Assignment 2

Erik Pak

2024-08-06

Load Libraries

```
# Load library
library(MASS)           # MASS package for inverses
library(tidyverse)      # Data manipulation package
library(ggplot2)        # used for ggplot
library(car)            # Variance Inflation Factor (VIF)
library(leaps)          # model selection methods
library(glmnet)         # regularization for linear regression
library(corrplot)       # correlation plot
```

Question 1:

```
# hand solution
knitr::include_graphics("hw02.png")
```

1)

$$Z = \begin{bmatrix} 1 & 2 \\ 1 & -3 \\ 1 & 4 \\ 1 & -1 \end{bmatrix} \quad Y = \begin{bmatrix} 0 \\ 1 \\ 4 \\ -3 \end{bmatrix}$$

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

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

2×4 $\quad$ 4×2

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

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

1. (20 points) Perform, by hand, the following calculations from linear algebra. For the following matrices and vectors. Submit a copy of your answers either in a high-quality scan or photo (unreadable or clipped answers will not receive credit) or you may format it carefully in Word with equation editor showing all work.

Question 2:

```
Z <- matrix(c(1,1,1,1,2,-3,4,-1), nrow = 4, ncol = 2)
Y <- matrix(c(0,1,4,-3))

# a) transpose Z
t(Z)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    1    1
## [2,]    2   -3    4   -1
```

```
# b) transpose Z * Z
t(Z) %*% Z
```

```
##      [,1] [,2]
## [1,]    4    2
## [2,]    2   30
```

```
# c) inverse of transpose Z * Z
ginv(t(Z) %*% Z)
```

```
##             [,1]        [,2]
## [1,]  0.25862069 -0.01724138
## [2,] -0.01724138  0.03448276
```

```
# d) transpose Z * Y
t(Z) %*% Y
```

```
##      [,1]
## [1,]    2
## [2,]   16
```

```
# e) calculate a regression formula
ginv(t(Z) %*% Z) %*% t(Z) %*% Y
```

```
##           [,1]
## [1,] 0.2413793
## [2,] 0.5172414
```

```
# f(det(transpose Z * Z))
det(t(Z) %*% Z)
```

```
## [1] 116
```

```r
# check it against what lm would give
X <-  c(2,-3,4,-1)
Y <-  c(0,1,4,-3)

# create data frame
df = data.frame(X, Y)

# display data frame
head(df)
```

```
##    X  Y
## 1  2  0
## 2 -3  1
## 3  4  4
## 4 -1 -3
```

```r
# model
fit = lm(Y ~ X, data=df)

# Check the "Estimate" for Beta_1 = X and Beta_0 = Intercept!
summary(fit)
```

```
##
## Call:
## lm(formula = Y ~ X, data = df)
##
## Residuals:
##      1      2      3      4
## -1.276  2.310  1.690 -2.724
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2414     1.4931   0.162    0.886
## X             0.5172     0.5452   0.949    0.443
##
## Residual standard error: 2.936 on 2 degrees of freedom
## Multiple R-squared:  0.3103, Adjusted R-squared:  -0.03448
## F-statistic:   0.9 on 1 and 2 DF,  p-value: 0.4429
```

2. (10 points) In R, write a script to compute each of the parts in problem 1 to check your answers. Submit both the `.r` commands and the output. Then, create a dataset with `x = <2, -3, 4, -1>` and `y = <0, 1, 4, -3>` and run a regression analysis on the data with "lm". Compare your value for `beta` in part e of the last problem, with the coefficients calculated by R's lm function.

Using matrix:

Bo = 0.2413793

B1 = 0.5172414

Using lm:

Bo = 0.2414

B1 = 0.5172

The betas represent the estimated coefficients that measure the relationship between the predictor variables and the response variable. The betas from the matrix are equal to the `lm` if we round the coefficients to four decimal places.

Question 3:

3. (10 pts) Use the dataset "mtcars" which is built-in RStudio. You can see the structure of the data by the command "head(mtcars)". Use the "mpg" column as your dependent variable Y, and do the following:

```
# view forst six rows
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
```

```
# view using glimpse
glimpse(mtcars)
```

```
## Rows: 32
## Columns: 11
## $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,~
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16~
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180~
## $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92,~
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.~
## $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18~
## $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,~
## $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,~
## $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,~
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

Copy Dataset

```
# select independent variable to A
A <- mtcars %>% select(cyl, disp, hp, wt, carb)

# dependent variable to Y
Y <- mtcars %>% select(mpg)

# display A & Y
glimpse(A)
```

```
## Rows: 32
## Columns: 5
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
```

```
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16~
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180~
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.~
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

```
glimpse(Y)
```

```
## Rows: 32
## Columns: 1
## $ mpg <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8, ~
```

a. Create a copy of the dataset called A with only the columns `{cyl, disp, hp, wt, carb}`. Use the column selection mechanism we covered in class to select these columns from the dataset.

```
# ones in the front with count as column name
A <- cbind(count = rep(1, nrow(A)), A)
```

b. Add a column of "ones" to A called "count".

```
# convert to matrix
A <- as.matrix(A)
Y <- as.matrix(Y)

# validate
print(class(A))
```

```
## [1] "matrix" "array"
```

```
print(class(Y))
```

```
## [1] "matrix" "array"
```

c. Use the `"as.matrix"` function to convert it to a matrix and assign it back to the variable A (so you are overwriting the data.frame here and converting it to a matrix)

```
# matrix calculation - regression
ginv(t(A) %*% A) %*% t(A) %*% Y
```

```
##               mpg
## [1,] 40.815359236
## [2,] -1.291898563
## [3,]  0.011485584
## [4,] -0.020352893
## [5,] -3.846949031
## [6,] -0.006746893
```

d. Compute the following multiple regression by manually computing the matrix operations:

Matrix Calculation: mpg = 40.815 - 1.292(cyl) + 0.011(disp) - 0.020(hp) - 3.847(wt) - 0.007(carb)

```r
# combine Y & A intercept coefficient (drop count column)
data <- cbind(Y, A[,-1])

# covert to data frame
data <- data.frame(data)

# create model
fit <- lm(mpg ~ . , data = data)

# summary of model
summary(fit)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = data)
##
## 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
```

e. Compute the regression with the RStudio "lm" command and compare with your results from d). Note any differences.

The matrix calculations can produce values with more decimal places than what is displayed by R's lm function. However, the values shown by `lm` are simply the default output format, which limits the number of decimal places displayed for readability.

The difference is the actual beta coefficients used by the `lm` function are used to calculate other metrics, such as the R-squared value, the F-statistic, and the p-values associated with each coefficient. These metrics can be used to determine the statistical significance of each predictor in the model and the overall performance of the model. However, it is still helpful to access the additional metrics provided by the `lm` function, as they can provide important information about the performance and validity of the model without taking further steps to produce in matrix calculation.

Question 4:

(20 pts) Return to the housing dataset. In the last homework you conducted a regression with various methods of feature selection. In this problem you will turn to evaluating regression's predictive power on this dataset, whether it is overfitting and investigate the performance of regularized regression. For this, I

have provided you with the data divided into test and training sets. Note that R's glmnet handles scaling internally, so you do not need to scale manually and then undo the scaling when computing betas so that the reported coefficients are applicable to the original data. All of this happens transparently. One other note here: Do not try to calculate an R2 on a test set. This measure applies only to a training set and has no meaning for out-of-sample prediction. On a test set, the so-called R2 value could be > 1 and could even be negative! This is one reason we use RMSE. One good way to compare the test and training RMSE's is to take their quotient (testRMSE / trainingRMSE) and that will tell you how much bigger test is than training %-wise.

