Fitting Linear Models

Requires assumptions about ϵ_is. Usual assumptions:

1. $\epsilon_1, \ldots, \epsilon_n$ are independent. (Sometimes we assume only that $\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$; that is we assume the errors are **uncorrelated**.)
2. Homoscedastic errors; all variances are equal:

$$\mathrm{Var}(\epsilon_1) = \mathrm{Var}(\epsilon_2) = \cdots = \sigma^2$$

3. Normal errors: $\epsilon_i \sim N(0, \sigma^2)$.

Remember: we already have assumed $\mathrm{E}(\epsilon_i) = 0$.

Notes

- Assumptions 1, 2 and 3 permit **Maximum Likelihood Estimation**.
- Assumptions 1 and 2 justify **least squares**.
- Assumption 3 can be replaced by other error distributions, but not in this course.
- With normal errors maximum likelihood estimates are the same as least squares estimates.
- Assumption 2 — Homoscedastic errors — can be relaxed; see STAT 402 "Generalized Linear Models" or "weighted least square".
- Assumption 1 can be relaxed; see STAT 804 for Time Series models.

Motivation for Least Squares

Choose $\hat{\beta}$ to make **fitted values** $\hat{\mu} = X\hat{\beta}$ as close to Ys as possible.
There are many possible choices for “close”:

- Mean Absolute Deviation: minimize

$$\frac{1}{n}\sum |Y_i - \hat{\mu}_i|$$

- Least Median of Squares: minimize

$$\text{median}\{|Y_i - \hat{\mu}_i|^2\}$$

- Least squares: minimize

$$\sum (Y_i - \hat{\mu}_i)^2$$

WE DO LS = least squares.

To minimize $\sum(Y_i - \hat\mu_i)^2$ take derivatives with respect to each $\hat\beta_j$ and set them equal to 0:

$$\frac{\partial}{\partial\hat\beta_j}\sum_{i=1}^n (Y_i-\hat\mu_i)^2 = \sum_{i=1}^n \frac{\partial}{\partial\hat\beta_j}(Y_i-\hat\mu_i)^2$$
$$= \sum_{i=1}^n \left[\frac{\partial}{\partial\hat\mu_i}(Y_i-\hat\mu_i)^2\right]\frac{\partial\hat\mu_i}{\partial\hat\beta_j}$$

But

$$\frac{\partial}{\partial\hat\mu_i}(Y_i-\hat\mu_i)^2 = -2(Y_i-\hat\mu_i)$$

and

$$\hat\mu_i = \sum_{j=1}^p x_{ij}\hat\beta_j$$

so

$$\frac{\partial\hat\mu_i}{\partial\hat\beta_j} = x_{ij}$$

Thus

$$\frac{\partial}{\partial \hat\beta_j} \sum_{i=1}^n (Y_i - \hat\mu_i)^2 = -2\sum_{i=1}^n x_{ij}(Y_i - \hat\mu_i)$$

Normal Equations

Set this equal to 0; Get so-called **normal equations**:

$$\sum_{i=1}^{n} Y_i x_{ij} = \sum_{i=1}^{n} \hat{\mu}_i x_{ij} \qquad j = 1, \ldots, p$$

Finally remember that $\hat{\mu}_i = \sum_{k=1}^{p} x_{ik} \hat{\beta}_k$ to get

$$\sum Y_i x_{ij} = \sum_{i=1}^{n} \sum_{k=1}^{p} x_{ij} x_{ik} \hat{\beta}_k \tag{1}$$

- Formula looks dreadful
- but it's just a bunch of matrix-vector multiplications written out in summation notation.
- Note that it is a set of p linear equations in p unknowns $\hat{\beta}_1, \ldots, \hat{\beta}_p$.

Normal equations in vector matrix form

First

$$\sum_{i=1}^{n} x_{ij} x_{ik}$$

is the dot product between the jth and kth columns of X. Another way to view this is as the jkth entry in the matrix $X^T X$:

$$X^T X = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \cdots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \cdots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$$

The jkth entry in this matrix product is clearly

$$x_{1j}x_{1k} + x_{2j}x_{2k} + \cdots + x_{nj}x_{nk}$$

so that the right hand side of (1) is

$$\sum_{k=1}^{p}(X^TX)_{jk}\hat\beta_k$$

which is just the matrix product

$$((X^TX)\hat\beta)_j$$

Now look at the left hand side of (1), namely, $\sum_{i=1}^n Y_i x_{ij}$ which is just the dot product of Y and the jth column of X or the jth entry of $X^T Y$:

$$\begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \cdots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} x_{11}Y_1 + x_{21}Y_2 + \cdots + x_{n1}Y_n \\ \vdots \\ x_{1p}Y_1 + x_{2p}Y_2 + \cdots + x_{np}Y_n \end{bmatrix}$$

So the normal equations are

$$(X^T Y)_j = (X^T X \hat\beta)_j$$

or just

$$X^T Y = X^T X \hat\beta$$

Last formula is usual way to write the normal equations.

