
The following three outputs, Output 12.2.11, Output 12.2.12, and Output 12.2.13, are generated from the three plot statements. The first plot, specified with the `PREDPLOT` statement, is the plot of the predicted probability against the single continuous variable `Dose`, which is specified by the `VAR=` option in the `PREDPLOT` statement. This single continuous variable must be in the `MODEL` statement. If the `VAR=` op-

tion is not used, the first single continuous variable in the MODEL statement is used. In this example, you would get the same plot if the VAR = dose was not used in the PREDPLOT statement. You can specify values of other independent variables in the MODEL statement using an XDATA= data set, or by using the default values.

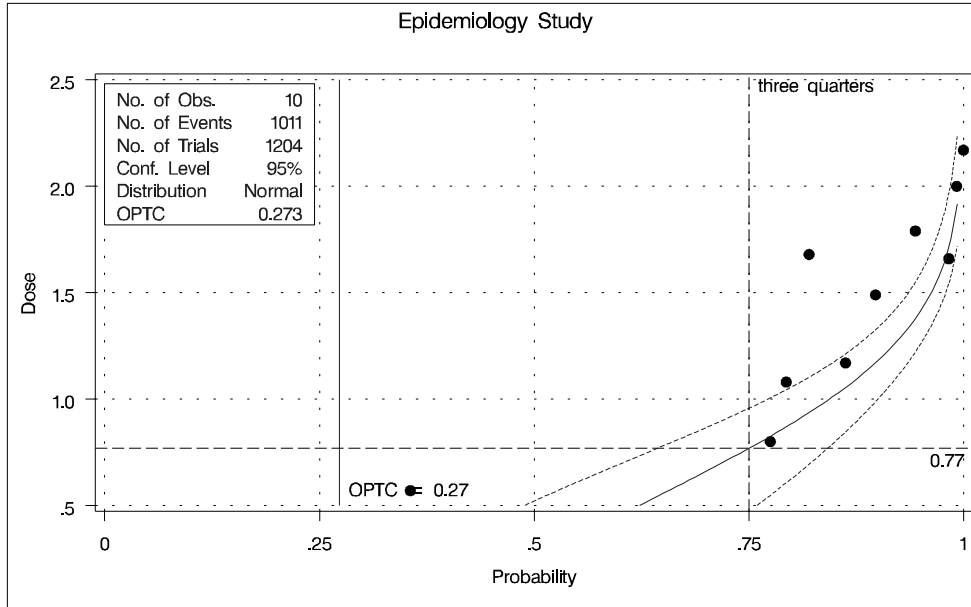
The second plot, specified with the IPPLOT statement, is the inverse of the predicted probability plot with the fiducial limits. It should be pointed out that the fiducial limits are *not* just the inverse of the confidence limits in the predicted probability plot; see the section “Inverse Confidence Limits” on page 278 for the computation of these limits. The third plot, specified with the LPREDPLOT statement, is the plot of the linear predictor  $\mathbf{x}'\beta$  against the first single continuous variable (or the single continuous variable specified by the VAR= option) with the Wald confidence intervals.

After each plot statement, an optional INSET statement is used to draw a box within the plot (inset box). In the inset box, information about the model fitting can be specified. See “INSET Statement” on page 250 for more detail.

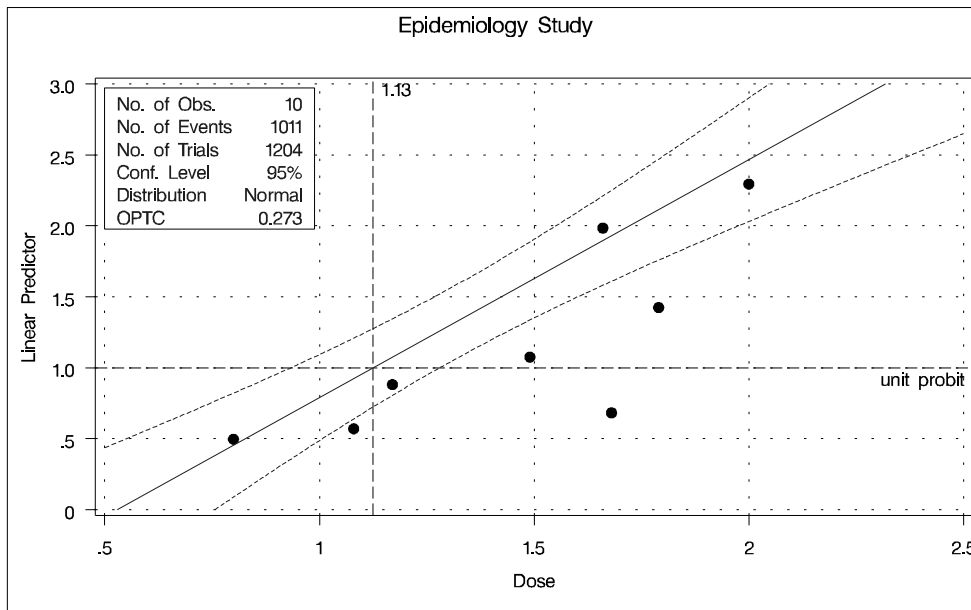
**Output 12.2.11.** Predicted Probability Plot



**Output 12.2.12.** Inverse Predicted Probability Plot



**Output 12.2.13.** Linear Predictor Plot



Combining INEST= data set and the MAXIT= option in the MODEL statement, the PROBIT procedure can do prediction, if the parameterizations for the models used for the training data and the validation data are exactly the same.

After the first invocation of PROC PROBIT, you have the estimated parameters and their covariance matrix in the data set OUTEST = Out1, and the fitted probabilities for the training data set epidemic in the data set OUTPUT = Out2. See Output 12.2.10 on page 293 for the data set Out1 and Output 12.2.14 on page 297 for the data set Out2.

The validation data are collected in data set validate. The second invocation of PROC PROBIT simply passes the estimated parameters from the training data set epidemic to the validation data set validate for prediction. The predicted probabilities are stored in the data set OUTPUT = Out3 (see Output 12.2.15 on page 297). The third invocation of PROC PROBIT passes the estimated parameters as initial values for a new fit of the validation data set using the same model. Predicted probabilities are stored in the data set OUTPUT = Out4 (see Output 12.2.16 on page 297). Goodness-of-Fit tests are computed based on the cells grouped by the AGGREGATE= group variable. Results are shown in Output 12.2.17 on page 298.

```

data validate;
    input treat $ dose sex n r group;
    datalines;
B 2.0 0 44 43 1
B 2.0 1 54 52 2
B 1.5 1 36 32 3
B 1.5 0 45 40 4
A 2.0 0 66 64 5
A 2.0 1 89 89 6
A 1.5 1 45 39 7
A 1.5 0 66 60 8
B 2.0 0 44 44 1
B 2.0 1 54 54 2
B 1.5 1 36 30 3
B 1.5 0 45 41 4
A 2.0 0 66 65 5
A 2.0 1 89 88 6
A 1.5 1 45 38 7
A 1.5 0 66 59 8
;

proc probit optc data = validate inest = out1;
    class treat sex;
    model r/n = dose treat sex sex*treat / maxit = 0 ;
    output out = out3 p =p;
run ;

proc probit optc lackfit data = validate inest = out1;
    class treat sex;
    model r/n = dose treat sex sex*treat / aggregate = group ;
    output out = out4 p =p;
run ;

```

Output 12.2.14. Out2

| Obs | treat | dose | n   | r   | sex | p       |
|-----|-------|------|-----|-----|-----|---------|
| 1   | A     | 2.17 | 142 | 142 | 0   | 0.99272 |
| 2   | A     | 0.57 | 132 | 47  | 1   | 0.35925 |
| 3   | A     | 1.68 | 128 | 105 | 1   | 0.81899 |
| 4   | A     | 1.08 | 126 | 100 | 0   | 0.77517 |
| 5   | A     | 1.79 | 125 | 118 | 0   | 0.96682 |
| 6   | B     | 1.66 | 117 | 115 | 1   | 0.97901 |
| 7   | B     | 1.49 | 127 | 114 | 0   | 0.90896 |
| 8   | B     | 1.17 | 51  | 44  | 1   | 0.89749 |
| 9   | B     | 2.00 | 127 | 126 | 0   | 0.98364 |
| 10  | B     | 0.80 | 129 | 100 | 1   | 0.76414 |

