Keiland Pullen DSC 424 | Fall 2021
Homework 2

1.)

Keiland Pullen

a.) $Z^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & -3 & 4 & -1 \end{bmatrix}$

b.) $Z^T Z = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & -3 & 4 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 1 & -3 \\ 1 & 4 \\ 1 & -1 \end{bmatrix} = \begin{matrix} (1*1)+(1*1)+(1*1)+(1*1) & (1*2)+(1*-3)+(1*4)+(1*-1) \\ (2*1)+(1*-3)+(4*1)+(-1*1) & (2*2)+(-3*-3)+(4*4)+(-1*-1) \end{matrix}$

$= \begin{matrix} 4 & 2+-3+4+-1 \\ 2+-3+4+-1 & 4+9+16+1 \end{matrix}$

$= \begin{bmatrix} 4 & 2 \\ 2 & 30 \end{bmatrix}$

c.) $(Z^T Z)^{-1} = \begin{bmatrix} 4 & 2 \\ 2 & 30 \end{bmatrix}^{-1}$

$= \dfrac{1}{(4*30)-(2*2)} \begin{bmatrix} 30 & -2 \\ -2 & 4 \end{bmatrix}$

$= \dfrac{1}{116} \begin{bmatrix} 30 & -2 \\ -2 & 4 \end{bmatrix}$

$= \begin{bmatrix} \frac{30}{116} & \frac{-2}{116} \\ \frac{-2}{116} & \frac{4}{116} \end{bmatrix}$

d.) $Z^T Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & -3 & 4 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 4 \\ -3 \end{bmatrix} = \begin{matrix} (1*\varnothing)+(1*1)+(1*4)+(1*-3) \\ 0+1+4-3 \\ 2 \\ (2*\varnothing)+(-3*1)+(4*4)+(-1*-3) \\ 0+-3+16+3 \\ 16 \end{matrix} = \begin{bmatrix} 2 \\ 16 \end{bmatrix}$

e.) $B = (Z^T Z)^{-1} Z^T Y = \begin{bmatrix} \frac{30}{116} & \frac{-2}{116} \\ \frac{-2}{116} & \frac{4}{116} \end{bmatrix} \begin{bmatrix} 2 \\ 16 \end{bmatrix} = \begin{matrix} \left(\frac{30}{116}*2\right)+\left(\frac{-2}{116}*16\right) \\ \left(\frac{-2}{116}*-2\right)+\left(\frac{4}{116}*16\right) \end{matrix} = \begin{matrix} \frac{60}{116}+\frac{-32}{116} \\ \frac{-4}{116}+\frac{64}{116} \end{matrix} =$

$= \begin{bmatrix} \frac{28}{116} \\ \frac{60}{116} \end{bmatrix}$

f.) $\det(Z^T Z) = \begin{bmatrix} 4 & 2 \\ 2 & 30 \end{bmatrix}$

$= (4*30)-(2*2)$

$= 116$

Keiland Pullen DSC 424 | Fall 2021
Homework 2

2.)
a.)

```
> Z = matrix(c(1,1,1,1,2,-3,4,-1), nrow=4, byrow=F)
> Z
[,1] [,2]
[1,]    1    2
[2,]    1   -3
[3,]    1    4
[4,]    1   -1
> t(Z)
 [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    2   -3    4   -1
```

b.)

```
> t(Z) %*% Z
 [,1] [,2]
[1,]    4    2
[2,]    2   30
>
```

c.)

```
> ginv(t(Z) %*% Z)
 [,1]        [,2]
[1,]  0.25862069 -0.01724138
[2,] -0.01724138  0.03448276
>
```

d.)

```
> Y = matrix(c(0,1,4,-3), nrow=4, byrow=F)
> Y
     [,1]
[1,]    0
[2,]    1
[3,]    4
[4,]   -3
> t(Z) %*% Y
[,1]
[1,]    2
[2,]   16
```

e.)

```
> ginv(t(Z) %*% Z) %*% t(Z) %*% Y
 [,1]
[1,] 0.2413793
[2,] 0.5172414
>
```

f.)

```
> det(t(Z) %*% Z)
[1] 116
```

3.)

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
>
> str(mtcars)
'data.frame':	32 obs. of  11 variables:
$ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
$ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
$ disp: num  160 160 108 258 360 ...
$ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
>
> A = mtcars[, c(2,3,4,6,11)]
> # A
> A$count <- 1
> # A
> A <- as.matrix(A)
> # A
> # t(A)
>
> View(mtcars)
>
> Y = mtcars[,c(1)]
> # Y
>
>
>
>
>
```

Keiland Pullen DSC 424 | Fall 2021
Homework 2

```
> ginv(t(A) %*% A) %*% t(A) %*% Y
            [,1]
[1,] -1.291898563
[2,]  0.011485584
[3,] -0.020352893
[4,] -3.846949031
[5,] -0.006746893
[6,] 40.815359236
>
> mpgModel = lm(mpg ~ cyl + disp + hp + wt + carb, data=mtcars)
> summary(mpgModel)

Call:
lm(formula = mpg ~ cyl + disp + hp + wt + carb, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0635 -1.4580 -0.4306  1.2927  5.8244

