A2

Jefferson Li, Arib Shaikh

15/02/2021

Set Seed

```
set.seed(1005057368)
```

Q1

a)

Note that

$$\hat{\beta_1} \sim N(\beta_1, \sigma^2/Sxx)$$

We know $\beta_1 = 4, \sigma^2 = 25$.

We can calculate Sxx from the following code

```
Xs = c(4, 8, 12, 16, 20)
Sxx = 0
for (x in Xs) {
  Sxx = Sxx + (x - mean(Xs))^2
}
Sxx
```

```
## [1] 160
```

Therefore

$$\hat{\beta_1} \sim N(4, 25/160)$$

We want to calculate,

$$\begin{aligned} P(|\hat{\beta_1} - \beta_1| > 1) &= P(|\hat{\beta_1} - 4| > 1) && (1)\\ &= P(\hat{\beta_1} - 4 > 1 \quad \vee \quad \hat{\beta_1} - 4 < -1) && (2)\\ &= P(\hat{\beta_1} > 5) + P(\hat{\beta_1} < 3) \qquad \text{Since they are disjoint} && (3) \end{aligned}$$

This probability can be calculated in r with the following code

```
prob = pnorm(3, 4, sqrt(25/Sxx)) + (1-pnorm(5, 4, sqrt(25/Sxx)))
```

Therefore,

$$P(|\hat{\beta_1} - \beta_1| > 1) = 0.011412$$

b)

```
Xs = c(4, 8, 12, 16, 20)

errors = rnorm(5,0,5)
errors
```

```
## [1]  5.910658  3.719423 -4.960113 -1.684409  5.930103
```

```
Ys = rep(0, 5)
for (i in 1:length(Ys)) {
    Ys[i] = 20 + 4*Xs[i] + errors[i]
}
Ys
```

```
## [1]  41.91066  55.71942  63.03989  82.31559 105.93010
```

```
Sxy = 0
for (i in 1:length(Ys)) {
    Sxy = Sxy + (Xs[i] - mean(Xs))*(Ys[i] - mean(Ys))
}
Sxy
```

```
## [1] 618.5402
```

```
Bhat1 = Sxy / Sxx
Bhat0 = mean(Ys) - Bhat1*mean(Xs)
Bhat1
```

```
## [1] 3.865876
```

```
Bhat0
```

```
## [1] 23.39261
```

```
X0 = 10
Yhat0 =  Bhat0 + Bhat1 * X0
Yhat0
```

```
## [1] 62.05138
```

```
fit = lm(formula = Ys ~ Xs)
CI = predict(fit, data.frame(Xs=10), interval="confidence")
```

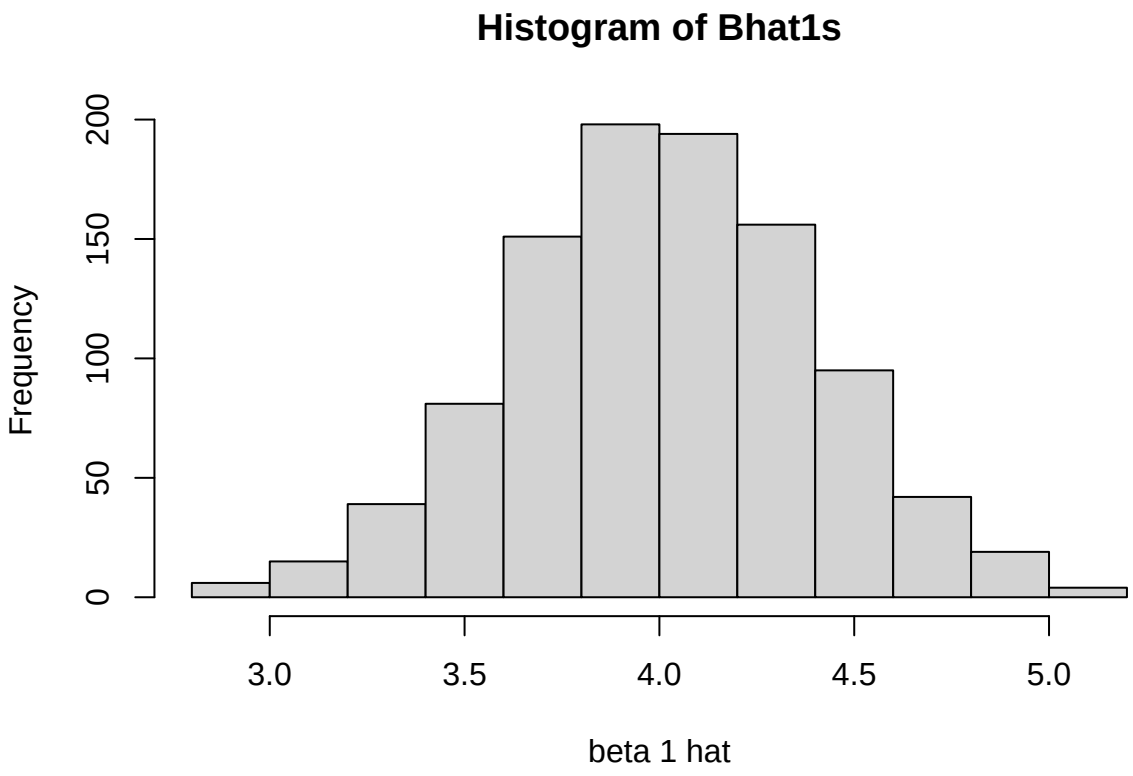
The 95% confidence interval for E(Y0) when X0 = 10 is given by