Output 12.2.15. Out3

| Obs | treat | dose | sex | n  | r  | group | p       |
|-----|-------|------|-----|----|----|-------|---------|
| 1   | B     | 2.0  | 0   | 44 | 43 | 1     | 0.98364 |
| 2   | B     | 2.0  | 1   | 54 | 52 | 2     | 0.99506 |
| 3   | B     | 1.5  | 1   | 36 | 32 | 3     | 0.96247 |
| 4   | B     | 1.5  | 0   | 45 | 40 | 4     | 0.91145 |
| 5   | A     | 2.0  | 0   | 66 | 64 | 5     | 0.98500 |
| 6   | A     | 2.0  | 1   | 89 | 89 | 6     | 0.91835 |
| 7   | A     | 1.5  | 1   | 45 | 39 | 7     | 0.74300 |
| 8   | A     | 1.5  | 0   | 66 | 60 | 8     | 0.91666 |
| 9   | B     | 2.0  | 0   | 44 | 44 | 1     | 0.98364 |
| 10  | B     | 2.0  | 1   | 54 | 54 | 2     | 0.99506 |
| 11  | B     | 1.5  | 1   | 36 | 30 | 3     | 0.96247 |
| 12  | B     | 1.5  | 0   | 45 | 41 | 4     | 0.91145 |
| 13  | A     | 2.0  | 0   | 66 | 65 | 5     | 0.98500 |
| 14  | A     | 2.0  | 1   | 89 | 88 | 6     | 0.91835 |
| 15  | A     | 1.5  | 1   | 45 | 38 | 7     | 0.74300 |
| 16  | A     | 1.5  | 0   | 66 | 59 | 8     | 0.91666 |

Output 12.2.16. Out4

| Obs | treat | dose | sex | n  | r  | group | p       |
|-----|-------|------|-----|----|----|-------|---------|
| 1   | B     | 2.0  | 0   | 44 | 43 | 1     | 0.98954 |
| 2   | B     | 2.0  | 1   | 54 | 52 | 2     | 0.98262 |
| 3   | B     | 1.5  | 1   | 36 | 32 | 3     | 0.86187 |
| 4   | B     | 1.5  | 0   | 45 | 40 | 4     | 0.90095 |
| 5   | A     | 2.0  | 0   | 66 | 64 | 5     | 0.98768 |
| 6   | A     | 2.0  | 1   | 89 | 89 | 6     | 0.98614 |
| 7   | A     | 1.5  | 1   | 45 | 39 | 7     | 0.88075 |
| 8   | A     | 1.5  | 0   | 66 | 60 | 8     | 0.88964 |
| 9   | B     | 2.0  | 0   | 44 | 44 | 1     | 0.98954 |
| 10  | B     | 2.0  | 1   | 54 | 54 | 2     | 0.98262 |
| 11  | B     | 1.5  | 1   | 36 | 30 | 3     | 0.86187 |
| 12  | B     | 1.5  | 0   | 45 | 41 | 4     | 0.90095 |
| 13  | A     | 2.0  | 0   | 66 | 65 | 5     | 0.98768 |
| 14  | A     | 2.0  | 1   | 89 | 88 | 6     | 0.98614 |
| 15  | A     | 1.5  | 1   | 45 | 38 | 7     | 0.88075 |
| 16  | A     | 1.5  | 0   | 66 | 59 | 8     | 0.88964 |

**Output 12.2.17.** Goodness-of-Fit Table

| Probit Procedure      |            |        |    |            |
|-----------------------|------------|--------|----|------------|
| Goodness-of-Fit Tests |            |        |    |            |
| Statistic             |            | Value  | DF | Pr > ChiSq |
| Pearson               | Chi-Square | 2.8101 | 2  | 0.2454     |
| L.R.                  | Chi-Square | 2.8080 | 2  | 0.2456     |

---

## References

- Agresti, A. (1990), *Categorical Data Analysis*, New York: John Wiley & Sons.
- Collett, D (1991), *Modelling Binary Data*, London: Chapman and Hall.
- Cox, D.R. (1970), *Analysis of Binary Data*, London: Chapman and Hall.
- Cox, D.R. and Oakes, D. (1984), *Analysis of Survival Data*, London: Chapman and Hall.
- Finney, D.J. (1971), *Probit Analysis*, Third Edition, London: Cambridge University Press.
- Hubert, J.J., Bohidar, N.R., and Peace, K.E. (1988), "Assessment of Pharmacological Activity," *Biopharmaceutical Statistics for Drug Development*, ed. K.E. Peace, New York: Marcel Dekker.



Chapter 13

# The REG Procedure

Chapter Table of Contents

---

|                           |     |
|---------------------------|-----|
| OVERVIEW . . . . .        | 301 |
| SYNTAX . . . . .          | 301 |
| MODEL Statement . . . . . | 301 |



# Chapter 13

## The REG Procedure

---

### Overview

The SCORR1 and SCORR2 options in the MODEL statement now enable you to specify TEST and SEQTEST options that request F-tests,  $p$ -values, and cumulative R-Square values as variables are sequentially added to a model.

---

### Syntax

---

#### MODEL Statement

The following new and updated options are available in the MODEL statement after a slash (/).

**PARTIALR2** < ( < **TESTS** > < **SEQTESTS** > ) >

See the SCORR1 option.

**SCORR1** < ( < **TESTS** > < **SEQTESTS** > ) >

displays the squared semi-partial correlation coefficients using Type I sums of squares. This is calculated as  $SS/SST$ , where  $SST$  is the corrected total  $SS$ . If the NOINT option is used, the uncorrected total  $SS$  is used in the denominator. The optional arguments TESTS and SEQTESTS request F-tests,  $p$ -values, and cumulative R-Square values as variables are sequentially added to a model. The F-test values are computed as the Type I sum of squares for the variable in question divided by a mean square error. If you specify the TESTS option, the denominator MSE is the residual mean square for the full model specified in the MODEL statement. If you specify the SEQTESTS option, the denominator MSE is the residual mean square for the model containing all the independent variables that have been added to the model up to and including the variable in question. The TESTS and SEQTESTS options are not supported if you specify model selection methods, or if you specify the RIDGE or PCOMIT options. Note that the PARTIALR2 option is a synonym for the SCORR1 option.

**SCORR2** < ( **TESTS** ) >

displays the squared semi-partial correlation coefficients using Type II sums of squares. These are calculated the same way as with the SCORR1 option, except that Type II  $SS$  are used instead of Type I  $SS$ . The optional TEST argument requests F-tests,  $p$ -values, and cumulative R-Square values as variables are sequentially added to a model. The F-test values are computed as the Type II sum of squares for the variable in question divided by the residual mean square for the full model specified in the MODEL statement. The TESTS option is not supported if you specify model selection methods, or if you specify the RIDGE or PCOMIT options.



# Chapter 14

## The SURVEYMEANS Procedure

### Chapter Table of Contents

---

|                             |     |
|-----------------------------|-----|
| <b>OVERVIEW</b>             | 305 |
| <b>SYNTAX</b>               | 305 |
| PROC SURVEYMEANS Statement  | 305 |
| RATIO Statement             | 307 |
| <b>DETAILS</b>              | 308 |
| Statistical Computations    | 308 |
| Displayed Output            | 310 |
| ODS Table Names             | 311 |
| <b>EXAMPLE</b>              | 311 |
| Example 14.1 Ratio Analysis | 311 |
| <b>REFERENCES</b>           | 314 |



# Chapter 14

## The SURVEYMEANS Procedure

---

### Overview

You can estimate ratios with the new RATIO statement in the SURVEYMEANS procedure. You can also obtain covariance coefficients for total estimators.

---

### Syntax

#### PROC SURVEYMEANS Statement

---

**PROC SURVEYMEANS** < *options* > < *statistic-keywords* > ;

In the PROC SURVEYMEANS statement, you can use *statistic-keywords* to specify statistics for the procedure to compute.

*statistic-keywords*

specifies the statistics for the procedure to compute. If you do not specify any statistic-keywords, PROC SURVEYMEANS computes the NOBS, MEAN, STDERR, and CLM statistics by default.

The statistics produced depend on the type of the analysis variable. If you name a numeric variable in the CLASS statement, then the procedure analyzes that variable as a categorical variable. The procedure always analyzes character variables as categorical.

PROC SURVEYMEANS computes MIN, MAX, and RANGE for numeric variables but not for categorical variables. For numeric variables, the keyword MEAN produces the mean, but for categorical variables it produces the proportion in each category or level. Also for categorical variables, the keyword NOBS produces the number of observations for each variable level, and the keyword NMISS produces the number of missing observations for each level. If you request the keyword NCLUSTER for a categorical variable, PROC SURVEYMEANS displays for each level the number of clusters with observations in that level. PROC SURVEYMEANS computes SUMWGT in the same way for both categorical and numeric variables, as the sum of the weights over all nonmissing observations.

PROC SURVEYMEANS performs univariate analysis, analyzing each variable separately. Thus, the number of nonmissing and missing observations may not be the same for all analysis variables.

If you use the keyword RATIO without the keyword MEAN, the keyword MEAN is implied.