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 40.815359   3.025568  13.490    3e-13 ***
cyl         -1.291899   0.679227  -1.902  0.06830 .
disp         0.011486   0.015375   0.747  0.46175
hp          -0.020353   0.020062  -1.015  0.31968
wt          -3.846949   1.192155  -3.227  0.00337 **
carb        -0.006747   0.574269  -0.012  0.99072
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.56 on 26 degrees of freedom
Multiple R-squared:  0.8486,  Adjusted R-squared:  0.8195
F-statistic: 29.15 on 5 and 26 DF,  p-value: 7.056e-10
```

The Beta values in the Model are the same as those that were computed using the matrix calculations.

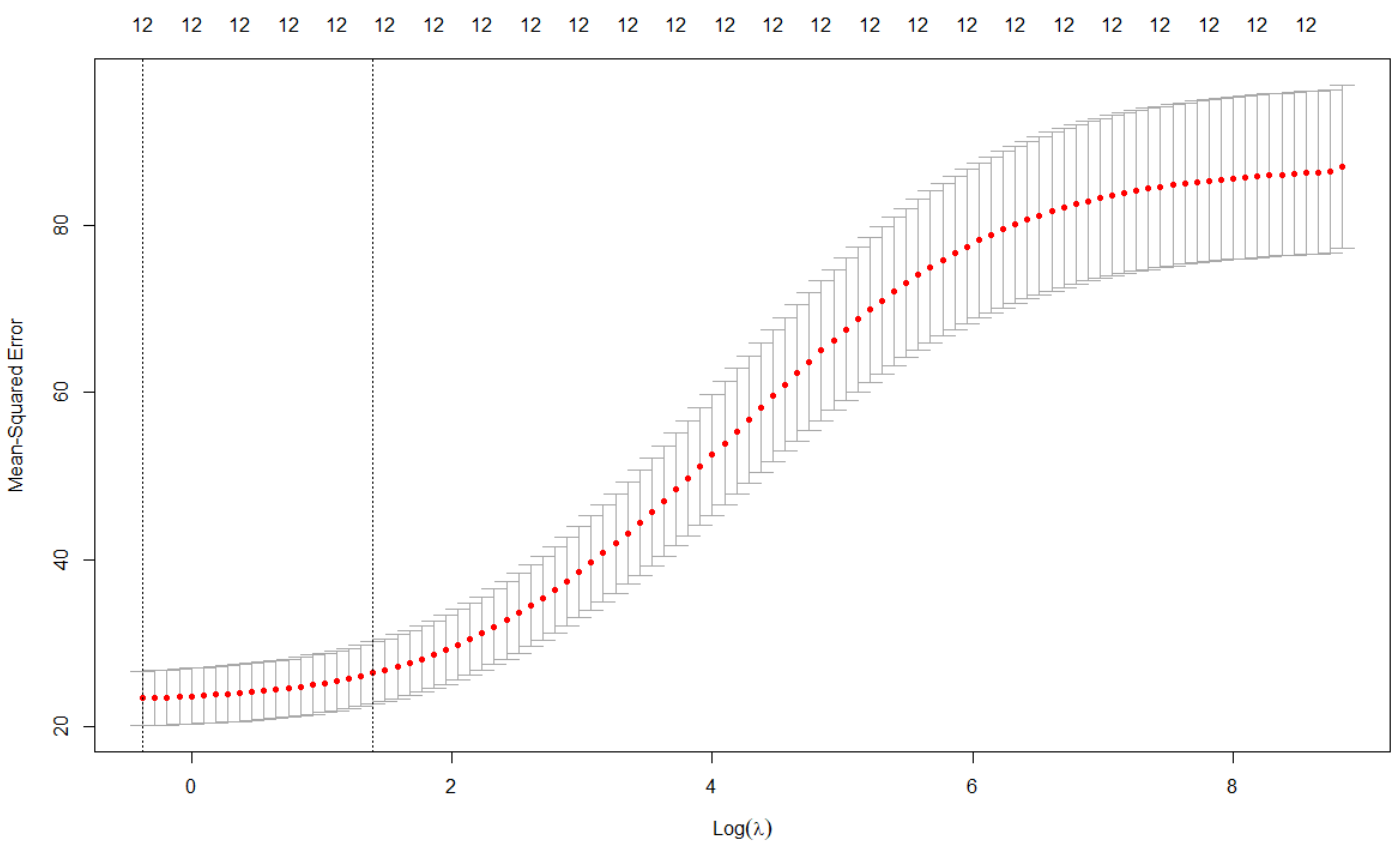
4.)
a.)

```
Call:
lm(formula = MEDV ~ ., data = housingTrain)

Residuals:
Min       1Q   Median       3Q      Max
-15.9605  -2.6653  -0.6272   1.7309  26.2670

Coefficients:
              Estimate      Std. Error    t value       Pr(>|t|)
(Intercept)   3.965e+01     5.610e+00     7.069 7.94e-12 ***
CRIM          -1.299e-01    3.412e-02     -3.807 0.000165 ***
ZN            4.341e-02     1.570e-02     2.764 0.005994 **
INDUS         6.302e-03     6.958e-02     0.091 0.927884
CHAS          3.594e+00     9.454e-01     3.802 0.000168 ***
NOX           -2.197e+01    4.377e+00     -5.021 8.05e-07 ***
RM            4.229e+00     4.898e-01     8.634  < 2e-16 ***
AGE           -1.268e-04    1.511e-02     -0.008 0.993307
DIS           -1.529e+00    2.318e-01     -6.598 1.46e-10 ***
RAD           2.665e-01     7.341e-02     3.630 0.000324 ***
TAX           -1.134e-02    4.130e-03     -2.746 0.006338 **
PTRATIO       -9.828e-01    1.506e-01     -6.526 2.24e-10 ***
LSTAT         -4.665e-01    6.094e-02     -7.655 1.73e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.661 on 367 degrees of freedom
Multiple R-squared:  0.7571,  Adjusted R-squared:  0.7491
F-statistic: 95.31 on 12 and 367 DF,  p-value: < 2.2e-16

> rmse_houseTrain = sqrt(mean(houseTrain$residuals^2))
> rmse_houseTrain
[1] 4.580769

> pred_values = predict(houseTrain, housingTest)
> rmse_houseTest = sqrt(mean((pred_values - housingTest$MEDV)^2))
> rmse_houseTest
[1] 5.263608

> (rmse_houseTest / rmse_houseTrain)
[1] 1.149067
```

Yes, there appears to be overfitting, because the RMSE value of the Test set is greater than the RMSE value of the training. Dividing the RMSEs (rmse_houseTest/rmse_houseTrain) tells us that the difference between the two is 114%.

b.)

```
library(glmnet)

XhousingTrain = as.matrix(housingTrain[,-13])
YhousingTrain = as.matrix(housingTrain[,13])

XhousingTrain
YhousingTrain

XhousingTest = as.matrix(housingTest[,-13])
YhousingTest = as.matrix(housingTest[,13])

XhousingTest
YhousingTest

lRange = seq(0, 3, .1)
ridge_housingTrain = cv.glmnet(XhousingTrain, YhousingTrain, alpha=0, nfolds=7)
ridge_housingTrain
ridge_housingTrain$lambda.min
ridge_housingTrain$lambda.1se
plot(ridge_housingTrain)
```



```
lRange = seq(0, 3, .1)
ridge_housingTrain = cv.glmnet(XhousingTrain, YhousingTrain, alpha=0, nfolds=7)

Call:  cv.glmnet(x = XhousingTrain, y = YhousingTrain, nfolds = 7, alpha = 0)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min  0.690   100   23.40 3.269      12
1se  4.041    81   26.42 3.714      12