$$[53.6645619, 70.4381973]$$

c)

```r
set.seed(1005057368)
Bhat0s = rep(0,1000)
Bhat1s = rep(0,1000)
lowers = rep(0,1000)
uppers = rep(0,1000)

Xs = c(4, 8, 12, 16, 20)

for (i in 1:length(Bhat0s)) {
  errors = rnorm(5,0,5)

  Ys = rep(0, 5)
  for (j in 1:length(Ys)) {
      Ys[j] = 20 + 4*Xs[j] + errors[j]
  }
  fit = lm(formula = Ys ~ Xs)

  Bhat0s[i] = fit$coefficients[1]
  Bhat1s[i] = fit$coefficients[2]
  CI = predict(fit, data.frame(Xs=10), interval="confidence")
  lowers[i] = CI[2]
  uppers[i] = CI[3]
 }
```

```r
hist(Bhat1s, xlab = "beta 1 hat")
```



```
mean(Bhat1s)
```

```
## [1] 4.013382
```

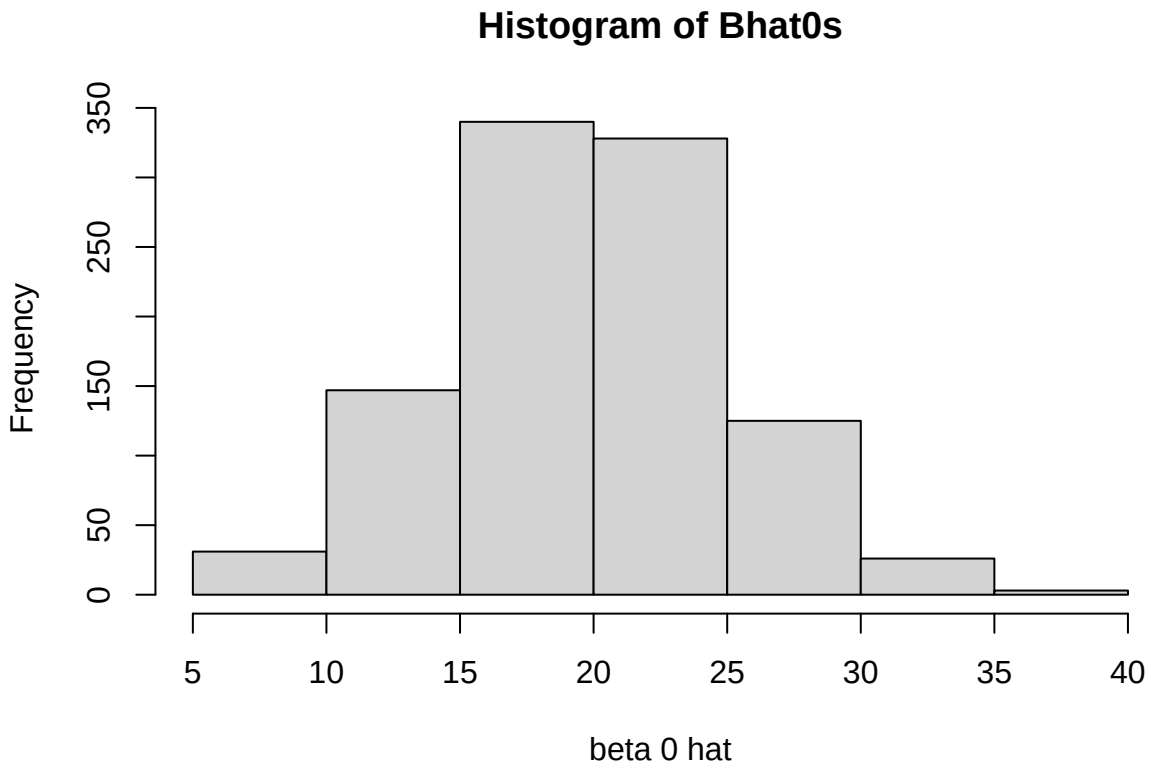
```
sd(Bhat1s)
```

```
## [1] 0.389623
```

