CS-E-106: Data Modeling

Assignment 9

Instructor: Hakan Gogtas
Submitted by: *Saurabh Kulkarni*

Due Date: *12/09/2019*

Question 1 Refer to Employee salaries data. A group of high-technology companies agreed to share employee salary information in an effort to establish salary ranges for technical positions in research and development. Data obtained for each employee included current salary (Y), a coded variable indicating highest academic degree obtained (1 = bachelor's degree, 2 = master's degree; 3 = doctoral degree), years of experience since last degree (X3), and the number of persons currently supervised (X4). (40 pts)

(a) Create two indicator variables for highest degree attained: (5pts)

Solution

```
employee_data = read.csv("Employee  Salaries.csv")
employee_data$Degree = as.factor(employee_data$Degree)
employee_data["X1"] = as.factor(ifelse(employee_data$Degree==2, 1, 0))
employee_data["X2"] = as.factor(ifelse(employee_data$Degree==3, 1, 0))
summary(employee_data)
```

```
##        Y          Degree       X3              X4          X1     X2
##  Min.   : 29.00   1:16   Min.   : 0.140   Min.   : 0.000   0:38   0:43
##  1st Qu.: 40.90   2:27   1st Qu.: 2.260   1st Qu.: 0.000   1:27   1:22
##  Median : 55.90   3:22   Median : 5.180   Median : 0.000
##  Mean   : 60.02          Mean   : 7.959   Mean   : 3.446
##  3rd Qu.: 70.60          3rd Qu.:12.880   3rd Qu.: 5.000
##  Max.   :163.70          Max.   :29.540   Max.   :42.000
```

(b) Regress Y on X1, X2, X3 and X4, using a first-order model and ordinary least squares, obtain the residuals. and plot them against Yˆ. What does the residual plot suggest? (5pts)

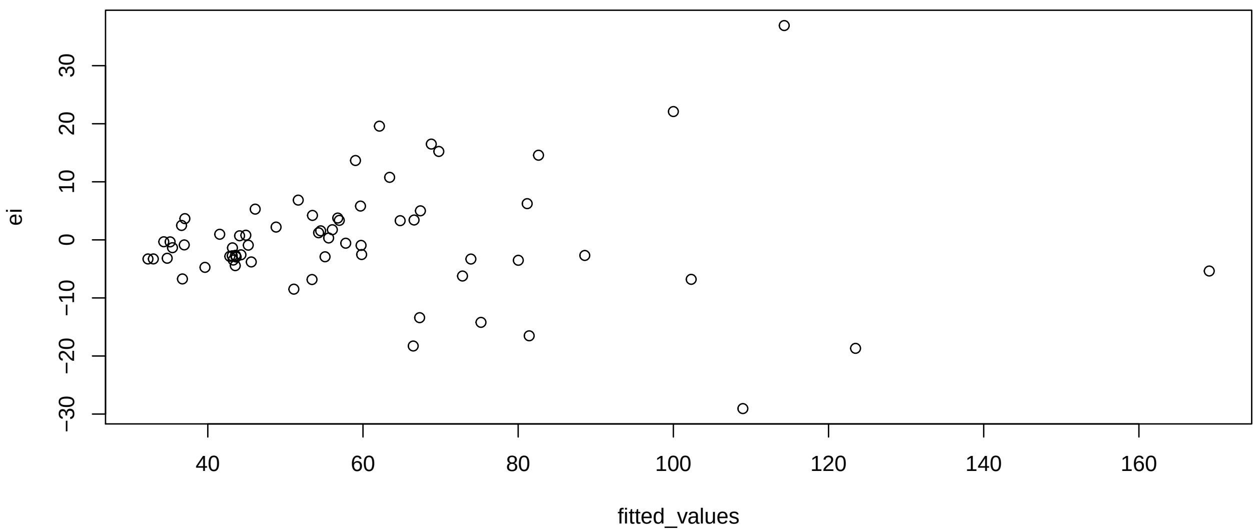
Solution

```
lm_employee = lm(Y~X1+X2+X3+X4, data=employee_data)
summary(lm_employee)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4, data = employee_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.058  -3.477  -0.915   3.417  36.909
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.4714     2.8691  10.969 5.73e-16 ***
## X11          10.8120     3.2183   3.360  0.00136 **
## X21          22.6307     3.4846   6.494 1.81e-08 ***
## X3            1.2581     0.2273   5.535 7.23e-07 ***
## X4            1.8523     0.2276   8.137 2.86e-11 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.14 on 60 degrees of freedom
## Multiple R-squared:  0.8633, Adjusted R-squared:  0.8542
## F-statistic: 94.76 on 4 and 60 DF,  p-value: < 2.2e-16
```

```r
ei = lm_employee$residuals
fitted_values = lm_employee$fitted.values
plot(fitted_values, ei)
```



Interpretation:

Residual plot shows a megaphone shape which impies non-constant variance. Also, it means that e_i regresses on $\hat{Y}$.

(c) Divide the cases into two groups, placing the 33 cases with the smallest fitted values (Y_i)ˆ into group 1 and the other 32 cases into group 2. Conduct the Brown-Forsythe test for constancy of the error variance, using = .01. State the decision rule and conclusion? (5 pts)

Brown-Forsythe Test

Null Hypothesis: H_0: Error variance is constant Alternate Hypothesis: H_1: Error variance is not constant

```r
ei = lm_employee$residuals
yHat = lm_employee$fitted.values
df = data.frame(cbind(employee_data,ei, yHat))
df = df[order(yHat),]
```

```r
df1 = df[1:33,]
df2 = df[34:nrow(df),]

med1 = median(df1[["ei"]])
med2 = median(df2[["ei"]])

#n1
n1 = nrow(df1)
print(n1)
```

```
## [1] 33
```

```r
#n2
n2 = nrow(df2)
print(n2)
```

```
## [1] 32
```

```r
d1 = abs(df1[["ei"]]-med1)
d2 = abs(df2[["ei"]]-med2)

#calculate means for our answer
mean_d1 = mean(d1)
print(mean_d1)
```

```
## [1] 2.759499
```

```r
mean_d2 = mean(d2)
print(mean_d2)
```

```
## [1] 10.11656
```

```r
s2 = (var(d1)*(n1-1)+var(d2)*(n2-1))/(n1+n2-2)
print(s2)
```

```
## [1] 40.50362
```

```r
#calculate s
s = sqrt(s2)
print(s)
```

```
## [1] 6.364245
```

```r
#testStastic = (mean.d1 - mean.d2) / (s * sqrt((1/n1)+1/n2)
testStastic = (mean_d1-mean_d2)/(s*sqrt((1/n1)+(1/n2)))
print(testStastic)
```

```
## [1] -4.659428
```

```r
t = qt(1-0.01, lm_employee$df.residual)
print(t)
```

```
## [1] 2.390119
```

Decision Rule:

