**6.0 – Vectors and Matrices**

Formulating and fitting regression models in matrix notation is much cleaner and computationally efficient. When fitting regression models statistical software uses matrix algebra to fit the models. If you are unfamiliar with matrix algebra this section will certainly be challenging from a mathematical perspective. Fortunately moving forward we won’t need to worry about much of the mathematics presented here. To help digest this material, I strongly suggest you read A.6 in the Weisberg (3rd and 4th ed.) and/or section 7.9 in the Cook & Weisberg text. I will also give you a handout showing matrix calculations done in R including fitting a regression model using matrices.

**6.1 – Simple Linear Regression Model in Matrix Notation**

Given a random sample the data model for a simple linear regression where the mean function is given by we have

Using vectors and matrices we can write this model as:

or

Using OLS to obtain parameter estimates we choose to minimize the RSS:

Using vectors and matrices we can write the RSS as

Multiplying this expression out we have

Taking the derivative with respect to the parameter vector ( and setting the equation to zero will allow us to solve for that minimizes **RSS(**.

**OLS Estimate for**

Thus the **fitted values** are given by

The matrix is called the ***Hat Matrix*** because it puts the hat () on the

The **residuals** are given by

The RSS is given by

Thus the estimated conditional variance is given by

**Variances and Standard Errors**

We won’t see the mathematical details but one can show that

In case of simple linear regression where this will be a 2 x 2 matrix with the following elements.

The square roots of the diagonal elements of this matrix will give the standard errors for the both parameter estimates. The off-diagonal elements are equal, so this is a symmetric matrix, and give the covariance the parameter estimates. Recall the covariance only tells us whether the association is positive or negative. To understand this in the case of simple linear regression see diagrams below.

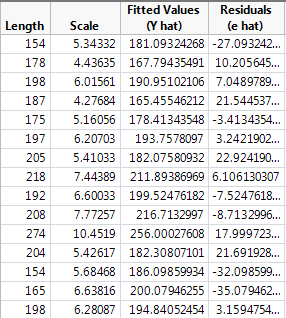
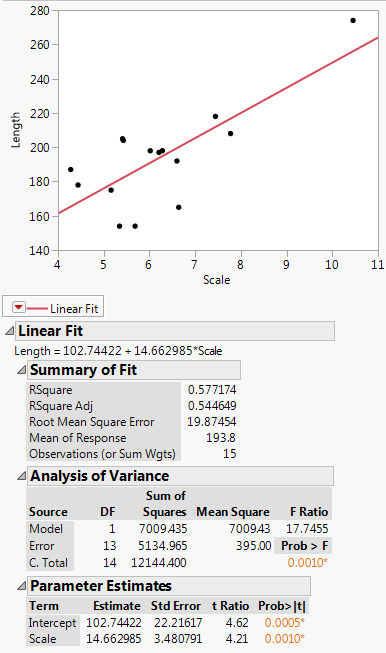
**Standard Errors for Confidence and Prediction**

For estimating the mean of the response given

For estimating the response value for an individual with

**Example 6.1: Length and Scale Radius of 4-year Old Smallmouth Bass**

Consider again the smallmouth bass data from West Bearskin Lake, but restricting our attention to only fish determined to be 4-years old. These data are given below. We will now use R to perform the matrix operations to obtain most the output shown in the regression summary from JMP. The model used will be and



> Bass4 = read.table(file.choose(),header=T,sep=",")

> names(Bass4)

[1] "Length" "Scale"

> dim(Bass4)

[1] 15 2

> Ones = rep(1,15)

> Ones

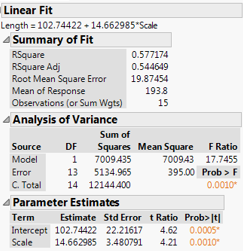
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

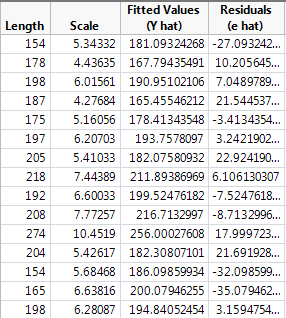
Form the model matrix () with a column of ones for the intercept and a column for scale radii.

> X = cbind(Ones,Scale=Bass4$Scale)

> X

Ones Scale





[1,] 1 5.34332

[2,] 1 4.43635

[3,] 1 6.01561

[4,] 1 4.27684

[5,] 1 5.16056

[6,] 1 6.20703

[7,] 1 5.41033

[8,] 1 7.44389

[9,] 1 6.60033

[10,] 1 7.77257

[11,] 1 10.45190

[12,] 1 5.42617

[13,] 1 5.68468

[14,] 1 6.63816

[15,] 1 6.28087

Form response vector ***Y***

> Y = Bass4$Length

Obtain parameter estimates

> betahat = solve((t(X)%\*%X))%\*%t(X)%\*%Y

> betahat

[,1]

Ones 102.74422

Scale 14.66299

> yhat = X%\*%betahat

> yhat

[,1]

[1,] 181.09

[2,] 167.79

[3,] 190.95

[4,] 165.46

[5,] 178.41

[6,] 193.76

[7,] 182.08

[8,] 211.89

[9,] 199.52

[10,] 216.71

[11,] 256.00

[12,] 182.31

[13,] 186.10

[14,] 200.08

[15,] 194.84

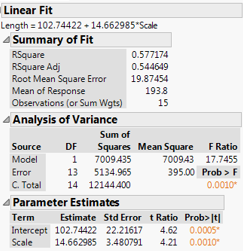
Using the Hat Matrix () and the fact that