The sample mean of 4.0133824 is consistent with the theoretical mean of 4.

The sample standard deviation of 0.389623 is consistent with the theoretical standard deviation of $\sqrt{\sigma^2/Sxx} = \sqrt{25/160} = 0.3952$.

```
hist(Bhat0s, xlab = "beta 0 hat")
```




```
mean(Bhat0s)
```

```
## [1] 19.83224
```

```
sd(Bhat0s)
```

```
## [1] 5.184326
```

The sample mean of 19.8322374 is consistent with the theoretical mean of 20.

The sample standard deviation of 5.1843258 is consistent with the theoretical standard deviation of $\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{Sxx}}\sigma = \sqrt{\frac{1}{5} + \frac{12^2}{160}}5 = 5.244$.

d)

```
numsWithinCI = 0

for (i in 1:length(lowers)) {
  Ey = 20 + 4*X0
  if (Ey > lowers[i] && Ey < uppers[i]){
    numsWithinCI = numsWithinCI + 1
  }
}
numsWithinCI
```

```
## [1] 951
```

This result of 951/1000 is consistent with the theoretical proportion of 950/1000

Q2

a)

```
Data <-read.csv("NHLhtwt.csv")
X <- Data$Height
Y <- Data$Weight
fit <- lm(Y~X)
summary(fit)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.109  -6.138  -0.131   6.876  39.876
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -163.3483    14.7026  -11.11   <2e-16 ***
## X              4.9927     0.2006   24.89   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.08 on 715 degrees of freedom
## Multiple R-squared:  0.4642, Adjusted R-squared:  0.4634
## F-statistic: 619.4 on 1 and 715 DF,  p-value: < 2.2e-16
```

```
b_0 = fit$coefficients[1]
b_0
```

```
## (Intercept)
##   -163.3483
```

```r
b_1 = fit$coefficients[2]
b_1
```

```
##        X
## 4.992666
```

```r
### By Hand Calcuation
x0 = 74
n=717
y0_hat = -163.348309 + 4.992666*(x0)
sxx = sum((X - mean(X))^2)
sigma_hat = 11.08
se = sigma_hat*sqrt(1/n + 1/sxx*(x0-mean(X))^2)
lower_int = y0_hat - qt(0.975, n-2)*se
lower_int
```

```
## [1] 205.246
```

```r
upper_int = y0_hat + qt(0.975, n-2)*se
upper_int
```

```
## [1] 206.9719
```

```r
#Using Built-in R Function

new_fit <- data.frame(X=74)
predict(fit, new_fit, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 206.1089 205.2458 206.9721
```

So we can conclude that our 95% Confidence Interval is [205.2458, 206.9721], and based on our calculations, we can see that the Confidence Interval calculation is the same whether we use by-hand, or the built-in R function.

b)

```r
# By "Hands"
spred = sigma_hat*sqrt(1+ 1/n + 1/sxx*(x0-mean(X))^2)
lower_int = y0_hat - qt(0.975, n-2)*spred
lower_int
```

```
## [1] 184.3386
```

```r
upper_int = y0_hat + qt(0.975, n-2)*spred
upper_int
```

```
## [1] 227.8793
```