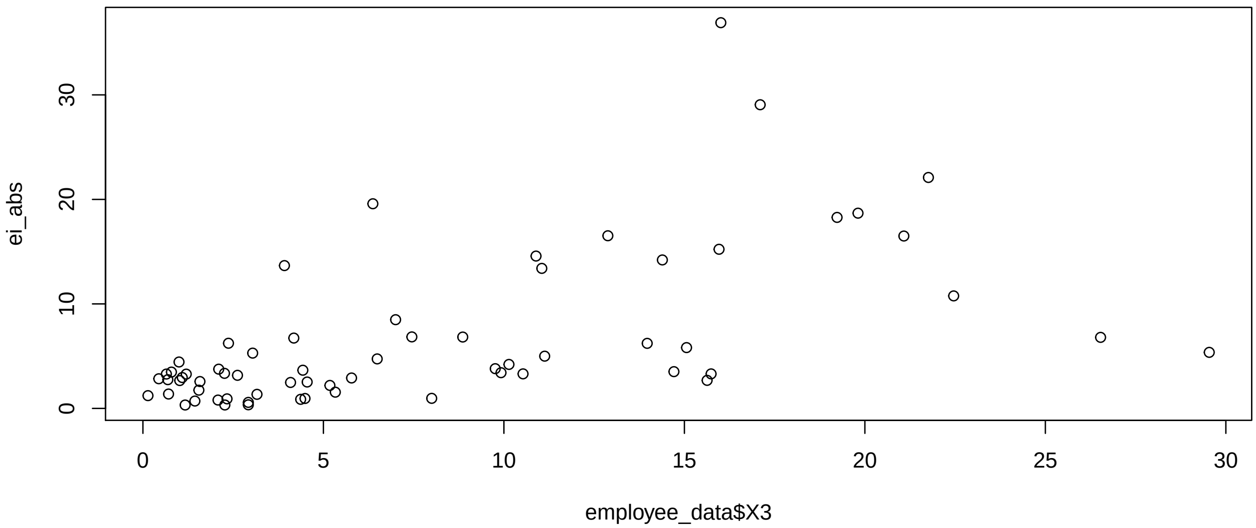
- If $|testStatistic| \leq t(1-\alpha/2, n-2)$, conclude H_0: constant error variance
- If $|testStatistic| > t(1-\alpha/2, n-2)$, conclude H_1: non-constant error variance

Result:

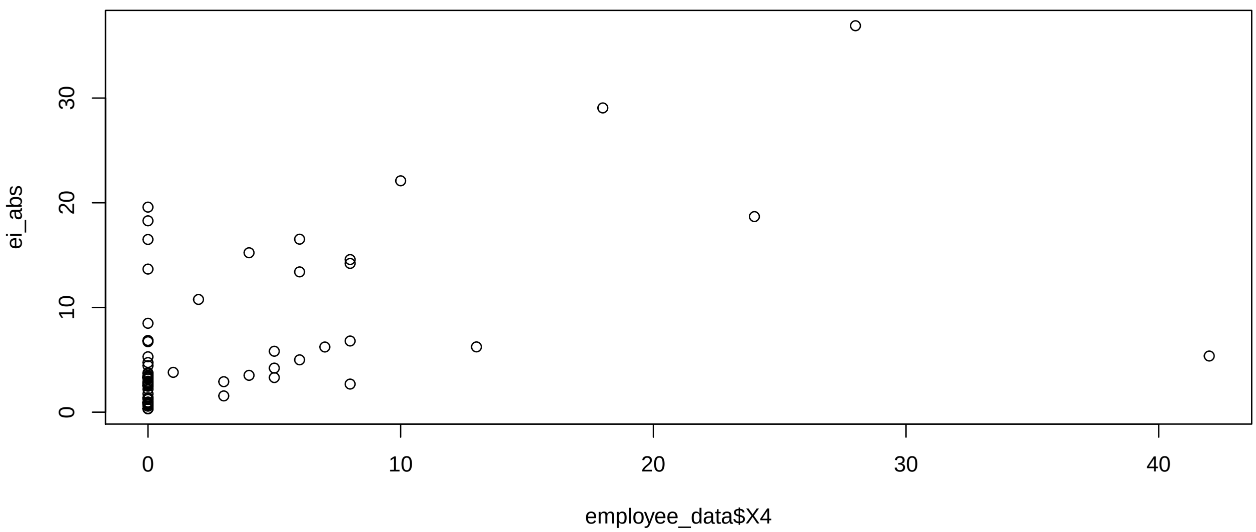
Since $|-4.659428| > 2.390119$ i.e. $|testStatistic| > t(1-\alpha/2, n-2)$, we conclude H_1. The error variance is not constant and thus varies with X.

(d)

```r
ei_abs = abs(ei)
plot(employee_data$X3, ei_abs)
```




```r
plot(employee_data$X4, ei_abs)
```



Interpretation:

We can see a megaphone shape indicating non-constant variance in the error term.

(e) Estimate the. standard deviation function by regressing the absolute residuals against X3 and X4 in first-order form, and then calculate the estimated weight for each case using equation 11.16a on the book. (5pts)

```r
lm_ei_1e = lm(ei_abs~employee_data$X3+employee_data$X4)
summary(lm_ei_1e)
```

```
##
## Call:
## lm(formula = ei_abs ~ employee_data$X3 + employee_data$X4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.180  -3.000  -0.635   1.621  20.545
##
## Coefficients:
```

```
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2.4204     1.1106   2.179  0.03312 *
## employee_data$X3    0.3996     0.1315   3.040  0.00346 **
## employee_data$X4    0.2695     0.1290   2.089  0.04080 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.883 on 62 degrees of freedom
## Multiple R-squared:  0.3697, Adjusted R-squared:  0.3493
## F-statistic: 18.18 on 2 and 62 DF,  p-value: 6.123e-07
```

```r
si = lm_ei_1e$fitted.values
wi = 1/(si^2)
```

(f) Using the estimated weights, obtain the weighted least squares fit of the regression model. Are the weighted least squares estimates of the regression coefficients similar to the ones obtained with ordinary least squares in part (b)? (5 pts)

```r
lm_1f = lm(Y~X3+X4, weights=wi, data=employee_data)
summary(lm_1f)
```

```
##
## Call:
## lm(formula = Y ~ X3 + X4, data = employee_data, weights = wi)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1560 -1.6304 -0.6045  1.3706  6.4719
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  42.0084     1.8415  22.812  < 2e-16 ***
## X3            1.2470     0.4503   2.769  0.00740 **
## X4            2.3880     0.7221   3.307  0.00157 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.619 on 62 degrees of freedom
## Multiple R-squared:  0.3967, Adjusted R-squared:  0.3772
## F-statistic: 20.38 on 2 and 62 DF,  p-value: 1.575e-07
```

Interpretation:

The weighted least squares estimates for the coefficients β_3 and β_4 appear very close.