> ridge_housingTrain$lambda.min
[1] 0.6899987

> ridge_housingTrain$lambda.1se
[1] 4.041337

> fitRidgeHousing = glmnet(XhousingTrain, YhousingTrain, alpha=0, lambda=4.041337)
> fitRidgeHousing

Call:  glmnet(x = XhousingTrain, y = YhousingTrain, alpha = 0, lambda = 4.041337)

Df %Dev Lambda
1 12 71.2  4.041
```

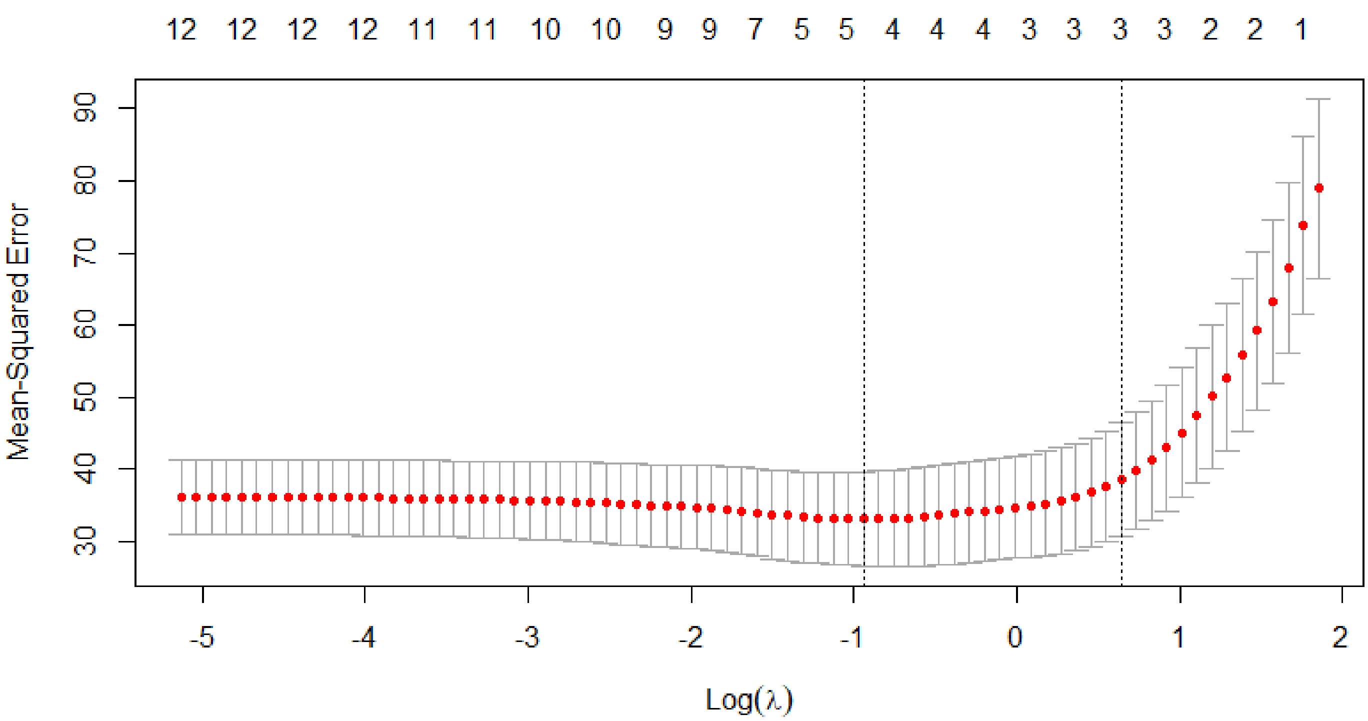
The R-squared value for the Training set is 71.2%

```
> rmse_ridge_housingTrain = sqrt(mean((pred_ridge_housingTest - YhousingTest)^2))
> rmse_ridge_housingTrain
[1] 5.587147

> rmse_ridge_housingTest = sqrt(mean((pred_ridge_housingTest - YhousingTest)^2))
> rmse_ridge_housingTest
[1] 6.409954

> (rmse_ridge_housingTest/rmse_houseTrain)
[1] 1.399318
>
```

In this case, I wonder if there is an error in calculations, because the difference in the RMSE seems to be increasing.

c.)

```
> lRange = seq(0, 3, .1)
> lasso_housingTrain = cv.glmnet(XhousingTrain, YhousingTrain, alpha=1)
> lasso_housingTrain

Call:  cv.glmnet(x = XhousingTrain, y = YhousingTrain, alpha = 1)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min 0.0237    62   23.36 4.252      10
1se 0.5100    29   27.52 4.708       8