Solving Normal Equations for $\hat\beta$

Let's look at the dimensions of the matrices first.

- X^T is $p \times n$,
- Y is $n \times 1$,
- X^TX is a $p \times n$ matrix multiplied by a $n \times p$ matrix which just produces a $p \times p$ matrix.
- If the matrix X^TX has rank p then X^TX is not singular and its inverse $(X^TX)^{-1}$ exists. So solve

$$X^TY = X^TX\hat\beta$$

  for $\hat\beta$ by multiplying both sides by $(X^TX)^{-1}$ to get

$$\hat\beta = (X^TX)^{-1}X^TY$$

This is the ordinary least squares estimator.
See assignment 1 for an example with $\mathrm{rank}(X) < p$.
See chapter 5 in the text for a review of matrices and vectors.

Normal Equations for Simple Linear Regression

Thermoluminescence Example

See *Introduction* for the framework. Here I consider two models:

- a straight-line model,

$$Y_i = \beta_1 + \beta_2 D_i + \epsilon_i$$

- a quadratic model,

$$Y_i = \beta_1 + \beta_2 D_i + \beta_3 D_i^2 + \epsilon_i\,.$$

First, general theoretical formulas, then numbers and arithmetic:

$$X = \begin{bmatrix} 1 & D_1 \\ \vdots & \vdots \\ 1 & D_n \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1 & \cdots & 1 \\ D_1 & \cdots & D_n \end{bmatrix} \begin{bmatrix} 1 & D_1 \\ \vdots & \vdots \\ 1 & D_n \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^n D_i \\ \sum_{i=1}^n D_i & \sum_{i=1}^n D_i^2 \end{bmatrix}$$

$$(X^T X)^{-1} = \frac{1}{n \sum D_i^2 - (\sum D_i)^2} \begin{bmatrix} \sum_{i=1}^n D_i^2 & -\sum_{i=1}^n D_i \\ -\sum_{i=1}^n D_i & n \end{bmatrix}$$

$$X^T Y = \begin{bmatrix} 1 & \cdots & 1 \\ D_1 & \cdots & D_n \end{bmatrix} \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n Y_i \\ \sum_{i=1}^n D_i Y_i \end{bmatrix}$$

$$(X^T X)^{-1} X^T Y = \begin{bmatrix} \frac{\sum Y_i \sum D_i^2 - \sum D_i \sum D_i Y_i}{n \sum D_i^2 - (\sum D_i)^2} \\ \\ \frac{n \sum D_i Y_i - (\sum D_l)(\sum Y_i)}{n \sum D_i^2 - (\sum D_i)^2} \end{bmatrix}$$

So

$$(X^TX)^{-1}X^TY = \begin{bmatrix} \frac{\bar{Y}\sum D_i^2 - \bar{D}\sum D_i Y_i}{\sum(D_i - \bar{D})^2} \\ \\ \frac{\sum(D_i-\bar{D})(Y_i-\bar{Y})}{\sum(D_i-\bar{D})^2} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\bar{Y}\left[\sum D_i^2 - n\bar{D}^2\right] - \bar{D}\left[\sum D_i Y_i - \bar{D}\bar{Y}\right]}{n\sum D_i^2 - (\sum D_i)^2} \\ \\ \frac{S_{DY}}{S_{DD}} \end{bmatrix}$$

$$= \begin{bmatrix} \bar{Y} - \frac{S_{DY}}{S_{DD}}\bar{D} \\ \\ \frac{S_{DY}}{S_{DD}} \end{bmatrix}$$

The data are

Dose	Count
0	27043
0	26902
0	25959
150	27700
150	27530
150	27460
420	30650
420	30150
420	29480
900	34790
900	32020
1800	42280
1800	39370
1800	36200
3600	53230
3600	49260
3600	53030

The design matrix for the linear model is

$$X = \begin{bmatrix} 1 & 27043 \\ 1 & 26902 \\ 1 & 25959 \\ 1 & 27700 \\ 1 & 27530 \\ 1 & 27460 \\ 1 & 30650 \\ 1 & 30150 \\ 1 & 29480 \\ 1 & 34790 \\ 1 & 32020 \\ 1 & 42280 \\ 1 & 39370 \\ 1 & 36200 \\ 1 & 53230 \\ 1 & 49260 \\ 1 & 53030 \end{bmatrix}$$

