FINAL_ASSIGNMENT

M.SOWMIA, ROLL NO:MA19M020

23/05/2020

QUESTION:1

PART:1-Load the dataset Seatbelts from datasets package in R.

```
#we start by loading the Seatbelts dataset through the following command
Seatbelts <- data.frame(Seatbelts)
#Use the str() function to view a compact display of the dataframe object
str(Seatbelts)
```

```
## 'data.frame':    192 obs. of  8 variables:
##  $ DriversKilled: num  107 97 102 87 119 106 110 106 107 134 ...
##  $ drivers      : num  1687 1508 1507 1385 1632 ...
##  $ front        : num  867 825 806 814 991 ...
##  $ rear         : num  269 265 319 407 454 427 522 536 405 437 ...
##  $ kms          : num  9059 7685 9963 10955 11823 ...
##  $ PetrolPrice  : num  0.103 0.102 0.102 0.101 0.101 ...
##  $ VanKilled    : num  12 6 12 8 10 13 11 6 10 16 ...
##  $ law          : num  0 0 0 0 0 0 0 0 0 0 ...
```

Thus we see that there are 192 observations of the following 8 variables
1)driversKilled: # ofCar drivers killed.
2)drivers: Same as UKDriverDeaths.
3)front:# of passengers in front-seat that are killed or seriously injured.
4)rear:# of passengers in rear-seat that are killed or seriously injured.
5)kms: Distance driven.
6)PetrolPrice: Petrol price.
7)VanKilled: # of Number of van drivers.
8)Law: Was the law in effect that month/not? This is a binary variable
To have a better understanding, we look at the first 15 records of the dataset:

```
head(Seatbelts,n=15)
```

```
##    DriversKilled drivers front rear   kms PetrolPrice VanKilled law
## 1            107    1687   867  269  9059   0.1029718        12   0
## 2             97    1508   825  265  7685   0.1023630         6   0
## 3            102    1507   806  319  9963   0.1020625        12   0
## 4             87    1385   814  407 10955   0.1008733         8   0
## 5            119    1632   991  454 11823   0.1010197        10   0
## 6            106    1511   945  427 12391   0.1005812        13   0
## 7            110    1559  1004  522 13460   0.1037740        11   0
## 8            106    1630  1091  536 14055   0.1040764         6   0
## 9            107    1579   958  405 12106   0.1037740        10   0
## 10           134    1653   850  437 11372   0.1030264        16   0
## 11           147    2152  1109  434  9834   0.1027301        13   0
```

```
## 12            180    2148  1113  437   9267   0.1019972        14   0
## 13            125    1752   925  316   9130   0.1012746        14   0
## 14            134    1765   903  311   8933   0.1007040         6   0
## 15            110    1717  1006  351  11000   0.1001396         8   0
```

We now see a summary of each sub-set below

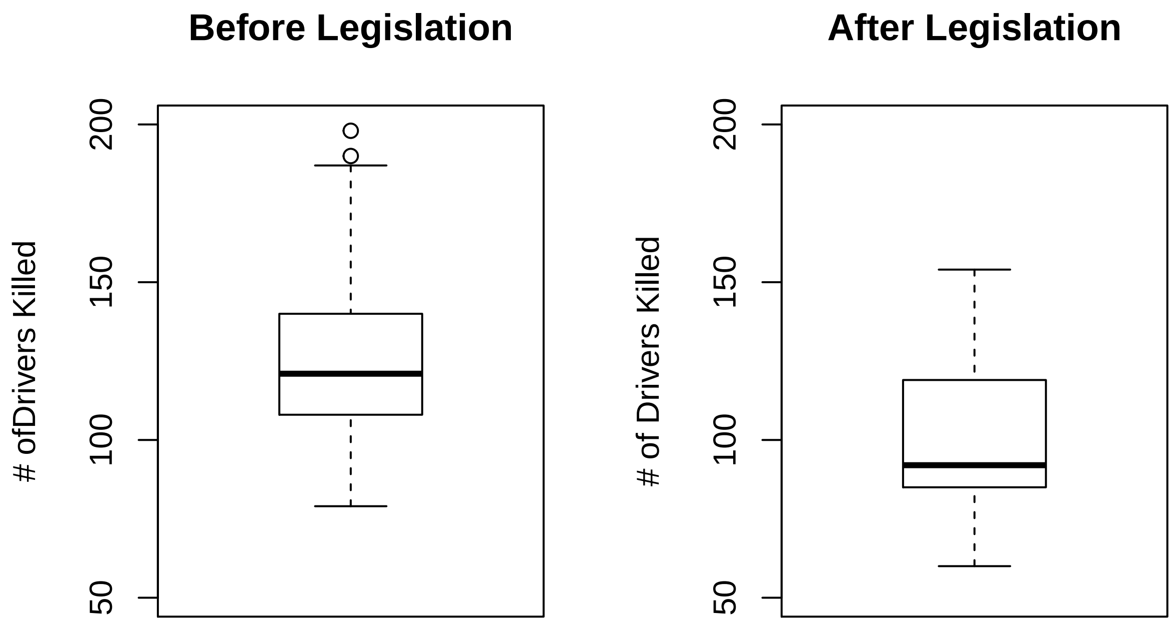
```r
summary(Seatbelts)
```

```
##  DriversKilled      drivers          front             rear      
##  Min.   : 60.0   Min.   :1057   Min.   : 426.0   Min.   :224.0  
##  1st Qu.:104.8   1st Qu.:1462   1st Qu.: 715.5   1st Qu.:344.8  
##  Median :118.5   Median :1631   Median : 828.5   Median :401.5  
##  Mean   :122.8   Mean   :1670   Mean   : 837.2   Mean   :401.2  
##  3rd Qu.:138.0   3rd Qu.:1851   3rd Qu.: 950.8   3rd Qu.:456.2  
##  Max.   :198.0   Max.   :2654   Max.   :1299.0   Max.   :646.0  
##       kms          PetrolPrice        VanKilled           law        
##  Min.   : 7685   Min.   :0.08118   Min.   : 2.000   Min.   :0.0000  
##  1st Qu.:12685   1st Qu.:0.09258   1st Qu.: 6.000   1st Qu.:0.0000  
##  Median :14987   Median :0.10448   Median : 8.000   Median :0.0000  
##  Mean   :14994   Mean   :0.10362   Mean   : 9.057   Mean   :0.1198  
##  3rd Qu.:17203   3rd Qu.:0.11406   3rd Qu.:12.000   3rd Qu.:0.0000  
##  Max.   :21626   Max.   :0.13303   Max.   :17.000   Max.   :1.0000  
```