> predLasso_train = predict(lasso_housingTrain, newx=XhousingTrain, s="lambda.1se")
> predLasso_test = predict(lasso_housingTrain, newx=XhousingTest, s="lambda.1se")
>
> rmseLasso_train = sqrt(mean((predLasso_train - YhousingTrain)^2))
> rmseLasso_train
[1] 5.014091
>
> rmseLasso_test = sqrt(mean((predLasso_test - YhousingTest)^2))
> rmseLasso_test
[1] 5.488786
```



The RMSE values using the Ridge technique are larger than the RMSE values using LASSO. Also, the difference between the RMSE values is smaller in the LASSO technique. LASSO regression suggests that the number of variables that have an impact on our Y-value is either 4 or 5 variables. In the previous assignment, using OLS, there were twice as many variables that were considered to be significant.

d.)

An eye comparison of the Ridge and LASSO plots looks similar between the Lambdas. The main difference between the two appears to be the number of variables. While Ridge has used all of the variables, LASSO has demonstrated variable selection by only using 5 variables. Both of these regularization techniques demonstrate, equally, that the cross-validated error can be reduced. This means that the overfitting in the original OLS model can be greatly reduced.

5.)
a.)

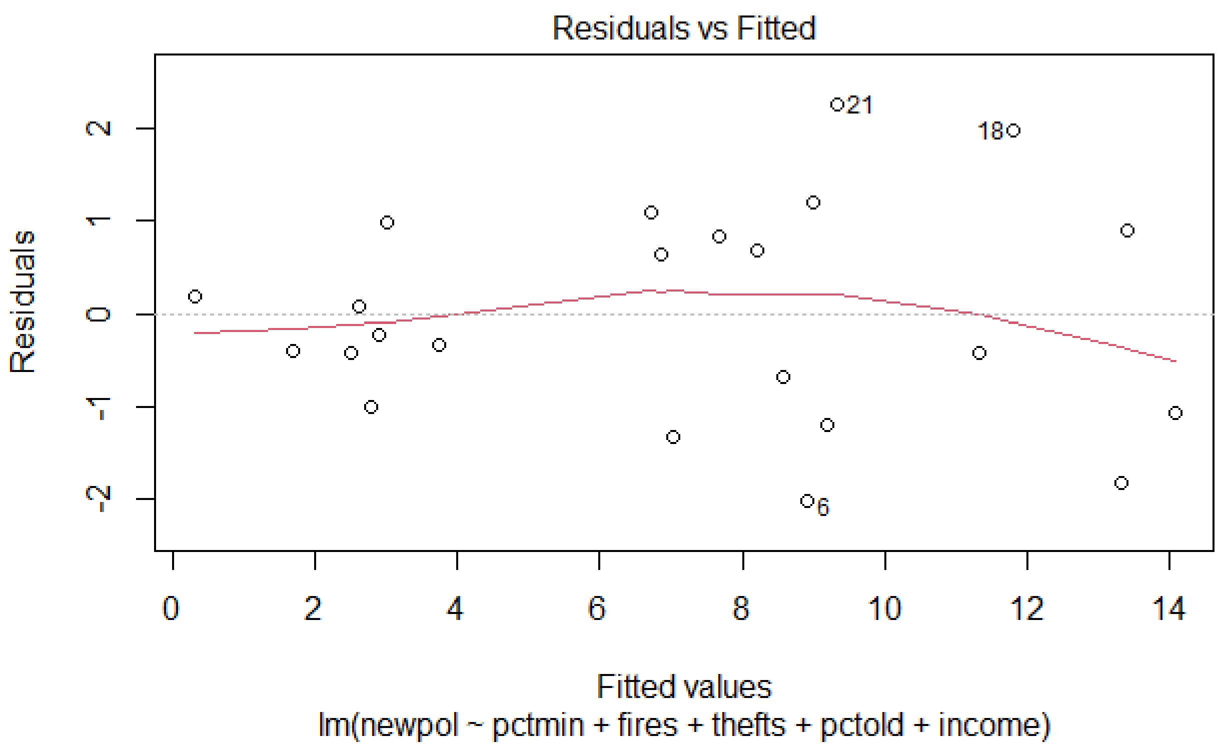
```
> insure_Model_train = lm(newpol ~ pctmin + fires + thefts + pctold + income , data=insureTrain)
> summary(insure_Model_train)

Call:
lm(formula = newpol ~ pctmin + fires + thefts + pctold + income, data = insureTrain)

Residuals:
Min      1Q  Median      3Q     Max
-2.0116 -0.8380 -0.2138  0.8646  2.2625

Coefficients:
             Estimate    Std. Error   t value Pr(>|t|)
(Intercept)  13.1561459  3.1181250    4.219   0.000577 ***
pctmin       -0.0506103  0.0203648    -2.485 0.023654 *
fires        -0.1148724  0.0532530    -2.157 0.045603 *
thefts       -0.0983249  0.0312860    -3.143 0.005934 **
pctold       -0.0628540  0.0215552    -2.916 0.009631 **
income       0.0003252   0.0002103    1.546 0.140407
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.303 on 17 degrees of freedom
Multiple R-squared:  0.9272,  Adjusted R-squared:  0.9058
F-statistic: 43.32 on 5 and 17 DF,  p-value: 4.387e-09

> rmse_train = sqrt(mean(insure_Model_train$residuals^2))
> rmse_train
[1] 1.120282
>
```

At the 0.5 level, there are two variables that are difference from zero, they are "pctmin" and "fires". There is one variable, "income" that appears to have no significance on the response variable. Additionally, the variables in this model that appear to have the best significance are "pct-old" and "thefts".

To view if any of the predictors displayed different signs than in a correlation table, the following correlation table was created with "cor(insureTrain)":

> cor(insureTrain)

	zipcode	pctmin	fires	thefts	pctold	newpol	fairpol	income
zipcode	1.00000000	-0.1722320	-0.2270551	-0.35221350	-0.21830656	0.2657576	-0.2820865	-0.03919324
pctmin	-0.17223198	1.0000000	0.7136507	0.22273302	0.31367646	-0.8398859	0.7135159	-0.70472179
fires	-0.22705509	0.7136507	1.0000000	0.25394242	0.46085852	-0.8040214	0.8996296	-0.51931579
thefts	-0.35221350	0.2227330	0.2539424	1.00000000	0.04065939	-0.3847511	0.0740055	0.29051791
pctold	-0.21830656	0.3136765	0.4608585	0.04065939	1.00000000	-0.6282507	0.4704286	-0.54942467
newpol	0.26575757	-0.8398859	-0.8040214	-0.38475109	-0.62825072	1.0000000	-0.7633393	0.66678452
fairpol	-0.28208651	0.7135159	0.8996296	0.07400550	0.47042858	-0.7633393	1.0000000	-0.61418974
income	-0.03919324	-0.7047218	-0.5193158	0.29051791	-0.54942467	0.6667845	-0.6141897	1.00000000



The above plot displays linearity in addition to several outliers.

b.)

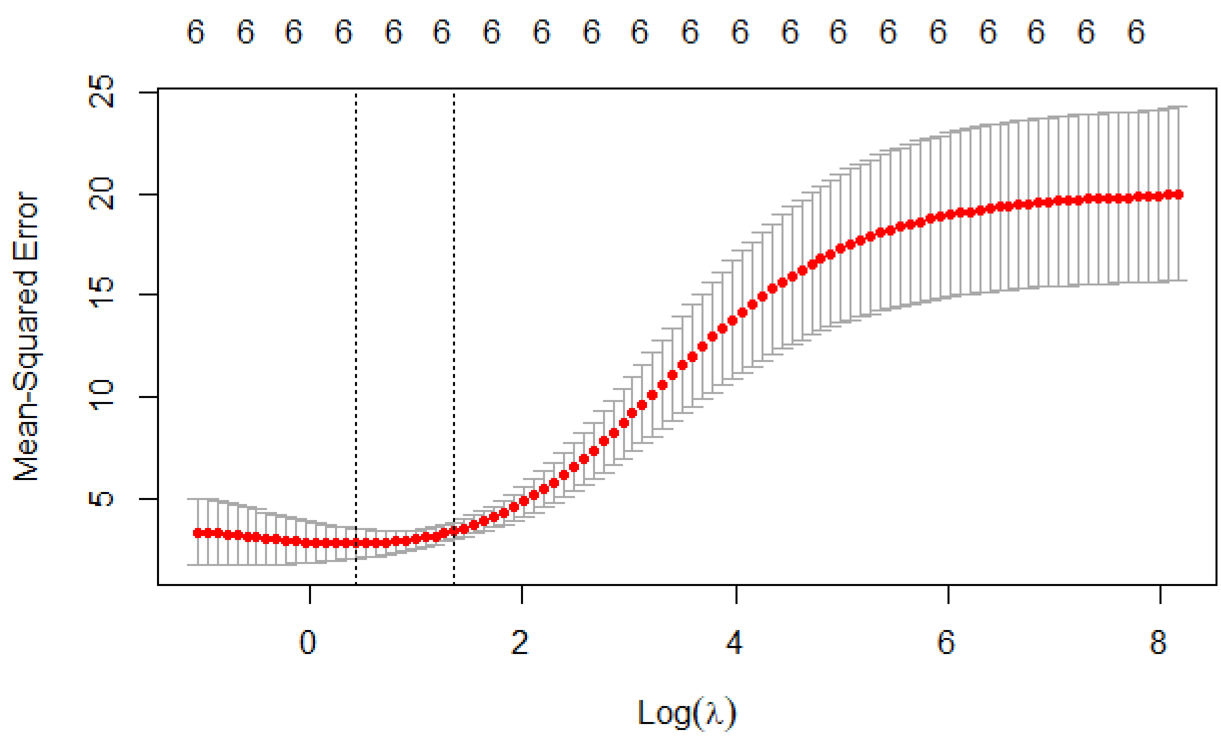
```
> insure_Model_test = lm(newpol ~ pctmin + fires + thefts + pctold + income , data=insureTest)
> summary(insure_Model_test)

