HW9-Solutions

Problem 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)

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)

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

d)Plot the absolute residuals against X3, and against X4. What do these plots suggest about the relation between the standard deviation of the error term and X3, and X4? (5pts)

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)

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)

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)

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)

a)

_Solution: see below.

```
library(knitr)
Employee..Salaries <- read.csv("/cloud/project/Employee  Salaries.csv")
Y<-Employee..Salaries$Y
X3<-Employee..Salaries$X3
X4<-Employee..Salaries$X4
X1<-1*(Employee..Salaries$Degree==2)
X2<-1*(Employee..Salaries$Degree==3)
```

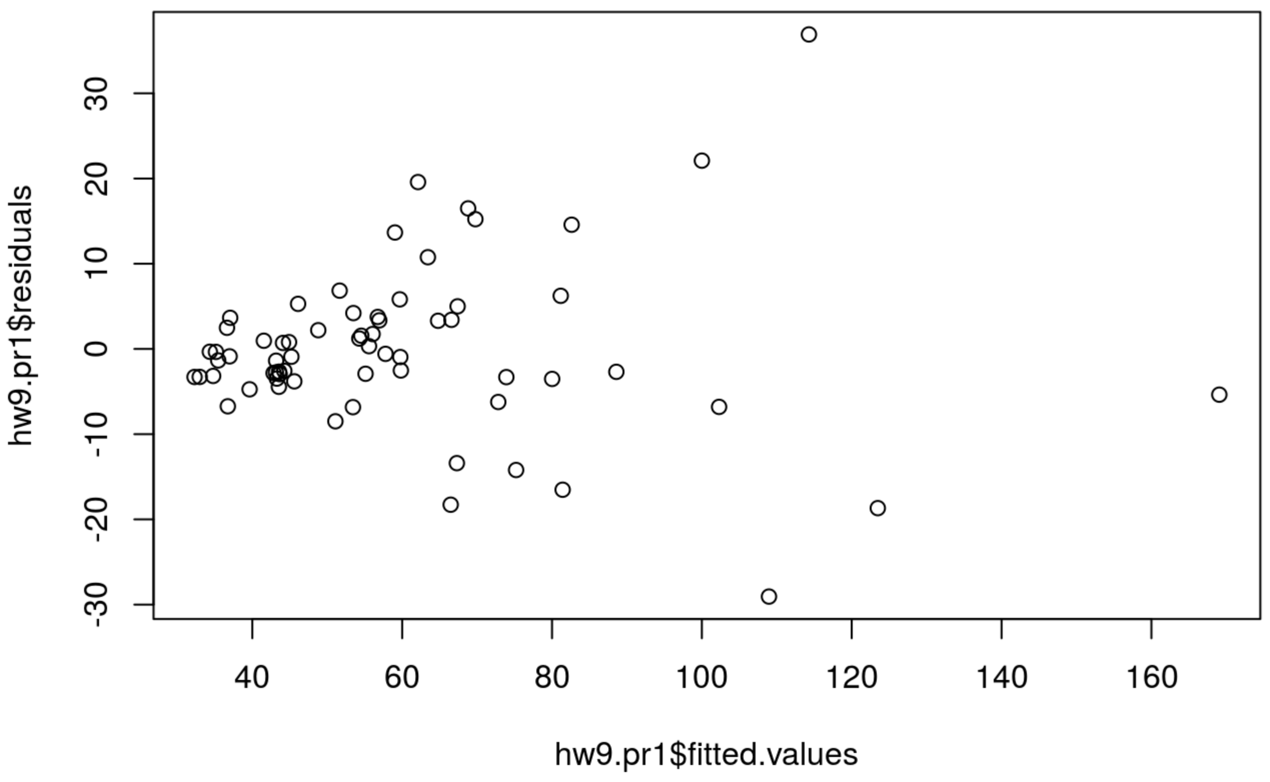
b)

_Solution: Unequal variances, Rsquare is 86%.

```
hw9.pr1<-lm(Y~X1+X2+X3+X4)
summary(hw9.pr1)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4)
##
## 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 ***
## X1           10.8120     3.2183   3.360  0.00136 **
## X2           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
```

```
plot(hw9.pr1$fitted.values,hw9.pr1$residuals)
```



c)

_Solution: Ho:Error variance is constant ### Ha:Error variance is Not Constant. Pvalue is less than 0.01. Reject Ho. Error variance is not constant. See below.

```
ei<-hw9.pr1$residuals
DM<-data.frame(cbind(hw9.pr1$fitted.values,ei))
DM.S<-DM[order(DM[,1]),]

DM1<-DM.S[1:32,]
DM2<-DM.S[33:65,]
M1<-median(DM1[,2])
M2<-median(DM2[,2])
N1<-length(DM1[,2])
N2<-length(DM2[,2])
d1<-abs(DM1[,2]-M1)
d2<-abs(DM2[,2]-M2)
s2<-sqrt((var(d1)*(N1-1)+var(d2)*(N2-1))/(N1+N2-2))
Den<- s2*sqrt(1/N1+1/N2)
Num<- mean(d1)-mean(d2)
T= Num/Den
T
```

```
## [1] -4.442422
```

```
2*pt(-4.442422,N1+N2-2)
```

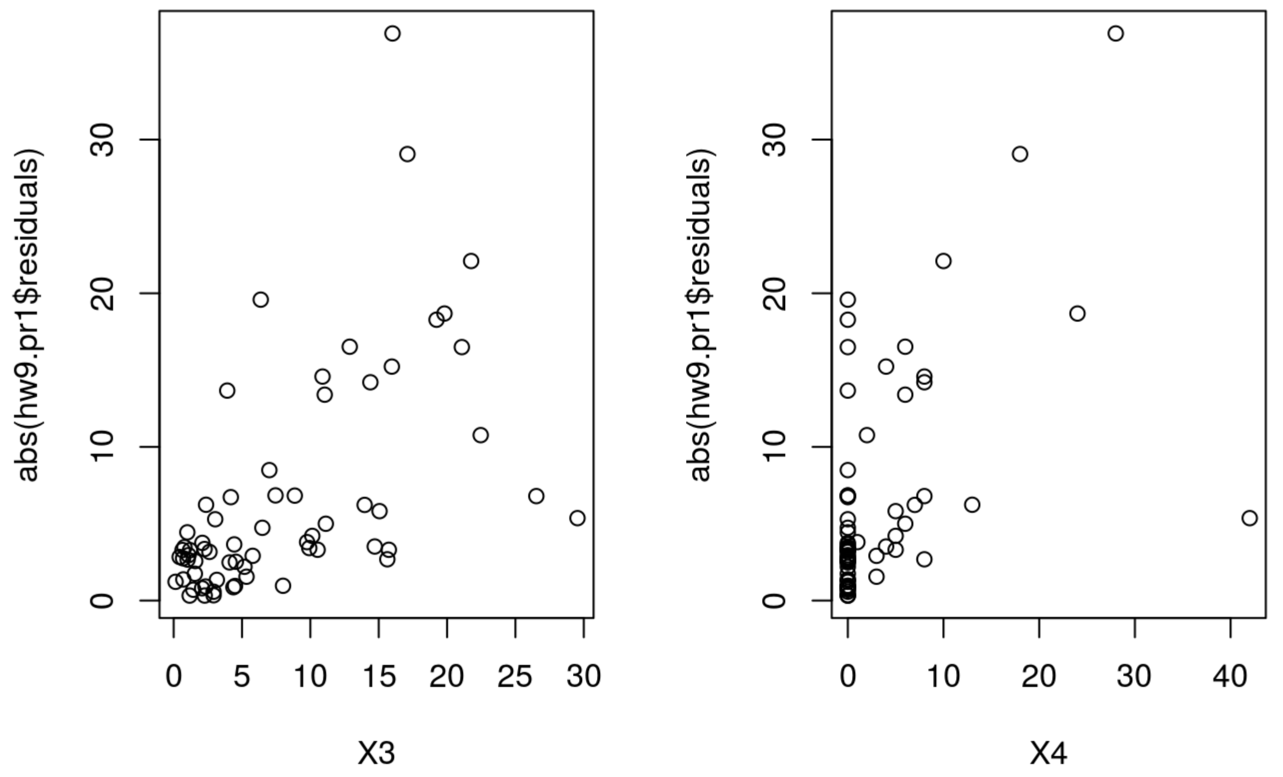
```
## [1] 3.6688e-05
```

d)