As mentioned, seat-belt legislation was introduced on January 31, 1983, so it makes sense to split the dataset into two (before the legislation and after the legislation.We carryout this division of dataset using the subset() function that returns subsets of vectors, matrices, or data frames which meet conditions.In our case The condition will be law == 0 for before the legislation, and law ==1 for after the legislation:

PART:2 Subdivide the dataset into two - Before Legislation and After Legislation. Obtain the boxplots for the Drivers Killed Before Legislation and Drivers Killed After Legislation

```r
BeforeLegislation <-subset(Seatbelts,law ==0)
AfterLegislation <- subset(Seatbelts,law==1)
#To check for any changes following the enforcement of the law,
#we can plot the boxplots of the two newly obtained data frames:
par(mfrow=c(1,2))
boxplot(BeforeLegislation$DriversKilled,ylim=c(50,200),main="Before Legislation",
        ylab="# ofDrivers Killed")
boxplot(AfterLegislation$DriversKilled,ylim=c(50,200),main="After Legislation",
        ylab="# of Drivers Killed")
```



PART:3 Analyse and compare both the plots obtained
The above figure shows the Drivers Killed boxplot Before Legislation (to the left), and the Drivers Killed boxplot After Legislation (to the right):From the above plots,it is evident that the # of drivers killed has reduced after the legislation was passed.We further calculate the standard deviation of the drivers killed in each of the subset:

```
sd (BeforeLegislation$DriversKilled)
```

```
## [1] 24.26088
```

```
sd (AfterLegislation$DriversKilled)
```

```
## [1] 22.2286
```

We also look at the summary of both the subsets

```
summary(BeforeLegislation)
```

```
##  DriversKilled      drivers         front             rear
##  Min.   : 79.0   Min.   :1309   Min.   : 567.0   Min.   :224.0
##  1st Qu.:108.0   1st Qu.:1511   1st Qu.: 767.0   1st Qu.:344.0
##  Median :121.0   Median :1653   Median : 860.0   Median :401.0
```

```
##  Mean   :125.9   Mean   :1718   Mean   : 873.5   Mean   :400.3
##  3rd Qu.:140.0   3rd Qu.:1926   3rd Qu.: 986.0   3rd Qu.:454.0
##  Max.   :198.0   Max.   :2654   Max.   :1299.0   Max.   :646.0
##       kms         PetrolPrice        VanKilled          law
##  Min.   : 7685   Min.   :0.08118   Min.   : 2.000   Min.   :0
##  1st Qu.:12387   1st Qu.:0.09078   1st Qu.: 7.000   1st Qu.:0
##  Median :14455   Median :0.10273   Median :10.000   Median :0
##  Mean   :14463   Mean   :0.10187   Mean   : 9.586   Mean   :0
##  3rd Qu.:16585   3rd Qu.:0.11132   3rd Qu.:13.000   3rd Qu.:0
##  Max.   :21040   Max.   :0.13303   Max.   :17.000   Max.   :0
```

```r
summary(AfterLegislation)
```