Call:
lm(formula = newpol ~ pctmin + fires + thefts + pctold + income, data = insureTest)

Residuals:
Min      1Q  Median      3Q     Max
-2.5751 -0.8723  0.0170  0.3921  4.2468

Coefficients:
             Estimate   Std. Error  t value  Pr(>|t|)
(Intercept)  2.3464760  4.9457686   0.474    0.6409
pctmin      -0.0266274  0.0158485  -1.680    0.1102
fires       -0.0530762  0.0629142  -0.844    0.4099
thefts       0.0294532  0.0179499   1.641    0.1182
pctold      -0.0504838  0.0189129  -2.669    0.0156 *
income       0.0007084  0.0003381   2.095    0.0506 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.721 on 18 degrees of freedom
Multiple R-squared:  0.827,    Adjusted R-squared:  0.779
F-statistic: 17.21 on 5 and 18 DF,  p-value: 2.651e-06

> rmse_test = sqrt(mean(insure_Model_test$residuals^2))
> rmse_test
[1] 1.490679
```

The RMSE value for the Test data is larger than the RMSE value of the Training data, which indicates overfitting.

c.)

```
XinsureTrain = as.matrix(insureTrain[, -c(1,6)])
YinsureTrain = as.matrix(insureTrain[,6])

XinsureTest = as.matrix(insureTest[,-c(1,6)])
YinsureTest = as.matrix(insureTest[,6])
```

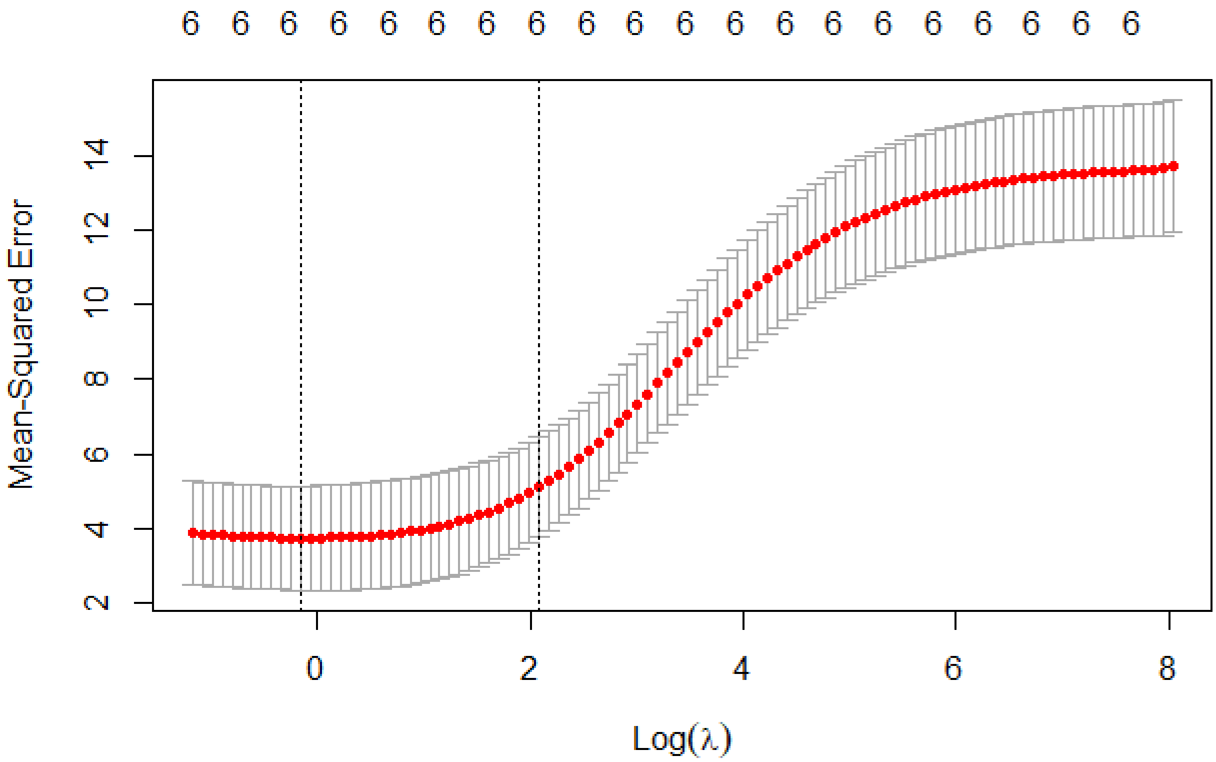
```
> lRange = seq(0, 3, .1)
> ridge_insureTrain = cv.glmnet(XinsureTrain, YinsureTrain, alpha=0, nfolds=7)
> ridge_insureTrain

Call:  cv.glmnet(x = XinsureTrain, y = YinsureTrain, nfolds = 7, alpha = 0)

Measure: Mean-Squared Error

    Lambda Index Measure     SE Nonzero
min  1.545    84   2.810 0.7420       6
1se  3.918    74   3.425 0.3994       6

> ridge_insureTrain$lambda.min
[1] 1.545409

> ridge_insureTrain$lambda.1se
[1] 3.918176
```



The training plot displays that regularization is definitely combating any overfitting of data.