_Solution: Both graphs show positive relationship.

```
par(mfrow=c(1,2))
plot(X3,abs(hw9.pr1$residuals))
plot(X4,abs(hw9.pr1$residuals))
```

e)



_Solution: See below.

```
abs.ei<-abs(hw9.pr1$residuals)
hw9.pr1e<-lm(abs.ei~X3+X4)
si<-hw9.pr1e$fitted.values
wi<-1/(si^2)
```

f)

_Solution: Coefficents are similar.See below.

```
hw9.pr1f<-lm(Y~X1+X2+X3+X4,weights=wi)
summary(hw9.pr1f)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4, weights = wi)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2414 -0.7531 -0.2709  0.6915  3.3246
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  29.4255     1.3617  21.610  < 2e-16 ***
## X1           10.8996     1.4918   7.307 7.50e-10 ***
## X2           26.6849     1.6686  15.992  < 2e-16 ***
## X3            1.4253     0.2002   7.118 1.57e-09 ***
## X4            1.7239     0.3206   5.377 1.31e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.15 on 60 degrees of freedom
## Multiple R-squared:  0.8874, Adjusted R-squared:  0.8799
## F-statistic: 118.2 on 4 and 60 DF,  p-value: < 2.2e-16
```

```
rbind(coef(hw9.pr1f),coef(hw9.pr1))
```

```
##      (Intercept)       X1       X2       X3       X4
## [1,]    29.42549 10.89964 26.68488 1.425330 1.723949
## [2,]    31.47143 10.81195 22.63073 1.258124 1.852313
```

g)

_Solution: Standart deviations are smaller in the model built part f than part b.

```
a1<-summary(hw9.pr1)
a2<-summary(hw9.pr1f)
rbind(a1$coefficients[,2],a2$coefficients[,2])
```

```
##      (Intercept)       X1       X2        X3        X4
## [1,]    2.869078 3.218307 3.484612 0.2272961 0.2276418
## [2,]    1.361659 1.491761 1.668615 0.2002390 0.3206416
```

h)

_Solution: See below, the coefficents and standart deviations are changed slighly in the second iteration from the first iteration.

```
abs.ei<-abs(hw9.pr1f$residuals)
fe2<-lm(abs.ei~X3+X4)
si<-fe2$fitted.values
wi<-1/(si^2)
hw9.pr1f1<-lm(Y~X1+X2+X3+X4,weights=wi)

a1<-summary(hw9.pr1)
a2<-summary(hw9.pr1f)
a3<-summary(hw9.pr1f1)
rbind(a1$coefficients[,2],a2$coefficients[,2],a3$coefficients[,2])
```

```
##      (Intercept)       X1       X2        X3        X4
## [1,]    2.869078 3.218307 3.484612 0.2272961 0.2276418
## [2,]    1.361659 1.491761 1.668615 0.2002390 0.3206416
## [3,]    1.206206 1.299832 1.457938 0.2011935 0.3372866
```

```
rbind(coef(hw9.pr1),coef(hw9.pr1f),coef(hw9.pr1f1))
```

```
##      (Intercept)       X1       X2       X3       X4
## [1,]    31.47143 10.81195 22.63073 1.258124 1.852313
## [2,]    29.42549 10.89964 26.68488 1.425330 1.723949
## [3,]    29.08323 11.00753 26.81422 1.490446 1.692183
```

Problem 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)

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)

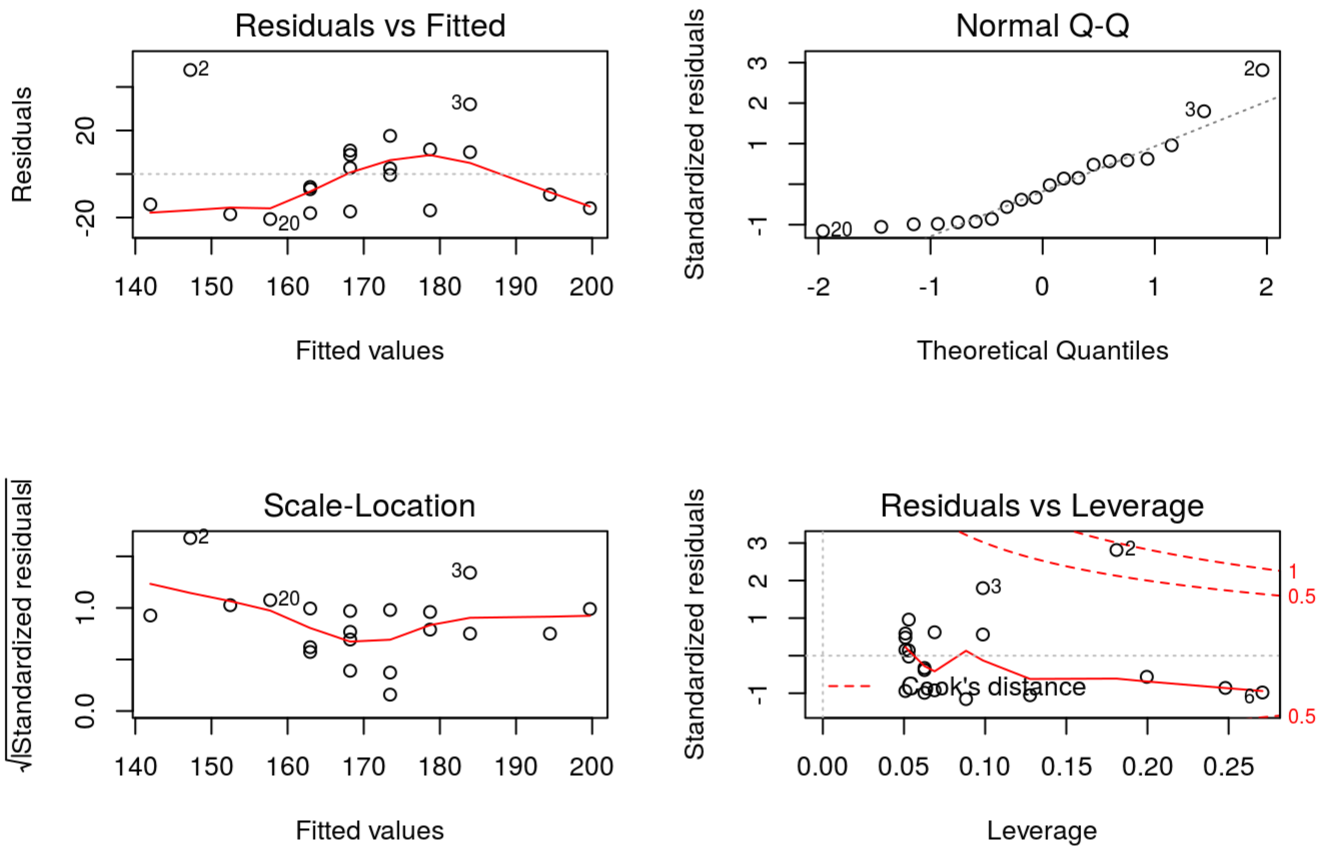
c) Using the weights calculated in part (b), obtain the weighed least squares estimates of the regression coefficients. How do these estimates compare to those found in part (a) using ordinary least squares? (5pts)

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)

a)