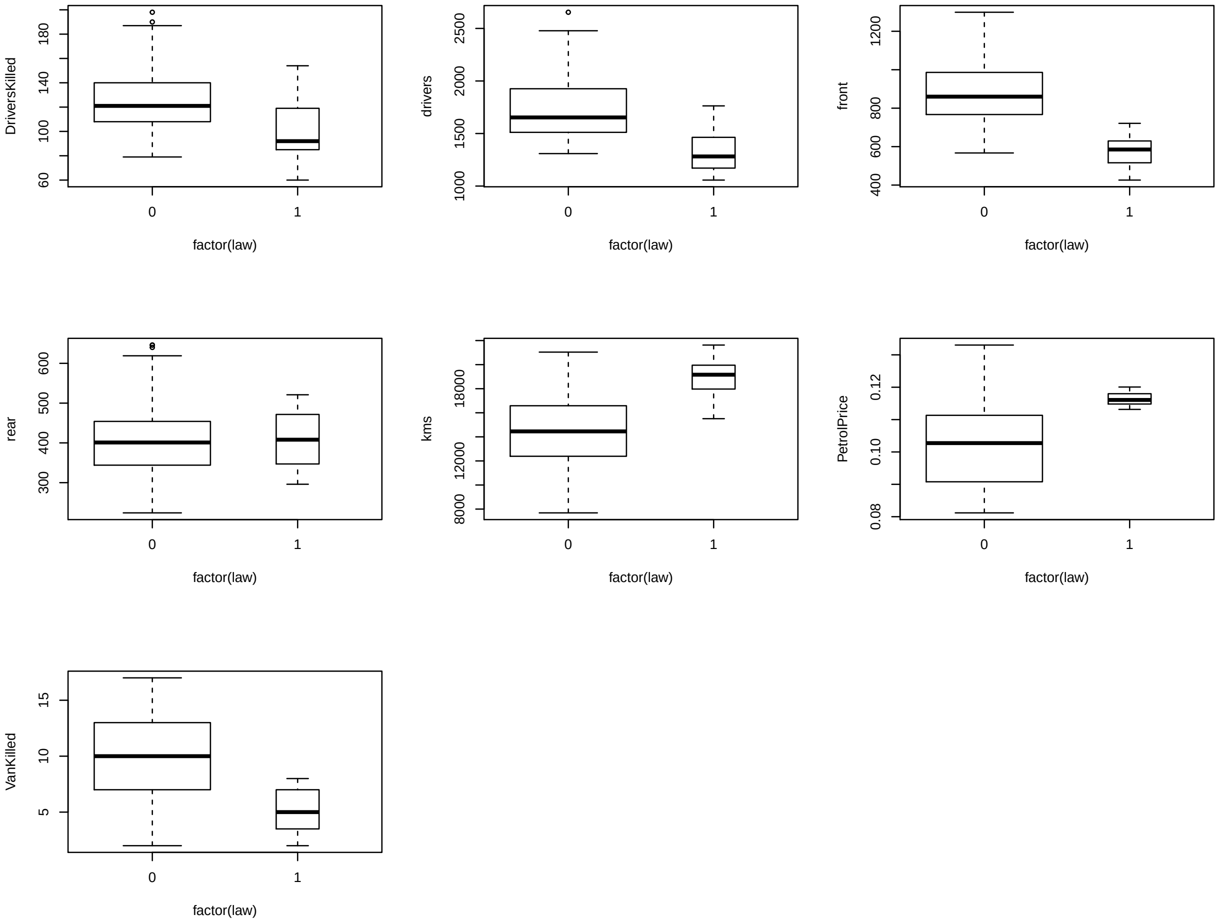
```
##  DriversKilled      drivers         front            rear
##  Min.   : 60.0   Min.   :1057   Min.   :426.0   Min.   :296.0
##  1st Qu.: 85.0   1st Qu.:1171   1st Qu.:516.0   1st Qu.:347.0
##  Median : 92.0   Median :1282   Median :585.0   Median :408.0
##  Mean   :100.3   Mean   :1322   Mean   :571.0   Mean   :407.7
##  3rd Qu.:119.0   3rd Qu.:1464   3rd Qu.:629.5   3rd Qu.:471.5
##  Max.   :154.0   Max.   :1763   Max.   :721.0   Max.   :521.0
##       kms         PetrolPrice       VanKilled         law
##  Min.   :15511   Min.   :0.1131   Min.   :2.000   Min.   :1
##  1st Qu.:17971   1st Qu.:0.1148   1st Qu.:3.500   1st Qu.:1
##  Median :19162   Median :0.1161   Median :5.000   Median :1
##  Mean   :18890   Mean   :0.1165   Mean   :5.174   Mean   :1
##  3rd Qu.:19952   3rd Qu.:0.1180   3rd Qu.:7.000   3rd Qu.:1
##  Max.   :21626   Max.   :0.1201   Max.   :8.000   Max.   :1
```

We observe that these facts given by sd calculation and summary agree with our above stated claim.