Other available statistics computed for a ratio are N, NCLU, SUMWGT, RATIO, STDERR, DF, T, PROBT, and CLM, as listed below. If no statistics are requested, the procedure will compute the ratio and its standard error by default for a RATIO statement.

The valid statistic-keywords are as follows:

|          |                                                                                                                                                                |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALL      | all statistics listed                                                                                                                                          |
| CLM      | $100(1 - \alpha)\%$ confidence limits for the MEAN, where $\alpha$ is determined by the ALPHA= option, and the default is $\alpha = 0.05$                      |
| CLSUM    | $100(1 - \alpha)\%$ confidence limits for the SUM, where $\alpha$ is determined by the ALPHA= option, and the default is $\alpha = 0.05$                       |
| CV       | coefficient of variation for MEAN                                                                                                                              |
| CVSUM    | coefficient of variation for SUM                                                                                                                               |
| DF       | degrees of freedom for the $t$ test                                                                                                                            |
| MAX      | maximum value                                                                                                                                                  |
| MEAN     | mean for a numeric variable, or the proportion in each category for a categorical variable                                                                     |
| MIN      | minimum value                                                                                                                                                  |
| NCLUSTER | number of clusters                                                                                                                                             |
| NMISS    | number of missing observations                                                                                                                                 |
| NOBS     | number of nonmissing observations                                                                                                                              |
| RANGE    | range, MAX–MIN                                                                                                                                                 |
| RATIO    | ratio of means or proportions                                                                                                                                  |
| STD      | standard deviation of the SUM. When you request SUM, the procedure computes STD by default.                                                                    |
| STDERR   | standard error of the MEAN or RATIO. When you request MEAN or RATIO, the procedure computes STDERR by default.                                                 |
| SUM      | weighted sum, $\sum w_i y_i$ , or estimated population total when the appropriate sampling weights are used                                                    |
| SUMWGT   | sum of the weights, $\sum w_i$                                                                                                                                 |
| T        | $t$ -value and its corresponding $p$ -value with DF degrees of freedom for<br>$H_0 : \theta = 0$ where $\theta$ is the population mean or the population ratio |
| VAR      | variance of the MEAN or RATIO                                                                                                                                  |
| VARSUM   | variance of the SUM                                                                                                                                            |



---

## RATIO Statement

**RATIO** < 'label' > *variables* / *variables* ;

The RATIO statement requests ratio analysis for means or proportions of analysis variables. A ratio statement names the variables whose means will be used as numerators or denominators in a ratio. Variables appearing before the slash (/), called *numerator variables*, are used for numerators. Variables appearing after the slash (/), called *denominator variables*, are used for denominators. These *variables* can be any number of analysis variables, either continuous or categorical, in the input data set.

You can optionally specify a label for each RATIO statement to identify the ratios in the output. Labels must be enclosed in single quotes.

If a RATIO statement does not have any numerator variable or denominator variable specified, the RATIO statement will be ignored.

A numerator or denominator variable must be an analysis variable. That is, if there is a VAR statement, then a numerator or denominator variable must appear in the VAR statement. If there is no VAR statement, a numerator or denominator variable must be on the default analysis variable list. If a numerator or denominator variable is not an analysis variable, it is ignored.

The computation of ratios depends on whether the numerator and denominator variables are continuous or categorical.

For continuous variables, ratios are calculated with the mean of the variables. For example, for continuous variables X, Y, Z, and T, the following RATIO statement requests the procedure to analyze the ratios  $\bar{x}/\bar{z}$ ,  $\bar{x}/\bar{t}$ ,  $\bar{y}/\bar{z}$ , and  $\bar{y}/\bar{t}$ :

**ratio x y / z t;**

If a continuous variable appears as both a numerator and a denominator variable, the ratio of this variable itself is ignored.

For categorical variables, ratios are calculated with the proportions for the categories of a categorical variable. For example, if categorical variable Gender has values “Male” and “Female,” with proportions  $p_m = \text{Pr}(\text{Gender}=\text{“Male”})$  and  $p_f = \text{Pr}(\text{Gender}=\text{“Female”})$ , and Y is a continuous variable, then the following RATIO statement requests the procedure to analyze the ratios  $p_m/p_f$ ,  $p_f/p_m$ ,  $\bar{y}/p_m$ , and  $\bar{y}/p_f$ :

**ratio Gender y / Gender;**

If a categorical variable appears as both a numerator and a denominator variable, then the ratios of the proportions for all categories are computed, except the ratio of each category with itself.

You may have more than one RATIO statement. Each RATIO statement produces ratios independently using its own numerator and denominator variables. Each RATIO statement also produces its own ratio analysis table.

Available statistics for a ratio are

- N, number of observations used to compute the ratio
- NCLU, number of clusters
- SUMWGT, sum of weights
- RATIO, ratio
- STDERR, standard error of ratio
- VAR, variance of ratio
- T,  $t$ -value of ratio
- PROBT,  $p$ -value of  $t$
- DF, degrees of freedom of  $t$
- CLM, confidence limits of ratio

The procedure will calculate these statistics based on the statistic-keywords described on page 305, which you specified in the PROC statement. If a statistic-keyword is not appropriate for RATIO statement, that statistic-keyword is ignored. If no valid statistics are requested for a RATIO statement, the procedure will compute the ratio and its standard error by default.

When calculating the means or proportions for the numerator and denominator variables in a ratio, an observation is excluded if it has a missing value in either the continuous numerator variable or the denominator variable. An observation with missing values is also excluded for the categorical numerator or denominator variables, unless the MISSING option is used.

---

## Details

---

### Statistical Computations

---

#### Definitions and Notation

For a stratified clustered sample design, together with the sampling weights, the sample can be represented by an  $n \times (P + 1)$  matrix

$$\begin{aligned} (\mathbf{w}, \mathbf{Y}) &= (w_{hij}, \mathbf{y}_{hij}) \\ &= \left( w_{hij}, y_{hij}^{(1)}, y_{hij}^{(2)}, \dots, y_{hij}^{(P)} \right) \end{aligned}$$

where

- $h = 1, 2, \dots, H$  is the stratum number, with a total of  $H$  strata

- $i = 1, 2, \dots, n_h$  is the cluster number within stratum  $h$ , with a total of  $n_h$  clusters
- $j = 1, 2, \dots, m_{hi}$  is the unit number within cluster  $i$  of stratum  $h$ , with a total of  $m_{hi}$  units
- $p = 1, 2, \dots, P$  is the analysis variable number, with a total of  $P$  variables
- $n = \sum_{h=1}^H \sum_{i=1}^{n_h} m_{hi}$  is the total number of observations in the sample
- $w_{hij}$  denotes the sampling weight for observation  $j$  in cluster  $i$  of stratum  $h$
- $\mathbf{y}_{hij} = (y_{hij}^{(1)}, y_{hij}^{(2)}, \dots, y_{hij}^{(P)})$  are the observed values of the analysis variables for observation  $j$  in cluster  $i$  of stratum  $h$ , including both the values of numerical variables and the values of indicator variables for levels of categorical variables.

For a categorical variable  $C$ , let  $l$  denote the number of levels of  $C$ , and denote the level values as  $c_1, c_2, \dots, c_l$ . Then there are  $l$  indicator variables associated with these levels. That is, for level  $C = c_k$  ( $k = 1, 2, \dots, l$ ), a  $y^{(q)}$  ( $q \in \{1, 2, \dots, P\}$ ) contains the values of the indicator variable for the category  $C = c_k$ , with the value of observation  $j$  in cluster  $i$  of stratum  $h$ :

$$y_{hij}^{(q)} = I_{\{C=c_k\}}(h, i, j) = \begin{cases} 1 & \text{if } C_{hij} = c_k \\ 0 & \text{otherwise} \end{cases}$$

Therefore, the total number of analysis variables,  $P$ , is the total number of numerical variables plus the total number of levels of all categorical variables.

Also,  $f_h$  denotes the sampling rate for stratum  $h$ . You can use the TOTAL= option or the RATE= option to input population totals or sampling rates. If you input stratum totals, PROC SURVEYMEANS computes  $f_h$  as the ratio of the stratum sample size to the stratum total. If you input stratum sampling rates, PROC SURVEYMEANS uses these values directly for  $f_h$ . If you do not specify the TOTAL= option or the RATE= option, then the procedure assumes that the stratum sampling rates  $f_h$  are negligible, and a finite population correction is not used when computing variances.

This notation is also applicable to other sample designs. For example, for a sample design without stratification, you can let  $H = 1$ ; for a sample design without clusters, you can let  $m_{hi} = 1$  for every  $h$  and  $i$ .

### Ratio

When you use a RATIO statement, the procedure produces statistics requested by the statistics-keywords in the PROC SURVEYMEANS statement.