```
library(glmnet)

XinsureTrain = as.matrix(insureTrain[, -c(1,6)])
YinsureTrain = as.matrix(insureTrain[,6])

XinsureTrain
YinsureTrain

XinsureTest = as.matrix(insureTest[,-c(1,6)])
YinsureTest = as.matrix(insureTest[,6])

XinsureTest
YinsureTest

lRange = seq(0, 3, .1)
ridge_insureTrain = cv.glmnet(XinsureTrain, YinsureTrain, alpha=0, nfolds=7)
ridge_insureTrain
ridge_insureTrain$lambda.min
ridge_insureTrain$lambda.1se
plot(ridge_insureTrain)

pred_ridge_Train = predict(ridge_insureTrain, XinsureTest, s="lambda.1se" )
pred_ridge_Train

rmse_ridge_Test = sqrt(mean((pred_ridge_Train - YinsureTest)^2))
rmse_ridge_Test

lRange = seq(0, 3, .1)
ridge_insureTest = cv.glmnet(XinsureTest, YinsureTest, alpha=0, nfolds=7)
ridge_insureTest
ridge_insureTest$lambda.min
ridge_insureTest$lambda.1se

pred_ridge_Test = predict(ridge_insureTest, XinsureTest, s="lambda.1se" )
pred_ridge_Test

rmse_ridge_Test = sqrt(mean((pred_ridge_Test - YinsureTest)^2))
rmse_ridge_Test


> rmse_ridge_Train = sqrt(mean((pred_ridge_Train - YinsureTest)^2))
> rmse_ridge_Train
[1] 2.656355
>

> rmse_ridge_Test = sqrt(mean((pred_ridge_Test - YinsureTest)^2))
> rmse_ridge_Test
[1] 2.131394
```

The following is the plot for the test data.



d.)

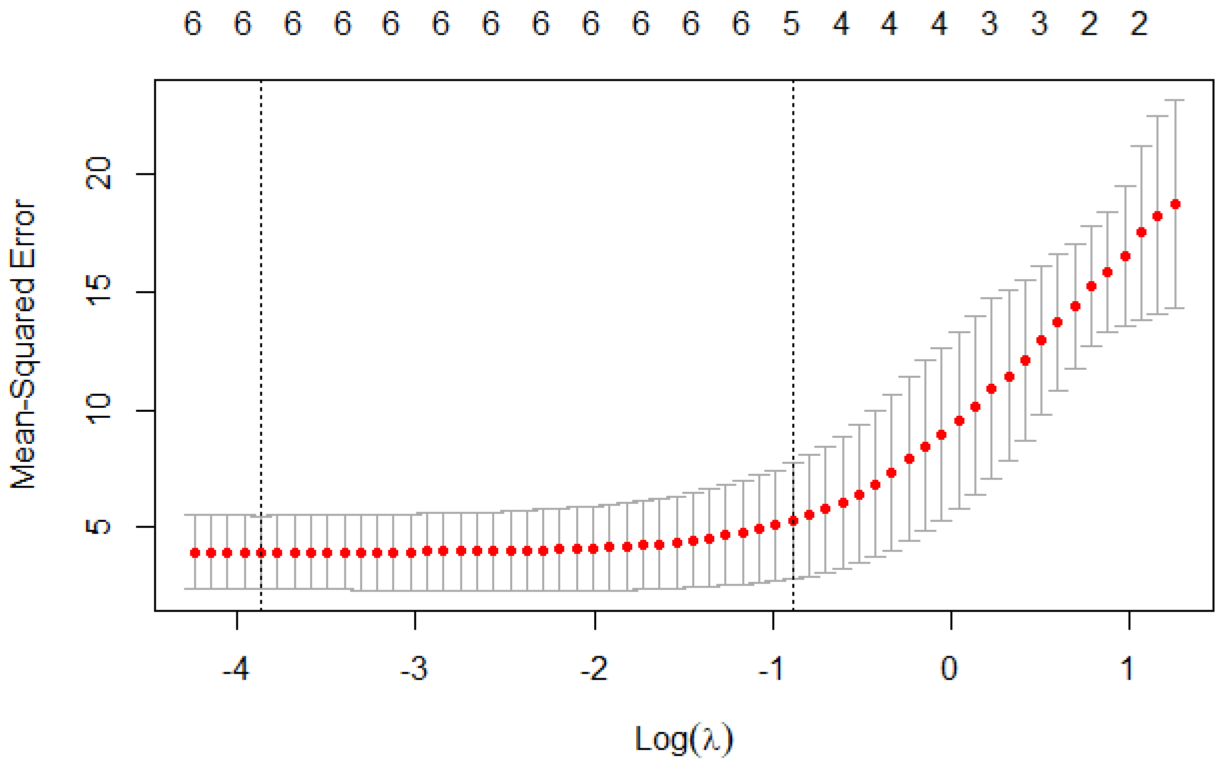
```
> lasso_insureTrain = cv.glmnet(XinsureTrain, YinsureTrain, alpha=1, nfolds=7)
> lasso_insureTrain

Call:  cv.glmnet(x = XinsureTrain, y = YinsureTrain, nfolds = 7, alpha = 1)

Measure: Mean-Squared Error

     Lambda Index Measure    SE Nonzero
min 0.0209    56   3.928 1.573       6
1se 0.4105    24   5.289 2.459       5
>
```