1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town
4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX: nitric oxides concentration (parts per 10 million)
6. RM: average number of rooms per dwelling
7. AGE: proportion of owner-occupied units built prior to 1940
8. DIS: weighted distances to five Boston employment centers
9. RAD: index of accessibility to radial highways
10. TAX: full-value property-tax rate per $10,000
11. PTRATIO: pupil-teacher ratio by town
12. LSTAT: % lower status of the population
13. MEDV: Median value of owner-occupied homes in $1,000's

Import Housing Datasets

```
# import housing training & test
housingTest  <- read.csv("housingTest.csv")
housingTrain <- read.csv("housingTrain.csv")

# display both
glimpse(housingTrain)
```

```
## Rows: 380
## Columns: 13
## $ CRIM    <dbl> 0.11432, 0.08826, 0.13642, 0.17134, 2.44668, 0.03041, 3.56868,~
## $ ZN      <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 22~
## $ INDUS   <dbl> 8.56, 10.81, 10.59, 10.01, 19.58, 5.19, 18.10, 18.10, 5.19, 13~
## $ CHAS    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
## $ NOX     <dbl> 0.520, 0.413, 0.489, 0.547, 0.871, 0.515, 0.580, 0.740, 0.515,~
## $ RM      <dbl> 6.781, 6.417, 5.891, 5.928, 5.272, 5.895, 6.437, 5.818, 5.968,~
## $ AGE     <dbl> 71.3, 6.6, 22.3, 88.2, 94.0, 59.6, 75.0, 92.4, 58.5, 92.4, 97.~
## $ DIS     <dbl> 2.8561, 5.2873, 3.9454, 2.4631, 1.7364, 5.6150, 2.8965, 1.8662~
## $ RAD     <int> 5, 4, 4, 6, 5, 5, 24, 24, 5, 5, 4, 24, 7, 6, 6, 24, 1, 2, 5, 5~
## $ TAX     <int> 384, 305, 277, 432, 403, 224, 666, 666, 224, 276, 437, 666, 33~
## $ PTRATIO <dbl> 20.9, 19.2, 18.6, 17.8, 14.7, 20.2, 20.2, 20.2, 20.2, 16.4, 21~
## $ LSTAT   <dbl> 7.67, 6.72, 10.87, 15.76, 16.14, 10.56, 14.36, 22.11, 9.29, 10~
## $ MEDV    <dbl> 26.5, 24.2, 22.6, 18.3, 13.1, 18.5, 23.2, 10.5, 18.7, 23.0, 17~
```

```
glimpse(housingTest)
```

```
## Rows: 126
## Columns: 13
```

```
## $ CRIM    <dbl> 0.00632, 0.11747, 0.62739, 1.05393, 0.85204, 1.38799, 0.06417,~
## $ ZN      <dbl> 18.0, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 75.0, 0.0, 0.0, 21.0~
## $ INDUS   <dbl> 2.31, 7.87, 8.14, 8.14, 8.14, 8.14, 5.96, 5.96, 2.95, 6.91, 6.~
## $ CHAS    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ NOX     <dbl> 0.5380, 0.5240, 0.5380, 0.5380, 0.5380, 0.5380, 0.4990, 0.4990~
## $ RM      <dbl> 6.575, 6.009, 5.834, 5.935, 5.965, 5.950, 5.933, 5.850, 6.595,~
## $ AGE     <dbl> 65.2, 82.9, 56.5, 29.3, 89.2, 82.0, 68.2, 41.5, 21.8, 2.9, 40.~
## $ DIS     <dbl> 4.0900, 6.2267, 4.4986, 4.4986, 4.0123, 3.9900, 3.3603, 3.9342~
## $ RAD     <int> 1, 5, 4, 4, 4, 4, 5, 5, 3, 3, 3, 4, 3, 5, 8, 8, 3, 4, 4, 4, 4,~
## $ TAX     <int> 296, 311, 307, 307, 307, 307, 279, 279, 252, 233, 233, 243, 46~
## $ PTRATIO <dbl> 15.3, 15.2, 21.0, 21.0, 21.0, 21.0, 19.2, 19.2, 18.3, 17.9, 17~
## $ LSTAT   <dbl> 4.98, 13.27, 8.47, 6.58, 13.83, 27.71, 9.68, 8.77, 4.32, 4.84,~
## $ MEDV    <dbl> 24.0, 18.9, 19.9, 23.1, 19.6, 13.2, 18.9, 21.0, 30.8, 26.6, 21~
```

Create Subset & Matrix

Full OLS Model

```
# full model
fullOLS <- lm(MEDV ~ . , data = housingTrain)

# summary
summary(fullOLS)
```

```
##
## 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
```

```r
# rmse train
rmseOLS_Train <- sqrt(mean(fullOLS$residuals^2))
rmseOLS_Train
```

```
## [1] 4.580769
```

```r
# predict test
predOLS <-  predict(fullOLS, housingTest)

# rmse test
rmseOLS_Test <-  sqrt(mean((predOLS - housingTest$MEDV)^2))
rmseOLS_Test
```

```
## [1] 5.263608
```

```r
# ratio
rmseOLS_Test / rmseOLS_Train
```

```
## [1] 1.149067
```

a. Rerun your full (all predictors) linear regression model of MEDV, using the training set and then calculate and report the R2 and RMSE of the residuals. Then predict the y-values in the test set using this model. What is the RMSE for the test set? Is there evidence of overfitting here?

The Multiple R-squared of 0.7571 for the full OLS model indicates that approximately 75.7% of the variance in the dependent variable can be explained by the independent variables in the model. The Adjusted R-squared of 0.7491 considers the number of independent variables in the model and penalizes the R-squared for overfitting.

The Root Mean Square Error (RMSE) of the training set is 4.58, and the RMSE made by the model is off by 4.58 units of the dependent variable. The RMSE for the testing set of 5.26 indicates that the model does not generalize as well to new data, which could be a sign of overfitting.

Create Matrix

```r
# Matrix
x_train <- as.matrix(housingTrain[ , -ncol(housingTrain)])
y_train <- as.matrix(housingTrain[ ,  ncol(housingTrain)])

x_test <- as.matrix(housingTest[ , -ncol(housingTest)])
y_test <- as.matrix(housingTest[ ,  ncol(housingTest)])

# count rows & columns check
cat("X Train Dataset: ", nrow(x_train), "Rows &" , ncol(x_train), "Columns",
    "\nY Train Dataset: ", nrow(y_train), "Rows &" , ncol(y_train), "Column")
```

```
## X Train Dataset:  380 Rows & 12 Columns
## Y Train Dataset:  380 Rows & 1 Column
```

```r
cat("\nX Test Dataset: ", nrow(x_test), "Rows &" , ncol(x_test), "Columns",
    "\nY Test Dataset: ", nrow(y_test), "Rows &" , ncol(y_test), "Column")
```

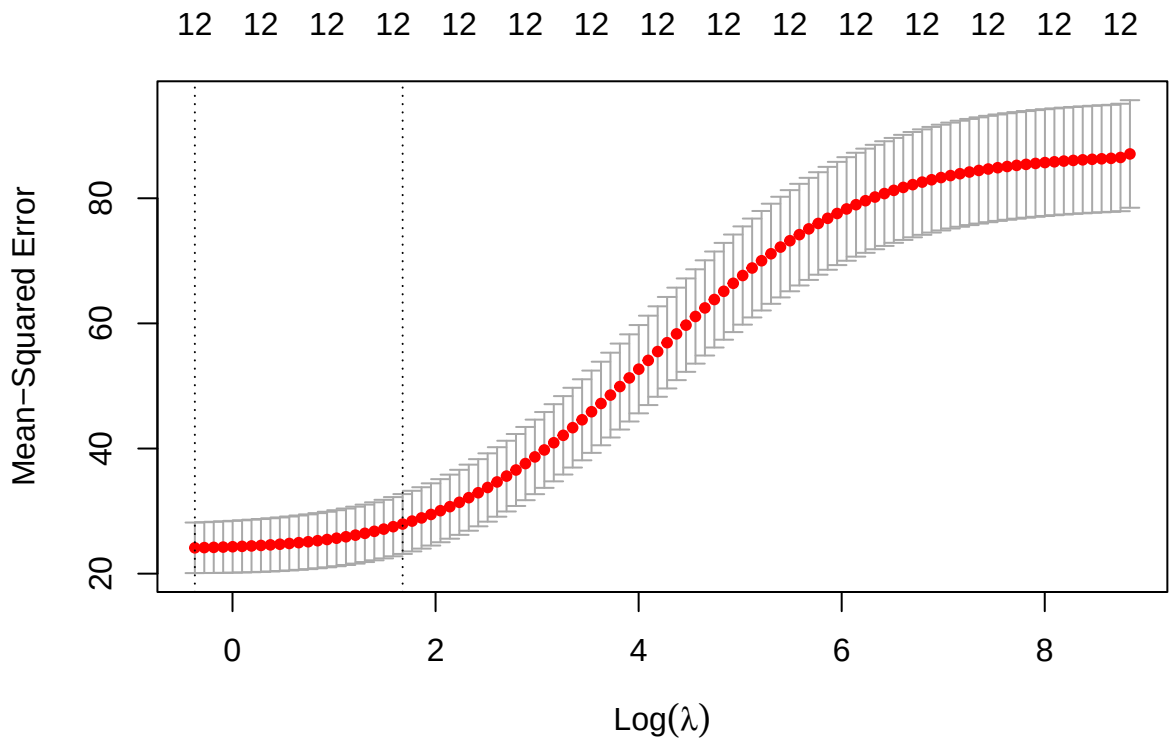
```
##
## X Test Dataset:  126 Rows & 12 Columns
## Y Test Dataset:  126 Rows & 1 Column
```