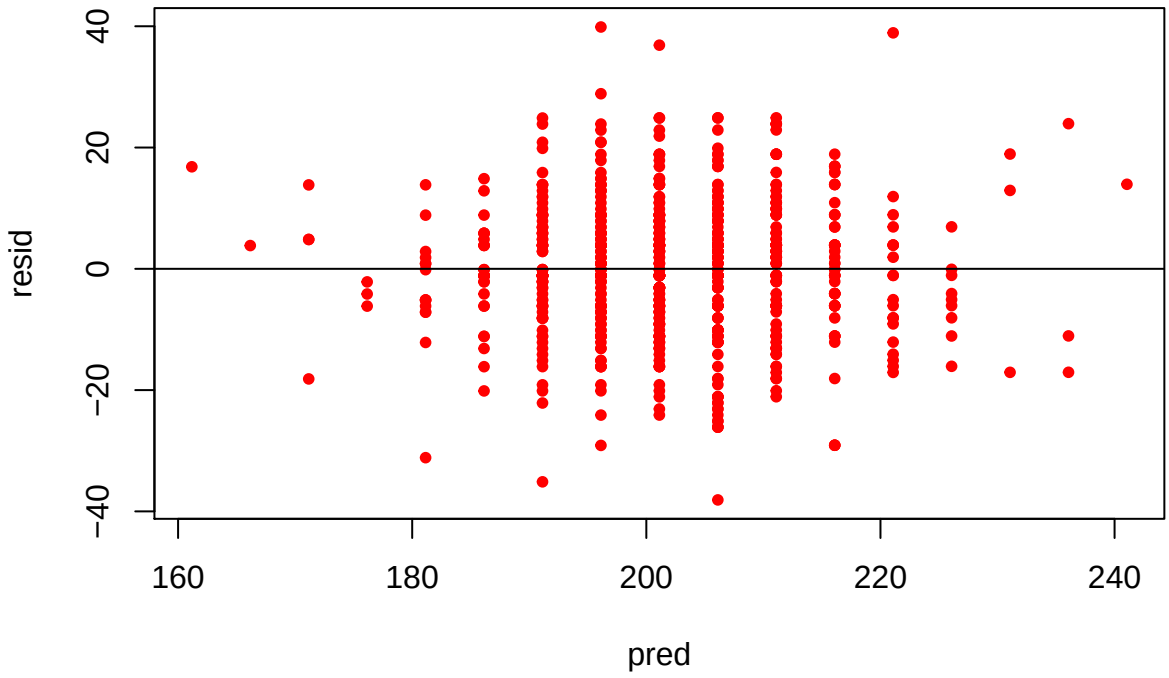
```r
#Using Built-in R function
predict(fit, new_fit, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 206.1089 184.3326 227.8853
```

So we can conclude that our prediction interval is [184.3326, 227.8853], and based on the values we got from using by hands and built-in R function, we ended up with the same value.