```
lasso_insureTrain = cv.glmnet(XinsureTrain, YinsureTrain, alpha=1, nfolds=7)
lasso_insureTrain
plot(lasso_insureTrain)
```

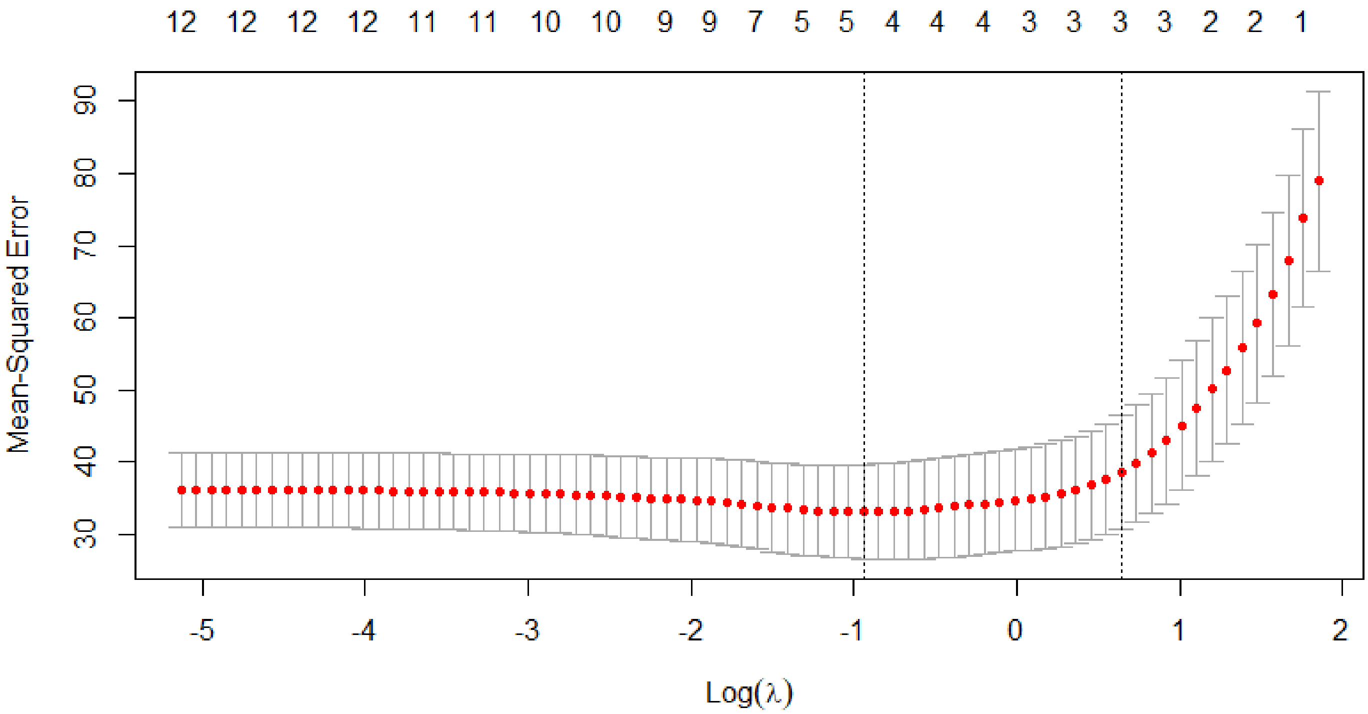
Keiland Pullen
Homework 2

DSC 424 | Fall 2021

The following is the LASSO plot for the training data.



lasso_insureTest = cv.glmnet(XinsureTest, YinsureTest, alpha=1, nfolds=7)
lasso_insureTest
plot(lasso_insureTrain)



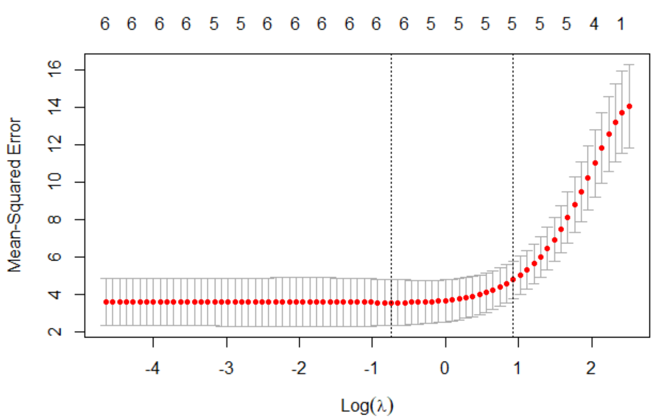
The above LASSO plots for the Test and Training data appear to be close to those of the Ridge plots. It is difficult to determine if there is any instability.

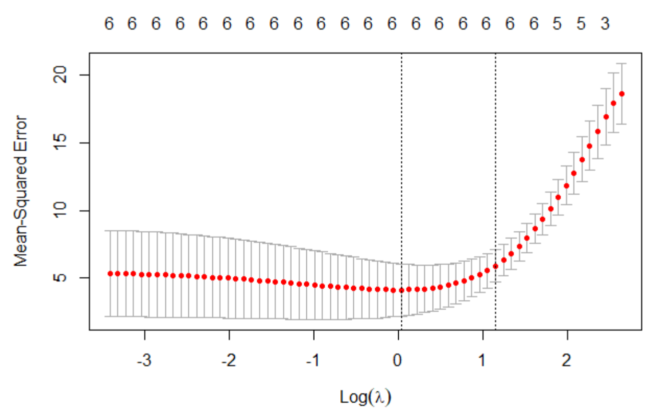
```
> predLasso_train = predict(lasso_insureTrain, newx=XinsureTrain, s="lambda.1se")
> predLasso_test = predict(lasso_insureTrain, newx=XinsureTest, s="lambda.1se")
>
> rmseLasso_train = sqrt(mean((predLasso_train - YinsureTrain)^2))
> rmseLasso_train
[1] 1.336242
>
> rmseLasso_test = sqrt(mean((predLasso_test - YinsureTest)^2))
> rmseLasso_test
[1] 2.71357
>
```

f.)

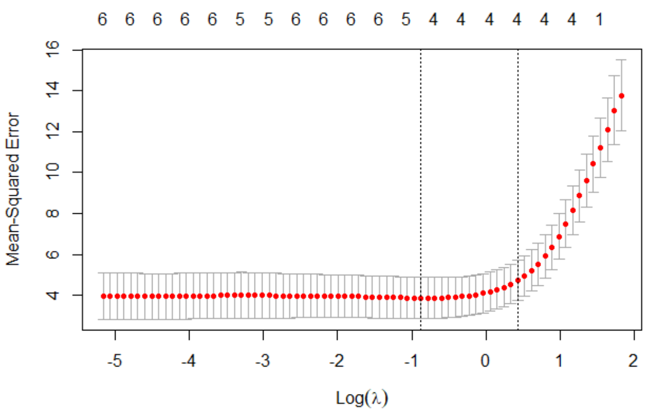
```
elastic1_insureTrain = cv.glmnet(XinsureTrain, YinsureTrain, alpha=0.25, nfolds=7)
elastic1_insureTrain