Suppose that you want to calculate the ratio of variable  $Y$  over variable  $X$ . Let  $x_{hij}$  be the value of variable  $X$  for the  $j$ th member in cluster  $i$  in the  $h$ th stratum.

The ratio of  $Y$  over  $X$  is

$$\hat{R} = \frac{\sum_{h=1}^H \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} w_{hij} y_{hij}}{\sum_{h=1}^H \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} w_{hij} x_{hij}}$$

PROC SURVEYMEANS uses the Taylor series expansion method to estimate the variance of the ratio  $\hat{R}$  as

$$\hat{V}(\hat{R}) = \sum_{h=1}^H \frac{n_h(1-f_h)}{n_h-1} \sum_{i=1}^{n_h} (g_{hi\cdot} - \bar{g}_{h\cdot\cdot})^2$$

where

$$g_{hi\cdot} = \frac{\sum_{j=1}^{m_{hi}} w_{hij} (y_{hij} - x_{hij} \hat{R})}{\sum_{h=1}^H \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} w_{hij} x_{hij}}$$

$$\bar{g}_{h\cdot\cdot} = \left( \sum_{i=1}^{n_h} g_{hi\cdot} \right) / n_h$$

The standard error of the ratio is the square root of the estimated variance.

$$\text{StdErr}(\hat{R}) = \sqrt{\hat{V}(\hat{R})}$$

### Coefficient of Variation

If you specify the keyword CV, PROC SURVEYMEANS computes the coefficient of variation, which is the ratio of the standard error of the mean to the estimated mean.

$$cv(\bar{Y}) = \text{StdErr}(\hat{Y}) / \hat{Y}$$

If you specify the keyword CVSUM, PROC SURVEYMEANS computes the coefficient of variation for the estimated total, which is the ratio of the standard deviation of the sum to the estimated total.

$$cv(Y) = \text{Std}(\hat{Y}) / \hat{Y}$$

---

## Displayed Output

If you have a RATIO statement, the procedure displays a “Ratio Analysis” table.

### Ratio Analysis

The “Ratio Analysis” table displays all of the statistics that you request with statistic-keywords in the PROC statement described on page 305. If you do not specify any statistic-keywords, then by default this table displays the ratio and its standard error. The “Ratio Analysis” table may contain the following information for each ratio, depending on which statistic-keywords you request:

- Numerator, which identifies the numerator variable of the ratio
- Denominator, which identifies the denominator variable of the ratio
- N, which is the number of observations used in the ratio analysis
- Number of Clusters
- Sum of Weights

- DF, which is the degrees of freedom for the  $t$  test
- Ratio
- Std Error of Ratio, which is the standard error of the ratio
- Var of Ratio, which is the variance of the ratio
- $t$  Value, for testing  $H_0 : \text{population RATIO} = 0$
- $\text{Pr} > |t|$ , which is the two-sided  $p$ -value for the  $t$  test
- Lower and Upper  $100(1 - \alpha)\%$  CL for Ratio, which are confidence limits for the Ratio

When you use the ODS output statement to create an output data set, if you use labels for your RATIO statement, these labels are saved in a variable `RatioLabel` in the output data set.

## ODS Table Names

PROC SURVEYMEANS assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

**Table 14.1.** ODS Tables Produced in PROC SURVEYMEANS

| ODS Table Name | Description             | Statement | Option  |
|----------------|-------------------------|-----------|---------|
| ClassVarInfo   | Class level information | CLASS     | default |
| Domain         | Statistics in domains   | DOMAIN    | default |
| Ratio          | Statistics for ratios   | RATIO     | default |
| Statistics     | Statistics              | PROC      | default |
| StrataInfo     | Stratum information     | STRATA    | LIST    |
| Summary        | Data summary            | PROC      | default |

## Example

### Example 14.1. Ratio Analysis

You are studying profiles of the 800 top-performing companies to provide information on their impact on the economy. In particular, suppose you are interested in the profit per employee and the sale per employee among these companies. You select a sample of 66 companies with unequal probability across market types. The data for these 66 companies is saved in the following data set:

```
data Company;
  length Type $14;
  input Type$ Asset Sale Value Profit Employee Weight;
  datalines;
Other          2764.0  1828.0  1850.3   144.0   18.7    9.6
Energy         13246.2  4633.5  4387.7   462.9   24.3   42.6
Finance        3597.7   377.8   93.0    14.0    1.1   12.2
```

|                |         |         |        |       |       |      |
|----------------|---------|---------|--------|-------|-------|------|
| Transportation | 6646.1  | 6414.2  | 2377.5 | 348.2 | 47.1  | 21.8 |
| HiTech         | 1068.4  | 1689.8  | 1430.2 | 72.9  | 4.6   | 4.3  |
| Manufacturing  | 1125.0  | 1719.4  | 1057.5 | 98.1  | 20.4  | 4.5  |
| Other          | 1459.0  | 1241.4  | 452.7  | 24.5  | 20.1  | 5.5  |
| Finance        | 2672.3  | 262.5   | 296.2  | 23.1  | 2.2   | 9.3  |
| Finance        | 311.0   | 566.2   | 932.0  | 52.8  | 2.7   | 1.9  |
| Energy         | 1148.6  | 1014.6  | 485.1  | 60.6  | 4.0   | 4.5  |
| Finance        | 5327.0  | 572.4   | 372.9  | 25.2  | 4.2   | 17.7 |
| Energy         | 1602.7  | 678.4   | 653.0  | 75.6  | 2.8   | 6.0  |
| Energy         | 5808.8  | 1288.4  | 2007.0 | 318.8 | 5.9   | 19.2 |
| Medical        | 268.8   | 204.4   | 820.9  | 45.6  | 3.7   | 1.8  |
| Transportation | 5222.6  | 2627.8  | 1910.0 | 245.6 | 22.8  | 17.4 |
| Other          | 872.7   | 1419.4  | 939.3  | 69.7  | 12.2  | 3.7  |
| Retail         | 4461.7  | 8946.8  | 4662.7 | 289.0 | 132.1 | 15.0 |
| HiTech         | 6719.2  | 6942.0  | 8240.2 | 381.3 | 85.8  | 22.1 |
| Retail         | 833.4   | 1538.8  | 1090.3 | 64.9  | 15.4  | 3.5  |
| Finance        | 415.9   | 167.3   | 1126.8 | 56.8  | 0.7   | 2.2  |
| HiTech         | 442.4   | 1139.9  | 1039.9 | 57.6  | 22.7  | 2.3  |
| Other          | 801.5   | 1157.0  | 664.2  | 56.9  | 15.5  | 3.4  |
| Finance        | 4954.8  | 468.8   | 366.4  | 41.7  | 3.0   | 16.5 |
| Finance        | 2661.9  | 257.9   | 181.1  | 21.2  | 2.1   | 9.3  |
| Finance        | 5345.8  | 530.1   | 337.4  | 36.4  | 4.3   | 17.8 |
| Energy         | 3334.3  | 1644.7  | 1407.8 | 157.6 | 6.4   | 11.4 |
| Manufacturing  | 1826.6  | 2671.7  | 483.2  | 71.3  | 25.3  | 6.7  |
| Retail         | 618.8   | 2354.7  | 767.7  | 58.6  | 19.0  | 2.9  |
| Retail         | 1529.1  | 6534.0  | 826.3  | 58.3  | 65.8  | 5.7  |
| Manufacturing  | 4458.4  | 4824.5  | 3132.1 | 28.9  | 67.0  | 15.0 |
| HiTech         | 5831.7  | 6611.1  | 9464.7 | 459.6 | 86.7  | 19.3 |
| Medical        | 6468.3  | 4199.2  | 3170.4 | 270.1 | 59.5  | 21.3 |
| Energy         | 1720.7  | 473.1   | 811.1  | 86.6  | 1.6   | 6.3  |
| Energy         | 1679.7  | 1379.9  | 721.1  | 91.8  | 4.5   | 6.2  |
| Retail         | 4018.2  | 16823.4 | 2038.3 | 178.1 | 162.0 | 13.6 |
| Other          | 227.1   | 575.8   | 1083.8 | 62.6  | 1.9   | 1.6  |
| Finance        | 3872.8  | 362.0   | 209.3  | 27.6  | 2.4   | 13.1 |
| Retail         | 3359.3  | 4844.7  | 2651.4 | 224.1 | 75.6  | 11.5 |
| Energy         | 1295.6  | 356.9   | 180.8  | 162.3 | 0.6   | 5.0  |
| Energy         | 1658.0  | 626.6   | 688.0  | 126.0 | 3.5   | 6.1  |
| Finance        | 12156.7 | 1345.5  | 680.7  | 106.6 | 9.4   | 39.2 |
| HiTech         | 3982.6  | 4196.0  | 3946.8 | 313.9 | 64.3  | 13.5 |
| Finance        | 8760.7  | 886.4   | 1006.9 | 90.0  | 7.5   | 28.5 |
| Manufacturing  | 2362.2  | 3153.3  | 1080.0 | 137.0 | 25.2  | 8.4  |
| Transportation | 2499.9  | 3419.0  | 992.6  | 47.2  | 25.3  | 8.8  |
| Energy         | 1430.4  | 1610.0  | 664.3  | 77.7  | 3.5   | 5.4  |
| Energy         | 13666.5 | 15465.4 | 2736.7 | 411.4 | 26.6  | 43.9 |
| Manufacturing  | 4069.3  | 4174.7  | 2907.6 | 289.2 | 38.2  | 13.7 |
| Energy         | 2924.7  | 711.9   | 1067.8 | 146.7 | 3.4   | 10.1 |
| Transportation | 1262.1  | 1716.0  | 364.3  | 71.2  | 14.5  | 4.9  |
| Medical        | 684.4   | 672.9   | 287.4  | 61.8  | 6.0   | 3.1  |
| Energy         | 3069.3  | 1719.0  | 1439.0 | 196.4 | 4.9   | 10.6 |
| Medical        | 246.5   | 318.8   | 924.1  | 43.8  | 3.1   | 1.7  |
| Finance        | 11562.2 | 1128.5  | 580.4  | 64.2  | 6.7   | 37.3 |
| Finance        | 9316.0  | 1059.4  | 816.5  | 95.9  | 8.0   | 30.2 |
| Retail         | 1094.3  | 3848.0  | 563.3  | 29.4  | 44.7  | 4.4  |
| Retail         | 1102.1  | 4878.3  | 932.4  | 65.2  | 47.3  | 4.4  |