CV Ridge Model

```
# set seed for reproducibility
set.seed(1776)

# full cv ridge model
fitRidge  <-  cv.glmnet(x_train, y_train, alpha=0, nfolds=10)

# plot cv ridge
plot(fitRidge)
```




```
# display cv ridge
fitRidge
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, nfolds = 10, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  0.690   100   24.13 4.039      12
## 1se  5.342    78   27.93 4.786      12
```

```r
# display lamdba min
coef(fitRidge, s="lambda.min")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  30.899492234
## CRIM         -0.105932093
## ZN            0.033729396
## INDUS        -0.051168174
## CHAS          3.851032791
## NOX         -15.024688568
## RM            4.340742506
## AGE          -0.004513003
## DIS          -1.154365464
## RAD           0.127417787
## TAX          -0.005750184
## PTRATIO      -0.860305488
## LSTAT        -0.427106315
```

```r
# display lamdba 1se
coef(fitRidge, s="lambda.1se")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 22.836638569
## CRIM        -0.070816112
## ZN           0.024553239
## INDUS       -0.082739495
## CHAS         3.355310581
## NOX         -6.264984029
## RM           3.566560023
## AGE         -0.009476738
## DIS         -0.412054940
## RAD          0.007901592
## TAX         -0.003165237
## PTRATIO     -0.616925711
## LSTAT       -0.304184241
```

```r
# extract lambda 1se
lambda1seR <- fitRidge$lambda.1se

# ridge model with lambda 1se to product %Dev is R2
fitR <-  glmnet(x_train, y_train, lambda=lambda1seR, alpha=0)
# display model info
fitR
```

```
##
## Call:  glmnet(x = x_train, y = y_train, alpha = 0, lambda = lambda1seR)
##
##   Df  %Dev Lambda
## 1 12 69.69  5.342
```

```r
# predict train with lambda 1se
predRidgeTrain <- predict(fitRidge, x_train, lambda=lambda1seR)

# predict test with lambda 1se
predRidgeTest <- predict(fitRidge, x_test, lambda=lambda1seR)

# ridge rmse train
rmseRidge_Train <- sqrt(mean((predRidgeTrain - y_train)^2))

# ridge rmse test
rmseRidge_Test <- sqrt(mean((predRidgeTest - y_test)^2))

# rmse for ridge & ols
cat("Train RMSE for Ridge: ", rmseRidge_Train ,
    "\nTest RMSE for Ridge:  ", rmseRidge_Test)
```

```
## Train RMSE for Ridge:  5.116238
## Test RMSE for Ridge:   5.681669
```

```r
# test & train ratio
cat("\nTest & Train ratio: " ,rmseRidge_Test / rmseRidge_Train)
```

```
##
## Test & Train ratio:  1.110517
```

b. Use cross-validated ridge regression on the training set and plot the relationship between the cross-validated error and log-lambda. Then use the model for predicting again the y's in the test set using the "lambda.1se". Report the R2 for the training set (how do you get this?) and the RMSE for both the training and test set. How do these compare to the OLS regression model? Are they improving prediction? If so, how?

The R2 for the training set for cross-validated ridge regression is 0.6969 compared to 0.7571 from full OLS, and R2 for training was found by using `glmnet` with lambda1se value from `cv.glmnet`. The percentage of prediction error according to the RMSE ratio, applying Lasso with cross-validation does provide better predictive power due to the lower RMSE ratio between test and train.

Stepwise Variable Selection

```r
# stepwise variable selection
# Compute the null model
fit_null = lm(MEDV ~ 1, data=housingTrain)

# Compute the full model
fit_full = lm(MEDV ~ ., data=housingTrain)

# stepwise variable selection
fit_stepwise  <-  step(fit_null, scope = list(lower=fit_null, upper=fit_full),
                direction="both", trace=F)

# summary of the model
summary(fit_stepwise)
```

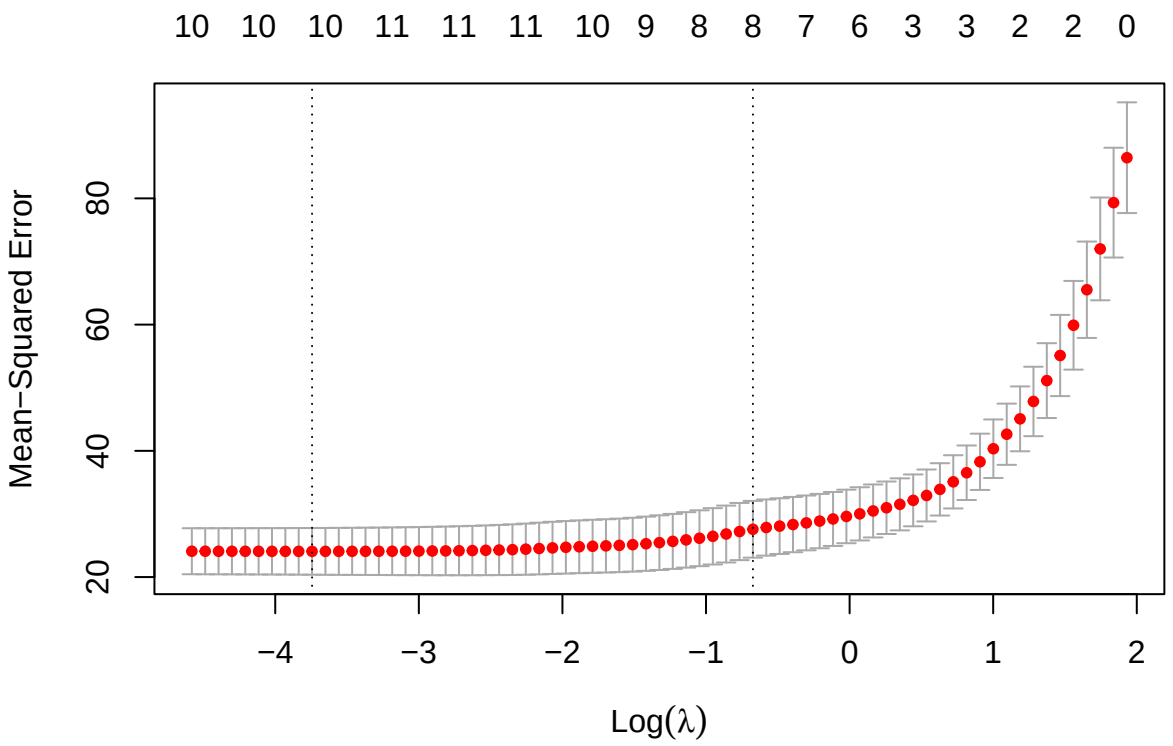
```
##
## Call:
## lm(formula = MEDV ~ LSTAT + RM + PTRATIO + CHAS + NOX + DIS +
##     CRIM + RAD + TAX + ZN, data = housingTrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.9570  -2.6567  -0.6346   1.7359  26.2691
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  39.62441    5.56671   7.118 5.75e-12 ***
## LSTAT        -0.46638    0.05578  -8.362 1.28e-15 ***
## RM            4.22267    0.47093   8.967  < 2e-16 ***
## PTRATIO      -0.98090    0.14815  -6.621 1.26e-10 ***
## CHAS          3.60370    0.93438   3.857 0.000136 ***
## NOX         -21.87056    4.07231  -5.371 1.39e-07 ***
## DIS          -1.53304    0.21260  -7.211 3.17e-12 ***
## CRIM         -0.12997    0.03402  -3.821 0.000156 ***
## RAD           0.26456    0.06954   3.804 0.000166 ***
## TAX          -0.01117    0.00367  -3.044 0.002502 **
## ZN            0.04331    0.01538   2.815 0.005133 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.649 on 369 degrees of freedom
## Multiple R-squared:  0.7571, Adjusted R-squared:  0.7505
## F-statistic:   115 on 10 and 369 DF,  p-value: < 2.2e-16
```

