SAS/IML® Software: Changes and Enhancements, Release 8.1

Table of Contents

Chapter 1
Changes and Enhancements

Chapter Table of Contents

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.

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.

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

$$D_{\mathrm{mod}} = \frac{\sum_{k=1}^{K} t_k \frac{c(k)}{(1+y)^{t_k}}}{P(1+y)}$$

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*:

$$D_{\mathrm{Mac}} = \frac{\sum_{k=1}^{K} t_k \frac{c(k)}{(1+y)^{t_k}}}{P}$$

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

$$f_1 = s_1$$

$$f_i = \left(\frac{(1 + s_i)^{t_i}}{(1 + s_{i-1})^{t_{i-1}}} \right)^{\frac{1}{t_i - t_{i-1}}} - 1; \quad i = 2, \ldots, n$$

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, $\{c(k)\}k = 1, \ldots, K$ 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

$$s_1 = f_1$$

$$s_i = \left(\Pi_{j=1}^{j=i} (1 + f_j)^{t_j - t_{j-1}} \right)^{\frac{1}{t_i}} - 1; \quad i = 2, \ldots, n$$

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.

times   is an n-dimensional column vector of times. Elements should be non-negative.

flows   is an n-dimensional 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. No negative values are allowed.

value   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, $\{c(k)\}k = 1, ..., K$ 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 $[(n + p + 1)/2] \le h \le n$. 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 $h = [.75n]$.

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, $p \ge 2$. 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_{merged}. In the merged set, for each of the $5 \times 10 = 50$ best solutions, carry out two C-steps using n_{merged} and $h_{\text{merged}} = [n_{\text{merged}}(h/n)]$ 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

                         Approx           Pr >
   Variable     Estimate  Std Err  t Value    |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

                                Best
       Subset     Singular     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        47.916666667
```

```
                    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      0.5833333333      4.3333333333

                                                              (replaced)
--------------------------------------------------------------------------

                        Resistant Diagnostic

                                    Resistant
            N               U       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


                            Approx              Pr >
   Variable      Estimate    Std Err  t Value    |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                          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


                            Approx            Pr >
    Variable     Estimate   Std Err  t Value    |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


                        Approx              Pr >
  Variable      Estimate    Std Err  t Value    |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 re-
quest 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 optimiza-
tion 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]= $h = \left[\frac{N+n+1}{2}\right]$. 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_{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 sub-
set, 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_{lower}	500	50	22	17	15	14	0	0	0	0

n	11	12	13	14	15
N_{lower}	0	0	0	0	0

If the number of cases (observations) N is smaller than N_{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
                 Mahalanobis              Robust
             N       Distances          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, Φ, 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

$$m_p = nrow(phi)/ncol(phi)$$

where *nrow(phi)* is the number of rows of the matrix *phi* and *ncol(phi)* is the number of columns of the matrix *phi*.

If you do not specify *p*, the default subset is $p = \{1, 2, \ldots, m_p\}$.

For example, consider a four-dimensional vector time series, and *phi* is a 4×4 matrix. If you specify *p*=1 (the default, since $m_p = 4/4 = 1$), the VARMACOV subroutine computes the theoretical cross-covariance matrices of VAR(1) as $\mathbf{y}_t = \Phi \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$.

If you specify *p*=2, the VARMACOV subroutine computes the cross-covariance matrices of VAR(2) as $\mathbf{y}_t = \Phi \mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$.

Let $phi = [\Phi'_1, \ \Phi'_2]'$ be an 8×4 matrix. If you specify $p = \{1, 3\}$, the VARMACOV subroutine computes the cross-covariance matrices of VAR(3) as $\mathbf{y}_t = \Phi_1 \mathbf{y}_{t-1} + \Phi_2 \mathbf{y}_{t-3} + \boldsymbol{\epsilon}_t$. If you do not specify *p*, the VARMACOV subroutine computes the cross-covariance matrices of VAR(2) as $\mathbf{y}_t = \Phi_1 \mathbf{y}_{t-1} + \Phi_2 \mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$.

q specifies the subset of the MA order. The quantity m_q is defined as

$$m_q = nrow(theta)/ncol(theta)$$

where *nrow(theta)* is the number of rows of matrix *theta* and *ncol(theta)* is the number of columns of matrix *theta*.

If you do not specify *q*, the default subset is $q = \{1, 2, \ldots, m_q\}$.

The usage of *q* is the same as that of *p*.

lag specifies the length of lags, which must be a positive number. If $lag = h$, the VARMACOV computes the cross-covariance matrices from lag zero to lag h. By default, $lag = 12$.

The VARMACOV subroutine returns the following value:

cov is a $k(lag + 1) \times k$ 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

$$\mathbf{y}_t = \Phi \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t - \Theta \boldsymbol{\epsilon}_{t-1}$$

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 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:

series 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.

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. 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 $k \times k$ covariance matrix of the innovation series. If you do not specify *sigma*, an identity matrix wis used.

p specifies the subset of the AR order. See the VARMACOV subroutine.

q specifies the subset of the MA order. See the VARMACOV subroutine.

opt specifies the method of computing the log-likelihood function:

opt=0 requests the multivariate innovations algorithm. This algorithm requires that the time series is stationary and does not contain missing observations.

opt=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.

opt=2 requests the Kalman filtering algorithm. This is the default and is used if the required conditions in *opt=0* and *opt=1* are not satisfied.

The VARMALIK subroutine returns the following value:

lnl   is a 3×1 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 VARMASIM(***series, phi, theta, mu, sigma, n <, p, q, initial, seed>***);**

The inputs to the VARMASIM 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 VARMACOV subroutine.

q   specifies the subset of the MA order. See the VARMACOV subroutine.

initial   specifies the initial values of random variables. If $initial = a_0$, then $\mathbf{y}_{-p+1}, \ldots, \mathbf{y}_0$ and $\boldsymbol{\epsilon}_{-q+1}, \ldots, \boldsymbol{\epsilon}_0$ 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 VARMASIM 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 $\boldsymbol{\mu}$, Σ, and Θ 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 μ and covariance matrix Σ, 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 VARMACOV subroutine.

q   specifies the subset of the MA order. See the VARMACOV 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 kq_{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

Syntax Index

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

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.**

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/selectext**, or contact our SelecText coordinators by sending e-mail to **selectext@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** or call 919-677-8000, then press 1-6479. See the SAS Institute Publishing Web page at **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**. 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** 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
Web page: 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.