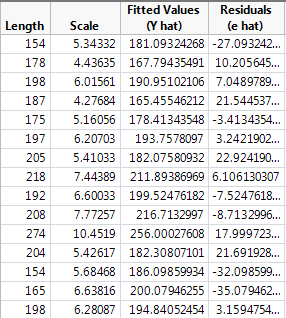
> H = X%\*%solve(t(X)%\*%X)%\*%t(X)

> yhat = H%\*%Y

> yhat

[,1]





[1,] 181.09

[2,] 167.79

[3,] 190.95

[4,] 165.46

[5,] 178.41

[6,] 193.76

[7,] 182.08

[8,] 211.89

[9,] 199.52

[10,] 216.71

[11,] 256.00

[12,] 182.31

[13,] 186.10

[14,] 200.08

[15,] 194.84

Form residual vector

> e = Y - yhat

> e

[,1]

[1,] -27.0932

[2,] 10.2056

[3,] 7.0490

[4,] 21.5445

[5,] -3.4134

[6,] 3.2422

[7,] 22.9242

[8,] 6.1061

[9,] -7.5248

[10,] -8.7133

[11,] 17.9997

[12,] 21.6919

[13,] -32.0986

[14,] -35.0795

[15,] 3.1595

Find

> RSS = t(e)%\*%e

> RSS

[,1]

[1,] 5134.9654

Find

> varhat = RSS/(15-2)

> varhat

[,1]

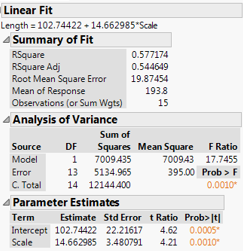
[1,] 394.99734

Find variance and standard errors of the parameter estimates

> varbetas = 394.997\*solve(t(X)%\*%X)

> varbetas

Ones Scale



Ones 493.557891 -75.238604

Scale -75.238604 12.115898

> SEbo = sqrt(493.558)

> SEb1 = sqrt(12.116)

> SEbo

[1] 22.216165

> SEb1

[1] 3.4808045

> lm1 = lm(Bass4$Length~Bass4$Scale)

> summary(lm1)

Call:

lm(formula = Bass4$Length ~ Bass4$Scale)

Residuals:

Min 1Q Median 3Q Max

-35.0795 -8.1190 3.2422 14.1027 22.9242

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 102.7442 22.2162 4.6247 0.0004758 \*\*\*

Bass4$Scale 14.6630 3.4808 4.2125 0.0010156 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 19.875 on 13 degrees of freedom

Multiple R-squared: 0.57717, Adjusted R-squared: 0.54465

F-statistic: 17.746 on 1 and 13 DF, p-value: 0.0010156

> SYY = sum((Y-mean(Y))^2)

> SYY

[1] 12144.4

> RSS

[,1]

[1,] 5134.9654

> Rsquare = (SYY-RSS)/SYY

> Rsquare

[,1]

[1,] 0.57717422

> sqrt(varhat)

[,1]

[1,] 19.87454

> Fstat = SSreg/varhat

> Fstat

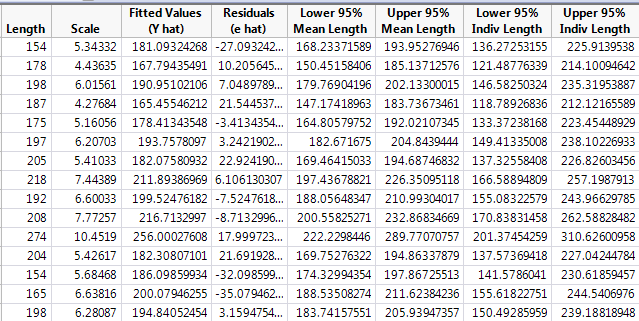
[,1]

[1,] 17.745523

> pf(17.7455,1,13,lower.tail=F)

[1] 0.0010155555

**Confidence and Prediction Intervals from JMP**



In R

> xnew = c(1,5.34332)

> yhatstar = xnew%\*%betahat

> yhatstar

[,1]

[1,] 181.09324

Compute

> sefit=sigmahat\*sqrt(t(xnew)%\*%solve(t(X)%\*%X)%\*%xnew)

> sefit

[,1]

[1,] 5.9524687

Find 🡨

> qt(.975,13)

[1] 2.1603687

Find LCL and UCL

> LCL = yhatstar - 2.1603687\*sefit

> UCL = yhatstar + 2.1603687\*sefit

> cbind(LCL,UCL)

[,1] [,2]

[1,] 168.23372 193.95277 🡨 this matches the results from JMP in the table above.

Compute

> sepred=sigmahat\*sqrt(1+t(xnew)%\*%solve(t(X)%\*%X)%\*%xnew)

> sepred

[,1]

[1,] 20.746788

Find LPL and UPL (P = Prediction)

> LPL = yhatstar - 2.1603687\*sepred

> UPL = yhatstar + 2.1603687\*sepred

> cbind(LPL,UPL)

[,1] [,2]

[1,] 136.27253 225.91395 🡨 this matches the results from JMP in the table above.