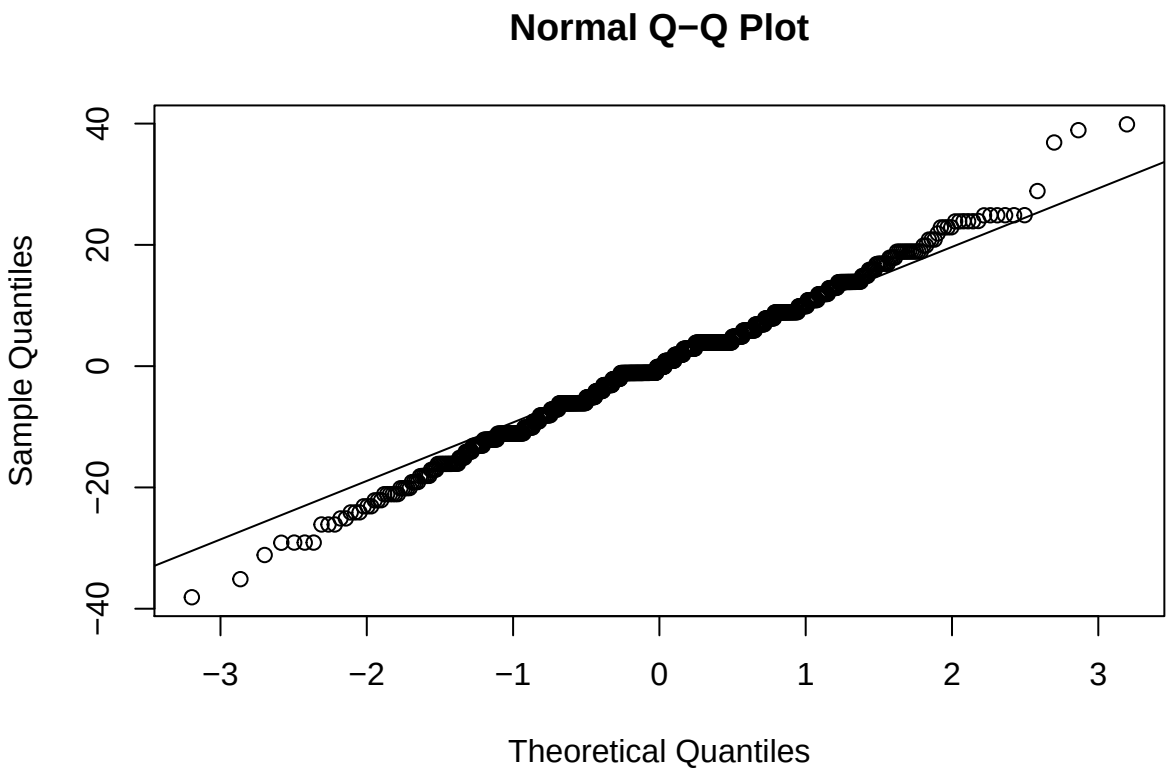
c)

```r
resid = fit$residuals
pred = fit$fitted.values
plot(pred, resid, pch=20, col="red")
abline(c(0,0))
```



d)

```r
qqnorm(resid)
qqline(resid)
```




```
shapiro.test(resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid
## W = 0.99407, p-value = 0.006511
```

Since we observed that the p-value of this is less than 0.005, by the Shapiro test, we reject the hypothesis that the data is normally distributed.

e)

```
btest <- function(resid_data, med_data){

resid1 <- subset(resid_data, Height>med_data)$resid
resid2 <- subset(resid_data, Height<=med_data)$resid
n1 = length(resid1)
n2 = length(resid2)

d1 = abs(resid1-median(resid1))
d2 = abs(resid2-median(resid2))
```

```
d1_mean = mean(d1)
d1_var = var(d1)

d2_mean = mean(d2)
d2_var = var(d2)

s2 = ((n1-1)*d1_var + (n2-1)*d2_var)/(n1+n2-2)

t1 = qt(0.975, n1+n2-2)

tBF = abs((d1_mean-d2_mean)/(sqrt(s2)*(sqrt((1/n1)+(1/n2)))))
return(list(tBF=tBF, t1=t1))

}
resid_data = data.frame(resid, Height = c(Data$Height))
med_data = median(Data$Height)
test <- btest(resid_data, med_data)
test$tBF
```

```
## [1] 2.00784
```

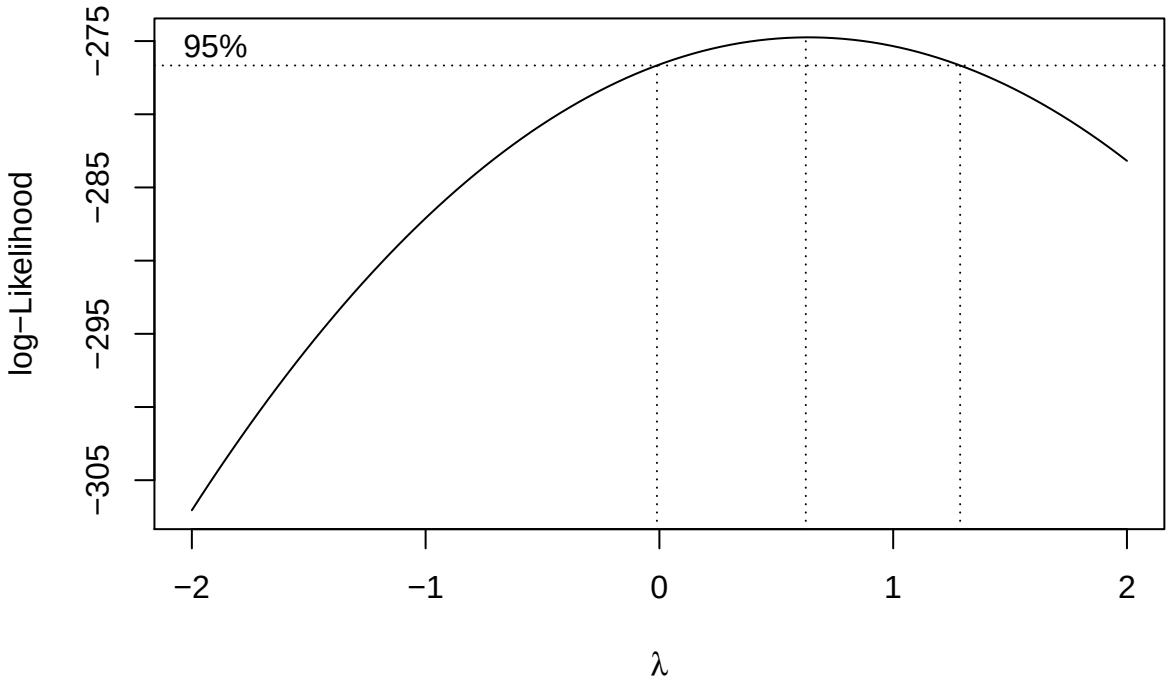
```
test$t1
```

```
## [1] 1.963287
```