_Solution: Y=-193.9+5.2X, Rsquare is 38% and the model is significant. Cooks distance indicates that there is leverage data point, obs=2.Obs=3 is also an outlier from QQ plot

```
Weight.and.Height <- read.csv("/cloud/project/Weight and Height.csv")
hw9.pr1<-lm(Y~X,data=Weight.and.Height)
par(mfrow=c(2,2))
plot(hw9.pr1)
```

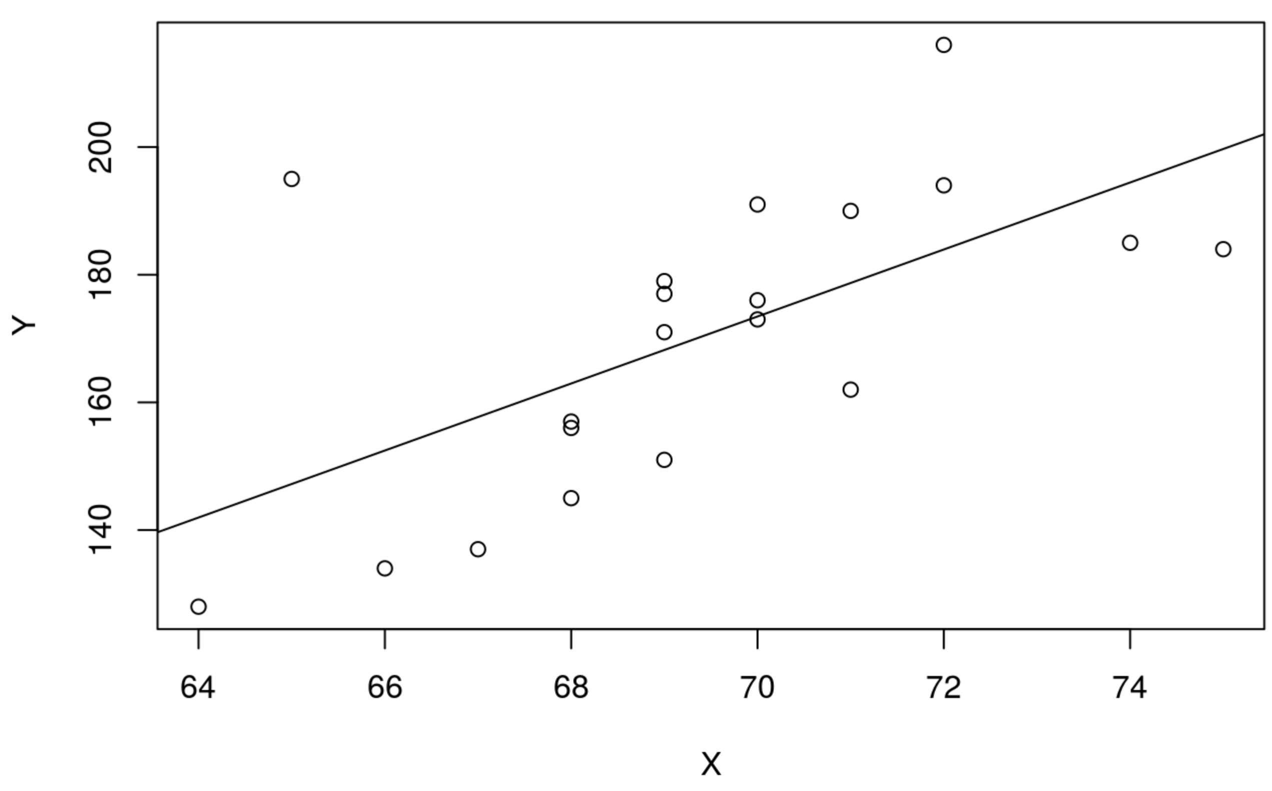


```
par(mfrow=c(1,1))
with(Weight.and.Height,plot(X,Y))
```

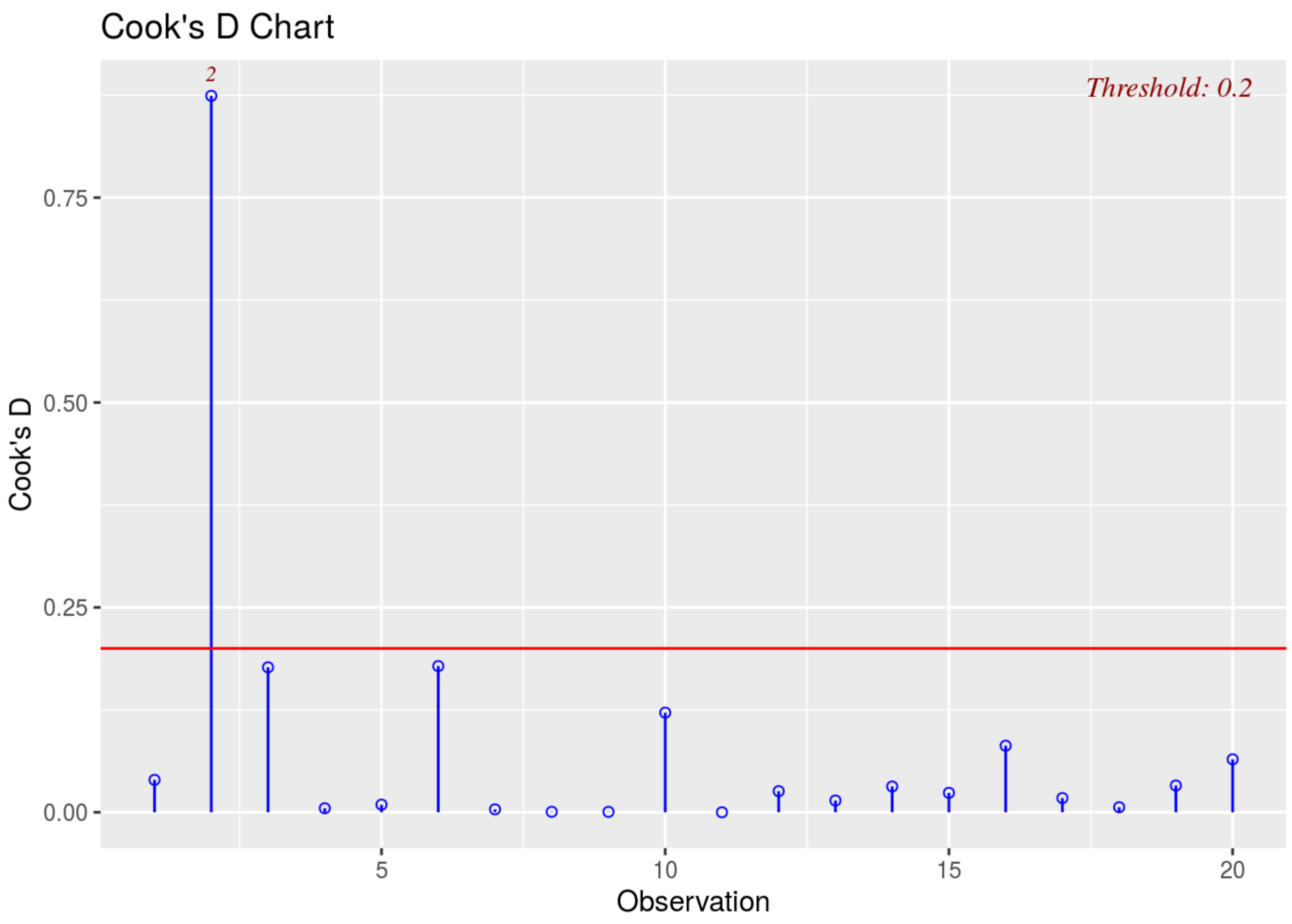
```
abline(hw9.pr1)
library(olsrr)
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```



```
ols_plot_cooksd_chart(hw9.pr1)
```



b)

_Solution: Obs=2 has the smallest weight. Obs=3 has the second smallest weight. They are outliers and it makes to give them smaller weights.