(g) Compare the estimated standard deviations of the weighted least squares coefficient estimates in part (f) with those for the ordinary least squares estimates in pan (b). What do you find? (5 pts)

```r
confint(lm_employee)
```

```
##                  2.5 %    97.5 %
## (Intercept) 25.7324201 37.210441
## X11          4.3743789 17.249525
## X21         15.6604647 29.600990
## X3           0.8034643  1.712784
## X4           1.3969615  2.307664
```

```r
confint(lm_1f)
```

```
##                   2.5 %    97.5 %
## (Intercept) 38.3273545 45.689444
## X3           0.3468172  2.147252
## X4           0.9445270  3.831472
```

Interpretation:

β_3 and β_4 seem to have overlapping intervals

(h) Iterate the steps in parts (e) and (f) one more time. Is there a substantial change in the estimated regression coefficients? If so, what should you do? (10 pts)

```r
ei_abs2 = abs(lm_1f$residuals)
lm_ei_1h = lm(ei_abs2~employee_data$X3+employee_data$X4)
summary(lm_ei_1h)
```

```
##
## Call:
## lm(formula = ei_abs2 ~ employee_data$X3 + employee_data$X4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.4704  -6.1480  -0.7165   4.9760  22.3043
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         8.6861     1.3543   6.414  2.2e-08 ***
## employee_data$X3    0.1189     0.1603   0.742    0.461
## employee_data$X4    0.2416     0.1573   1.536    0.130
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.173 on 62 degrees of freedom
## Multiple R-squared:  0.1055, Adjusted R-squared:  0.0766
## F-statistic: 3.655 on 2 and 62 DF,  p-value: 0.0316
```

```r
si = lm_ei_1h$fitted.values
wi = 1/(si^2)
lm_1h = lm(Y~X3+X4, weights=wi, data=employee_data)
summary(lm_1h)
```

```
##
## Call:
## lm(formula = Y ~ X3 + X4, data = employee_data, weights = wi)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8939 -0.8966 -0.2527  1.0205  3.3252
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.8457     2.2032  18.993  < 2e-16 ***
## X3            1.3269     0.3089   4.296 6.24e-05 ***
## X4            2.2168     0.4157   5.333 1.44e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.238 on 62 degrees of freedom
## Multiple R-squared:  0.636,  Adjusted R-squared:  0.6242
## F-statistic: 54.16 on 2 and 62 DF,  p-value: 2.481e-14
```

Interpretation:

There doesn't seem to be much difference in the coefficients in the models `lm_1f` (part f model) and `lm_1h` (part h model). However, there seems to be a significant improvement in the R^2.

So we should probably do one more iteration and see if it gives us an improvement, or else stop and use the weights identified part (h).

Question 2 Refer to the Weight and height. The weights and heights of twenty male 'Students in a freshman class are recorded in order to see how well weight (Y, in pounds) can be predicted from height (X, in inches). Assume that first-order regression is appropriate. (30 pts)

(a) Fit a simple linear regression model using ordinary least squares, and plot the data together with the fitted regression function. Also, obtain an Index plot of Cook s distance. What do these plots suggests? (5pts)

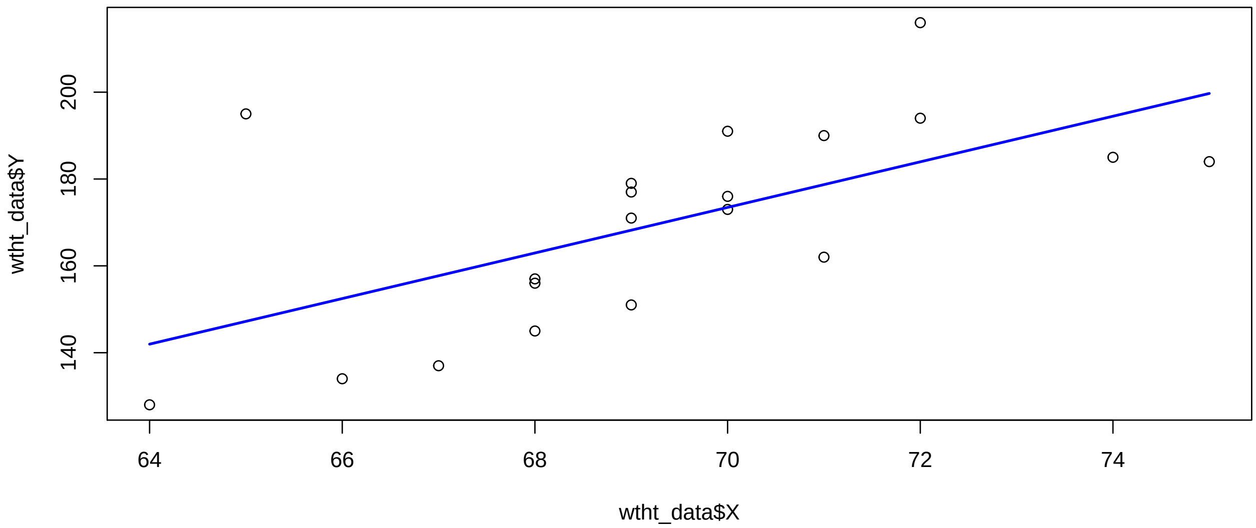
```r
wtht_data = read.csv("Weight and Height.csv")
summary(wtht_data)
```

```
##        Y               X
##  Min.   :128.0   Min.   :64.00
##  1st Qu.:154.8   1st Qu.:68.00
##  Median :174.5   Median :69.00
##  Mean   :170.1   Mean   :69.35
##  3rd Qu.:186.2   3rd Qu.:71.00
##  Max.   :216.0   Max.   :75.00
```

```r
lm_wtht = lm(Y~X, data=wtht_data)
summary(lm_wtht)
```

```
##
## Call:
## lm(formula = Y ~ X, data = wtht_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.716 -15.955  -3.213  10.228  47.780
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -193.924    108.313  -1.790  0.09022 .
## X              5.248      1.561   3.363  0.00346 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.76 on 18 degrees of freedom
## Multiple R-squared:  0.3859, Adjusted R-squared:  0.3517
## F-statistic: 11.31 on 1 and 18 DF,  p-value: 0.003464
```