As we can see, the tBF value is greater than our t-test, so we can reject our H0 that states that there is equal variance among errors.

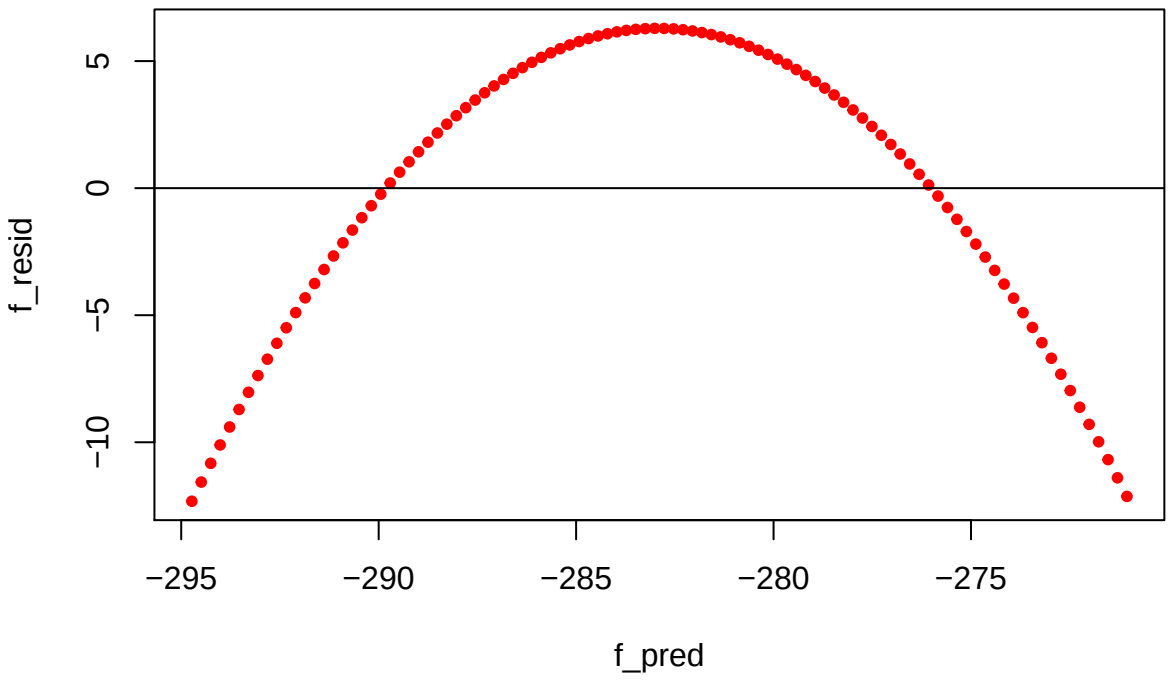
f)

```
library(MASS)
result = boxcox(fit)
```