Create CV Lasso

```r
# set seed for reproducibility
set.seed(1776)

# full cv lasso model
fitLasso <- cv.glmnet(x_train, y_train, alpha=1, nfolds=10)

# plot cv lasso
plot(fitLasso)
```




```
# extract lambda 1se
lambda1seL <- fitLasso$lambda.1se

# display coefficients
coef(fitLasso, s=lambda1seL)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept) 15.1825848298
## CRIM        -0.0387765851
## ZN           .
## INDUS        .
## CHAS         2.7824120165
## NOX         -3.5313238674
## RM           4.6230660026
## AGE          .
## DIS         -0.0252707663
## RAD          .
## TAX         -0.0002268827
## PTRATIO     -0.7571344065
## LSTAT       -0.4521358555
```

```
# lasso model with lambda 1se to product %Dev is R2
fitL <-  glmnet(x_train, y_train, lambda=lambda1seL, alpha=1)
# display model info
fitL
```

```
##
## Call:  glmnet(x = x_train, y = y_train, alpha = 1, lambda = lambda1seL)
##
##   Df  %Dev Lambda
## 1  8 70.91   0.51
```

```
# predict train with lambda 1se
predLassoTrain <- predict(fitLasso, x_train, lambda=lambda1seL)

# predict test with lambda 1se
predLassoTest <- predict(fitLasso, x_test, lambda=lambda1seL)

# lasso rmse train
rmseLasso_Train <- sqrt(mean((predLassoTrain - y_train)^2))

# lasso rmse test
rmseLasso_Test <- sqrt(mean((predLassoTest - y_test)^2))

# rmse for ridge & ols
cat("Train RMSE for Lasso: ", rmseLasso_Train ,
    "\nTest RMSE for Lasso:  ", rmseLasso_Test)
```

```
## Train RMSE for Lasso:  5.014091
## Test RMSE for Lasso:   5.488786
```

```
# test & train ratio
cat("\nLasso Test & Train ratio: " ,rmseLasso_Test / rmseLasso_Train)
```

```
##
## Lasso Test & Train ratio:  1.094672
```

c. Do the same as in b) for Lasso using "lambda.1se". How do the results compare to Ridge and OLS regression? Also, for Lasso, evaluate how the number of variables changes with lambda. How many variables are selected at lambda.1se? Finally, how do the variables selected, and their betas computed compare with the model you got last week with stepwise regression?

It appears that the Lasso model with a highly regularized `lambda.1se` value performed best according to the lowest test-train ratio. The Lasso model also selected fewer variables than the stepwise variable selection method.

The Lasso model is known for its ability to perform variable selection by shrinking the coefficients of less important variables towards zero as the regularization parameter lambda increases. This can help to eliminate independent variables that are not contributing much to the predictive power of the model.

Finally, it seems that the Lasso model with a highly regularized lambda.1se value provided better predictive performance than the stepwise variable selection method, while also selecting fewer variables and shrinking the beta coefficients towards zero. Furthermore, it is clear that Lasso betas are closer to zero, but there were few variables which did increase.

d. Revisit the cross-validated error graphs for both Ridge and Lasso, and evaluate how well regularization is working for this set? Is there a strong indication that cross-validated error can be reduced by regularization in this case? What do you think this mean for overfitting in the original model?

Based on the information provided, there was a negligible difference in the R2 between the Lasso and Ridge regression models. However, in terms of the RMSE ratio, the Lasso model showed approximately a four percent improvement over the Ridge model, indicating that the Lasso model provides better predictive power than Ridge.

Comparing the performance of the OLS model to the regularized models, regularization provides more robust performance in this dataset. Regularization can prevent overfitting, which occurs when the model is overly complex and fits the training data too closely. By adding a penalty term to the loss function, regularization can simplify the model and reduce the impact of noisy or irrelevant predictors, leading to better generalization performance.

In summary, the Lasso model provides better predictive power than the Ridge model in this dataset. In addition, regularization offers a more robust performance compared to the OLS model by helping to prevent overfitting in this dataset.

Question 5

5. (20 points) The data in the files insurTest.csv & insurTrain.csv are collected from 47 zip-code areas in the Illinois area. There are 8 columns in the data file but not all are relevant here. The response variable of interest is the number of new home insurance policies (NEWPOL) (minus canceled policies) per 100 housing units. The predictor variables are the percent minority population living in the area (PCTMINOR), the number of fires per 1000 housing units (FIRES), the number of thefts per 1000 in population (THEFTS), the percent of housing units built before 1940 (PCTOLD), and the median income (INCOME).

```
# import insurance training & test
insuranceTest  <- read.csv("insurTest.csv")
insuranceTrain <- read.csv("insurTrain.csv")

# view datasets
glimpse(insuranceTrain)
```

```
## Rows: 23
## Columns: 8
## $ zipcode <int> 60655, 60632, 60628, 60616, 60617, 60625, 60636, 60644, 60611,~
## $ pctmin  <dbl> 1.0, 4.4, 35.1, 62.3, 36.4, 7.1, 48.8, 59.8, 4.9, 1.8, 17.3, 9~
## $ fires   <dbl> 4.8, 5.6, 15.6, 12.2, 10.8, 6.9, 28.6, 16.5, 11.0, 5.4, 7.7, 2~
## $ thefts  <int> 19, 23, 28, 46, 34, 18, 27, 40, 75, 27, 37, 31, 29, 39, 34, 14~
## $ pctold  <dbl> 15.2, 71.5, 57.8, 48.0, 58.0, 78.5, 78.1, 72.7, 42.6, 85.1, 66~
## $ newpol  <dbl> 13.0, 8.0, 7.5, 3.4, 7.8, 6.9, 4.0, 2.7, 7.9, 8.9, 5.7, 0.5, 2~
## $ fairpol <dbl> 0.0, 0.3, 1.0, 0.6, 0.9, 0.0, 1.4, 0.8, 0.0, 0.0, 0.5, 0.9, 1.~
## $ income  <int> 13323, 11230, 11260, 8212, 11156, 11104, 9742, 9784, 21480, 11~
```

```
glimpse(insuranceTest)
```

```
## Rows: 24
## Columns: 8
## $ zipcode <int> 60626, 60640, 60613, 60614, 60610, 60618, 60647, 60646, 60656,~
## $ pctmin  <dbl> 10.0, 22.2, 19.6, 24.5, 54.0, 5.3, 21.5, 1.0, 1.7, 1.6, 13.4, ~
## $ fires   <dbl> 6.2, 9.5, 10.5, 8.6, 34.1, 7.3, 15.1, 5.7, 2.0, 2.5, 15.1, 18.~
## $ thefts  <int> 29, 44, 36, 53, 68, 31, 25, 11, 11, 22, 30, 32, 41, 147, 4, 15~
## $ pctold  <dbl> 60.4, 76.5, 73.5, 81.4, 52.6, 90.1, 89.8, 27.9, 7.7, 63.8, 89.~
## $ newpol  <dbl> 5.3, 3.1, 4.8, 5.9, 4.0, 7.6, 3.1, 12.1, 10.9, 10.7, 5.2, 1.2,~
## $ fairpol <dbl> 0.0, 0.1, 1.2, 0.7, 0.3, 0.4, 1.1, 0.0, 0.0, 0.0, 0.8, 1.8, 1.~
## $ income  <int> 11744, 9323, 9948, 9730, 8231, 10694, 9631, 16250, 13686, 1240~
```

Multiple Regression

```
# remove unwanted independent variables
insuranceTrain <- select(insuranceTrain, c(-zipcode, -fairpol))
insuranceTest  <- select(insuranceTest, c(-zipcode, -fairpol))

# fit train dataset
fit_newpol <- lm(newpol ~ . , data = insuranceTrain)

# summary of tran dataset
summary(fit_newpol)
```