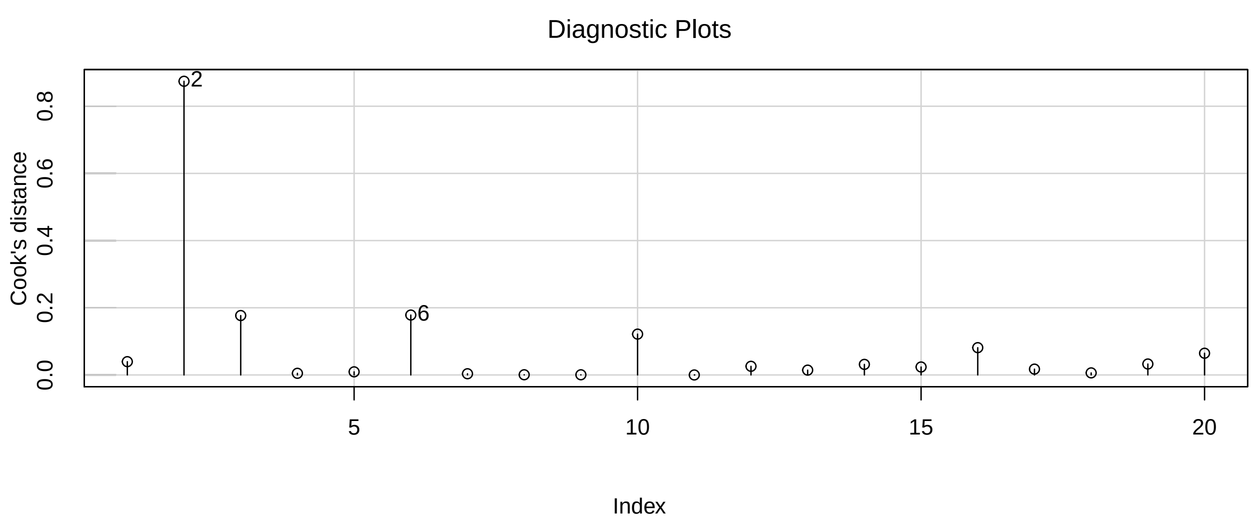
```r
par(mfrow=c(1,1))
plot(wtht_data$X, wtht_data$Y)
regLine(lm_wtht)
```




```r
cooks.distance(lm_wtht)
```

```
##            1            2            3            4            5
## 3.955002e-02 8.742274e-01 1.768963e-01 4.908339e-03 9.326783e-03
##            6            7            8            9           10
## 1.783961e-01 3.600040e-03 6.225684e-04 5.399948e-04 1.215796e-01
##           11           12           13           14           15
## 1.784134e-05 2.577509e-02 1.437125e-02 3.147983e-02 2.374945e-02
##           16           17           18           19           20
## 8.123881e-02 1.737446e-02 6.188856e-03 3.265639e-02 6.466418e-02
```

```r
influenceIndexPlot(lm_wtht, vars=c("Cook"))
```



Interpretation:

Fitted Regression Function: The plot suggests some non-linearity along with a few outliers as seen in the plot.

Cook's Distance: The plot suggests observation #2 is a clear outlier and #3, #6 need further investigation.

(b) Obtain the scaled residuals in equation 11.47 and use the Huber weight function (equation 11.44) to obtain case weights for a first iteration of IRLS robust regression. Which cases receive the smallest Huber weights? Why? (10 pts)

```r
ei = lm_wtht$residuals
MAD = (1/0.6745)*(median(abs(ei-median(ei))))
ui = ei/MAD
wi = ifelse(abs(ui)>1.345, (1.345/abs(ui)), 1)
wi
```

```
##         1         2         3         4         5         6         7 
## 1.0000000 0.5582278 0.8324212 1.0000000 1.0000000 1.0000000 1.0000000 
##         8         9        10        11        12        13        14 
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 
##        15        16        17        18        19        20 
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

Interpretation:

Observations 2 and 3 receive the smallest huber weights, because they are farthest away from the median residual value, suggesting an outlier.

(c) Using the weights calculated in part (b), obtain the weighed least squares estimates of the regression coefficients. How do these estimates compare to those found in part (a) using ordinary least squares? (5pts)

```r
lm_2c = lm(Y~X, weights=wi, data=wtht_data)
summary(lm_2c)
```

```
## 
## Call:
## lm(formula = Y ~ X, data = wtht_data, weights = wi)
## 
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max 
## -17.919 -15.210  -1.596  10.735  38.671 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -236.259    100.427  -2.353  0.03022 *  
## X              5.838      1.445   4.039  0.00077 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 16.79 on 18 degrees of freedom
## Multiple R-squared:  0.4754, Adjusted R-squared:  0.4463 
## F-statistic: 16.31 on 1 and 18 DF,  p-value: 0.0007697
```

Interpretation:

β_1 from both the models seem to be close to each other. But there seems to be a good difference in the intercept.

(d) Continue the IRLS procedure for two more iterations. Which cases receive the smallest weights in the final iteration? How do the final IRLS robust regression estimates compare to the ordinary least squares estimates obtained in part (a)? (10 pts)

Iteration #2

```r
ei = lm_2c$residuals
MAD = (1/0.6745)*(median(abs(ei-median(ei))))
ui = ei/MAD
wi = ifelse(abs(ui)>1.345, (1.345/abs(ui)), 1)
```

```
wi
```

```
##         1         2         3         4         5         6         7
## 1.0000000 0.5164069 0.8381745 1.0000000 1.0000000 1.0000000 1.0000000
##         8         9        10        11        12        13        14
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##        15        16        17        18        19        20
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

```
lm_2d1 = lm(Y~X, weights=wi, data=wtht_data)
summary(lm_2d1)
```

```
##
## Call:
## lm(formula = Y ~ X, data = wtht_data, weights = wi)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -17.942 -14.905  -1.461  10.832  37.507
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -241.577     99.665  -2.424 0.026112 *
## X              5.914      1.434   4.123 0.000639 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.61 on 18 degrees of freedom
## Multiple R-squared:  0.4857, Adjusted R-squared:  0.4571
## F-statistic:    17 on 1 and 18 DF,  p-value: 0.0006387
```

Iteration #3

```
ei = lm_2d1$residuals
MAD = (1/0.6745)*(median(abs(ei-median(ei))))
ui = ei/MAD
wi = ifelse(abs(ui)>1.345, (1.345/abs(ui)), 1)
wi
```

```
##         1         2         3         4         5         6         7
## 1.0000000 0.5049234 0.8287764 1.0000000 1.0000000 1.0000000 1.0000000
##         8         9        10        11        12        13        14
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##        15        16        17        18        19        20
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