PART:4 Plot the boxplot of each variable in one single plot and give a short analysis of each variable

We can draw multiple boxplots in a single plot, by passing in a list, data frame or multiple vectors.We try melting the data frame first and then sticking to base graphics.

```r
par(mfrow=c(3,3))
plot(DriversKilled~factor(law),data=Seatbelts,varwidth=TRUE)
plot(drivers~factor(law),data=Seatbelts,varwidth=TRUE)
plot(front~factor(law),data=Seatbelts,varwidth=TRUE)
plot(rear~factor(law),data=Seatbelts,varwidth=TRUE)
plot(kms~factor(law),data=Seatbelts,varwidth=TRUE)
plot(PetrolPrice ~factor(law),data=Seatbelts,varwidth=TRUE)
plot( VanKilled  ~factor(law),data=Seatbelts,varwidth=TRUE)
```



From the above boxplots,we make the following observations
1)The Drivers(UK drivers deaths),Driverskilled,front-seat passengers killed or seriously injured,Vanskilled have all decreased after the legislation has passed.
2)We also notice that the passing of legislation does not have much effect on the back-seat passengers killed or seriously injured.
3)Kms travelled and the PetrolPrice has both increased after the legislation has been passed
4)We also further note that fluctuation in petrol-price has decreased considerably after the passing of the legislation.

PART:5 • **Aim: To predict the Drivers Killed Before Legislation using a multi-linear regression model with 6 predictor variables (exclude Drivers and Law variable) using Ridge and Lasso regression**

RIDGE-REGRESSION:

Ridge regression is very similar to least squares, except that the Ridge coefficients are estimated by minimizing a slightly different quantity. In particular, the Ridge re- gression coefficients β are the values that minimize the following quantity:

$$\sum_{i=1}^{n}(y_i - \beta_1 x_i + \beta_0)^2 + \lambda\beta_1^2 = RSS + \lambda\beta_1^2$$

Here, $\lambda \geq 0$ is a tuning parameter which is to be determined separately. The term $\lambda\beta_1^2$ is a shrinkage penalty that decreases when the β parameters withdraw (shrink) towards the zero. Parameter λ controls the relative impact of the two components: RSS and the penalty term.

As we can see
1)If $\lambda = 0$ the Ridge regression coincides with the least squares regression model
2)If $\lambda \to \infty$ all estimated coefficients tend to zero.

To perform Ridge regression, we will use the glmnet package that provides methods to algorithm regularization.The main function in the package is glmnet(). This function fits a generalized linear model (GLM) via penalized maximum likelihood. The regularization path is computed for the Ridge penalty at a grid of values for the regularization parameter lambda. Can deal with all shapes of data, including very large sparse data matrices.

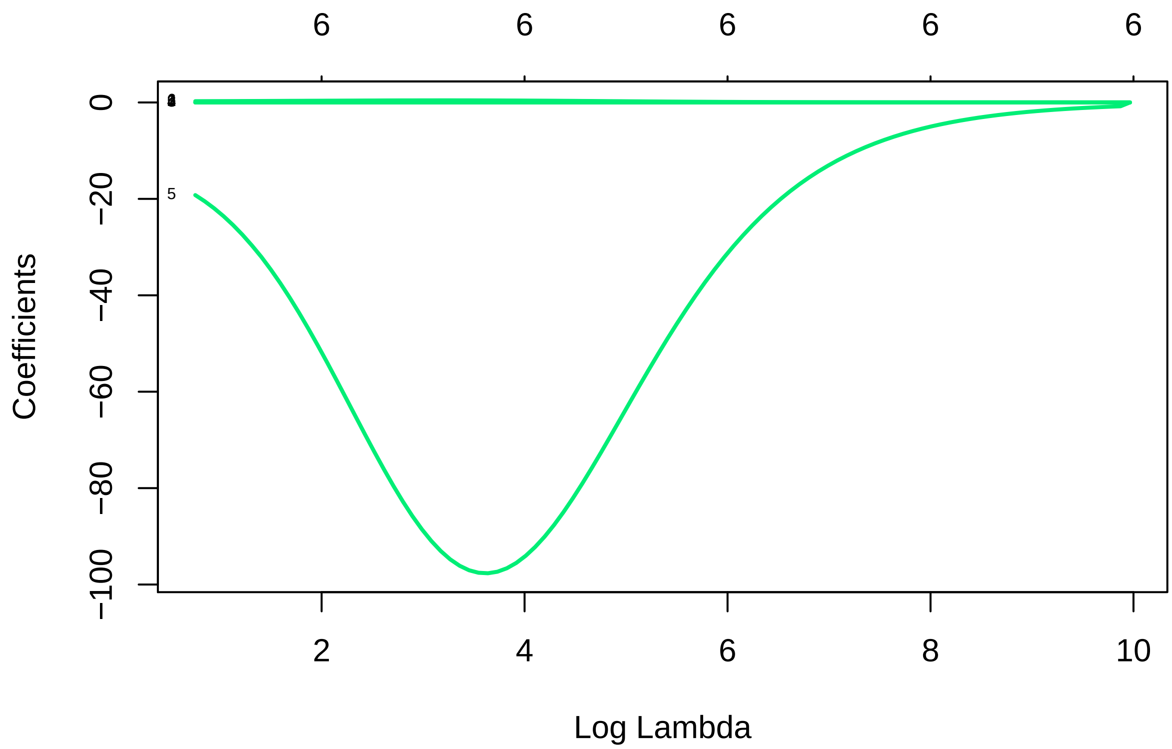
```
#Loading  the library through the library command:
library(glmnet)
#To use the glmnet() fn,we must first set the input matrix and the response variable:
x <- model.matrix(DriversKilled~., BeforeLegislation)[,-c(1,8)]
y <- BeforeLegislation$DriversKilled
```