```
ei<-hw9.pr1$residuals
mei<-median(ei)
mad<-(1/0.6745)*median(abs(ei-mei))
ui<- ei/mad
abs.ui<-abs(ui)
wi<-ei
for (i in 1:20){if(abs.ui[i]<= 1.345) {wi[i]=1} else {wi[i]=1.345/abs.ui[i]}}
#the r function below calculates the weights as well
#psi.huber(ui, k = 1.345, deriv = 0)
```

c)

_Solution:Y= -236.26+5.84X. The coefficents are slightly changed and Rsquare is inceased to 47% from 38%.

```
hw9.pr2<-lm(Y~X,weights=wi,data=Weight.and.Height)
```

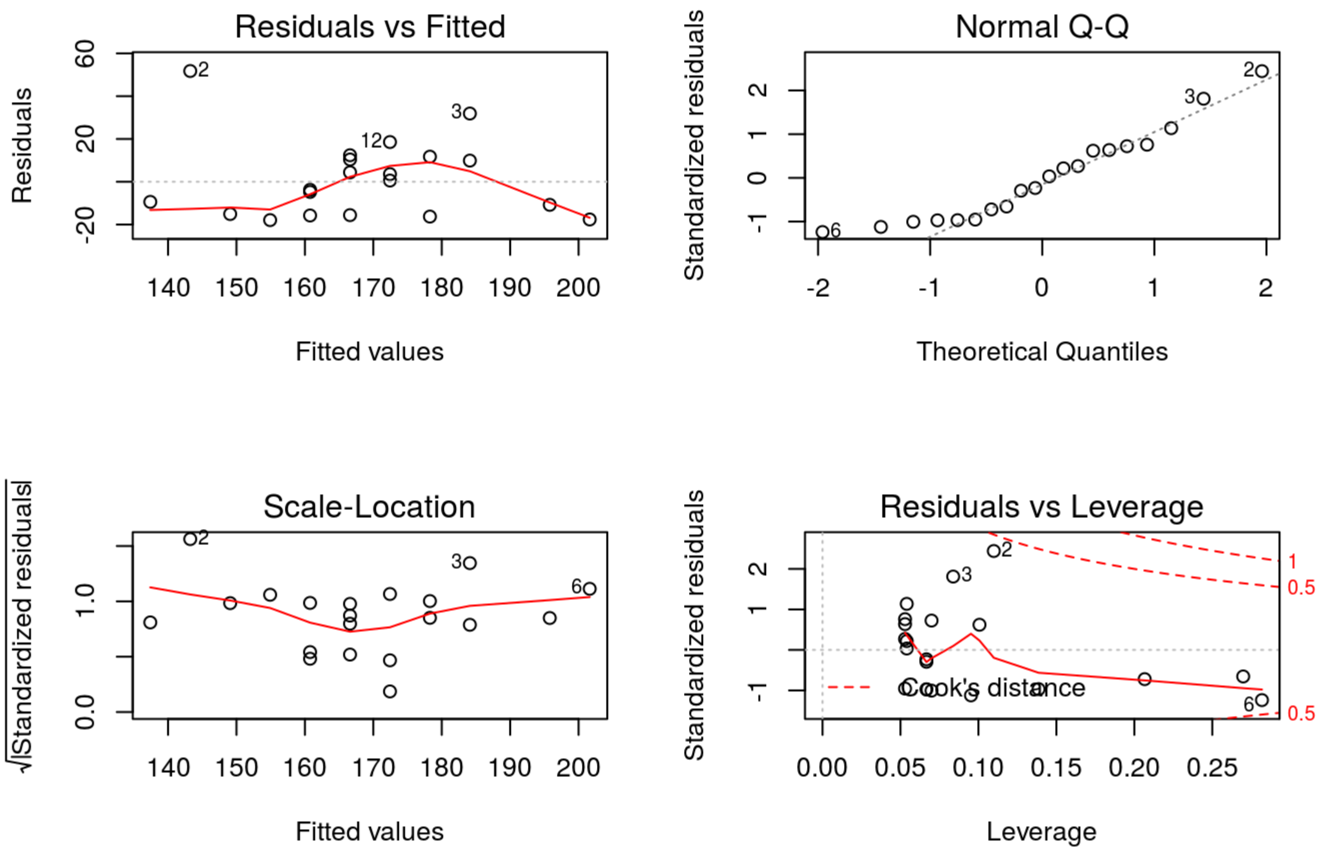
```
summary(hw9.pr2)
```

```
##
## Call:
## lm(formula = Y ~ X, data = Weight.and.Height, 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
```

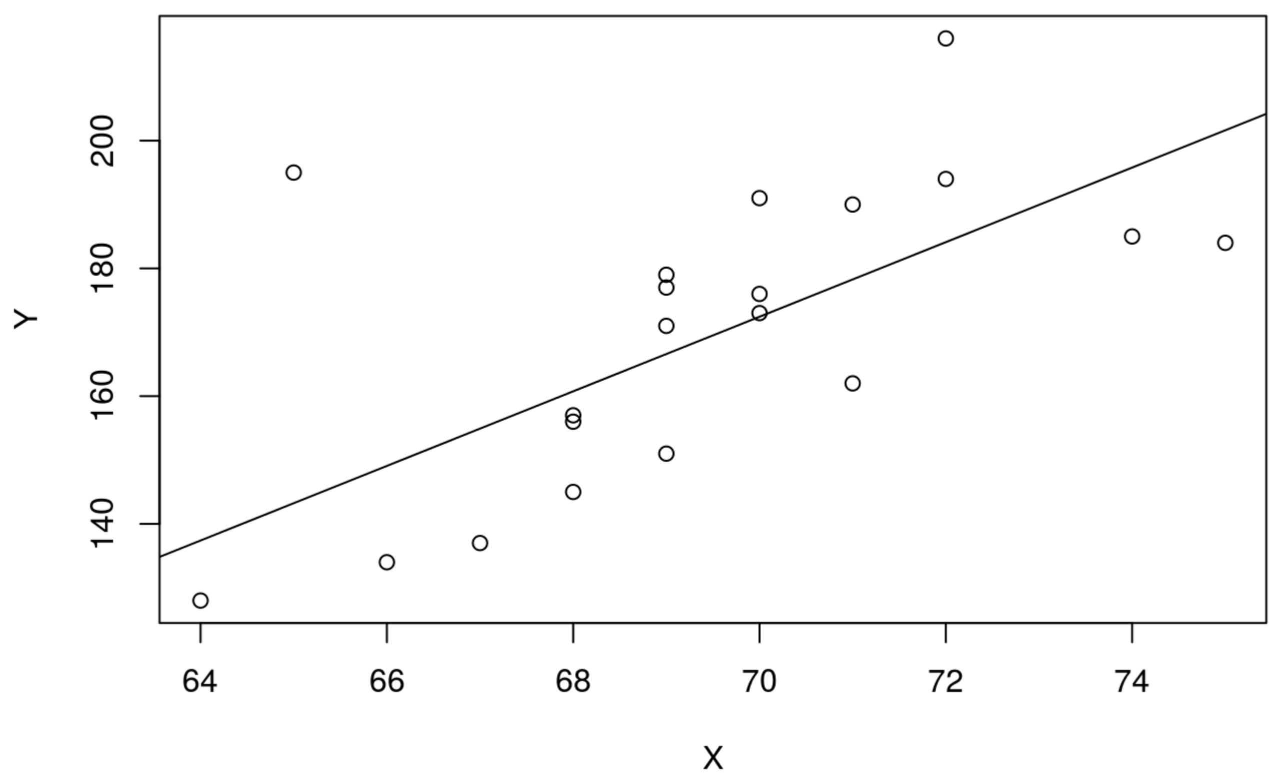
```
hw9.pr2$coefficients
```