- Compute $X^T X$ in `Minitab` or `Splus` or `R`.
- That matrix has 4 numbers each of which is computed as the dot product of 2 columns of X.
- For instance the first column dotted with itself produces $1^2 + \cdots + 1^2 = 17$.
- Here is an example S session which reads in the data, produces the design matrices for the two models and computes $X^T X$.

```
[36]skekoowahts% S
# The data are in a file called linear. The !
# tells S that what follows is not an S command but a standard
# UNIX (or DOS) command
#
> !more linear
  Dose Count
     0 27043
     0 26902
     0 25959
   150 27700
   150 27530
   150 27460
   420 30650
   420 30150
   420 29480
   900 34790
   900 32020
  1800 42280
  1800 39370
  1800 36200
  3600 53230
  3600 49260
  3600 53030
#
```

```
# The function help(function) produces help for
# a function such as
# > help(read.table)
#
# Read in the data from a file.  The file has 18 lines:
# 17 lines of data and a first line which has the names
# of the variables. The function read.table reads such
# data and header=T warns S that the first line is
# variable names.  The first argument of read.table is
# a character string containing the name of the file
# to read from.
#
> dat <- read.table("linear",header=T)
```

```
> dat
   Dose Count
 1    0 27043
 2    0 26902
 3    0 25959
 4  150 27700
 5  150 27530
 6  150 27460
 7  420 30650
 8  420 30150
 9  420 29480
10  900 34790
11  900 32020
12 1800 42280
13 1800 39370
14 1800 36200
15 3600 53230
16 3600 49260
17 3600 53030
#
```

```
# the design matrix has a column of 1s and also
# a column consisting of the first column of dat
# which is just the list of covariate values
# The notation dat[,1] picks out the first column of dat
#
> design.mat <- cbind(rep(1,17),dat[,1])
#
# To print out an object you type its name!
#
```

```
> design.mat
      [,1] [,2]
 [1,]    1    0
 [2,]    1    0
 [3,]    1    0
 [4,]    1  150
 [5,]    1  150
 [6,]    1  150
 [7,]    1  420
 [8,]    1  420
 [9,]    1  420
[10,]    1  900
[11,]    1  900
[12,]    1 1800
[13,]    1 1800
[14,]    1 1800
[15,]    1 3600
[16,]    1 3600
[17,]    1 3600
#
```

```
# Compute X^T X  -- uses %*% to multiply matrices
# and t(x) to compute the transpose of a matrix x.
#
> xprimex <- t(design.mat)%*% design.mat
> xprimex
      [,1]     [,2]
[1,]    17    19710
[2,] 19710 50816700
#
# Compute X^T Y
#
> xprimey <- t(design.mat)%*% dat[,2]
> xprimey
          [,1]
[1,]    593054
[2,] 882452100
```

```
#
# Next compute least squares estimates by solving
# normal equations
#
> solve(xprimex,xprimey)
             [,1]
[1,] 26806.734691
[2,]     6.968012
#
# solve(A,b) computes solution of Ax=b for A a
# square matrix and b a vector. Note x=A^{-1}b.
#
```

```
#
# The next piece of code regresses the variable
# Count on Dose taking the data from dat.
#
> lm( Count~Dose,data=dat)
Call:
lm(formula = Count ~ Dose, data = dat)

Coefficients:
 (Intercept)       Dose
    26806.73 6.968012

Degrees of freedom: 17 total; 15 residual
Residual standard error: 1521.238
#
# Notice the estimates agree with our calculations
# Residual standard error is usual estimate of sigma
# namely the square root of the Mean Square for Error.
#
```

```
#
# Now add a third column to fit the quadratic model
#
> design.mat2_cbind(design.mat,design.mat[,2]^2)
#
# Here is X^T X
#
> t(design.mat2)%*% design.mat2
         [,1]          [,2]         [,3]
[1,]       17         19710 5.081670e+07
[2,]    19710      50816700 1.591544e+11
[3,] 50816700 159154389000 5.367847e+14
```

```
#
# Here is X^T Y
#
> t(design.mat2)%*% dat[,2]
             [,1]
[1,] 5.930540e+05
[2,] 8.824521e+08
[3,] 2.469275e+12
#
# But the following illustrates the dangers
# of doing computations blindly on the computer.
# The trouble is that the design matrix has a
# third column which is so much larger that
# the first two.
#
> solve(t(design.mat2)%*% design.mat2,
        t(design.mat2)%*% dat[,2])
Error in solve.qr(a, b): apparently singular matrix
Dumped
```

```
#
# However, good packages know numerical techniques
#  which avoid the danger.
#
> lm(Count ~ Dose+Dose^2,data=dat)
Call:
lm(formula = Count ~ Dose + Dose^2, data = dat)

Coefficients:
 (Intercept)       Dose      I(Dose^2)
    26718.11 7.240314 -7.596867e-05

Degrees of freedom: 17 total; 14 residual
Residual standard error: 1571.277
```

```
#
# WARNING: you can't tell from the size of the
# estimate of an estimate such as that of beta_3
# whether or not it is important -- you have to
# compare it to values of the corresponding
# covariate and to its standard error
#
> q()
# Used to quit S:  pay attention to () --
#  that part is essential!
```