We used the model.matrix() function to create a model matrix by expanding the factors to a set of dummy variables (depending upon the contrasts) and expanding interactions similarly. We added this term [,-c(1,8)] inorder to remove the variables-Drivers killed and law. Then we set the response variable as driverskilled before the legislation. Now we can use the glmnet() function to build the linear regression model with Ridge regularization. The glmnet() function provides an alpha argument that determines what method is used. If alpha=0, then Ridge regression is used.

```
#nlambda=100: Set the number of lambda values (the default is 100)
#lambda.min.ratio=0.0001: Set the smallest value for lambda, as a fraction of lambda.max,
#the(data derived) entry value(the smallest value for which all coefficients are zero)
Seatbelt.Ridge <- glmnet(x, y, alpha=0, nlambda=100,lambda.min.ratio=0.0001)
```

Let us now see how the coefficient values change according to the λ value. To do that, we will use the plot.glmnet() function that produces a coefficient profile plot of the coefficient paths for a fitted glmnet object:

```
plot(Seatbelt.Ridge,xvar="lambda",label=TRUE,col="springgreen2",lwd=2)
```



We make the following observations from the above plot
1)When lambda is very large or equivaletly when $log(\lambda)$ approaches 10, the regularization effect dominates the squared loss function and the coefficients tend to zero.
2)Whereas at small values of lambda($log(\lambda)$ approaches 0),the solution tends to be same as that of the Ordinary Least Square method
3)Thus we observe that the coefficients exhibit big oscillations meaning they are unregularized.But it is necessary to tune lambda in such a way that a balance is maintained between both.
Therefore,we need to rely on methods such as cross-validation inorder to select the best lambda.
Let us take a look at the summary of the glmnet path at each step by using the print function

```
print(Seatbelt.Ridge)
```