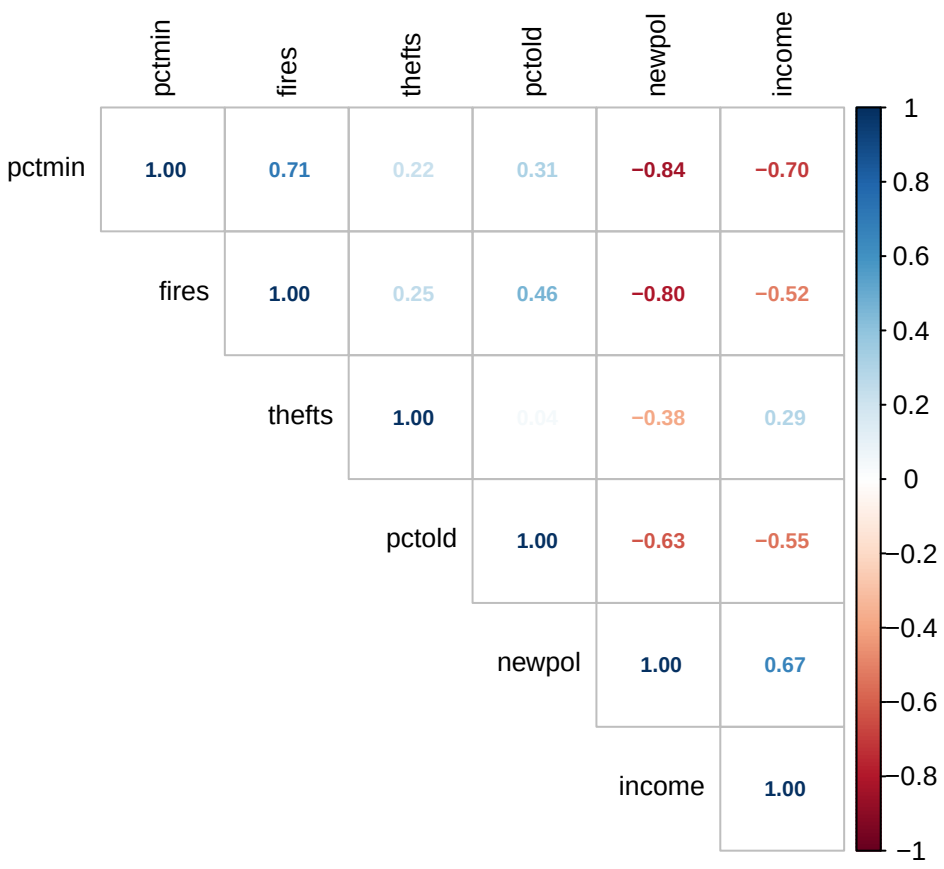
```
##
## Call:
## lm(formula = newpol ~ ., data = insuranceTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0116 -0.8380 -0.2138  0.8646  2.2625
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.1561459  3.1181250   4.219 0.000577 ***
## 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
rmseOLS_Train <- sqrt(mean(fit_newpol$residuals^2))
rmseOLS_Train
```

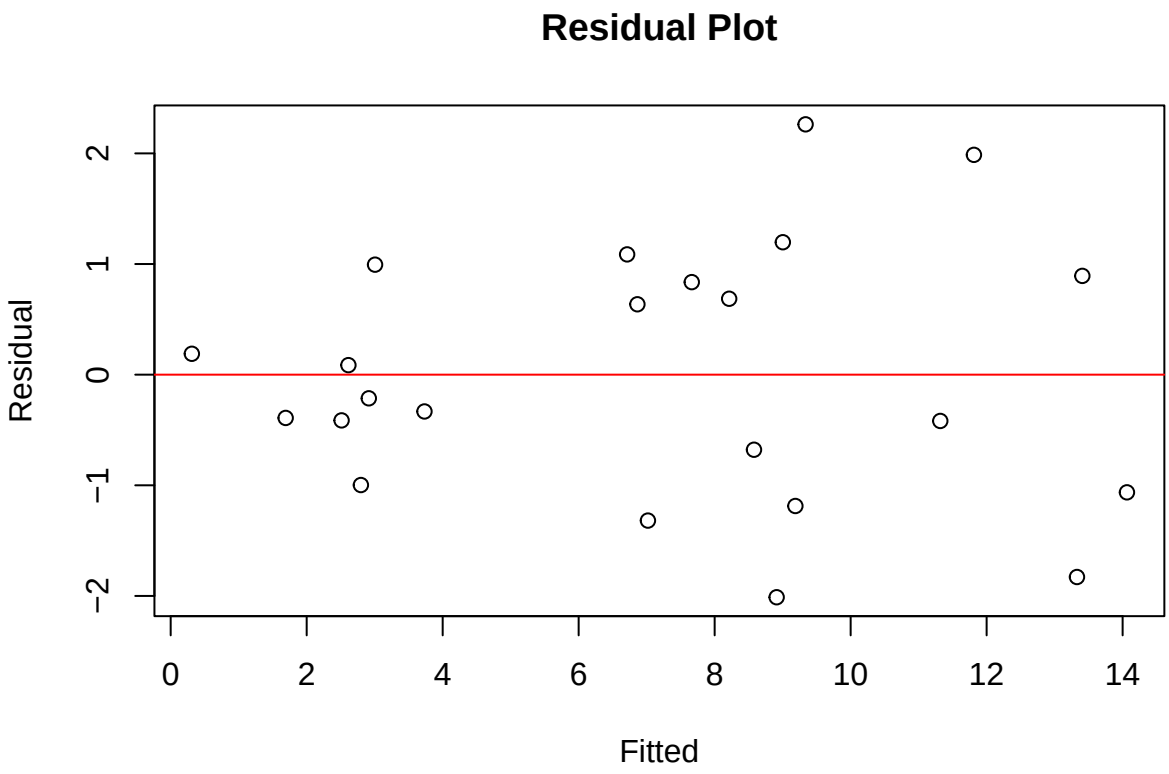
```
## [1] 1.120282
```

```
# correlation plot
corr <- cor(insuranceTrain)

# plot correlation
corrplot(corr, method = 'number', addCoef.col = 'green', tl.col = "black",
        type = 'upper', number.cex = 0.7, tl.cex = 0.8,)
```




```
# residual vs fitted
plot(fit_newpol$fitted, fit_newpol$residuals, main = "Residual Plot",
     ylab = "Residual", xlab = "Fitted")
abline(a=0, b=0, col="red")
```



a. Run a multiple regression of NEWPOL on the variables: PCT-MINOR FIRES THEFTS PCTOLD INCOME NEWPOL.

b. What is the overall fit (F-score, R^2) and what is the RMSE on the training set?

The F-statistic is 43.32 on 5 and 17 DF, and the p-value is 4.387e-09, which is very small, stating it is significant and the model is a good fit for the data. The Multiple R-squared is 0.9272, and the Adjusted R-squared is 0.9058 on the training dataset. The RMSE for the training dataset is 1.12, and the predictions made by the model for the training data are off by 1.12 units from the actual values.

ii. Which predictors have coefficients significantly different from zero at the .05 level?

A statistical analysis of the data found that four predictors (pctmin, fires, thefts, and pctold) are statistically significant at the 0.05 level or lower. Specifically, two predictors (pctmin and fires) are significant at the 0.05 level, and two predictors (thefts and pctold) are significant at the more stringent 0.01 level.

iii. Do any of the predictors have signs that are different than suggested by their simple correlations? If so, explain what may be happening.

In summary, a variable can have a correlation coefficient that suggests a linear relationship with the response variable (`newpol`). Still, the relationship may not be statistically significant due to factors such as small sample size, high variability in the data, or the presence of other variables.

iv. Examine a plot of residuals versus predicted values. Do you see any problems?

The residual plot shows randomness, which is generally a good indication of a well-fitting regression model able to explain the variation in the dependent variable from the independent variables, but the presence of possible outliers in the residual plot should be investigated further. Outliers can significantly impact the regression model, including distorting the estimated coefficients and affecting the model's goodness of fit.

```
# predict test
predOLS <-  predict(fit_newpol, insuranceTest)

# rmse test
rmseOLS_Test <-  sqrt(mean((predOLS - insuranceTest$newpol)^2))

# display rmse test
cat("\nOLS RMSE Test: " ,rmseOLS_Test)
```

```
##
## OLS RMSE Test:  4.260169
```

```
# OLS test & train ratio
cat("\nOLS Test & Train ratio: " ,rmseOLS_Test / rmseOLS_Train)
```

```
##
## OLS Test & Train ratio:  3.802763
```

b. Use the model from a) to predict the test set. What is the RMSE here? Is there evidence of overfitting?