|                |         |         |        |       |       |      |
|----------------|---------|---------|--------|-------|-------|------|
| HiTech         | 466.4   | 675.8   | 845.7  | 64.5  | 5.2   | 2.4  |
| Manufacturing  | 10839.4 | 5468.7  | 1895.4 | 232.8 | 47.8  | 35.0 |
| Manufacturing  | 733.5   | 2135.3  | 96.6   | 10.9  | 2.7   | 3.2  |
| Manufacturing  | 10354.2 | 14477.4 | 5607.2 | 321.9 | 188.5 | 33.5 |
| Energy         | 1902.1  | 2697.9  | 329.3  | 34.2  | 2.2   | 6.9  |
| Other          | 2245.2  | 2132.2  | 2230.4 | 198.9 | 8.0   | 8.0  |
| Transportation | 949.4   | 1248.3  | 298.9  | 35.4  | 10.4  | 3.9  |
| Retail         | 2834.4  | 2884.6  | 458.2  | 41.2  | 49.8  | 9.8  |
| Retail         | 2621.1  | 6173.8  | 1992.7 | 183.7 | 115.1 | 9.2  |

;

For each company in your sample, you have the following information in the data set:

| Variable | Description                                            |
|----------|--------------------------------------------------------|
| Type     | the type of market of the company                      |
| Asset    | the company's assets in millions of dollars            |
| Sale     | the market value of the company in millions of dollars |
| Profit   | the company's profit in millions of dollars            |
| Employee | the number of employees in thousands                   |
| Weight   | the sampling weight                                    |

The following SAS statements illustrate how you can use PROC SURVEYMEANS to estimate these ratios:

```

title1 'Ratio Analysis in Top Companies Profile Study';
proc surveymeans data=Company total=800 ratio;
  var Profit Sale Employee;
  weight Weight;
  ratio Profit Sale / Employee;
run;

```

The RATIO statement requests the ratio of the profit and the sale to the number of employees.

#### Output 14.1.1. Estimate Ratios

| Ratio Analysis in Top Companies Profile Study |             |            |           |
|-----------------------------------------------|-------------|------------|-----------|
| The SURVEYMEANS Procedure                     |             |            |           |
| Ratio Analysis                                |             |            |           |
| Numerator                                     | Denominator | Ratio      | Std Err   |
| Sale                                          | Employee    | 114.332497 | 20.502742 |
| Profit                                        | Employee    | 5.119698   | 1.058939  |

Output 14.1.1 shows the estimated ratios and their standard errors. Because the profit and the sale figures are in millions of dollars, and the employee figures in thousands, the profit per employee is estimated as \$5,120 with a standard error of \$1,059, and the sale per employee is \$114,332 with a standard error of \$20,503.

---

## References

- Cochran, W.G. (1977), *Sampling Techniques*, Third Edition, New York: John Wiley & Sons, Inc.
- Fuller, W.A. (1975), “Regression Analysis for Sample Survey,” *Sankhyā*, 37, Series C, Pt. 3, 117–132.
- Fuller, W.A., Kennedy, W., Schnell, D., Sullivan, G., and Park, H.J. (1989), *PC CARP*, Ames, IA: Statistical Laboratory, Iowa State University.
- Hansen, M.H., Hurwitz, W.N., and Madow, W.G. (1953), *Sample Survey Methods and Theory*, Volumes I and II, New York: John Wiley & Sons, Inc.
- Hidiroglou, M.A., Fuller, W.A., and Hickman, R.D. (1980), *SUPER CARP*, Ames, IA: Statistical Laboratory, Iowa State University.
- Kish, L. (1965), *Survey Sampling*, New York: John Wiley & Sons, Inc.



# Subject Index

## A

- Akaike's information criterion
  - generalized logit model, 122
- annotating
  - cdf plots, 246
  - ipp plots, 256
  - lpred plots, 263
  - pplot plots, 81
  - predicted probability plots, 273

## B

- biological assay data, 278
- boot
  - MI procedure, 146
- BOXPLOT procedure
  - overlays, 5
  - URLs associated with box-and-whisker plots, 3

## C

- cdf plots
  - annotating, 246
  - axes, color, 246
  - font, specifying, 247
  - options summarized by function, 243, 261
  - reference lines, options, 246–250
  - threshold lines, options, 249
- cdfplot
  - PROBIT procedure, 242
- censoring
  - LIFEREG procedure, 80
- classification variable
  - SURVEYMEANS procedure, 309
- coefficient of variation
  - SURVEYMEANS procedure, 310
- confidence intervals
  - LIFEREG procedure, 87
- confidence limits
  - CATMOD, 11
- confidence limits for predicted probabilities
  - generalized logit model, 121
- converge
  - MI procedure, 142, 146

## D

- deviance
  - PROBIT procedure, 269, 280
- dispersion parameter
  - PROBIT procedure, 280

## E

- event category
  - LOGISTIC procedure, 118
- EXACT computational resources
  - LOGISTIC procedure, 123

## F

- fiducial limits, 279

## G

- GAM procedure
  - comparing PROC GAM with PROC LOESS, 63
  - comparing PROC GAM with PROC TPSPLINE, 50
  - generalized additive model with binary data, 43
  - ODS table names, 43
  - Poisson regression analysis of component reliability, 55
- generalized logit model
  - Akaike's information criterion, 122
  - example (LOGISTIC), 124
  - LOGISTIC procedure, 117, 120
  - Newton-Raphson algorithm, 120
  - Schwartz criterion, 122
- graphics
  - saving output (MI), 145
- graphics catalog, specifying
  - LIFEREG procedure, 77
  - PROBIT procedure, 241

## I

- INEST= data sets
  - LIFEREG procedure, 92
- input data set
  - MI procedure, 138, 146, 207
- inset
  - LIFEREG procedure, 78
  - PROBIT procedure, 250
- intercept names
  - LOGISTIC procedure, 122–123
- inverse confidence limits
  - PROBIT procedure, 278
- ipp plots
  - annotating, 256
  - axes, color, 256
  - font, specifying, 256
  - options summarized by function, 253
  - reference lines, options, 256–260
  - threshold lines, options, 259

ippplot  
 PROBIT procedure, 252

## L

lack of fit tests, 279  
 lifereg analysis  
   insets, 78  
 LIFEREG procedure  
   censoring, 80  
   confidence intervals, 87  
   INEST= data sets, 92  
   inset, 78  
   syntax, 77  
   XDATA= data sets, 93  
 likelihood ratio chi-square test, 279  
 link function  
   LOGISTIC procedure, 119  
 LOGISTIC procedure  
   event category, 118  
   EXACT computational resources, 123  
   generalized logit model, 117, 120  
   link function, 119  
   nominal response, 117  
   output data sets, 123  
   reference category, 117  
   response level ordering, 118  
   response variable options, 117  
   reverse response level ordering, 118  
 lpred plots  
   annotating, 263  
   axes, color, 263  
   font, specifying, 264  
   reference lines, options, 264–267  
   threshold lines, options, 266  
 lpredplot  
   PROBIT procedure, 260