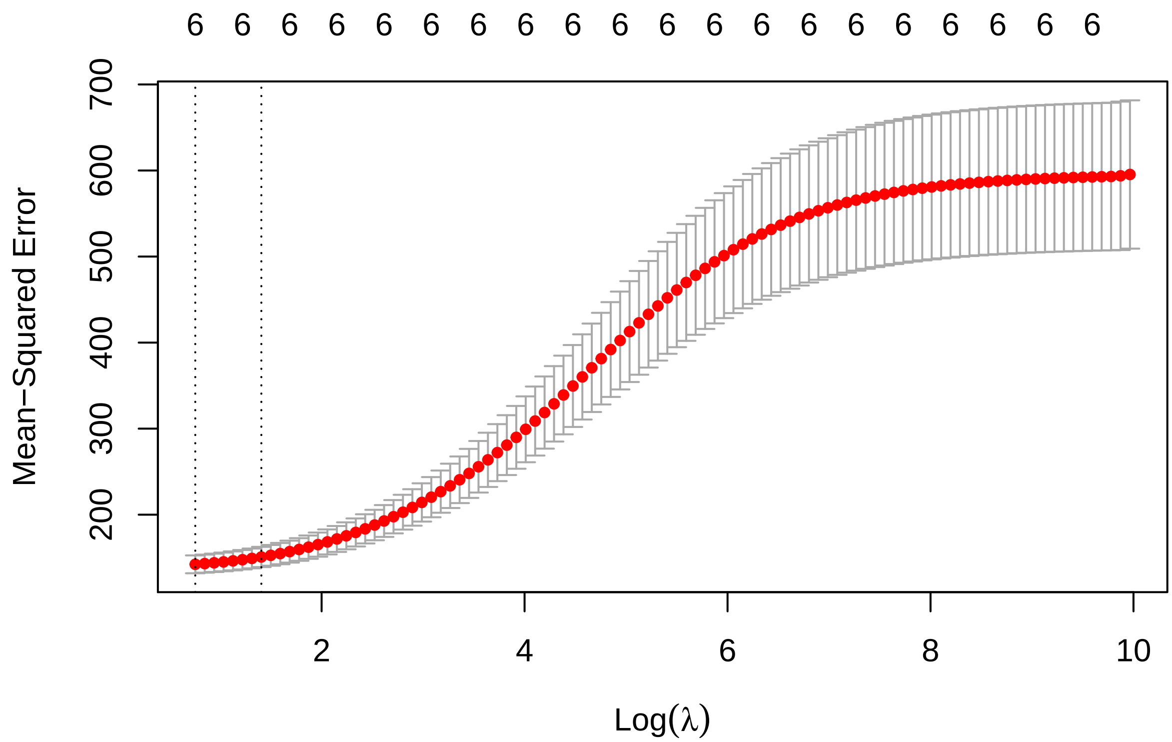
```
##
## Call:  glmnet(x = x, y = y, alpha = 0, nlambda = 100, lambda.min.ratio = 1e-04)
##
##      Df    %Dev  Lambda
## 1     6 0.00000 21280.0
## 2     6 0.00408 19390.0
## 3     6 0.00448 17660.0
## 4     6 0.00491 16090.0
## 5     6 0.00539 14660.0
## 6     6 0.00591 13360.0
## 7     6 0.00648 12170.0
## 8     6 0.00711 11090.0
## 9     6 0.00780 10110.0
```

```
## 10   6 0.00855  9210.0
## 11   6 0.00937  8392.0
## 12   6 0.01028  7646.0
## 13   6 0.01126  6967.0
## 14   6 0.01235  6348.0
## 15   6 0.01353  5784.0
## 16   6 0.01483  5270.0
## 17   6 0.01624  4802.0
## 18   6 0.01779  4375.0
## 19   6 0.01949  3987.0
## 20   6 0.02134  3633.0
## 21   6 0.02336  3310.0
## 22   6 0.02557  3016.0
## 23   6 0.02798  2748.0
## 24   6 0.03061  2504.0
## 25   6 0.03347  2281.0
## 26   6 0.03659  2079.0
## 27   6 0.03999  1894.0
## 28   6 0.04369  1726.0
## 29   6 0.04770  1572.0
## 30   6 0.05206  1433.0
## 31   6 0.05679  1305.0
## 32   6 0.06192  1190.0
## 33   6 0.06746  1084.0
## 34   6 0.07346   987.5
## 35   6 0.07993   899.8
## 36   6 0.08690   819.9
## 37   6 0.09441   747.0
## 38   6 0.10250   680.7
## 39   6 0.11110   620.2
## 40   6 0.12040   565.1
## 41   6 0.13020   514.9
## 42   6 0.14070   469.2
## 43   6 0.15190   427.5
## 44   6 0.16380   389.5
## 45   6 0.17630   354.9
## 46   6 0.18940   323.4
## 47   6 0.20330   294.6
## 48   6 0.21770   268.5
## 49   6 0.23280   244.6
## 50   6 0.24850   222.9
## 51   6 0.26470   203.1
## 52   6 0.28140   185.0
## 53   6 0.29860   168.6
## 54   6 0.31610   153.6
## 55   6 0.33390   140.0
## 56   6 0.35190   127.5
## 57   6 0.37010   116.2
## 58   6 0.38830   105.9
## 59   6 0.40650    96.5
## 60   6 0.42460    87.9
## 61   6 0.44240    80.1
## 62   6 0.46000    73.0
## 63   6 0.47720    66.5
```

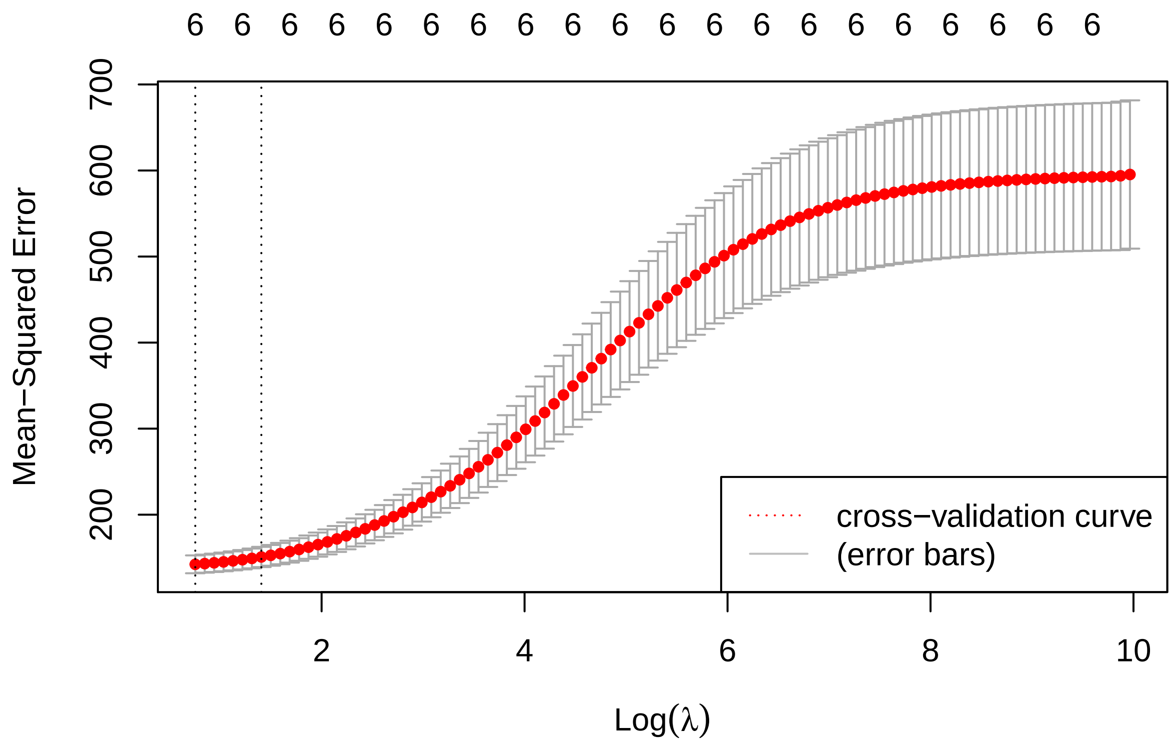
```
## 64   6 0.49410    60.6
## 65   6 0.51040    55.2
## 66   6 0.52630    50.3
## 67   6 0.54160    45.8
## 68   6 0.55640    41.8
## 69   6 0.57060    38.1
## 70   6 0.58410    34.7
## 71   6 0.59710    31.6
## 72   6 0.60960    28.8
## 73   6 0.62140    26.2
## 74   6 0.63270    23.9
## 75   6 0.64340    21.8
## 76   6 0.65360    19.8
## 77   6 0.66330    18.1
## 78   6 0.67240    16.5
## 79   6 0.68110    15.0
## 80   6 0.68930    13.7
## 81   6 0.69700    12.5
## 82   6 0.70430    11.3
## 83   6 0.71110    10.3
## 84   6 0.71750     9.4
## 85   6 0.72350     8.6
## 86   6 0.72910     7.8
## 87   6 0.73420     7.1
## 88   6 0.73900     6.5
## 89   6 0.74340     5.9
## 90   6 0.74740     5.4
## 91   6 0.75110     4.9
## 92   6 0.75440     4.5
## 93   6 0.75740     4.1
## 94   6 0.76020     3.7
## 95   6 0.76260     3.4
## 96   6 0.76480     3.1
## 97   6 0.76680     2.8
## 98   6 0.76850     2.6
## 99   6 0.77000     2.3
## 100  6 0.77140     2.1
```