The model's RMSE (Root Mean Square Error) is 4.26, which means that, on average, the model's predicted values differ from the actual values by 4.26 units.

The RMSE test and train ratio is roughly 280 percent, which indicates overfitting. This means that the model has learned the noise in the training data rather than the underlying patterns or relationships and will predict poorly on new or unseen data.

Convert to Matrix

```
# Matrix
x_insuranceTrain <- as.matrix(select(insuranceTrain, c(-newpol)))
y_insuranceTrain <- as.matrix(select(insuranceTrain, c(newpol)))

x_insuranceTest <- as.matrix(select(insuranceTest, c(-newpol)))
y_insuranceTest <- as.matrix(select(insuranceTest, c(newpol)))

# count rows & columns check
cat("X Train Dataset: ", nrow(x_insuranceTrain), "Rows &" ,
    ncol(x_insuranceTrain), "Columns", "\nY Train Dataset: ",
    nrow(y_insuranceTrain), "Rows &" , ncol(y_insuranceTrain), "Column")
```

```
## X Train Dataset:  23 Rows & 5 Columns
## Y Train Dataset:  23 Rows & 1 Column
```

```
cat("\nX Test Dataset: ", nrow(x_insuranceTest), "Rows &" ,
    ncol(x_insuranceTest), "Columns", "\nY Test Dataset: ",
    nrow(y_insuranceTest), "Rows &" , ncol(y_insuranceTest), "Column")
```

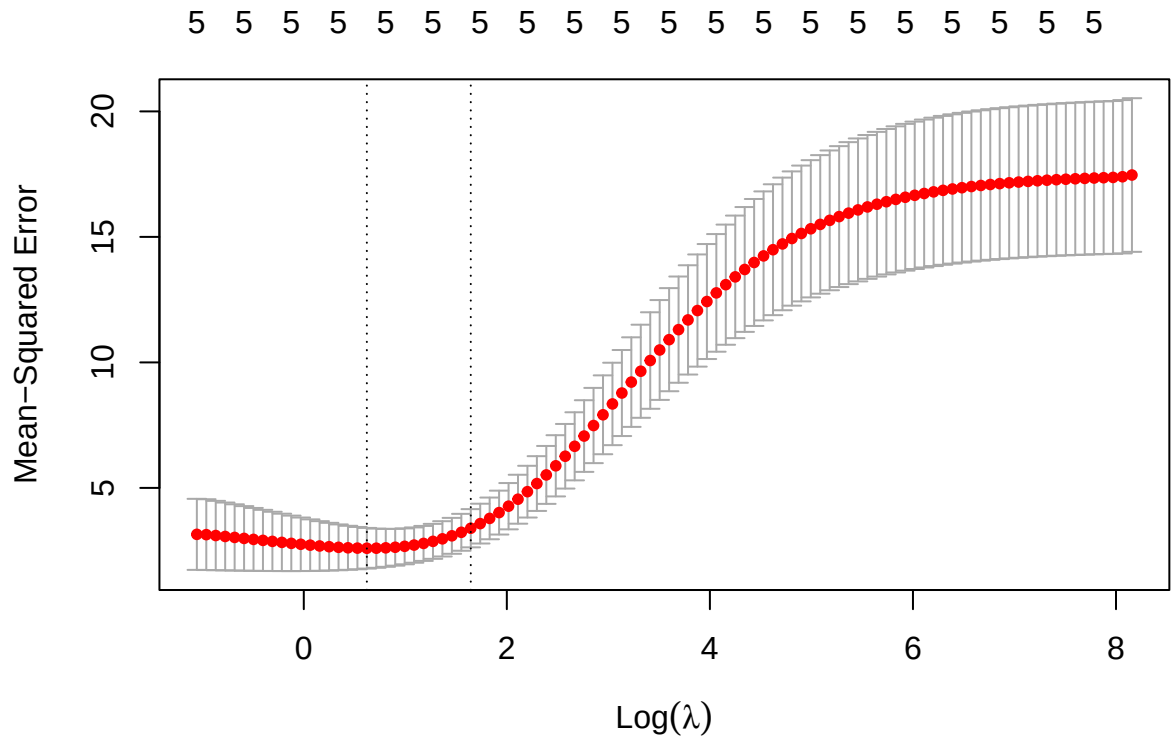
```
##
## X Test Dataset:  24 Rows & 5 Columns
## Y Test Dataset:  24 Rows & 1 Column
```

Insurance Ridge

```
# set seed for reproducibility
set.seed(1776)

# full cv ridge model
fitRidgeInsurance <- cv.glmnet(x_insuranceTrain, y_insuranceTrain,
                               alpha=0, nfolds=7)

# plot cv ridge
plot(fitRidgeInsurance)
```




```
# display cv ridge
fitRidgeInsurance
```

```
##
## Call:  cv.glmnet(x = x_insuranceTrain, y = y_insuranceTrain, nfolds = 7,      alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
```

```
## min  1.861    82   2.595 0.8103       5
## 1se  5.180    71   3.393 0.7610       5
```

```
# display lamdba min
coef(fitRidgeInsurance, s="lambda.min")
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept) 11.9091933523
## pctmin      -0.0401925485
## fires       -0.1240827743
## thefts      -0.0684219034
## pctold      -0.0501207030
## income       0.0002686264
```

```
# display lamdba 1se
coef(fitRidgeInsurance, s="lambda.1se")
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 10.623326354
## pctmin      -0.031092938
## fires       -0.104793414
## thefts      -0.046240147
## pctold      -0.038594364
## income       0.000217181
```

```
# extract lambda 1se
lambda1seRInsurance <- fitRidgeInsurance$lambda.1se

# ridge model with lambda 1se to product %Dev is R2
fitRInsurance <-  glmnet(x_insuranceTrain, y_insuranceTrain,
                         lambda=lambda1seRInsurance, alpha=0)
# display model info
fitRInsurance
```

```
##
## Call:  glmnet(x = x_insuranceTrain, y = y_insuranceTrain, alpha = 0,      lambda = lambda1seRInsuran
##
##   Df  %Dev Lambda
## 1  5 81.71   5.18
```

```
# predict train with lambda 1se
predRidgeTrainInsurance <- predict(fitRidgeInsurance, x_insuranceTrain,
                           lambda=lambda1seRInsurance)

# predict test with lambda 1se
predRidgeTestInsurance <- predict(fitRidgeInsurance, x_insuranceTest,
                          lambda=lambda1seRInsurance)

# ridge rmse train
rmseRidge_Train_Insurance <- sqrt(mean((predRidgeTrainInsurance -
```

```
                                          y_insuranceTrain)^2))

# ridge rmse test
rmseRidge_Test_Insurance <- sqrt(mean((predRidgeTestInsurance -
                                         y_insuranceTest)^2))

# rmse for ridge & ols
cat("Train RMSE for Ridge: ", rmseRidge_Train_Insurance ,
    "\nTest RMSE for Ridge:  ", rmseRidge_Test_Insurance)
```

```
## Train RMSE for Ridge:  1.775932
## Test RMSE for Ridge:   2.594884
```

```
# test & train ratio
cat("\nRidge Test & Train ratio: " , rmseRidge_Test_Insurance /
      rmseRidge_Train_Insurance)
```

```
##
## Ridge Test & Train ratio:  1.461139
```

c. Run a ridge regression on the training set, produce the lambda plot, and report the RMSE for both the training and test sets using lambda.1se. From the lambda plot, how well is regularization working here? Look at the shape of the plot for a significant dip before it rises.

The Ridge regularization used in the model has helped to reduce overfitting, as evidenced by the drop in the difference between the training and test RMSEs. However, a test and train ratio of 46.11% still indicates overfitting.

