

## Multiple Regression in R

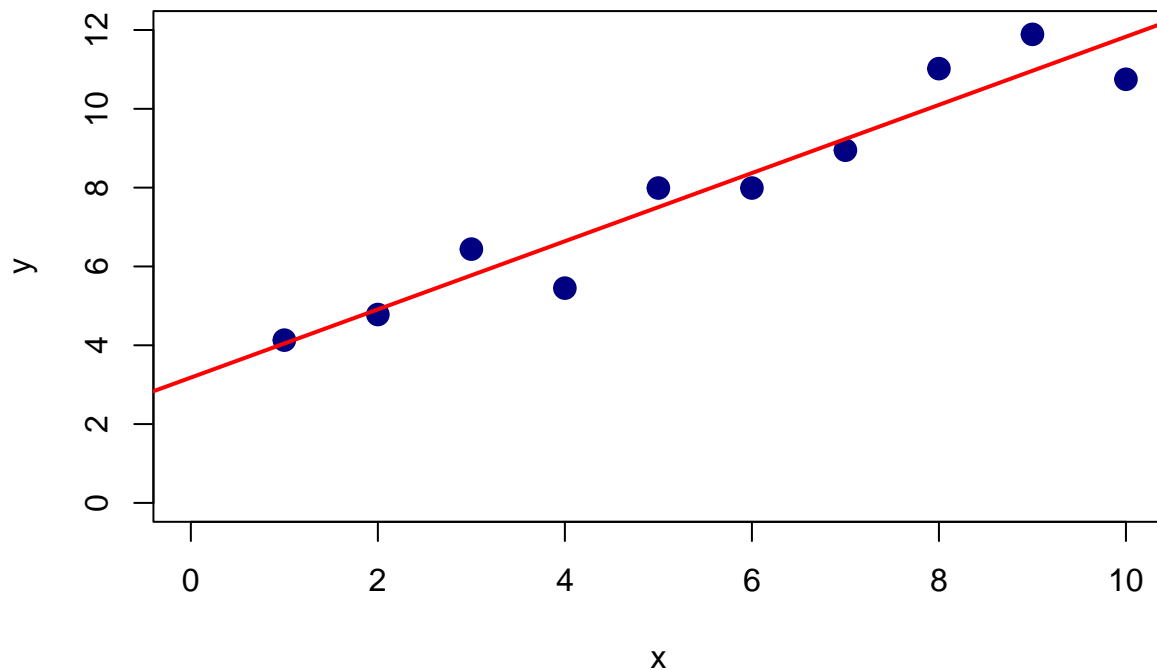
Again we the following observations  $Y_1, Y_2, \dots, Y_n$  following linear model  $E(Y) = \beta_0 + \beta_1 x$ :

$x$	1	2	3	4	5	6	7	8	9	10
$y$	4.13	4.78	6.44	5.45	7.99	7.99	8.95	11.02	11.89	10.75

In an earlier example, we fit the least squares model

$$\hat{y} = 3.18 + 0.865x$$

```
x <- 1:10
y <- c(4.13, 4.78, 6.44, 5.45, 7.99, 7.99, 8.95, 11.02, 11.89, 10.75)
plot(x, y, xlim = c(0,10), ylim = c(0,12), col = "navy", pch = 19, cex = 1.5)
abline(3.18, 0.865, col = "red", lwd = 2)
```



## Least Squares via Matrix Algebra

We could also have fit the least-squares line by solving the linear system

$$(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{Y},$$

where

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \hat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix}.$$

We can create the coefficient matrices  $\mathbf{X}'\mathbf{Y}$  and  $\mathbf{X}'\mathbf{X}$  using the following code.

```
# Store X as matrix.
n <- length(x)
X = cbind(rep(x = 1, times = n), x)

# Form coefficient matrices.
XX = t(X) %*% X
XY = t(X) %*% y
```

Here, the function `t(X)` yields the transpose of the matrix  $\mathbf{X}$  and the operator `%*%` is used to perform matrix-matrix multiplication. This gives the matrices:

$$\mathbf{X}'\mathbf{X} = \begin{bmatrix} 10 & 55 \\ 55 & 385 \end{bmatrix} \quad \mathbf{X}'\mathbf{Y} = \begin{bmatrix} 79.39 \\ 508.02 \end{bmatrix}$$

### Solving using Matrix Inversion

There are several equivalent processes for solving the linear system  $(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{Y}$  and, thus, calculating the least squares estimators. For example, we could calculate the inverse  $(\mathbf{X}'\mathbf{X})^{-1}$  and evaluate