## M

MI procedure  
   boot, 146  
   converge, 142, 146  
   input data set, 138, 146, 207  
   introductory example, 133  
   ODS table names, 176  
   output data sets, 139, 142, 147  
   output parameter estimates, 147  
   random number generators, 140  
   saving graphics output, 145  
   singularity, 140  
   suppressing output, 139  
   syntax, 137  
 MIANALYZE procedure  
   introductory example, 203  
   ODS table names, 216  
   syntax, 206  
 multiple imputations analysis, 131, 203

## N

Newton-Raphson algorithm

generalized logit model, 120  
 nominal response  
   LOGISTIC procedure, 117

## O

output data sets  
   LOGISTIC procedure, 123  
   MI procedure, 139, 142, 147  
 output parameter estimates  
   MI procedure, 147  
 output table names  
   SURVEYMEANS procedure, 311  
 overdispersion  
   PROBIT procedure, 268

## P

Pearson chi-square test, 279  
 Pearson's chi-square  
   PROBIT procedure, 268–269, 280  
 pplot plots  
   annotating, 81  
   axes, color, 81  
   font, specifying, 82  
   reference lines, options, 82–84, 86  
 predicted probability plots  
   annotating, 273  
   axes, color, 273  
   font, specifying, 274  
   options summarized by function, 270  
   reference lines, options, 273–277  
   threshold lines, options, 276  
 predpplot  
   PROBIT procedure, 269  
 probit analysis  
   insets, 251  
 PROBIT procedure  
   cdfplot, 242  
   deviance, 269, 280  
   deviance statistic, 279  
   dispersion parameter, 280  
   goodness-of-fit, 268–269  
   goodness-of-fit tests, 268, 279  
   inset, 250  
   inverse confidence limits, 278  
   ippplot, 252  
   lpredplot, 260  
   overdispersion, 268  
   Pearson's chi-square, 268–269, 279–280  
   predpplot, 269  
   subpopulation, 268–269, 280

## R

random number generators  
   MI procedure, 140  
 ratio  
   SURVEYMEANS procedure, 307, 309  
 ratio analysis  
   SURVEYMEANS procedure, 307, 309–310  
 reference category

- LOGISTIC procedure, 117, 119
- response level ordering
  - LOGISTIC procedure, 118
- response variable options
  - LOGISTIC procedure, 117
- reverse response level ordering
  - LOGISTIC procedure, 118

## S

- Schwartz criterion
  - generalized logit model, 122
- singularity
  - MI procedure, 140
- standard error of ratio
  - SURVEYMEANS procedure, 309
- statistic-keywords
  - SURVEYMEANS procedure, 305
- statistical computation
  - SURVEYMEANS procedure, 308
- subpopulation
  - PROBIT procedure, 268–269, 280
- suppressing output
  - MI procedure, 139
- SURVEYMEANS procedure
  - categorical variable, 309
  - coefficient of variation, 310
  - denominator variable, 307
  - indicator variable, 309
  - numerator variable, 307
  - ODS table names, 311
  - output table names, 311
  - ratio, 307, 309
  - ratio analysis, 307, 309–310
  - standard error of ratio, 309
  - statistic-keywords, 305
  - statistical computation, 308
- syntax
  - LIFEREG procedure, 77

## X

- XDATA= data sets
  - LIFEREG procedure, 93



# Syntax Index

## A

AGGREGATE= option  
    MODEL statement (PROBIT), 268  
ALPHA= option  
    MODEL statement (CATMOD), 11  
    MODEL statement (GAM), 31  
    MODEL statement (LIFEREG), 80

## B

BINOMIAL option  
    TABLES statement (FREQ), 20  
BINOMIALC option  
    TABLES statement (FREQ), 20  
BOOTSTRAP option  
    PROC MI statement, 146  
BOXCOX transformation  
    TRANSFORM statement (MI), 150  
BOXPLOT procedure, PLOT statement  
    CCOVERLAY= option, 3  
    COVERLAY= option, 3  
    HTML= option, 3  
    LOVERLAY= option, 3  
    NOOVERLAYLEGEND option, 3  
    OVERLAY= option, 3  
    OVERLAYHTML= option, 4  
    OVERLAYLEGLAB= option, 4  
    OVERLAYSYM= option, 4  
    OVERLAYSYMHT= option, 4  
    WOVERLAY= option, 4  
BY statement  
    GAM procedure, 28  
    MI procedure, 141  
    MIANALYZE procedure, 208

## C

C= option  
    TRANSFORM statement (MI), 151  
CATMOD procedure, MODEL statement  
    ALPHA= option, 11  
    CLPARM option, 11  
CCOVERLAY= option  
    PLOT statement (BOXPLOT), 3  
CDFPLOT statement  
    See PROBIT procedure, CDFPLOT statement  
    options summarized by function, 243  
    PROBIT procedure, 242  
CLASS statement  
    GAM procedure, 29  
CLPARM option

    MODEL statement (CATMOD), 11  
CONVERGE option  
    PROC MI statement, 142, 146  
CONVERGE= option  
    MODEL statement (LIFEREG), 80  
    MODEL statement (PROBIT), 268  
COVERLAY= option  
    PLOT statement (BOXPLOT), 3

## D

DATA= option  
    PROC GAM statement, 28  
    PROC MI statement, 138  
    PROC MIANALYZE statement, 207  
    SCORE statement (GAM), 33  
DESCENDING option  
    MODEL statement (LOGISTIC), 118  
DIST = option  
    MODEL statement (GAM), 31

## E

EM statement  
    MI procedure, 141  
EPSILON = option  
    MODEL statement (GAM), 31  
EPSSCORE = option  
    MODEL statement (GAM), 31  
EVENT= option  
    MODEL statement (LOGISTIC), 118  
EXACT statement  
    FREQ procedure, 19  
    NPARIWAY procedure, 237  
EXP transformation  
    TRANSFORM statement (MI), 150

## F

FACTOR procedure, PROC FACTOR statement  
    FLAG= option, 15  
    FUZZ= option, 15  
    ROUND option, 15  
FLAG= option  
    PROC FACTOR statement, 15  
FREQ procedure  
    syntax, 19  
FREQ procedure, EXACT statement, 19  
    POINT option, 19  
FREQ procedure, TABLES statement, 20  
    BINOMIAL option, 20  
    BINOMIALC option, 20

RISKDIFFC option, 20  
 FREQ statement  
   GAM procedure, 29  
   MI procedure, 142  
 FUZZ= option  
   PROC FACTOR statement, 15

**G**

GAM procedure, 28  
   syntax, 28  
 GAM procedure, BY statement, 28  
 GAM procedure, CLASS statement, 29  
 GAM procedure, FREQ statement, 29  
 GAM procedure, ID statement, 30  
 GAM procedure, MODEL statement, 30  
   ALPHA= option, 31  
   DIST= option, 31  
   EPSILON= option, 31  
   EPSSCORE= option, 31  
   ITPRINT option, 31  
   MAXITER= option, 31  
   MAXITSCORE= option, 31  
   METHOD= option, 32  
   NOTEST option, 32  
 GAM procedure, OUTPUT statement, 32  
   OUT= option, 32  
 GAM procedure, PROC GAM statement, 28  
   DATA= option, 28  
 GAM procedure, SCORE statement, 33  
   DATA= option, 33  
   OUT= option, 33  
 GOUT= option  
   MCMC statement, 145  
   PROC LIFEREG statement, 77  
   PROC PROBIT statement, 241

**H**

HTML= option  
   PLOT statement (BOXPLOT), 3

**I**

ID statement  
   GAM procedure, 30  
 INEST= option  
   PROC LIFEREG statement, 77  
   PROC PROBIT statement, 241  
 INSET statement  
   LIFEREG procedure, 78  
   PROBIT procedure, 250  
 IPPLOT statement  
   options summarized by function, 253  
   PROBIT procedure, 252  
 ITPRINT option  
   MCMC statement (MI), 146  
   MODEL statement (GAM), 31  
   PROC MI statement, 142