```
## (Intercept)           X
## -236.259077    5.838481
```

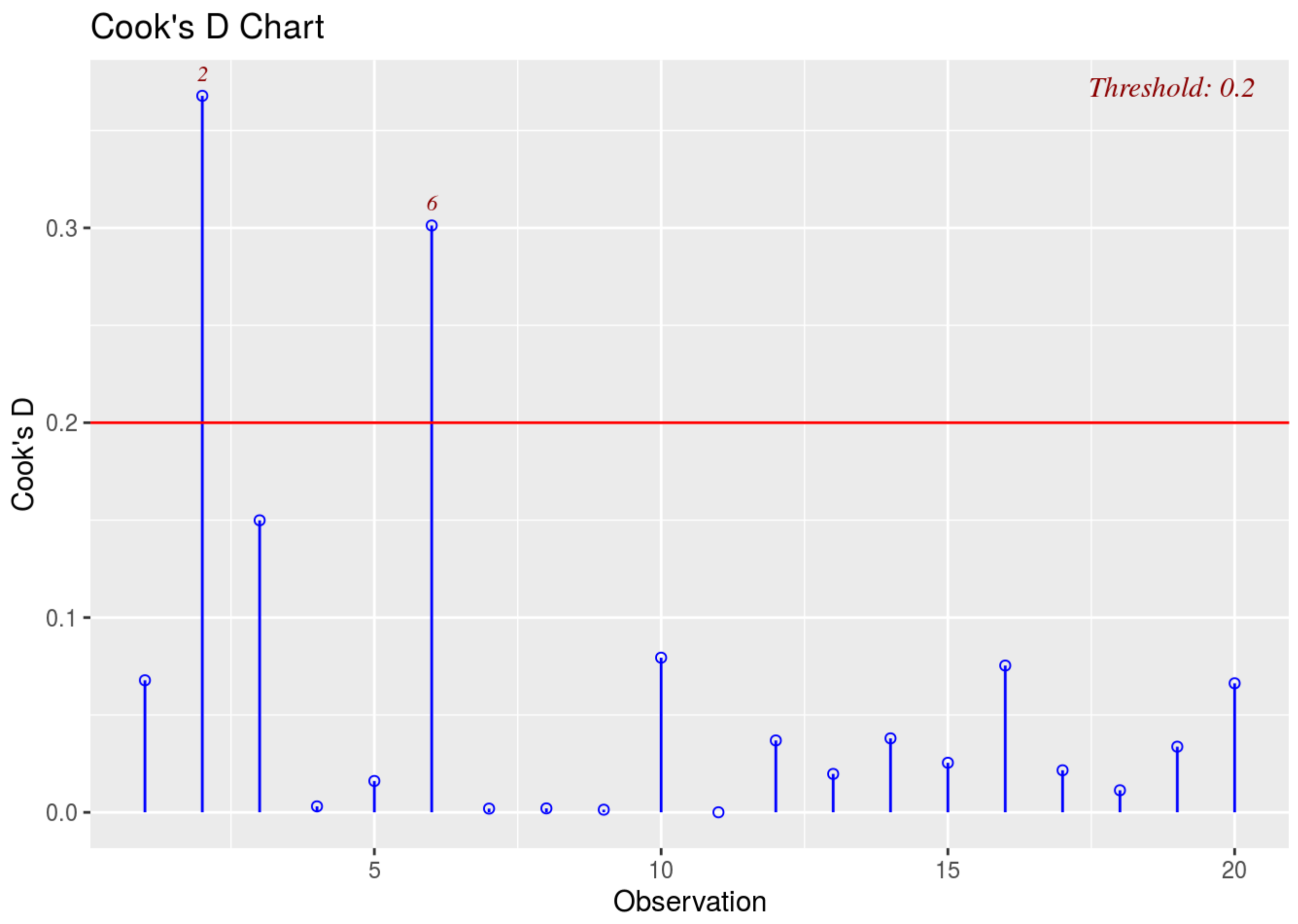
```
par(mfrow=c(2,2))
plot(hw9.pr2)
```



```
par(mfrow=c(1,1))
with(Weight.and.Height,plot(X,Y))
abline(hw9.pr2)
```



```
ols_plot_cooksd_chart(hw9.pr2)
```



d)

_Solution: Same as part b. Obs=2 and Obs=3 have the smallest weights after 2 more iterations.

```
#first iteration
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:olsrr':
##
##     cement
```

```
ei<-hw9.pr2$residuals
mei<-median(ei)
mad<-(1/0.6745)*median(abs(ei-mei))
ui<- ei/mad
wi=psi.huber(ui, k = 1.345, deriv = 0)
hw9.pr3<-lm(Y~X,weights=wi,data=Weight.and.Height)
#second iteration
```

```
ei<-hw9.pr3$residuals
mei<-median(ei)
mad<-(1/0.6745)*median(abs(ei-mei))
ui<- ei/mad
wi=psi.huber(ui, k = 1.345, deriv = 0)
wi
```

```
##  [1] 1.0000000 0.5049234 0.8287764 1.0000000 1.0000000 1.0000000 1.0000000
##  [8] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [15] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

Problem 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)

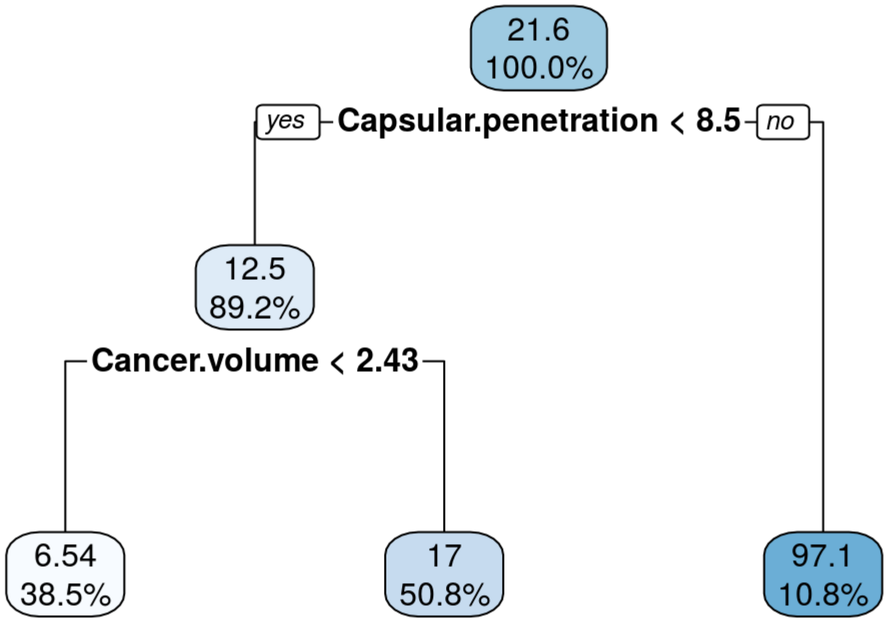
b) Compare the performance of your regression tree model with that of the best regression model obtained in HW7. Which model is more easily interpreted and why? (5pts)

a)

_Solution: Performance on the holdout is better than the development sample. Rsquare increased to 55% from 44%.

```
Prostate.Cancer <- read.csv("/cloud/project/Prostate Cancer.csv")
set.seed(1023)
ind <- sample(1:nrow(Prostate.Cancer), size =65)
dev <- Prostate.Cancer[ind,]
holdout <- Prostate.Cancer[-ind,]
```

```
library(rpart)
library(rpart.plot)
tmod<-rpart(PSA.level~.,dev)
rpart.plot(tmod, digits = 3)
```



```
sse.dev<-sum((predict(tmod)-dev$PSA.level)^2)
R2.dev<-1-sum(residuals(tmod)^2)/sum((dev$PSA.level-mean(dev$PSA.level))^2)
res.hold<-predict(tmod,holdout)-holdout$PSA.level
R2.hold<-1-sum(res.hold^2)/sum((holdout$PSA.level-mean(holdout$PSA.level))^2)
cbind(R2.dev,R2.hold)
```

```
##         R2.dev    R2.hold
## [1,] 0.4581377 -0.2008684
```

b)