```
lm_2d2 = lm(Y~X, weights=wi, data=wtht_data)
summary(lm_2d2)
```

```
##
## Call:
## lm(formula = Y ~ X, data = wtht_data, weights = wi)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -17.975 -14.821  -1.408  10.878  37.165
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -242.606     99.363  -2.442 0.025175 *
## X              5.928      1.430   4.145 0.000608 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.54 on 18 degrees of freedom
## Multiple R-squared:  0.4884, Adjusted R-squared:   0.46
## F-statistic: 17.19 on 1 and 18 DF,  p-value: 0.0006075
```

Interpretation:

The same observations (#2 and #3) recieve the smallest weights as is the previous two iterations.

The β_1 again seem to be close to each other. But the intercept seems to be far apart.

Question 3 Refer to the Prostate Cancer data set in Appendix C.5 and Homework 7&8. Select a random sample of 65 observations to use as the model-building data set (use set.seed(1023)). Use the remaining observations for the test data. (10 pts)

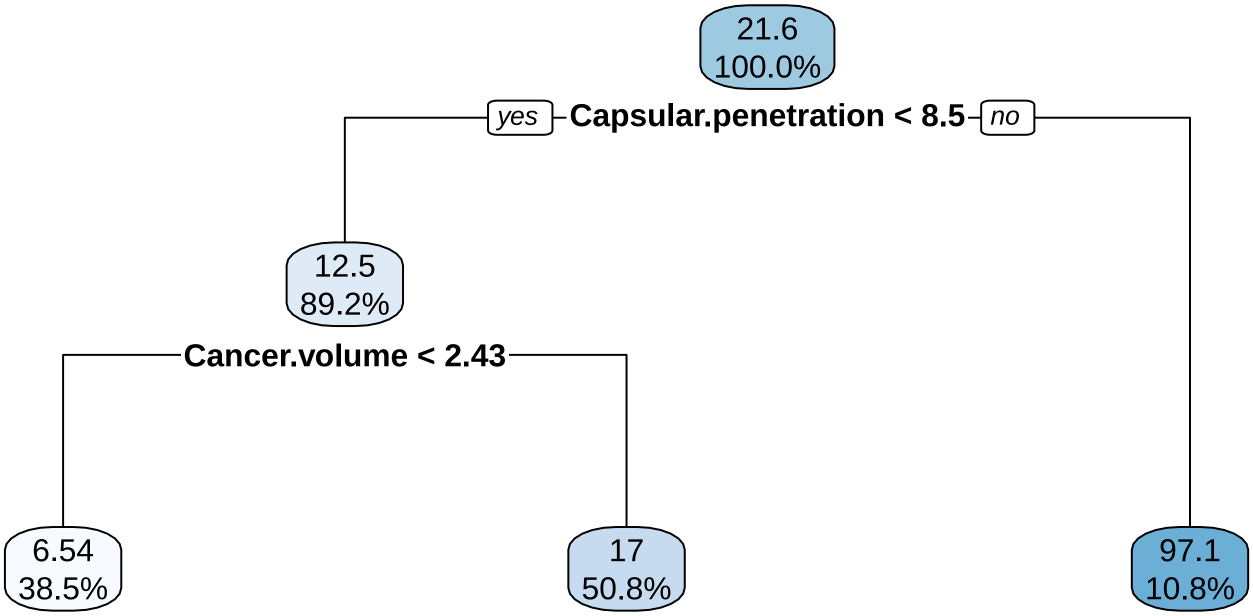
(a) Develop a regression tree for predicting PSA. Justify your choice of number of regions (tree size), and interpret your regression tree. Test the performance of the model on the test data. (5 pts)

```r
prostate_data = read.csv("Prostate Cancer.csv")
prostate_data$Seminal.vesicle.invasion = as.factor(prostate_data$Seminal.vesicle.invasion)
prostate_data$Gleason.score = as.factor(prostate_data$Gleason.score)
summary(prostate_data)
```

```
##    PSA.level       Cancer.volume        Weight            Age
##  Min.   :  0.651   Min.   : 0.2592   Min.   : 10.70   Min.   :41.00
##  1st Qu.:  5.641   1st Qu.: 1.6653   1st Qu.: 29.37   1st Qu.:60.00
##  Median : 13.330   Median : 4.2631   Median : 37.34   Median :65.00
##  Mean   : 23.730   Mean   : 6.9987   Mean   : 45.49   Mean   :63.87
##  3rd Qu.: 21.328   3rd Qu.: 8.4149   3rd Qu.: 48.42   3rd Qu.:68.00
##  Max.   :265.072   Max.   :45.6042   Max.   :450.34   Max.   :79.00
##  Benign.prostatic.hyperplasia Seminal.vesicle.invasion
##  Min.   : 0.000               0:76
##  1st Qu.: 0.000               1:21
##  Median : 1.350
##  Mean   : 2.535
##  3rd Qu.: 4.759
##  Max.   :10.278
##  Capsular.penetration Gleason.score
##  Min.   : 0.0000      6:33
##  1st Qu.: 0.0000      7:43
##  Median : 0.4493      8:21
##  Mean   : 2.2454
##  3rd Qu.: 3.2544
##  Max.   :18.1741
```

```r
set.seed(1023)
train_ind = sample(1:nrow(prostate_data), 65)
test_ind = setdiff(1:nrow(prostate_data), train_ind)
train_df = prostate_data[train_ind,]
test_df = prostate_data[test_ind,]
```

```r
library(rpart.plot)
tree_prostate = rpart(PSA.level~., data=train_df)
rpart.plot(tree_prostate, digits = 3)
```



Interpretation:

- We can see that the tree model identified three significant regions based on two most significant predictors variables - `Cancer.volume` and `Capsular.penetration`.
- The numbers on each of the leaf nodes idicate the mean of the response variable in the region and the % value indicates the amount of variation explained.

```r
yHat_tree_te = predict(tree_prostate, test_df)
SSE_tree = sum((yHat_tree_te-test_df$PSA.level)^2)
SSE_tree
```

```
## [1] 69339.32
```

(b) Compare the performance of your regression tree model with that of the best regression model obtained in HW7. Which model is more easily interpreted and why? (5pts)

```r
hw7_best = lm(PSA.level~Cancer.volume+Capsular.penetration, data=train_df)
summary(hw7_best)
```

```
##
## Call:
## lm(formula = PSA.level ~ Cancer.volume + Capsular.penetration,
##     data = train_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -69.286  -7.829   1.323   5.499 137.416
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.2003     4.4328   0.045  0.96411
## Cancer.volume          1.8205     0.6013   3.028  0.00359 **
```

```
## Capsular.penetration   3.7938     1.2180   3.115  0.00279 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.77 on 62 degrees of freedom
## Multiple R-squared:  0.5266, Adjusted R-squared:  0.5113
## F-statistic: 34.48 on 2 and 62 DF,  p-value: 8.572e-11
```

```
yHat_lm_test = predict(hw7_best, test_df)
SSE_hw7 = sum((yHat_lm_test-test_df$PSA.level)^2)
SSE_hw7
```

```
## [1] 47256.33
```

Interpretation:

Thus, the best linear model identified in HW 7 has lower SSE and preforms better on the test data than the tree model.

Question 4 Refer to Cement composition. The variables collected were the amount of tricalcium aluminate (X1), the amount of tricalcium silicate (X2), the amount of tetracalcium alumino ferrite (X3), the amount of dicalcium silicate (X4), and the heat evolved in calories per gram of cement (Y). (20pts)

(a) Fit regression model for four predictor variables to the data. State the estimated regression function. (5pts)

```
cement_data = read.csv("Cement Composition.csv")
summary(cement_data)
```

```
##        Y                X1               X2              X3
##  Min.   : 72.50   Min.   : 1.000   Min.   :26.00   Min.   : 4.00
##  1st Qu.: 83.80   1st Qu.: 2.000   1st Qu.:31.00   1st Qu.: 8.00
##  Median : 95.90   Median : 7.000   Median :52.00   Median : 9.00
##  Mean   : 95.42   Mean   : 7.462   Mean   :48.15   Mean   :11.77
##  3rd Qu.:109.20   3rd Qu.:11.000   3rd Qu.:56.00   3rd Qu.:17.00
##  Max.   :115.90   Max.   :21.000   Max.   :71.00   Max.   :23.00
##        X4
##  Min.   : 6
##  1st Qu.:20
##  Median :26
##  Mean   :30
##  3rd Qu.:44
##  Max.   :60
```