10-FOLD CROSS-VALIDATION FOR BEST LAMBDA:
The Ridge regression draws a whole model path;but we neede to select the best.So we use the cv.glmnet() function available in the glmnet package.This function does k-fold cross-validation for glmnet, produces a plot, and returns the best λ value

```r
Seatbelt.Ridge.cv<- cv.glmnet(x, y, alpha=0, nlambda=100,lambda.min.ratio=0.0001)
#Now we will plot the cross-validation curve produced by plot.cv.glmnet():
plot(Seatbelt.Ridge.cv)
```




```r
legend("bottomright",c("cross-validation curve","(error bars)"),lty=c("dotted", "solid"),
       col=c("red","gray"))
```



The function cv.glmnet() runs glmnet nfolds+1 times; the first iteration is to get the lambda sequence, and then the rest of them is to compute the fit with each of the folds omitted. The error is accumulated, and the average error and standard deviation over the folds is computed.
We also observe that for very large values of λ, the MSE is very high, and the coefficients are restricted to be too small; and then at some point, it kind of levels off.This seems to indicate that the full model is doing a good job.
There are two vertical dotted-lines at two λ values:
1)The first vertical dotted-line is at the λ value where the mean cross-validated error is minimum
2)The second vertical dotted-line is at a λ value where the cross-validated error is within one standard error of the minimum
We now choose the best λ value based on the 10-fold cross-validation performed above

```
Seatbelt.Ridge.bestlambda <- Seatbelt.Ridge.cv$lambda.min
Seatbelt.Ridge.bestlambda
```

```
## [1] 2.127603
```

obtaining the coefficients of the Ridge using predict()

Ridge regression addresses the problem by estimating regression coefficients using the following equation:

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

where
$\lambda := Ridge \quad parameter$
$I := Identity \quad matrix$

```
predict(Seatbelt.Ridge,s=Seatbelt.Ridge.bestlambda, type="coefficients")
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -1.219728e+01
## drivers      6.655047e-02
## front        1.851418e-02
## rear        -2.842390e-03
## kms          5.644939e-04
## PetrolPrice -1.921260e+01
## VanKilled    2.617725e-01
```

A model built with the above coefficients is the best model for our dataset

LASSO REGRESSION:

The Lasso regression is also a shrinkage method just like Ridge, with subtle but important differences. The Lasso estimate is defined by the following equation:

$$\sum_{i=1}^{n}(y_i - \beta_1 x_i + \beta_0)^2 + \lambda|\beta_1| = RSS + \lambda|\beta_1|$$

Here, $\lambda \geq 0$ is a tuning parameter which is to be determined separately. The term $\lambda|\beta_1|$ is a shrinkage penalty.It is clear that Ridge and Lasso regression use two different penalty functions. Ridge uses L2- norm, whereas Lasso goes with L1-norm.
Let us set up the response variable as drivers killed before legislation and the 6 variables as predictors for our model