_Solution: Rsquare from Tree is slightly higher than regression model (44% vs.42%).The regression model is easier to explain than tree method.Given that Rsquares are close. I would choose the regression model.

```
Prostate.Cancer <- read.csv("/cloud/project/Prostate Cancer.csv")
hw9.pr3<-lm(PSA.level~Cancer.volume+Capsular.penetration,data=Prostate.Cancer)
summary(hw9.pr3)
```

```
##
## Call:
## lm(formula = PSA.level ~ Cancer.volume + Capsular.penetration,
##     data = Prostate.Cancer)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.346  -8.324  -1.205   4.159 183.843
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1.3276     4.2861   0.310    0.757
## Cancer.volume          2.4139     0.5655   4.269 4.69e-05 ***
## Capsular.penetration   2.4533     1.1779   2.083    0.040 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.48 on 94 degrees of freedom
## Multiple R-squared:  0.4165, Adjusted R-squared:  0.4041
## F-statistic: 33.55 on 2 and 94 DF,  p-value: 1.01e-11
```

```
anova(hw9.pr3)
```

```
## Analysis of Variance Table
##
## Response: PSA.level
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## Cancer.volume         1  62202   62202  62.757 4.654e-12 ***
## Capsular.penetration  1   4300    4300   4.338   0.03999 *
## Residuals            94  93170     991
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
R2.reg<-0.4165
data.frame(cbind(R2.reg,R2.dev))
```

```
##   R2.reg    R2.dev
## 1 0.4165 0.4581377
```

Problem 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)

b)Obtain the estimated ridge standardized regression coefficients, variance inflation factors, and R2 for the following biasing constants: c = .000, .002, .004, .006, .008, .02, .04, .06,.08, .10. Suggest a reasonable value for the biasing constant c based on the ridge trace, VIF values, and R2 values.(5pts)

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)

d)Fit Lasso and Elastic Net models and compare it against the Ridge regression model results. (5pts)

a)

_Solution: Y= 62.41+1.55*X1+0.51*X2+0.10*X3-0.14*X4. No variable is significant and Rsquare us 98%. There is a strong multicollinearity in the data(VIF>10). X2 and X4; X1 and X3 are highly correlated.

```
Cement.Composition <- read.csv("/cloud/project/Cement Composition.csv")
hw9.pr4<-lm(Y~X1+X2+X3+X4,data=Cement.Composition)
summary(hw9.pr4)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4, data = Cement.Composition)
##
## 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
```

```
round(hw9.pr4$coefficients,2)
```

```
## (Intercept)          X1          X2          X3          X4
##       62.41        1.55        0.51        0.10       -0.14
```

```
library(faraway)
```

```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
##
## Attaching package: 'faraway'
```

```
## The following object is masked from 'package:rpart':
##
##     solder
```

```
## The following object is masked from 'package:olsrr':
##
##     hsb
```

```
vif(hw9.pr4)
```

```
##        X1        X2        X3        X4
##  38.49621 254.42317  46.86839 282.51286
```

```
round(cor(Cement.Composition),2)
```

```
##        Y    X1    X2    X3    X4
## Y   1.00  0.73  0.82 -0.53 -0.82
## X1  0.73  1.00  0.23 -0.82 -0.25
## X2  0.82  0.23  1.00 -0.14 -0.97
## X3 -0.53 -0.82 -0.14  1.00  0.03
## X4 -0.82 -0.25 -0.97  0.03  1.00
```

b)