```
lm_cement = lm(Y~., data=cement_data)
summary(lm_cement)
```

```
##
## Call:
## lm(formula = Y ~ ., data = cement_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1750 -1.6709  0.2508  1.3783  3.9254
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  62.4054    70.0710   0.891   0.3991
```

```
## X1            1.5511     0.7448   2.083   0.0708 .
## X2            0.5102     0.7238   0.705   0.5009
## X3            0.1019     0.7547   0.135   0.8959
## X4           -0.1441     0.7091  -0.203   0.8441
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.446 on 8 degrees of freedom
## Multiple R-squared:  0.9824, Adjusted R-squared:  0.9736
## F-statistic: 111.5 on 4 and 8 DF,  p-value: 4.756e-07
```

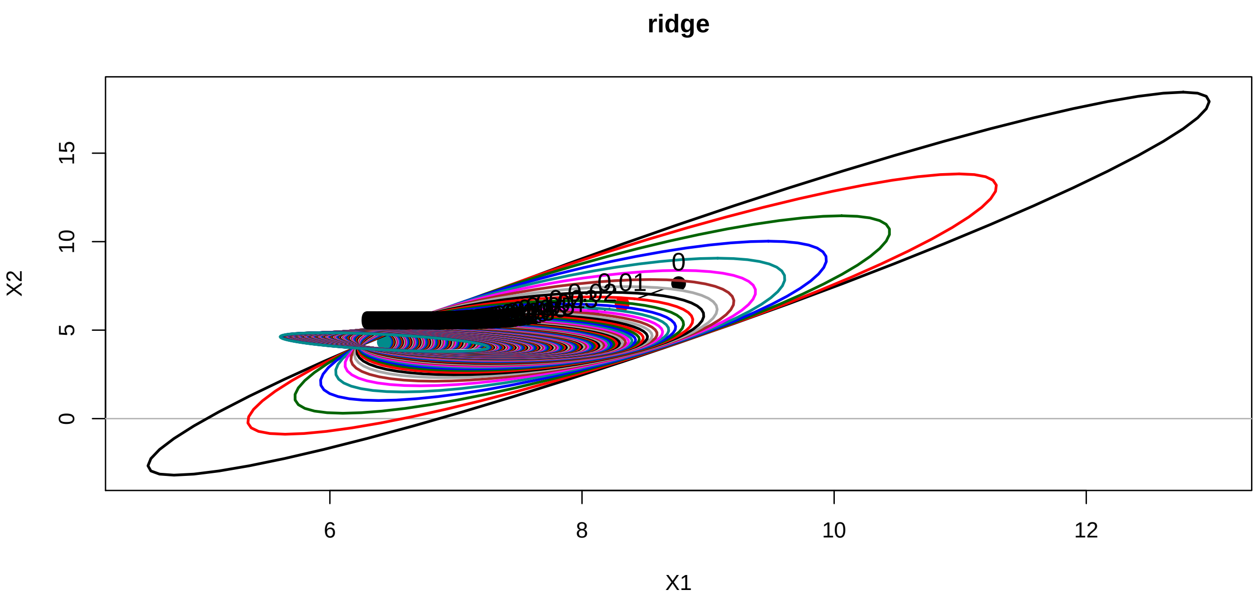
Estimated Regression Function: $Y = 62.4054 + 1.5511 * X_1 + 0.5102 * X_2 + 0.1019 * X_3 - 0.1441 * X_4$

(b) Obtain the estimated ridge standardized regression coefficients, variance inflation factors, and R2. Suggest a reasonable value for the biasing constant c (use seq(0,1,by=0.01)) based on the ridge trace, VIF values, and R2 values. (5pts) (hint: use vif.ridge function under library(genride), you can also get MSEs under this library)

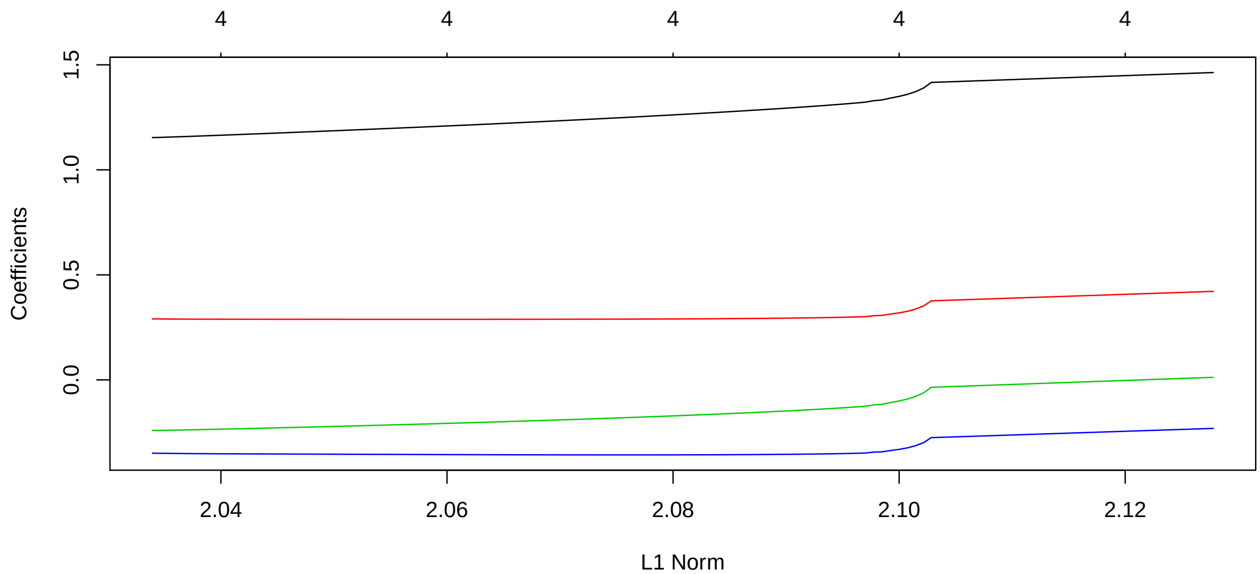
```
Y = as.matrix(cement_data$Y)
X = as.matrix(cement_data[,-which(names(cement_data) %in% c("Y"))])
lm_ridge = ridge(Y~., data=cement_data, lambda = seq(0,1,by=0.01))
```

```
## Warning in model.matrix.default(Terms, m, contrasts): non-list contrasts
## argument ignored
```