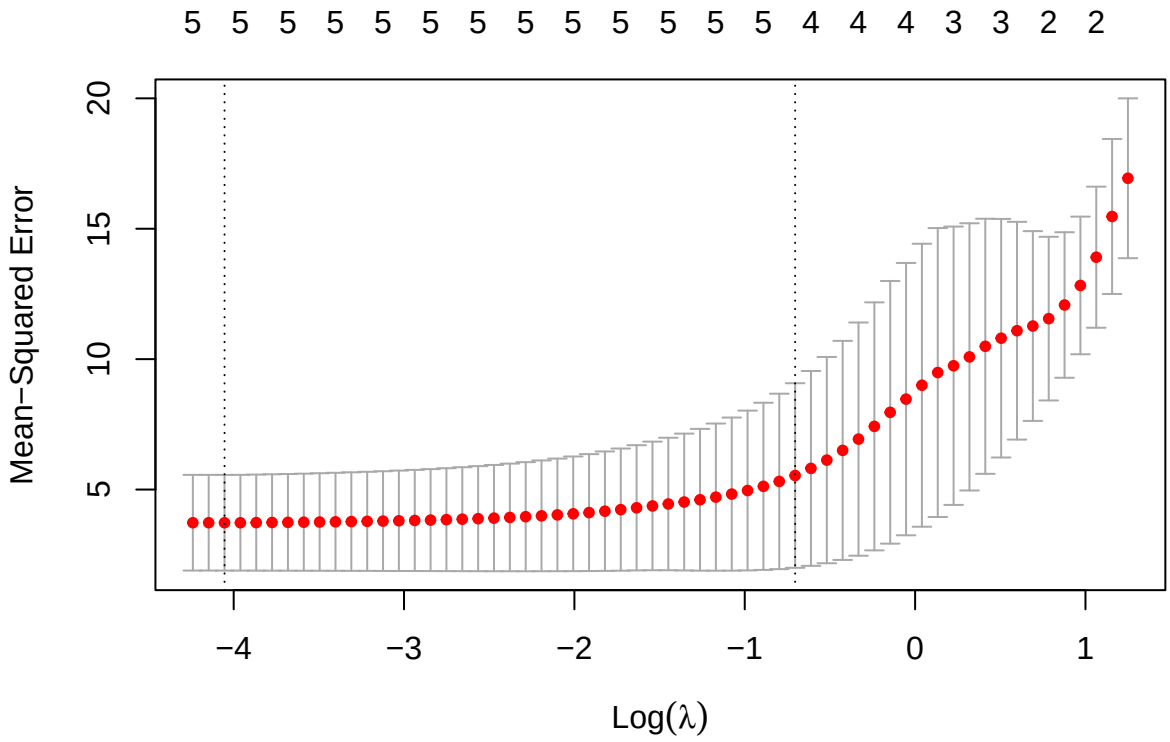
It's essential to remember that regularization is not a silver bullet for overfitting, and other strategies may need to be employed, such as feature selection and feature engineering or increasing the size of the training set.

Additionally, it may be worth exploring a range of lambda values to see if a different choice yields better results by utilizing the shape of the plot due to a slightly lower error between the min and 1se lambda values.

```
# set seed for reproducibility
set.seed(1776)

# full cv lasso model
fitLassoInsurance <- cv.glmnet(x_insuranceTrain, y_insuranceTrain,
                               alpha=1, nfolds=7)

# plot cv ridge
plot(fitLassoInsurance)
```




```
# display cv lasso
fitLassoInsurance
```

```
##
## Call:  cv.glmnet(x = x_insuranceTrain, y = y_insuranceTrain, nfolds = 7,      alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.0174    58   3.727 1.834       5
## 1se 0.4944    22   5.538 3.539       4
```

```
# display lamdba min
coef(fitLassoInsurance, s="lambda.min")
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept) 13.2646175000
## pctmin      -0.0513902677
## fires       -0.1144543964
## thefts      -0.0956365105
## pctold      -0.0630093042
## income       0.0003108343
```

```r
# display lamdba 1se
coef(fitLassoInsurance, s="lambda.1se")
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 15.05718340
## pctmin      -0.06586481
## fires       -0.10524689
## thefts      -0.03169408
## pctold      -0.06145707
## income       .
```

```r
# extract lambda 1se
lambda1seLInsurance <- fitLassoInsurance$lambda.1se

# lasso model with lambda 1se to product %Dev is R2
fitLInsurance <-  glmnet(x_insuranceTrain, y_insuranceTrain,
                         lambda=lambda1seLInsurance, alpha=1)
# display model info
fitLInsurance
```

```
##
## Call:  glmnet(x = x_insuranceTrain, y = y_insuranceTrain, alpha = 1,      lambda = lambda1seLInsuran
##
##   Df  %Dev Lambda
## 1  4 88.59 0.4944
```

```r
# predict train with lambda 1se
predLassoTrainInsurance <- predict(fitLassoInsurance, x_insuranceTrain,
                           lambda=lambda1seLInsurance)

# predict test with lambda 1se
predLassoTestInsurance <- predict(fitLassoInsurance, x_insuranceTest,
                          lambda=lambda1seLInsurance)

# lasso rmse train
rmseLasso_Train_Insurance <- sqrt(mean((predLassoTrainInsurance -
                                        y_insuranceTrain)^2))

# lasso rmse test
rmseLasso_Test_Insurance <- sqrt(mean((predLassoTestInsurance -
                                       y_insuranceTest)^2))

# rmse for lasso & ols
cat("Train RMSE for Lasso: ", rmseLasso_Train_Insurance ,
    "\nTest RMSE for Lasso:  ", rmseLasso_Test_Insurance)
```

```
## Train RMSE for Lasso:  1.403101
## Test RMSE for Lasso:   2.541902
```

```r
# test & train ratio
cat("\nLasso Test & Train ratio: " , rmseLasso_Test_Insurance /
      rmseLasso_Train_Insurance)
```

```
##
## Lasso Test & Train ratio:  1.811631
```

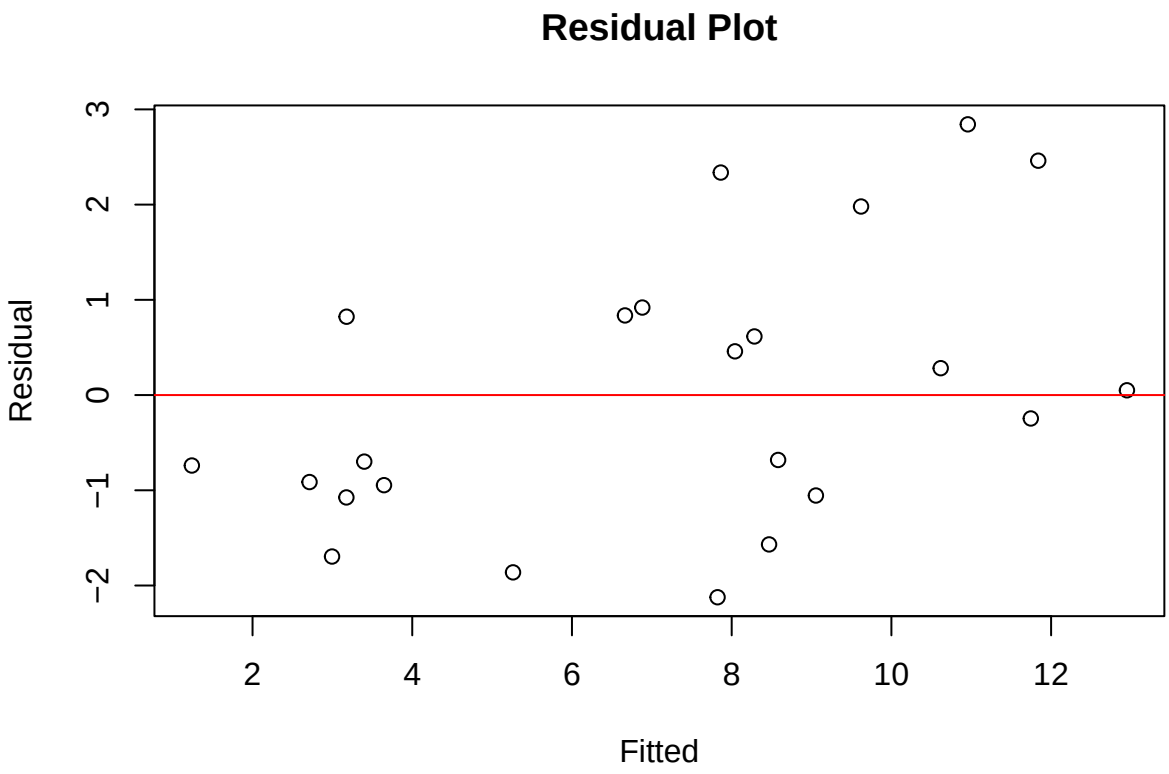
d. Run a lasso regression on the training set, produce the lambda plot, and report the RMSE for both the training and test sets using lambda.1se. Compare them with those you got in c). Are they closer together, more stable, or have they worsened overall?