**K**

KS option  
   EXACT statement (NPAR1WAY), 237

**L**

LAMBDA= option  
   TRANSFORM statement (MI), 151  
 LIFEREG procedure, INSET statement, 78  
   keywords, 78  
 LIFEREG procedure, MODEL statement, 80  
   ALPHA= option, 80  
   CONVERGE= option, 80  
 LIFEREG procedure, OUTPUT statement, 80  
 LIFEREG procedure, PLOT statement  
   ANNOTATE= option, 81  
   CAXIS= option, 81  
   CCENSOR option, 81  
   CENBIN option, 81  
   CENCOLOR option, 81  
   CENSYMBOL option, 82  
   CFIT= option, 82  
   CFRAME= option, 82  
   CGRID= option, 82  
   CHREF= option, 82  
   CTEXT= option, 82  
   CVREF= option, 82  
   DESCRIPTION= option, 82  
   FONT= option, 82  
   HCL, 82  
   HEIGHT= option, 82  
   HLOWER= option, 82  
   HOFFSET= option, 83  
   HREF= option, 83  
   HREFLABELS= option, 83  
   HREFLABPOS= option, 83  
   HUPPER= option, 83  
   INBORDER option, 83  
   INTERTILE option, 83  
   ITPRINTEM option, 83  
   JITTER option, 83  
   LFIT option, 83  
   LGRID option, 84  
   LHREF= option, 84  
   LVREF= option, 84  
   MAXITEM= option, 84  
   NAME= option, 84  
   NOCENPLOT option, 84  
   NOCONF option, 84  
   NODATA option, 84  
   NOFIT option, 84  
   NOFRAME option, 84  
   NOGRID option, 84  
   NOHLABEL option, 84  
   NOHTICK option, 84  
   NOPOLISH option, 84  
   NOVLABEL option, 84  
   NOVTICK option, 84  
   NPINTERVALS option, 85  
   PCTLIST option, 85  
   PLOWER= option, 85  
   PPOS option, 85  
   PPOUT option, 85  
   PRINTPROBS option, 85

PROBLIST option, 85  
 PUPPER= option, 85  
 ROTATE option, 85  
 SQUARE option, 85  
 TOLLIKE option, 85  
 TOLPROB option, 86  
 VAXISLABEL= option, 86  
 VREF= option, 86  
 VREFLABELS= option, 86  
 VREFLABPOS= option, 86  
 WAXIS= option, 86  
 WFIT= option, 86  
 WGRID= option, 86  
 WREFL= option, 86  
 LIFEREG procedure, PROBLOT statement, 81  
 LIFEREG procedure, PROC LIFEREG statement, 77  
   GOUT= option, 77  
   INEST= option, 77  
   NAMELEN= option, 78  
   XDATA= option, 78  
 LINK= option  
   MODEL statement (LOGISTIC), 119  
 LOESS procedure, MODEL statement  
   SELECT= option, 107  
 LOG transformation  
   TRANSFORM statement (MI), 150  
 LOGISTIC procedure, MODEL statement  
   DESCENDING option, 118  
   EVENT= option, 118  
   LINK= option, 119  
   ORDER= option, 118  
   REFERENCE= option, 119  
 LOGIT transformation  
   TRANSFORM statement (MI), 150  
 LOVERLAY= option  
   PLOT statement (BOXPLOT), 3  
 LPREDPLOT statement  
   options summarized by function, 261  
   PROBIT procedure, 260

## M

MAXITER = option  
   MODEL statement (GAM), 31  
 MAXITER= option  
   PROC MI statement, 142, 146  
 MAXITSCORE = option  
   MODEL statement (GAM), 31  
 MCMC statement  
   MI procedure, 143  
 METHOD= option  
   MODEL statement (GAM), 32  
 MI procedure, BY statement, 141  
 MI procedure, EM statement, 141  
 MI procedure, FREQ statement, 142  
 MI procedure, MCMC statement, 143  
   GOUT= option, 145  
   ITPRINT option, 146  
 MI procedure, MONOTONE statement, 149  
 MI procedure, PROC MI statement, 138

BOOTSTRAP option, 146  
 CONVERGE option, 142, 146  
 DATA= option, 138  
 ITPRINT option, 142  
 MAXITER= option, 142, 146  
 NBITER= option, 147  
 NIMPUTE= option, 139  
 NITER= option, 147  
 NOPRINT option, 139  
 OUT= option, 139  
 OUTEM= option, 142  
 OUTEST= option, 147  
 OUTITER= option, 142, 147  
 SEED option, 140  
 SIMPLE, 140  
 SINGULAR option, 140  
 MI procedure, PROC MIANALYZE statement  
   DATA= option, 207  
 MI procedure, TRANSFORM statement, 150  
   BOXCOX transformation, 150  
   C= option, 151  
   EXP transformation, 150  
   LAMBDA= option, 151  
   LOG transformation, 150  
   LOGIT transformation, 150  
   POWER transformation, 150  
 MI procedure, VAR statement, 151, 209  
 MIANALYZE procedure, BY statement, 208  
 MIANALYZE procedure, PROC MIANALYZE statement, 207  
 MODEL statement  
   GAM procedure, 30  
   LIFEREG procedure, 80  
 MONOTONE statement  
   MI procedure, 149

## N

NAMELEN= option  
   PROC LIFEREG statement, 78  
   PROC PROBIT statement, 242  
 NBITER= option  
   PROC MI statement, 147  
 NIMPUTE= option  
   PROC MI statement, 139  
 NITER= option  
   PROC MI statement, 147  
 NOOVERLAYLEGEND option  
   PLOT statement (BOXPLOT), 3  
 NOPRINT option  
   PROC MI statement, 139  
 NOTEST option  
   MODEL statement (GAM), 32  
 NPAR1WAY procedure  
   syntax, 237  
 NPAR1WAY procedure, EXACT statement, 237  
   KS option, 237  
   POINT option, 237

**O**

## options

- CDFPLOT statement (PROBIT), 243
- IPPPLOT statement (PROBIT), 253
- LPREDPLOT statement (PROBIT), 261
- PREDPPLOT statement (PROBIT), 270
- ORDER= option
  - MODEL statement (LOGISTIC), 118
- OUT= option
  - OUTPUT statement (GAM), 32
  - PROC MI statement, 139
  - SCORE statement (GAM), 33
- OUTEM= option
  - PROC MI statement, 142
- OUTEST= option
  - PROC MI statement, 147
- OUTITER= option
  - PROC MI statement, 142, 147
- OUTPUT statement
  - GAM procedure, 32
  - LIFEREG procedure, 80
- OVERLAY= option
  - PLOT statement (BOXPLOT), 3
- OVERLAYHTML= option
  - PLOT statement (BOXPLOT), 4
- OVERLAYLEGLAB= option
  - PLOT statement (BOXPLOT), 4
- OVERLAYSYM= option
  - PLOT statement (BOXPLOT), 4
- OVERLAYSYMHT= option
  - PLOT statement (BOXPLOT), 4

**P**

- PARTIALR2 option
  - MODEL statement (REG), 301
- POINT option
  - EXACT statement (FREQ), 19
  - EXACT statement (NPAR1WAY), 237
- POWER transformation
  - TRANSFORM statement (MI), 150
- PREDPPLOT statement
  - options summarized by function, 270
  - PROBIT procedure, 269
- PROBIT procedure
  - syntax, 241
- PROBIT procedure, CDFPLOT statement, 242
  - ANNOTATE= option, 246
  - CAXIS= option, 246
  - CFIT= option, 246
  - CFRAME= option, 246
  - CGRID= option, 246
  - CHREF= option, 246
  - CLABBOX= option, 246
  - CTEXT= option, 246
  - CVREF= option, 246
  - DESCRIPTION= option, 246
  - FONT= option, 247
  - HAXIS= option, 247
  - HEIGHT= option, 247

- HLOWER= option, 247
- HOFFSET= option, 247
- HREF= option, 247
- HREFLABELS= option, 247
- HREFLABPOS= option, 248
- HUPPER= option, 247
- INBORDER option, 248
- LEVEL option, 248
- LFIT option, 248
- LGRID option, 248
- LHREF= option, 248
- LVREF= option, 248
- NAME= option, 248
- NOFIT option, 248
- NOFRAME option, 248
- NOGRID option, 248
- NOHLABEL option, 248
- NOHTICK option, 249
- NOTHRESH option, 249
- NOVLABEL option, 249
- NOVTICK option, 249
- options, 243
- THRESHLABPOS= option, 249
- VAR= option, 242
- VAXIS= option, 249
- VAXISLABEL= option, 249
- VLOWER= option, 249
- VREF= option, 249
- VREFLABELS= option, 249
- VREFLABPOS= option, 250
- VUPPER= option, 250
- WAXIS= option, 250
- WFIT= option, 250
- WGRID= option, 250
- WREFL= option, 250
- PROBIT procedure, INSET statement, 250–251
  - keywords, 251
- PROBIT procedure, IPPPLOT statement, 252
  - ANNOTATE= option, 256
  - CAXIS= option, 256
  - CFIT= option, 256
  - CFRAME= option, 256
  - CGRID= option, 256
  - CHREF= option, 256
  - CTEXT= option, 256
  - CVREF= option, 256
  - DESCRIPTION= option, 256
  - FONT= option, 256
  - HAXIS= option, 257
  - HEIGHT= option, 257
  - HLOWER= option, 257
  - HOFFSET= option, 257
  - HREF= option, 257
  - HREFLABELS= option, 257
  - HREFLABPOS= option, 257
  - HUPPER= option, 257
  - INBORDER option, 258
  - LFIT option, 258
  - LGRID option, 258