```
plot(lm_ridge, main="ridge")
```



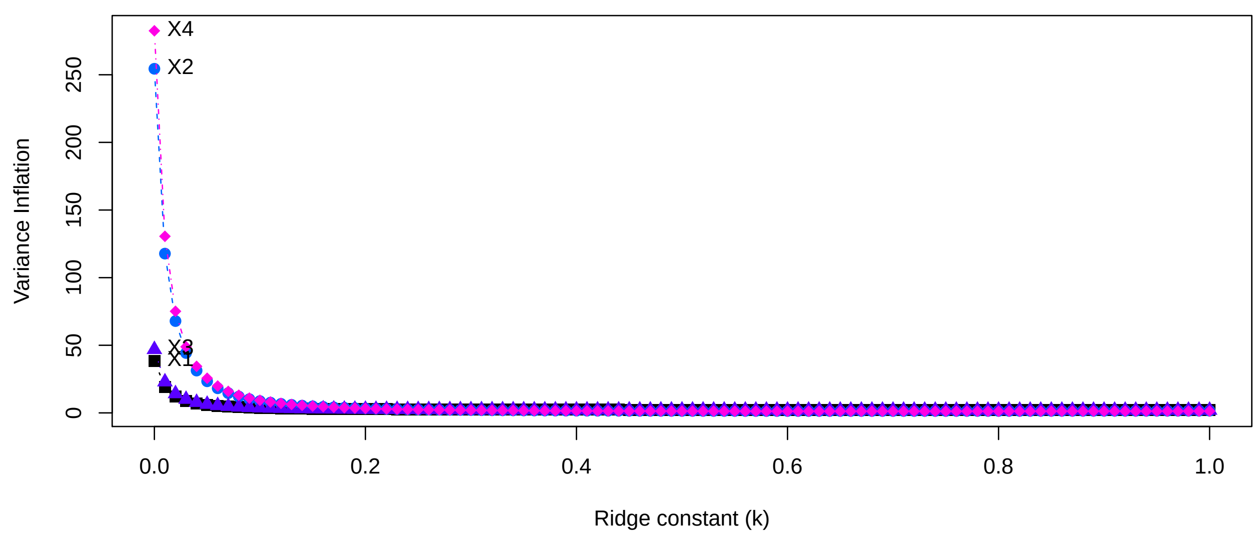

```
glm_ridge = glmnet(X,Y, lambda=seq(0,1,by=0.01), alpha=0)
plot(glm_ridge)
```



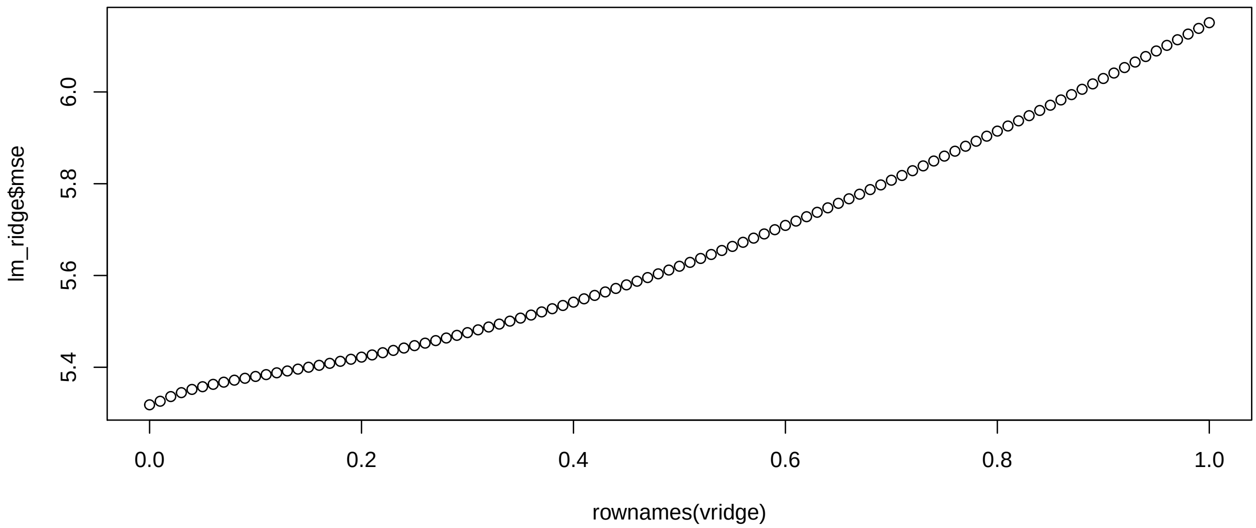

```r
vridge = vif(mod = lm_ridge)

pch <- c(15:18, 7, 9)
clr <- c("black", rainbow(5, start=.6, end=.1))

matplot(rownames(vridge), vridge, type='b',
    xlab='Ridge constant (k)', ylab="Variance Inflation",
    xlim=c(0, 1),
    col=clr, pch=pch, cex=1.2)
text(0.0, vridge[1,], colnames(vridge), pos=4)
```




```r
par(mfrow=c(1,1))
plot(rownames(vridge), lm_ridge$mse)
```



Interpretation:

We select $\lambda = 0.1$ since all the measures seem to be stablilizing at that point (elbow effect).

```
lm_ridge_best = lm.ridge(Y~., data=cement_data, lambda=0.1)
lm_ridge_best$coef
```

```
##        X1        X2        X3        X4
##  7.517712  4.691142 -0.714124 -5.398689
```

```
lm_ridge_best1 = glmnet(X,Y, alpha = 0, lambda = 0.1)
lm_ridge_best1$beta
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##             s0
## X1  1.35224906
## X2  0.33381239
## X3 -0.09432759
## X4 -0.31632419
```

(c) Transform the estimated standardized ridge regression coefficients selected in part (b) to the original variables and obtain the fitted values for the 13 cases. How similar are these fitted values to those obtained with the ordinary least squares fit ill part (a)? (5pts)

```
coef(lm_ridge_best)
```

```
##                     X1         X2         X3         X4
## 81.8251068  1.3301867  0.3137801 -0.1160452 -0.3357076
```

```
coef(lm_ridge_best1)
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 79.85875697
## X1           1.35224906
## X2           0.33381239
## X3          -0.09432759
## X4          -0.31632419
```

```
pred_ridge_best = predict(lm_ridge_best1, s=0.1, newx=X)
pred_ridge_best
```

```
##               1
##  [1,]  78.45821
##  [2,]  73.02779
##  [3,] 106.34589
##  [4,]  89.45982
##  [5,]  95.67808
##  [6,] 105.28510
##  [7,] 104.11467
##  [8,]  75.56572
##  [9,]  91.93210
## [10,] 115.34343
## [11,]  81.63894
## [12,] 112.12028
## [13,] 111.52998
```