A Lasso regression model produced smaller RMSE values on both the training (1.40) and testing (2.54) sets than the Ridge regularization model with training (1.78) and testing (2.59) sets. However, the testing and training ratio is roughly 35% greater in the Lasso regularization, and the ratio between the test & training set is 81.16% which suggests that the Lasso model is overfitting to the training data. In contrast, the Ridge regularization model has RMSE values that are closer together, which indicates that it may be more effective at reducing overfitting and improving the model's generalization performance; therefore, Ridge is more stable.

Lasso Residual Plot

```r
# residual vs fitted training
plot(predLassoTrainInsurance,  y_insuranceTrain - predLassoTrainInsurance,
     main = "Residual Plot",
     ylab = "Residual", xlab = "Fitted")

# add abline on residual plot
abline(a=0, b=0, col="red")
```



e. Plot the residuals vs the fitted (predicted values), you can do this with plot(fitted, residuals) since you've calculated both in order to get the RMSE (remember, residuals are the difference between actual and predicted). Does the lasso add any significant bias into the model? Note that it may or may not show up as a slope difference. It could also appear as a mean residual different from 0 as well.

The Lasso regularization methods introduce bias into the model to reduce variances because the regularization result in residuals plot clearly shows that on the left side of the residual plot, there are more points below zero. On the right side, there are more points above zero. Finally, the mean of the residual is larger in magnitude than the OLS, which indicates that Lasso is introducing bias in the model, which is displayed in the broader spread of residuals from the mean from the residual plot.

```
# set seed for reproducibility
set.seed(1776)

# alpha vector
alpha <- c(0.25, 0.50, 0.75)

# create data frame with 0 rows and 6 columns
df <- data.frame(alpha.value=numeric(),
                 lambda.1se=numeric(),
                 R2=numeric(),
                 RMSE.test=numeric(),
                 RMSE.train=numeric(),
                 test.train.ratio=numeric())
```

```
# for loop alpha values
for (i in alpha) {
  #
  elsticNet <- cv.glmnet(x_insuranceTrain, y_insuranceTrain,
                              alpha=i, nfolds=7)

  # lambda.1se
  lambda.1se <- elsticNet$lambda.1se

  # fit glmnet
  fit <-  glmnet(x_insuranceTrain, y_insuranceTrain,
                         lambda=lambda.1se, alpha=i)

  # R2 value
  r2 <- fit$dev.ratio

  # predict train with lambda 1se
  predTrain <- predict(elsticNet, x_insuranceTrain,
                          lambda=lambda.1se)

  # predict test with lambda 1se
  predTest <- predict(elsticNet, x_insuranceTest,
                         lambda=lambda.1se)

  # elastic net rmse train
  rmseTrain <- sqrt(mean((predTrain - y_insuranceTrain)^2))

  # elastic net rmse test
  rmseTest <- sqrt(mean((predTest - y_insuranceTest)^2))

  # test & train ratio
  ratio <- rmseTest / rmseTrain

  # create vector
  output <-  c(i, lambda.1se, r2, rmseTest, rmseTrain, ratio)

  # add row to dataframe
  df[nrow(df) + 1,] <- output

}

# print test.train.ratio ascending
df[order(df$test.train.ratio),]
```

```
##   alpha.value lambda.1se        R2 RMSE.test RMSE.train test.train.ratio
## 2        0.50  1.7280182 0.7893123  2.145784   1.906236         1.125665
## 1        0.25  2.8692622 0.7801561  2.285533   1.947278         1.173707
## 3        0.75  0.4543773 0.9054303  2.993267   1.277122         2.343759
```

f. Use Elastic Net regression and determine if there is an alpha between 0 and 1 that will give a better result with this test and training set. Try alphas of .25, .5 and .75. Does it appear that there might be an alpha that does better than either ridge or lasso?

A significantly lower RMSE ratio for Elastic Net regression with an alpha value of 0.50 compared to Ridge and Lasso suggests that Elastic Net with alpha=0.5 performs substantially better than Ridge and Lasso for the given dataset. Finally, RMSE values for testing & training are closer together in the Elastic Net and a better performing model.

g. Is there a practical reason to try and mix lasso and ridge here? Explain your answer.

Yes, there are practical reasons to use Elastic Net regression because Elastic Net regression is a hybrid approach that combines the strengths of both Lasso and Ridge regression. For example, the Elastic Net combines the Lasso and Ridge penalties and allows for a mixture of variable selection and variable shrinkage, depending on the values of the regularization parameters. In addition, by balancing the Lasso and Ridge penalties, Elastic Net can help to overcome some of the limitations and potential downsides of each method and select a stable subset of predictors while avoiding overfitting per the insurance dataset.

Question 6

6. (20 points) Read and review the posted paper "Adding bias to reduce variance in psychological results." Most of the mathematics here will be similar to what we've gone over in class but pay particular attention to Section 3: Examples. Answer the following questions, in detail. You should be able to write a detailed paragraph about each of at least three or four complete sentences with

a. What is the size of the dataset relative to the number of independent variables?

The dataset contains 395 rows with 39 variables due to four categorical predictors, but students captured the actual independent variables were 30.

b. Is there evidence of overfitting in their dataset?

When the MSPE values are consistently high, such as values roughly eight and nine as shown on the report, and are obtained from multiple iterations of the model using different samples with higher MSPE means poorer prediction accuracy, a sign of overfitting. In addition, the small sample size relative to the feature size can exacerbate the problem of overfitting. Overfitting occurs when the model is too complex relative to the amount of data available, causing it to fit the noise or random fluctuations in the data instead of the underlying patterns.

c. How are regularized regression techniques being used in their examples?

In this paper all three regularized regression techniques Ridge & Lasso & Elastic Net were used. In Lasso and Elastic Net regularization, a penalty term lambda.min & lamdba.1se was added to the linear regression model.

The two values, lambda.min and lambda.1se, are selected based on the results of cross-validation, which is used to find the optimal value of lambda that results in the best predictive performance of the model.

Lambda.min is the value of lambda that results in the smallest cross-validation error, while lambda.1se is the largest value of lambda that is within one standard error of the minimum cross-validation error.

d. How do the results of regularized regression differ from the OLS model?

The results of regularized regression overall performed better than the OLS model, especially Elastic Net was the best performing model for both lambda.min & lambda.1se as the strength of the penalty led to more coefficients being shrunk to zero.

e. How do they evaluate the performance of each of the regularized regression techniques?

The dataset was split into training and testing sets using multiple random samples. This procedure was repeated for 100 different random splits of the data into training and testing datasets. Finally, 100 splits for each of the ten different methods are shown in Fig 4.

f. Are there any issues that you can identify with the way that they are evaluating this performance?

The paper suggests that regularization techniques outperform the stepwise variable selection method, another common variable selection technique in linear regression models. On the other hand, regularization techniques can select the most informative variables while avoiding overfitting and unstable estimates of regression coefficients.

The paper also suggests Elastic Net, which combines the strengths of both Ridge and Lasso, created a better-performing and more stable model in this dataset, including outperforming traditional OLS regression.

In summary, these regularization techniques can outperform traditional methods such as OLS and stepwise variable selection. The choice of regularization technique depends on the nature of the data.

7. (10 Points) Post a comment to the "Lecture 1 & 2" discussion forum regarding a topic from lectures 1 & 2 and homeworks 1 & 2. In your post, describe what you found easiest to understand and also what you found most difficult. Think about topics that you found most interesting, topics that you would like to hear more about, or topics that you found confusing and you would like more clarification. Please also take the time to respond to your classmates' questions and comments (respectfully of course).

Refer to the DSC 424 D2L