- LHREF= option, 258
- LVREF= option, 258
- NAME= option, 258
- NOCONF option, 258
- NODATA option, 258
- NOFIT option, 258
- NOFRAME option, 258
- NOGRID option, 258
- NOHLABEL option, 258
- NOHTICK option, 258
- NOTHRESH option, 259
- NOVLABEL option, 259
- NOVTICK option, 259
- options, 253
- THRESHLABPOS= option, 259
- VAR= option, 252
- VAXIS= option, 259
- VAXISLABEL= option, 259
- VLOWER= option, 259
- VREF= option, 259
- VREFLABELS= option, 259
- VREFLABPOS= option, 260
- VUPPER= option, 260
- WAXIS= option, 260
- WFIT= option, 260
- WGRID= option, 260
- WREFL= option, 260
- PROBIT procedure, LPREDPLOT statement, 260
  - ANNOTATE= option, 263
  - CAXIS= option, 263
  - CFIT= option, 263
  - CFRAME= option, 263
  - CGRID= option, 264
  - CHREF= option, 264
  - CTEXT= option, 264
  - CVREF= option, 264
  - DESCRIPTION= option, 264
  - FONT= option, 264
  - HAXIS= option, 264
  - HEIGHT= option, 264
  - HLOWER= option, 264
  - HOFFSET= option, 264
  - HREF= option, 265
  - HREFLABELS= option, 265
  - HREFLABPOS= option, 265
  - HUPPER= option, 265
  - INBORDER option, 265
  - LEVEL option, 265
  - LFIT option, 265
  - LGRID option, 265
  - LHREF= option, 266
  - LVREF= option, 266
  - NAME= option, 266
  - NOCONF option, 266
  - NODATA option, 266
  - NOFIT option, 266
  - NOFRAME option, 266
  - NOGRID option, 266
  - NOHLABEL option, 266
  - NOHTICK option, 266
  - NOTHRESH option, 266
  - NOVLABEL option, 266
  - NOVTICK option, 266
  - options, 261
  - THRESHLABPOS= option, 266
  - VAR= option, 260
  - VAXIS= option, 267
  - VAXISLABEL= option, 267
  - VLOWER= option, 267
  - VREF= option, 267
  - VREFLABELS= option, 267
  - VREFLABPOS= option, 267
  - VUPPER= option, 267
  - WAXIS= option, 267
  - WFIT= option, 268
  - WGRID= option, 268
  - WREFL= option, 268
- PROBIT procedure, MODEL statement, 268
  - AGGREGATE= option, 268
  - ALPHA= option, 268
  - CONVERGE= option, 268
  - SCALE= option, 268
- PROBIT procedure, PREDPLOT statement
  - LEVEL option, 275
- PROBIT procedure, PREDPLOT statement, 269
  - ANNOTATE= option, 273
  - CAXIS= option, 273
  - CFIT= option, 273
  - CFRAME= option, 273
  - CGRID= option, 273
  - CHREF= option, 273
  - CTEXT= option, 273
  - CVREF= option, 273
  - DESCRIPTION= option, 273
  - FONT= option, 274
  - HAXIS= option, 274
  - HEIGHT= option, 274
  - HLOWER= option, 274
  - HOFFSET= option, 274
  - HREF= option, 274
  - HREFLABELS= option, 274
  - HREFLABPOS= option, 274
  - HUPPER= option, 274
  - INBORDER option, 275
  - LFIT option, 275
  - LGRID option, 275
  - LHREF= option, 275
  - LVREF= option, 275
  - NAME= option, 275
  - NOCONF option, 275
  - NODATA option, 275
  - NOFIT option, 275
  - NOFRAME option, 275
  - NOGRID option, 276
  - NOHLABEL option, 276
  - NOHTICK option, 276
  - NOTHRESH option, 276
  - NOVLABEL option, 276
  - NOVTICK option, 276

NOVTICK option, 276  
 options, 270  
 THRESHLABPOS= option, 276  
 VAR= option, 269  
 VAXIS= option, 276  
 VAXISLABEL= option, 276  
 VLOWER= option, 276  
 VREF= option, 276  
 VREFLABELS= option, 277  
 VREFLABPOS= option, 277  
 VUPPER= option, 277  
 WAXIS= option, 277  
 WFIT= option, 277  
 WGRID= option, 277  
 WREFL= option, 277  
 PROBIT procedure, PROC PROBIT statement, 241  
   GOUT= option, 241  
   INEST= option, 241  
   NAMELEN= option, 242  
   XDATA= option, 242  
 PROBLOT statement  
   LIFEREG procedure, 81  
 PROC GAM statement  
   See GAM procedure  
 PROC LIFEREG statement  
   See LIFEREG procedure  
 PROC MI statement  
   See MI procedure  
 PROC MIANALYZE statement  
   See MIANALYZE procedure  
 PROC PROBIT statement  
   See PROBIT procedure  
 PROC SURVEYMEANS statement  
   See SURVEYMEANS procedure

**R**

RATIO statement  
   SURVEYMEANS procedure, 307  
 REFERENCE= option  
   MODEL statement (LOGISTIC), 119  
 REG procedure, MODEL statement  
   PARTIALR2 option, 301  
   SCORR1 option, 301  
   SCORR2 option, 301  
 RISKDIFFC option  
   TABLES statement (FREQ), 20  
 ROUND option  
   PROC FACTOR statement, 15

**S**

SCALE= option  
   MODEL statement (PROBIT), 268  
 SCORE statement, GAM procedure, 33  
 SCORR1 option  
   MODEL statement (REG), 301  
 SCORR2 option  
   MODEL statement (REG), 301  
 SEED option  
   PROC MI statement, 140

SELECT= option  
   MODEL statement (LOESS), 107  
 SIMPLE option  
   PROC MI statement, 140  
 SINGULAR option  
   PROC MI statement, 140  
 SURVEYMEANS procedure  
   PROC SURVEYMEANS statement, 305  
   RATIO statement, 307  
   syntax, 305

**T**

TABLES statement  
   FREQ procedure, 20  
 TRANSFORM statement  
   MI procedure, 150

**V**

VAR statement  
   MI procedure, 151  
   MIANALYZE procedure, 209  
 VAR= option  
   CDFPLOT statement (PROBIT), 242  
   IPPPLOT statement (PROBIT), 252  
   LPREDPLOT statement (PROBIT), 260  
   PREDPPLOT statement (PROBIT), 269

**W**

WOVERLAY= option  
   PLOT statement (BOXPLOT), 4

**X**

XDATA= option  
   PROC LIFEREG statement, 78  
   PROC PROBIT statement, 242

# Your Turn

---

If you have comments or suggestions about *SAS/STAT® Software: Changes and Enhancements, Release 8.2*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute Inc.  
SAS Publishing  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [yourturn@sas.com](mailto:yourturn@sas.com)

For suggestions about the software, please return the photocopy to

SAS Institute Inc.  
Technical Support Division  
SAS Campus Drive  
Cary, NC 27513  
**E-mail:** [suggest@sas.com](mailto:suggest@sas.com)



*Welcome \* Bienvenue \* Willkommen \* Yohkoso \* Bienvenido*

## **SAS® Institute Publishing Is Easy to Reach**

**Visit our Web page located at [www.sas.com/pubs](http://www.sas.com/pubs)**

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

**Explore all the services that SAS Institute Publishing has to offer!**

### **Your Listserv Subscription Automatically Brings the News to You**

Do you want to be among the first to learn about the latest books and services available from SAS Institute Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS release(s) that each book addresses.

To subscribe,

1. Send an e-mail message to **[listserv@vm.sas.com](mailto:listserv@vm.sas.com)**.
2. Leave the "Subject" line blank.
3. Use the following text for your message:

**subscribe NEWDOCNEWS-L *your-first-name your-last-name***

For example: subscribe NEWDOCNEWS-L John Doe

## Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at [www.sas.com/selecttext](http://www.sas.com/selecttext), or contact our SelecText coordinators by sending e-mail to [selecttext@sas.com](mailto:selecttext@sas.com).

## You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at [www.sas.com/pubs](http://www.sas.com/pubs) for complete information.

## See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at [www.sas.com/obs](http://www.sas.com/obs). Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to [sasbbu@sas.com](mailto:sasbbu@sas.com) or call 919-677-8000, then press 1-6479.

## Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

---

SAS Institute  
SAS Campus Drive  
Cary, NC 27513-2414  
Fax 919-677-4444

E-mail: [sasbook@sas.com](mailto:sasbook@sas.com)  
Web page: [www.sas.com/pubs](http://www.sas.com/pubs)  
To order books, call Fulfillment Services at 800-727-3228\*  
For other SAS Institute business, call 919-677-8000\*

\* **Note:** Customers outside the U.S. should contact their local SAS office.