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}.$$

We can calculate the inverse  $(\mathbf{X}'\mathbf{X})^{-1}$  using R function `solve` and calculate  $\hat{\boldsymbol{\beta}}$  using the following code.

```
# Calculate inverse.
invXX <- solve(XX)

# Solve for hatbeta
hBeta <- invXX %*% XY
```

$$(\mathbf{X}'\mathbf{X})^{-1} = \begin{bmatrix} 0.466666666666667 & -0.066666666666667 \\ -0.066666666666667 & 0.012121212121212 \end{bmatrix} \quad \hat{\boldsymbol{\beta}} = \begin{bmatrix} 3.18066666666667 \\ 0.865151515151515 \end{bmatrix}$$

Note that the entries of  $\hat{\boldsymbol{\beta}}$  agree with the values of the  $\hat{\beta}_0$  and  $\hat{\beta}_1$  calculated earlier.

### Solving via Gaussian Elimination

We can also solve the system  $(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{Y}$  by reducing the augmented matrix

$$[\mathbf{X}'\mathbf{X} \quad \mathbf{X}'\mathbf{Y}]$$

to reduced row echelon form using Gaussian elimination. The resulting matrix will have final column equal to  $\hat{\boldsymbol{\beta}}$ . The following code uses the R function `rref` (part of the `pracma` package) to perform the necessary elementary row operations.

```
# Load pracma package.
library(pracma)

# Form augmented matrix.
Ab <- cbind(XX, XY)

# Reduce to RREF.
rrefAB <- rref(Ab)
```

$$\begin{bmatrix} 10 & 55 & 79.39 \\ 55 & 385 & 508.02 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 3.180666666666667 \\ 0 & 1 & 0.865151515151515 \end{bmatrix}$$

Notice that the final column of the reduced matrix agrees with the value of  $\hat{\beta}$  calculated the using the **inv** function and the values of  $\hat{\beta}_0$  and  $\hat{\beta}_1$  calculated directly using the least-squares equations.

### Using the Solve Function

We can also use the **solve** function to solve the linear system for the least squares estimator using Gaussian elimination (without explicitly forming the inverse  $(\mathbf{X}'\mathbf{X})^{-1}$ . Indeed, the command **solve(A,b)** calls R's linear system solving routines (based on the LAPACK package <https://www.netlib.org/lapack/>) to solve the system  $\mathbf{Ax} = \mathbf{b}$ . This syntax differs from our earlier use of **solve(XX)**: the command **solve(A)** yields the inverse of the matrix **A** when given a single matrix argument as input. The following code specializes the process to solving the least-squares system.

```
hBetaS <- solve(XX, XY)
```

This yields

$$\hat{\beta} = \begin{bmatrix} 3.180666666666667 \\ 0.865151515151515 \end{bmatrix}$$

Note that this gives the same least-squares estimators as before (as should be expected).