elastic1_insureTest = cv.glmnet(XinsureTest, YinsureTest, alpha=0.25, nfolds=7)
elastic1_insureTest

elastic2_insureTrain = cv.glmnet(XinsureTrain, YinsureTrain, alpha=0.5, nfolds=7)
elastic2_insureTrain

elastic2_insureTest = cv.glmnet(XinsureTest, YinsureTest, alpha=0.5, nfolds=7)
elastic2_insureTest

elastic3_insureTrain = cv.glmnet(XinsureTrain, YinsureTrain, alpha=0.5, nfolds=7)
elastic3_insureTrain

elastic3_insureTest = cv.glmnet(XinsureTest, YinsureTest, alpha=0.5, nfolds=7)
elastic3_insureTest

>
> elastic1_insureTest = cv.glmnet(XinsureTest, YinsureTest, alpha=0.25, nfolds=7)
> elastic1_insureTest
```

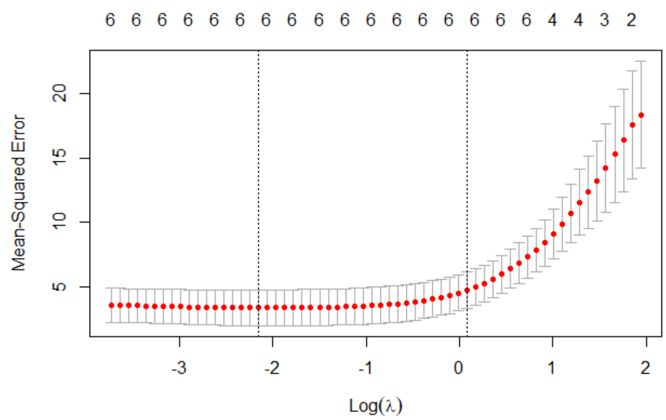
The following are the plots for the Test and Train sets with alpha=.025



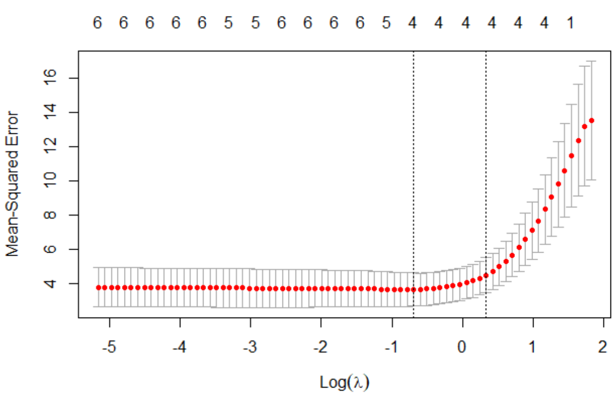


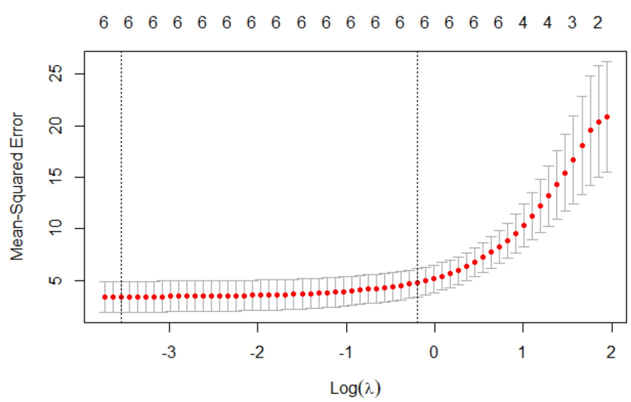
The following are the plots for the Elastic Test and Training sets at alpha=.5





The following are the plots for the Test and Training sets with alpha=0.75





In this particular data set, a case could be made for alpha=0.75 as a better choice over Ridge and LASSO.

g.)

In general, Elastic Net regression offers the benefits of LASSO and Ridge regression. Elastic offers a solution to modeling that could overcome any limitations of LASSO and Ridge regression. For this particular dataset, I'm unsure. I will have to read and ask why isn't Elastic a preferred solution over LASSO and Ridge.

6.)

a.)

In this article, the dataset is defined as n=395 instances. The number of predictor variables is 30. The variables are organized into binary, categorical and integer. The binary and integer are a mixture of discrete and continuous type variables. For the size of this dataset, there should be some concern regarding the relatively large number of variable.

b.)

There may have been overfitting when using the classic OLS method. The article states that OLS performed well in only 1% of the time. It is also stated that the preferred regularization technique was LASSO, however, Elastic Net and Ridge also performed well.

c.)

The article states that Ridge regression was useful with shrinking the solutions that were generated with the OLS method. The LASSO model offered the best performance because it was able to create a parsimonious model using a smaller number of variables. The Elastic Net was or is a hybrid of Ridge and LASSO, however, it is noted that an additional regularization parameter increases the computational burden of the problem.

d.)

The OLS and Stepwise models predicted that males would score more points on the math exam than females. The article does state that this model includes all of the coefficients in the model. In the Ridge model, the number of males that would score more points than females was smaller. It also stated that Lasso and Elastic Net selected larger models than OLS and Stepwise selected. Additionally, the Ridge, Lasso and Elastic Net models used a much smaller number of predictors.

e.)

The article states that the mean-squared prediction error (MPSE) is used to determine which model offers the best fit across 100 random splits. The data presented displays that LASSO, Ridge and Elastic Net had MPSE values that were consistently lower than those of the OLS model. The article also states that the cross-validation models are able to estimate the MSPE with less computational work.

f.)

I believe that there are several issues with the evaluation of the models. First, the only measurement of ranking the models seems to be the models that consistently return a smaller MSPE. The article suggests that the use of bias is encouraged and adding this bias allows the model to produce results that are more favorable. Bias may be true, but shouldn't it be factored early in the evaluations? The article states early that the OLS methods are using smaller datasets than those of the regularization techniques. This suggests that the size of the training and test datasets may not be as random as mentioned.