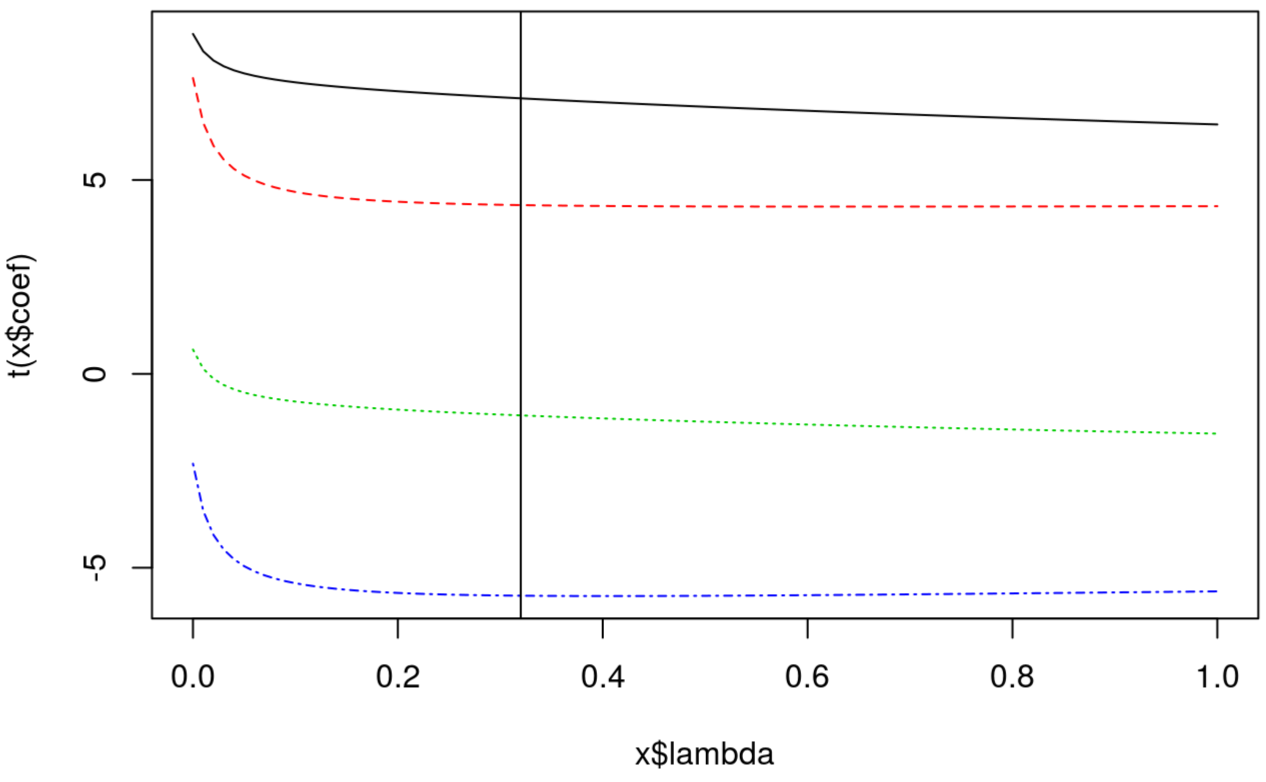
Solution: Choice of c or lamda is subjective. C wit the lowest cross validation error was selected. However, you could choose any point between 0.10 and 0.32 as VIF<10.

```
library(MASS)
lmd<-seq(0,1,by=0.01)
rgmod <- lm.ridge(Y~X1+X2+X3+X4,data=Cement.Composition, lambda =lmd)

plot(rgmod)
which.min(rgmod$GCV)
```

```
## 0.32
##   33
```

```
abline(v=0.32)
```



```
# get the VIFs
```

```
library(genridge)
```

```
## Loading required package: car
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following objects are masked from 'package:faraway':
##
##     logit, vif
```

```
X<-model.matrix(hw9.pr4)[,-1]
Y<-Cement.Composition$Y
f1<-ridge(Y, X, lambda=lmd)
v<-vif.ridge(f1)
#R Squares
SSE<- 8*f1$mse
SST<-(12)*var(Y)
RSquare<- (1-SSE/SST)
cbind(lmd,v,RSquare)
```

```
##       lmd        X1         X2        X3         X4   RSquare
## 0.00 0.00 38.496211 254.423166 46.868386 282.512865 0.9843339
## 0.01 0.01 19.197052 117.758126 22.935737 130.549055 0.9843111
## 0.02 0.02 12.144405  67.897644 14.193439  75.110465 0.9842810
## 0.03 0.03  8.797415  44.296965 10.047482  48.871528 0.9842558
## 0.04 0.04  6.946266  31.293385  7.756845  34.415703 0.9842351
## 0.05 0.05  5.813094  23.374373  6.356683  25.613305 0.9842177
## 0.06 0.06  5.067162  18.196435  5.436762  19.858527 0.9842025
## 0.07 0.07  4.548451  14.625890  4.798600  15.890814 0.9841887
## 0.08 0.08  4.171867  12.059971  4.336651  13.039955 0.9841759
## 0.09 0.09  3.888764  10.154159  3.990579  10.922890 0.9841637
## 0.10 0.10  3.669694   8.699940  3.723859   9.307785 0.9841518
## 0.11 0.11  3.495983   7.565110  3.513331   8.047655 0.9841401
## 0.12 0.12  3.355324   6.662561  3.343730   7.045651 0.9841284
## 0.13 0.13  3.239336   5.932997  3.204664   6.235853 0.9841167
## 0.14 0.14  3.142153   5.334924  3.088856   5.572130 0.9841048
## 0.15 0.15  3.059569   4.838600  2.991092   5.021419 0.9840927
## 0.16 0.16  2.988503   4.422253  2.907551   4.559517 0.9840803
## 0.17 0.17  2.926657   4.069648  2.835384   4.168378 0.9840677
## 0.18 0.18  2.872288   3.768483  2.772431   3.834327 0.9840547
## 0.19 0.19  2.824056   3.509292  2.717029   3.546845 0.9840414
## 0.20 0.20  2.780912   3.284702  2.667881   3.297735 0.9840278
## 0.21 0.21  2.742033   3.088893  2.623963   3.080531 0.9840138
## 0.22 0.22  2.706758   2.917229  2.584459   2.890081 0.9839994
## 0.23 0.23  2.674555   2.765974  2.548711   2.722234 0.9839846
## 0.24 0.24  2.644993   2.632094  2.516183   2.573620 0.9839695
## 0.25 0.25  2.617716   2.513102  2.486436   2.441476 0.9839540
## 0.26 0.26  2.592431   2.406945  2.459107   2.323521 0.9839380
## 0.27 0.27  2.568894   2.311916  2.433893   2.217860 0.9839217
## 0.28 0.28  2.546898   2.226586  2.410540   2.122905 0.9839050
## 0.29 0.29  2.526271   2.149749  2.388835   2.037319 0.9838879
## 0.30 0.30  2.506866   2.080385  2.368595   1.959970 0.9838705
## 0.31 0.31  2.488556   2.017627  2.349666   1.889895 0.9838526
## 0.32 0.32  2.471234   1.960729  2.331913   1.826267 0.9838344
## 0.33 0.33  2.454806   1.909052  2.315223   1.768377 0.9838158
## 0.34 0.34  2.439192   1.862043  2.299496   1.715612 0.9837968
## 0.35 0.35  2.424320   1.819222  2.284644   1.667439 0.9837774
## 0.36 0.36  2.410130   1.780170  2.270593   1.623396 0.9837576
## 0.37 0.37  2.396567   1.744522  2.257275   1.583077 0.9837375
## 0.38 0.38  2.383582   1.711957  2.244632   1.546127 0.9837171
## 0.39 0.39  2.371135   1.682193  2.232612   1.512234 0.9836962
## 0.40 0.40  2.359186   1.654979  2.221168   1.481120 0.9836750
## 0.41 0.41  2.347703   1.630094  2.210261   1.452541 0.9836535
## 0.42 0.42  2.336655   1.607341  2.199852   1.426281 0.9836316
## 0.43 0.43  2.326014   1.586545  2.189910   1.402144 0.9836094
## 0.44 0.44  2.315758   1.567549  2.180404   1.379958 0.9835868
## 0.45 0.45  2.305864   1.550212  2.171308   1.359567 0.9835639
## 0.46 0.46  2.296311   1.534408  2.162598   1.340833 0.9835406
## 0.47 0.47  2.287081   1.520023  2.154251   1.323630 0.9835171
## 0.48 0.48  2.278159   1.506955  2.146248   1.307846 0.9834931
## 0.49 0.49  2.269529   1.495110  2.138571   1.293377 0.9834689
## 0.50 0.50  2.261177   1.484405  2.131203   1.280132 0.9834444
## 0.51 0.51  2.253091   1.474763  2.124129   1.268027 0.9834195
## 0.52 0.52  2.245258   1.466114  2.117335   1.256984 0.9833944
## 0.53 0.53  2.237668   1.458394  2.110807   1.246935 0.9833689
## 0.54 0.54  2.230310   1.451547  2.104535   1.237816 0.9833431
## 0.55 0.55  2.223177   1.445518  2.098507   1.229570 0.9833170
## 0.56 0.56  2.216258   1.440259  2.092713   1.222142 0.9832907
## 0.57 0.57  2.209546   1.435725  2.087143   1.215483 0.9832640
## 0.58 0.58  2.203034   1.431875  2.081788   1.209551 0.9832370
## 0.59 0.59  2.196715   1.428671  2.076642   1.204302 0.9832098
## 0.60 0.60  2.190581   1.426078  2.071695   1.199699 0.9831822
## 0.61 0.61  2.184627   1.424065  2.066941   1.195707 0.9831544
## 0.62 0.62  2.178848   1.422600  2.062372   1.192294 0.9831264
## 0.63 0.63  2.173236   1.421657  2.057984   1.189429 0.9830980
## 0.64 0.64  2.167789   1.421209  2.053769   1.187085 0.9830694
## 0.65 0.65  2.162500   1.421233  2.049723   1.185236 0.9830405
## 0.66 0.66  2.157365   1.421706  2.045839   1.183858 0.9830113
## 0.67 0.67  2.152380   1.422608  2.042114   1.182929 0.9829819
## 0.68 0.68  2.147541   1.423920  2.038541   1.182428 0.9829522
## 0.69 0.69  2.142843   1.425624  2.035118   1.182336 0.9829223
## 0.70 0.70  2.138284   1.427702  2.031839   1.182634 0.9828921
## 0.71 0.71  2.133859   1.430139  2.028701   1.183307 0.9828616
## 0.72 0.72  2.129565   1.432920  2.025700   1.184338 0.9828309
## 0.73 0.73  2.125399   1.436031  2.022832   1.185712 0.9828000
## 0.74 0.74  2.121358   1.439460  2.020094   1.187417 0.9827688
## 0.75 0.75  2.117439   1.443194  2.017482   1.189438 0.9827374
## 0.76 0.76  2.113639   1.447222  2.014994   1.191763 0.9827058
## 0.77 0.77  2.109955   1.451533  2.012626   1.194382 0.9826739
## 0.78 0.78  2.106385   1.456116  2.010376   1.197283 0.9826417
## 0.79 0.79  2.102927   1.460962  2.008240   1.200457 0.9826094
## 0.80 0.80  2.099577   1.466063  2.006217   1.203894 0.9825768
## 0.81 0.81  2.096334   1.471409  2.004303   1.207584 0.9825440
## 0.82 0.82  2.093196   1.476992  2.002496   1.211520 0.9825110
## 0.83 0.83  2.090159   1.482805  2.000794   1.215694 0.9824777
## 0.84 0.84  2.087223   1.488840  1.999194   1.220097 0.9824443
## 0.85 0.85  2.084385   1.495091  1.997695   1.224723 0.9824106
## 0.86 0.86  2.081643   1.501550  1.996294   1.229565 0.9823767
## 0.87 0.87  2.078996   1.508213  1.994989   1.234617 0.9823426
## 0.88 0.88  2.076441   1.515072  1.993779   1.239872 0.9823083
## 0.89 0.89  2.073977   1.522122  1.992661   1.245324 0.9822737
## 0.90 0.90  2.071602   1.529359  1.991634   1.250969 0.9822390
## 0.91 0.91  2.069315   1.536776  1.990695   1.256801 0.9822041
## 0.92 0.92  2.067113   1.544369  1.989844   1.262815 0.9821689
## 0.93 0.93  2.064996   1.552134  1.989078   1.269007 0.9821336
## 0.94 0.94  2.062962   1.560065  1.988396   1.275371 0.9820980
## 0.95 0.95  2.061009   1.568159  1.987796   1.281904 0.9820623
## 0.96 0.96  2.059136   1.576412  1.987278   1.288601 0.9820264
## 0.97 0.97  2.057342   1.584820  1.986838   1.295459 0.9819903
## 0.98 0.98  2.055625   1.593379  1.986477   1.302474 0.9819539
## 0.99 0.99  2.053984   1.602085  1.986192   1.309643 0.9819174
## 1.00 1.00  2.052418   1.610936  1.985982   1.316961 0.9818807
```

```
t(rgmod$coef)["0.32",]
```

```
##        X1        X2        X3        X4
##  7.105196  4.351939 -1.070960 -5.720153
```