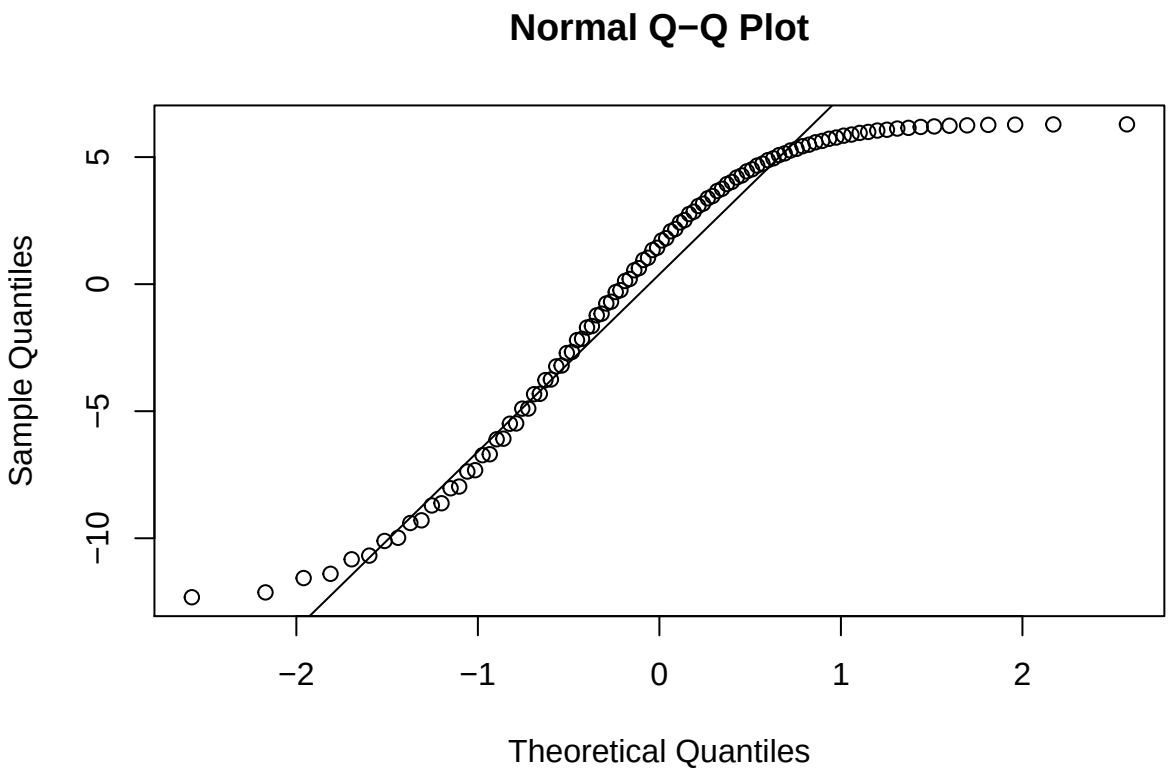



```
rfit <- lm(result$y~result$x)
```

```
f_resid = rfit$residuals
f_pred = rfit$fitted.values
plot(f_pred, f_resid, pch=20, col="red")
abline(c(0,0))
```




```
qqnorm(f_resid)
qqline(f_resid)
```




```
shapiro.test(f_resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  f_resid
## W = 0.8946, p-value = 7.945e-07
```

Since we observed that the p-value of this is less than 0.005, by the Shapiro test, we reject the hypothesis that the data is normally distributed.

```
f_resid_data = data.frame(resid = f_resid, Height = c(result$x))
f_med_data = median(result$x)
test <- btest(resid_data = f_resid_data, med_data = f_med_data)
test$tBF
```

```
## [1] 0.04996998
```

```
test$t1
```

```
## [1] 1.984467
```

Since we see that our tBF, 0.04996998 is less than 1.984467, we do not reject

$$H_0$$

, which states that there is equal variance among errors.