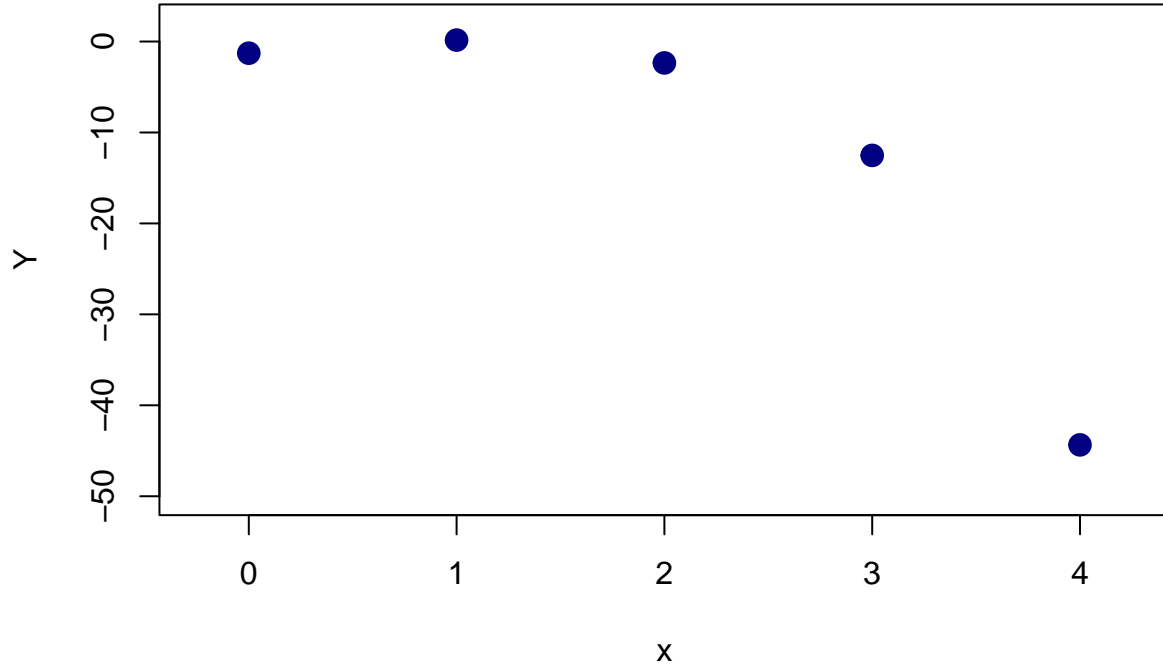
## A Nonlinear Example

Suppose that the response variable  $Y$  follows the model:

$$Y = \beta_0 + \beta_1 x + \beta_2 \sqrt{x} + \beta_3 e^x + \epsilon..$$

We have the following observations:

$x$	0	1	2	3	4
$y$	-1.28	0.16	-2.35	-12.52	-44.35



We can calculate the least squares estimates  $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3$  by solving the linear system

$$(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{Y},$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & \sqrt{x_1} & e^{x_1} \\ 1 & x_2 & \sqrt{x_2} & e^{x_2} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & \sqrt{x_n} & e^{x_n} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \hat{\boldsymbol{\beta}} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}.$$

The following code creates  $\mathbf{X}$  and  $\mathbf{Y}$  for the given data.

```
# Load data.
x2 <- 0:4
y2 <- c(-1.28, 0.16, -2.35, -12.52, -44.35)
n2 <- length(x2)
# Create X.
X2 <- cbind(rep(x = 1, times = n2), x2, sqrt(x2), exp(x2))
```

This gives following the matrix  $\mathbf{X}$  of independent variable values.

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 2.71828182845905 \\ 1 & 2 & 1.4142135623731 & 7.38905609893065 \\ 1 & 3 & 1.73205080756888 & 20.0855369231877 \\ 1 & 4 & 2 & 54.5981500331442 \end{bmatrix}.$$

After forming  $\mathbf{X}$ , we can create the coefficient matrices  $\mathbf{X}'\mathbf{X}$  and  $\mathbf{X}'\mathbf{Y}$  for the least squares linear system using the following code.

```
XX2 <- t(X2) %*% X2
XY2 <- t(X2) %*% y2
```

$$\mathbf{X}'\mathbf{X} = \begin{bmatrix} 5 & 10 & 6.14626436994197 & 85.7910248837216 \\ 10 & 30 & 17.0245795474528 & 296.14560492846 \\ 6.14626436994197 & 17.0245795474528 & 10 & 157.153455691253 \\ 85.7910248837216 & 296.14560492846 & 157.153455691253 & 3447.37398666654 \end{bmatrix}$$

$$\mathbf{X}'\mathbf{Y} = \begin{bmatrix} -60.34 \\ -219.5 \\ -113.548677982339 \\ -2691.10823298819 \end{bmatrix}$$

We're ready to calculate  $\hat{\beta}$  using the `solve` function.

```
hBetaN <- solve(XX2, XY2)
```

$$\hat{\beta} = \begin{bmatrix} -0.296697312837938 \\ 1.61004038580053 \\ 1.45214440571281 \\ -0.977750071506314 \end{bmatrix}$$

We can plot the curve corresponding to the least squares estimators using the following code.

```
# Plot observations.
plot(x = 0:4, xlab = "x",
     y = c(-1.28, 0.16, -2.35, -12.52, -44.35), ylab = "Y",
     xlim = c(-1/4, 4 + 1/4),
     ylim = c(-50, 2), col = "navy",
     pch = 19, cex = 1.5)

# Generate sequence of independent variables.
xs <- seq(from = 0, to = 4.25, by = 0.25)

# Predict points on curve using least-squares model
ys <- hBetaN[1] + hBetaN[2]*xs + hBetaN[3]*sqrt(xs) + hBetaN[4]*exp(xs)

# Plot least-squares curve
lines(xs, ys, col = "red", lwd = 2)
```