c)

_Solution: Y=84.7444062+1.2571960*X1+0.2910916*X2-0.1740310*X3-0.3556972*X4, see below for the fitted values, and errors.

```
#orgignal coefficents
coef(rgmod)["0.32",]
```

```
##                     X1         X2         X3         X4
## 84.7444062  1.2571960  0.2910916 -0.1740310 -0.3556972
```

```
ypred <- model.matrix(hw9.pr4) %*% coef(rgmod)["0.32",]
ei.ridge<-ypred-Y
round(cbind(ypred,Y,ei.ridge),2)
```

```
##                Y
## 1    78.73  78.5  0.23
## 2    73.34  74.3 -0.96
## 3   106.37 104.3  2.07
## 4    89.49  87.6  1.89
## 5    95.90  95.9  0.00
## 6  105.19 109.2 -4.01
## 7  104.09 102.7  1.39
## 8   75.55  72.5  3.05
## 9   92.02  93.1 -1.08
## 10 114.88 115.9 -1.02
## 11  81.55  83.8 -2.25
## 12 111.95 113.3 -1.35
## 13 111.45 109.4  2.05
```

d)

Solution: Lasso has the lowest SSE with 48.36, Use Lasso.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-1
```

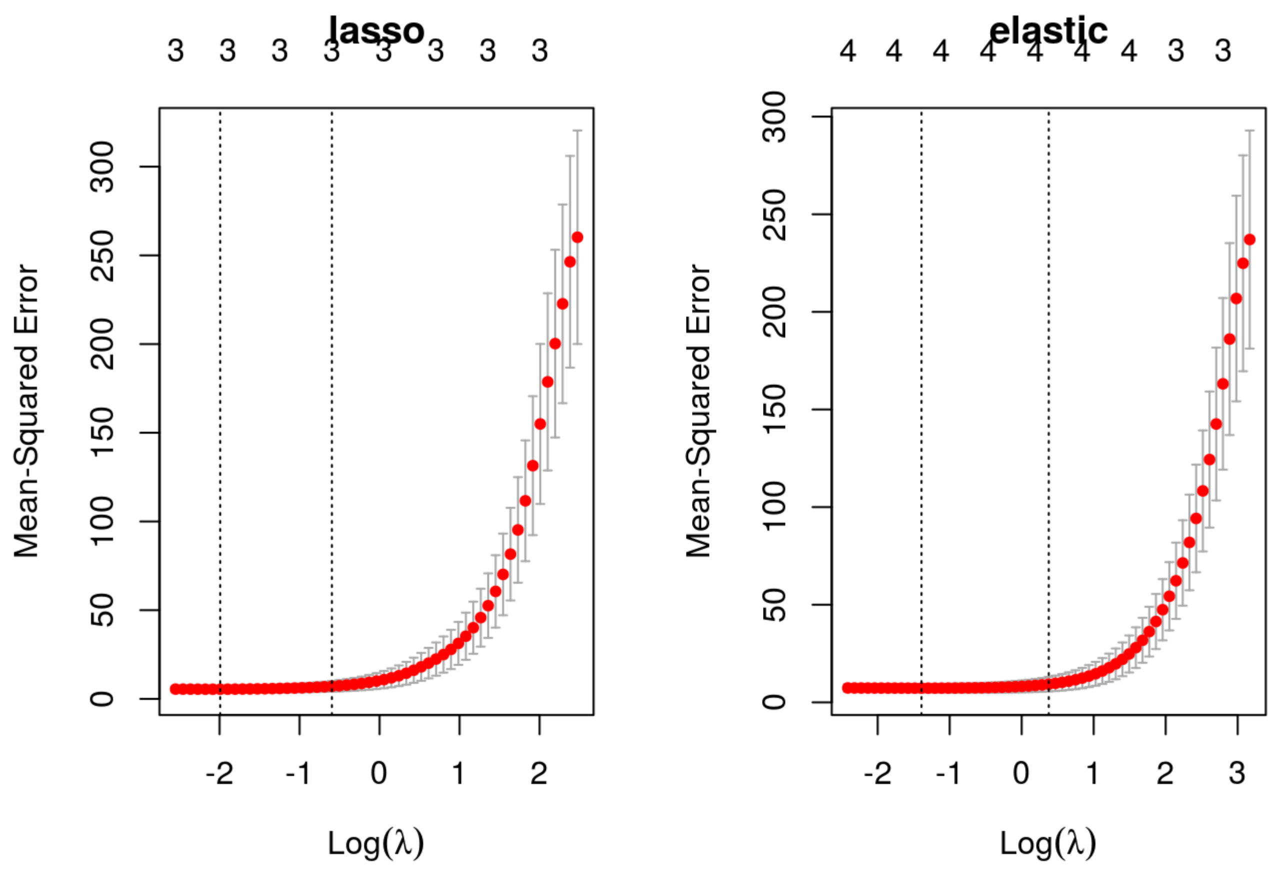
```
fit.lasso1 <- cv.glmnet(X,Y, standardize=TRUE,alpha=1)
```

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

```
fit.elnet1 <- cv.glmnet(X,Y, standardize=TRUE,alpha=.5)
```

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

```
par(mfrow=c(1,2))
plot(fit.lasso1, main="lasso")
plot(fit.elnet1, main="elastic")
```



```r
#best lamda
fit.lasso1$lambda.min
```

```
## [1] 0.136485
```

```r
fit.elnet1$lambda.min
```

```
## [1] 0.24872
```

```r
coef(fit.lasso1, s = "lambda.min")
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 71.9705570
## X1           1.4323974
## X2           0.4110560
## X3           .
## X4          -0.2343099
```

```r
coef(fit.elnet1, s = "lambda.min")
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 79.8239602
## X1           1.3431953
## X2           0.3303881
## X3          -0.0823684
## X4          -0.3121077
```

```
ei.lasso<-predict(fit.lasso1, newx = X, s = "lambda.min")-Y
ei.net<-predict(fit.elnet1, newx = X, s = "lambda.min")-Y

sse.lasso<-sum(ei.lasso^2)
sse.net<-sum(ei.net^2)
sse.ridge<-sum(ei.ridge^2)
data.frame(cbind(sse.lasso,sse.net,sse.ridge))
```

```
##   sse.lasso  sse.net sse.ridge
## 1  48.36563 48.91264  49.38976
```