Q3

a)

Note that the least squares estimator of

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

We will use R to create and compute the matrices as given in the question.

```
X = matrix(
   c(1, -9,  3,
     1, -7, -7,
     1, -5,  7,
     1, -3, -9,
     1, -1,  1,
     1,  1,  5,
     1,  3, -1,
     1,  5, -3,
     1,  7, -5,
     1,  9,  9), # the data elements
   nrow=10,              # number of rows
   ncol=3,              # number of columns
   byrow = TRUE)        # fill matrix by rows `
#t(X)
Y = matrix(c(34,16,26,35,32,11,24,1,-3,15),nrow = 10,ncol=1, byrow=TRUE)

A = (1/1067840)*matrix(
  c(106784,   0,   0,
    0     ,3300,-460,
    0     ,-460,3300), nrow = 3, ncol = 3, byrow = TRUE)
betahat = A %*% t(X) %*% Y
betahat
```

```
##             [,1]
## [1,] 19.1000000
## [2,] -1.5215013
## [3,]  0.4151184
```

From this, we can see that our least square estimators,

$$\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2$$

are given by

$$\hat{\beta} = [19.1, -1.5215013, 0.4151184]$$

b)

Note that the formula to find the vector of fitted values

$$\hat{Y} = X\hat{\beta}$$

, so we will compute this first using R. After, to find an unbiased estimate of error variance, we use this formula:

$$\hat{\sigma^2} = 1/(n-3) * e'e$$

, where

$$e = Y - \hat{Y}$$

We will use R to compute as needed.

```
Yhat = X %*% betahat
Yhat
```

```
##             [,1]
##  [1,] 34.038867
##  [2,] 26.844681
##  [3,] 29.613335
##  [4,] 19.928439
##  [5,] 21.036620
##  [6,] 19.654091
##  [7,] 14.120378
##  [8,] 10.247138
##  [9,]  6.373899
## [10,]  9.142553
```

```
ei = Y - Yhat
var = (t(ei)%*%ei)
sigma2 = var/7
sigma2
```

```
##          [,1]
## [1,] 122.6003
```

Our fitted value vector is given by

$[34.0388672, 26.8446809, 29.6133353, 19.9284387, 21.0366197, 19.6540905, 14.1203776, 10.2471381, 6.3738987, 9.1425532]$

. Our unbiased estimate for error variance is 122.600274.

Q4

a)

1)

```
Xs = c(3,5,4,6,7)
Ys = c(4,6.5,5,7,7.5)
fit = lm(formula = Ys ~ Xs,x=TRUE)
designMatrix = matrix(model.matrix(fit), nrow = 5, ncol = 2, byrow = FALSE)
designMatrix
```

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

2)

```
coeffs = solve(t(designMatrix)%*%designMatrix)%*%t(designMatrix)%*%Ys
coeffs
```

```
##      [,1]
## [1,]  1.5
## [2,]  0.9
```

3)

```
sigma2=4
varCovarMatrix = solve(t(designMatrix)%*%designMatrix) *sigma2
varCovarMatrix
```

```
##      [,1] [,2]
## [1,] 10.8 -2.0
## [2,] -2.0  0.4
```

b)

1)

```
varCovarMatrix[1,2]
```

```
## [1] -2
```

2)

```
varCovarMatrix[1,1]
```

```
## [1] 10.8
```

3)

```
varCovarMatrix[2,2]
```

```
## [1] 0.4
```

c)

```
H = designMatrix%*%solve(t(designMatrix)%*%designMatrix)%*%t(designMatrix)
H
```

```
##               [,1] [,2]         [,3]         [,4]          [,5]
## [1,]  6.000000e-01  0.2 4.000000e-01 4.440892e-16 -2.000000e-01
## [2,]  2.000000e-01  0.2 2.000000e-01 2.000000e-01  2.000000e-01
## [3,]  4.000000e-01  0.2 3.000000e-01 1.000000e-01  3.330669e-16
## [4,]  4.440892e-16  0.2 1.000000e-01 3.000000e-01  4.000000e-01
## [5,] -2.000000e-01  0.2 4.440892e-16 4.000000e-01  6.000000e-01
```

d)

```
sigma2 = 4
VarE = (diag(5) - H)*sigma2
VarE
```

```
##               [,1] [,2]          [,3]          [,4]          [,5]
## [1,]  1.600000e+00 -0.8 -1.600000e+00 -1.776357e-15  8.000000e-01
## [2,] -8.000000e-01  3.2 -8.000000e-01 -8.000000e-01 -8.000000e-01
## [3,] -1.600000e+00 -0.8  2.800000e+00 -4.000000e-01 -1.332268e-15
## [4,] -1.776357e-15 -0.8 -4.000000e-01  2.800000e+00 -1.600000e+00
## [5,]  8.000000e-01 -0.8 -1.776357e-15 -1.600000e+00  1.600000e+00
```