```
SSE_ridge = sum((pred_ridge_best-cement_data$Y)^2)
SSE_ridge
```

```
## [1] 48.31297
```

```
lm_a_c = lm(pred_ridge_best~lm_cement$fitted.values)
summary(lm_a_c)
```

```
##
## Call:
## lm(formula = pred_ridge_best ~ lm_cement$fitted.values)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22034 -0.14431 -0.00178  0.11021  0.41390
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.352371   0.361960   0.974    0.351
## lm_cement$fitted.values 0.996307   0.003751 265.598   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1938 on 11 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9998
## F-statistic: 7.054e+04 on 1 and 11 DF,  p-value: < 2.2e-16
```

Interpretation

We can see that the R^2 between the fitted values in part (a) and part (c) is very high. So the fitted values in part (a) and part (c) are very similar.

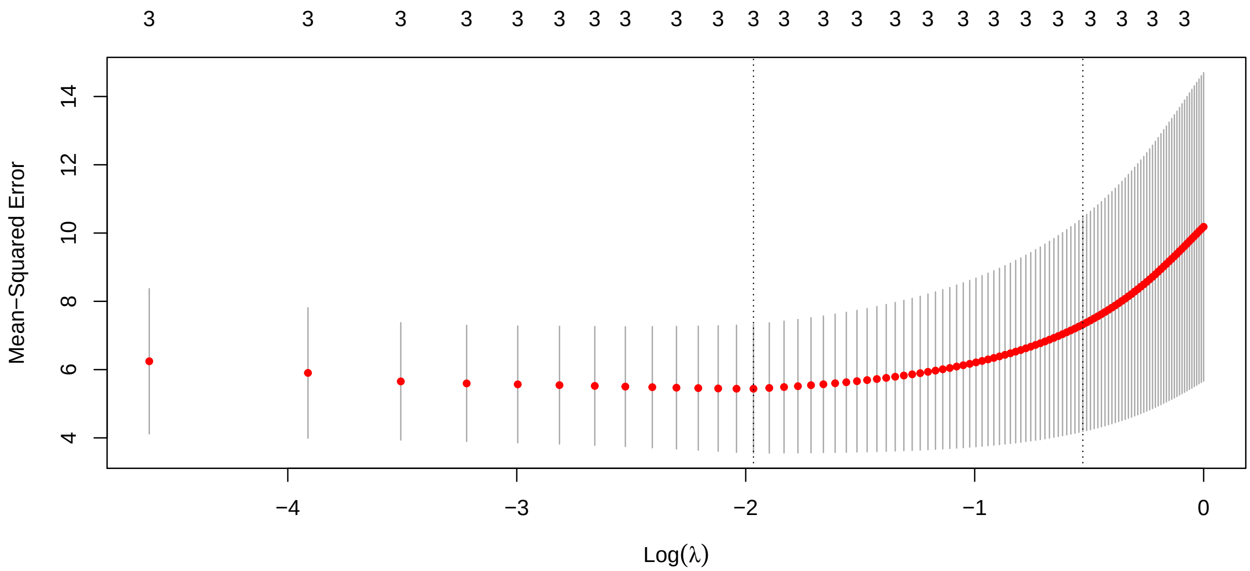
(d) Fit Lasso and Elastic net models and compare it against the Ridge regression model results. (5pts) (Hint: Calculate SSEs for each model)

LASSO

```
cv_lasso = cv.glmnet(X,Y, alpha = 1, lambda = seq(0,1,by=0.01))
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations
## per fold
```

```r
plot(cv_lasso)
```




```r
cv_lasso$lambda.min
```

```
## [1] 0.14
```

```r
lasso_best = glmnet(X,Y, alpha = 1, lambda=cv_lasso$lambda.min)
coef(lasso_best)
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept) 71.6710973
## X1           1.4320249
## X2           0.4149961
## X3           .
## X4          -0.2305596
```

```r
pred_lasso_best = predict(lasso_best, s=cv_lasso$lambda.min, newx=X)
SSE_lasso = sum((pred_lasso_best-cement_data$Y)^2)
SSE_lasso
```

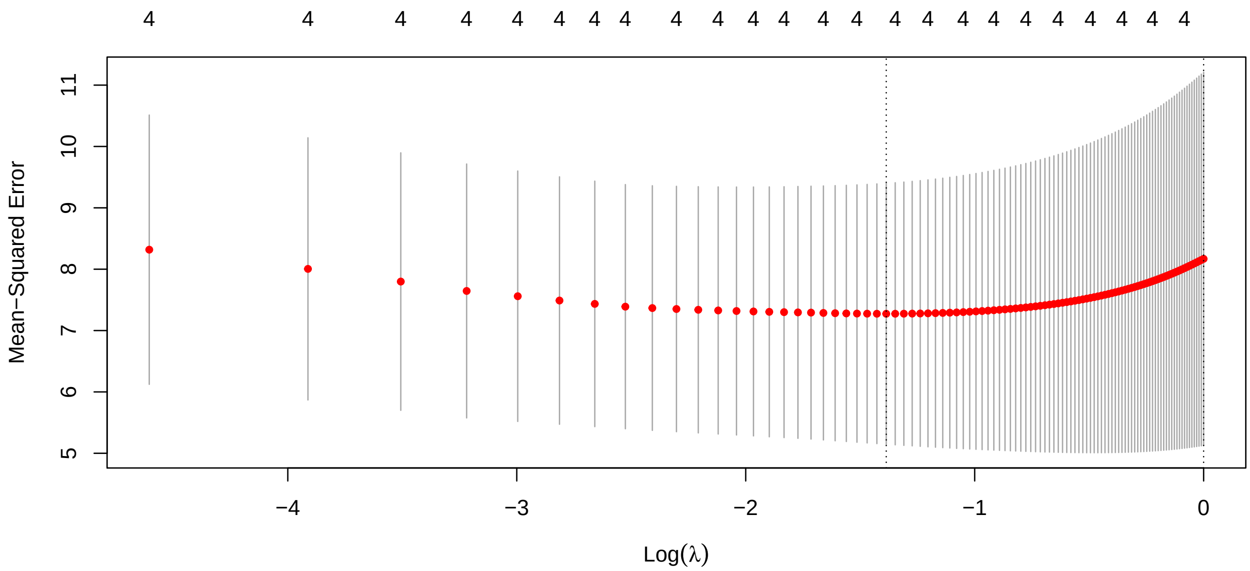
```
## [1] 48.38196
```

Elastic Net

```r
cv_elnet = cv.glmnet(X,Y, alpha = 0.5, lambda = seq(0,1,by=0.01))
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations
## per fold
```

```r
plot(cv_elnet)
```




```r
cv_elnet$lambda.min
```

```
## [1] 0.25
```

```r
elnet_best = glmnet(X,Y, alpha = 0.5, lambda=cv_elnet$lambda.min)
coef(elnet_best)
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 77.75584694
## X1           1.36438331
## X2           0.35180550
## X3          -0.06080439
## X4          -0.29127785
```

```r
pred_elnet_best = predict(elnet_best, s=cv_elnet$lambda.min, newx=X)
SSE_elnet = sum((pred_elnet_best-cement_data$Y)^2)
SSE_elnet
```

```
## [1] 48.81749
```

Interpretation:

Here, we find the best lambda for LASSO and Elastic Net individually. The SSEs for the best Ridge, LASSO and Elastic Net are very similar to each other.