11.3: OLS Regression in Matrix Form

As was the case with simple regression, we want to minimize the sum of the squared errors, e. In matrix notation, the OLS model is $y=Xb+e$, where $e=y-Xb$. The sum of the squared e is:

$$\sum e_i^2 = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} = e'e \tag{11.1}$$

Therefore, we want to find the b that minimizes this function:

$$e'e=(y-Xb)'(y-Xb)=y'y-b'X'y-y'Xb+b'X'Xb=y'y-2b'X'y+b'X'Xb$$

To do this we take the derivative of $e'e$ w.r.t b and set it equal to 0.

$$\frac{\partial e'e}{\partial b}=-2X'y+2X'Xb=0$$ To solve this we subtract $2X'Xb$ from both sides: $$-2X'Xb=-2X'y$$

Then to remove the -2's, we multiply each side by $-1/2$. This leaves us with:

$$(X'X)b=X'y$$

To solve for b we multiply both sides by the inverse of $X'X,(X'X)^{-1}$. Note that for matrices this is equivalent to dividing each side by $X'X$. Therefore:

$$b=(X'X)^{-1}X'y \tag{11.2}$$

The $X'X$ matrix is square, and therefore invertible (i.e., the inverse exists). However, the $X'X$ matrix can be non-invertible (i.e., singular) if $n<k$—the number of k independent variables exceeds the n-size—or if one or more of the independent variables is perfectly correlated with another independent variable. This is termed perfect **multicollinearity** and will be discussed in more detail in Chapter 14. Also note that the $X'X$ matrix contains the basis for all the necessary means, variances, and covariances among the X's.

$$X'X=\begin{bmatrix} n & \sum X_1 & \sum X_2 & \sum X_3 \\ \sum X_1 & \sum X_1^2 & \sum X_1X_2 & \sum X_1X_3 \\ \sum X_2 & \sum X_2X_1 & \sum X_2^2 & \sum X_2X_3 \\ \sum X_3 & \sum X_3X_1 & \sum X_3X_2 & \sum X_3^2 \end{bmatrix}$$

Regression in Matrix Form

Assume a model using n observations, k parameters, and $k-1$, X_i (independent) variables.

$$y=Xb+e$$
$$\hat{y}=Xb$$
$$b=(X'X)^{-1}X'y$$

- $y=n*1$ column vector of observations of the DV, Y
- $\hat{y}=n*1$ column vector of predicted Y values
- $X=n*k$ matrix of observations of the IVs; first column 1s
- $b=k*1$ column vector of regression coefficients; first row is A
- $e=n*1$ column vector of n residual values

Using the following steps, we will use `R` to calculate b, a vector of regression coefficients; $\hat{y}$, a vector of predicted y values; and e, a vector of residuals.

We want to fit the model $y=Xb+e$ to the following matrices:

$$y=\begin{bmatrix} 6 \\ 11 \\ 4 \\ 3 \\ 5 \\ 9 \\ 10 \end{bmatrix} \quad X=\begin{bmatrix} 1 & 4 & 5 & 4 \\ 1 & 7 & 2 & 3 \\ 1 & 2 & 6 & 4 \\ 1 & 1 & 9 & 6 \\ 1 & 3 & 4 & 5 \\ 1 & 7 & 3 & 4 \\ 1 & 8 & 2 & 5 \end{bmatrix}$$

Create two objects, the y matrix and the X matrix.

```
y <- matrix(c(6,11,4,3,5,9,10),7,1)
y
```

```
##      [,1]
## [1,]    6
## [2,]   11
## [3,]    4
## [4,]    3
## [5,]    5
## [6,]    9
## [7,]   10
```

```
X <- matrix(c(1,1,1,1,1,1,1,4,7,2,1,3,7,8,5,2,6,9,4,3,2,4,3,4,6,5,4,5),7,4)
X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    5    4
## [2,]    1    7    2    3
## [3,]    1    2    6    4
## [4,]    1    1    9    6
## [5,]    1    3    4    5
## [6,]    1    7    3    4
## [7,]    1    8    2    5
```

Calculate bb: b=(X′X)−1X′yb=(X′X)−1X′y.

We can calculate this in `R` in just a few steps. First, we transpose XX to get X′X′.

```
X.prime <- t(X)
X.prime
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    1    1    1    1    1    1    1
## [2,]    4    7    2    1    3    7    8
## [3,]    5    2    6    9    4    3    2
## [4,]    4    3    4    6    5    4    5
```

Then we multiply XX by X′X′; (X′XX′X).

```
X.prime.X <- X.prime %*% X
X.prime.X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    7   32   31   31
## [2,]   32  192  104  134
## [3,]   31  104  175  146
## [4,]   31  134  146  143
```

Next, we find the inverse of X′XX′X; X′X−1X′X−1

```
X.prime.X.inv<-solve(X.prime.X)
X.prime.X.inv
```

```
##             [,1]        [,2]        [,3]        [,4]
## [1,] 12.2420551 -1.04528602 -1.01536017 -0.63771186
## [2,] -1.0452860  0.12936970  0.13744703 -0.03495763
## [3,] -1.0153602  0.13744703  0.18697034 -0.09957627
## [4,] -0.6377119 -0.03495763 -0.09957627  0.27966102
```

Then, we multiply X′X−1X′X−1 by X′X′.

```
X.prime.X.inv.X.prime<-X.prime.X.inv %*% X.prime
X.prime.X.inv.X.prime
```

```
##             [,1]        [,2]        [,3]       [,4]       [,5]       [,6]
## [1,]  0.43326271  0.98119703  1.50847458 -1.7677436  1.8561970 -0.6718750
## [2,]  0.01959746  0.03032309 -0.10169492  0.1113612 -0.2821769  0.1328125
## [3,]  0.07097458  0.02198093 -0.01694915  0.2073623 -0.3530191  0.1093750
## [4,] -0.15677966 -0.24258475 -0.18644068  0.1091102  0.2574153 -0.0625000
##             [,7]
## [1,] -1.33951271
## [2,]  0.08977754
## [3,] -0.03972458
## [4,]  0.28177966
```

Finally, to obtain the bb vector we multiply X′X−1X′X′X−1X′ by yy.

```
b<-X.prime.X.inv.X.prime %*% y
b
```

```
##             [,1]
## [1,]  3.96239407
## [2,]  1.06064619
## [3,]  0.04396186
## [4,] -0.48516949
```

We can use the `lm` function in `R` to check and see whether our “by hand” matrix approach gets the same result as does the “canned” multiple regression routine:

```
lm(y~0+X)
```

```
##
## Call:
## lm(formula = y ~ 0 + X)
##
## Coefficients:
##       X1        X2        X3        X4
##  3.96239   1.06065   0.04396  -0.48517
```

Calculate ^yy^: ^y=Xby^=Xb.

To calculate the ^yy^ vector in `R` , simply multiply `X` and `b` .

```
y.hat <- X %*% b
y.hat
```

```
##            [,1]
## [1,]  6.484110
## [2,] 10.019333
## [3,]  4.406780
## [4,]  2.507680
## [5,]  4.894333
## [6,]  9.578125
## [7,] 10.109640
```

Calculate ee.

To calculate ee, the vector of residuals, simply subtract the vector yy from the vector ^yy^.

```
e <- y-y.hat
e
```

```
##             [,1]
## [1,] -0.4841102
## [2,]  0.9806674
## [3,] -0.4067797
## [4,]  0.4923199
## [5,]  0.1056674
## [6,] -0.5781250
## [7,] -0.1096398
```