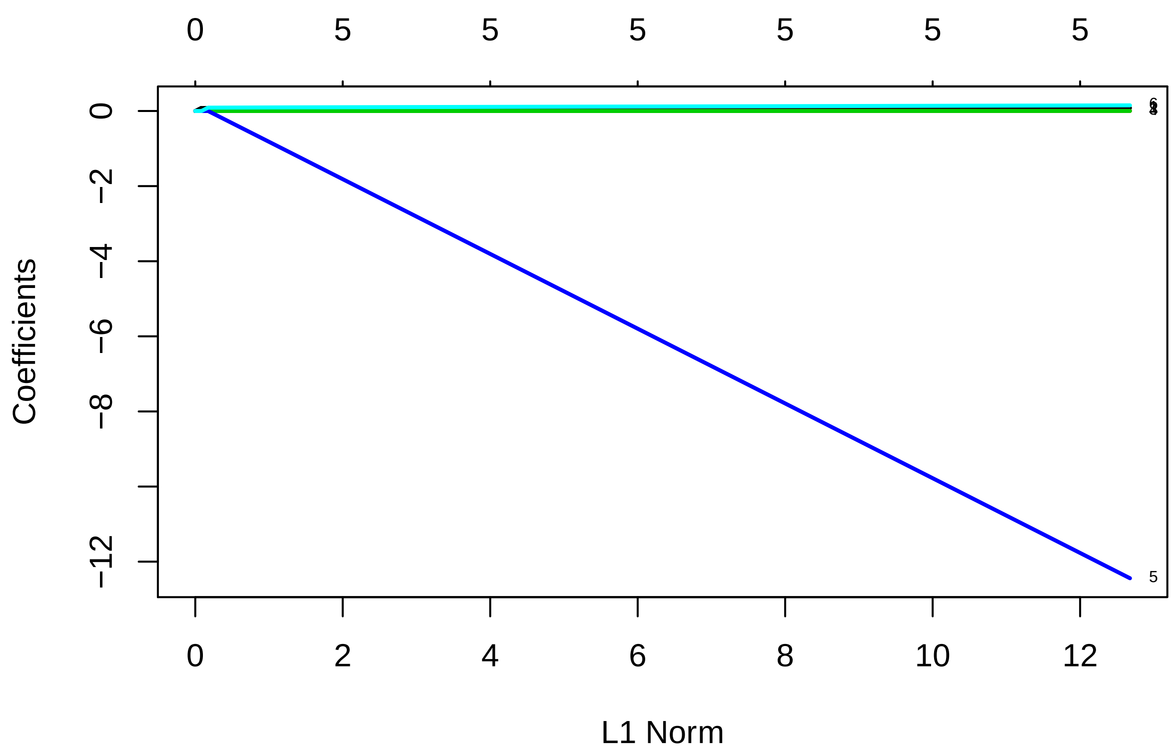
```
#To use the glmnet() function, we must first set the input matrix and the response variable:
x <- model.matrix(DriversKilled~., BeforeLegislation)[,-c(1,8)]
y <- BeforeLegislation$DriversKilled
```

Now we once again use the glmnet() function to build the linear regression model with Lasso regularization. The glmnet() function provides an alpha argument that determines what method is used. If alpha=1, the Lasso method is used

```
#nlambda=100: Set the number of lambda values (the default is 100)
#lambda.min.ratio=0.0001: Set the smallest value for lambda, as a fraction of lambda.max,
#the (data derived) entry value (that is, the smallest value for which all coefficients are zero)
Seatbelt.Lasso <- glmnet(x, y, alpha=1, nlambda=100,lambda.min.ratio=0.0001)
```

Let us now see how the coefficient values change according to the lambda value. To do that, we will use the plot.glmnet() function that produces a coefficient profile plot of the coefficient paths for a fitted glmnet object:

```r
#we use the norm option that plots coefficients against the L1-norm
plot(Seatbelt.Lasso,xvar="norm",label=TRUE,lwd=2)
```



In the previous figure, each curve corresponds to a variable. This plot shows the path of its coefficient against the L1-norm of the whole coefficient vector when λvarying. The previous axis indicates the number of nonzero coefficients at the current λ, which is the effective degrees of freedom for the Lasso model.
Let us looks at the summary of the glmnet path

```r
print(Seatbelt.Lasso)
```

```
##
## Call:  glmnet(x = x, y = y, alpha = 1, nlambda = 100, lambda.min.ratio = 1e-04)
##
##    Df   %Dev  Lambda
## 1   0 0.0000 21.2800
## 2   1 0.1314 19.3900
## 3   1 0.2404 17.6600
## 4   1 0.3309 16.0900
## 5   1 0.4061 14.6600
## 6   1 0.4685 13.3600
## 7   1 0.5203 12.1700
## 8   1 0.5633 11.0900
## 9   1 0.5990 10.1100
## 10  1 0.6287  9.2100
## 11  1 0.6533  8.3920
```

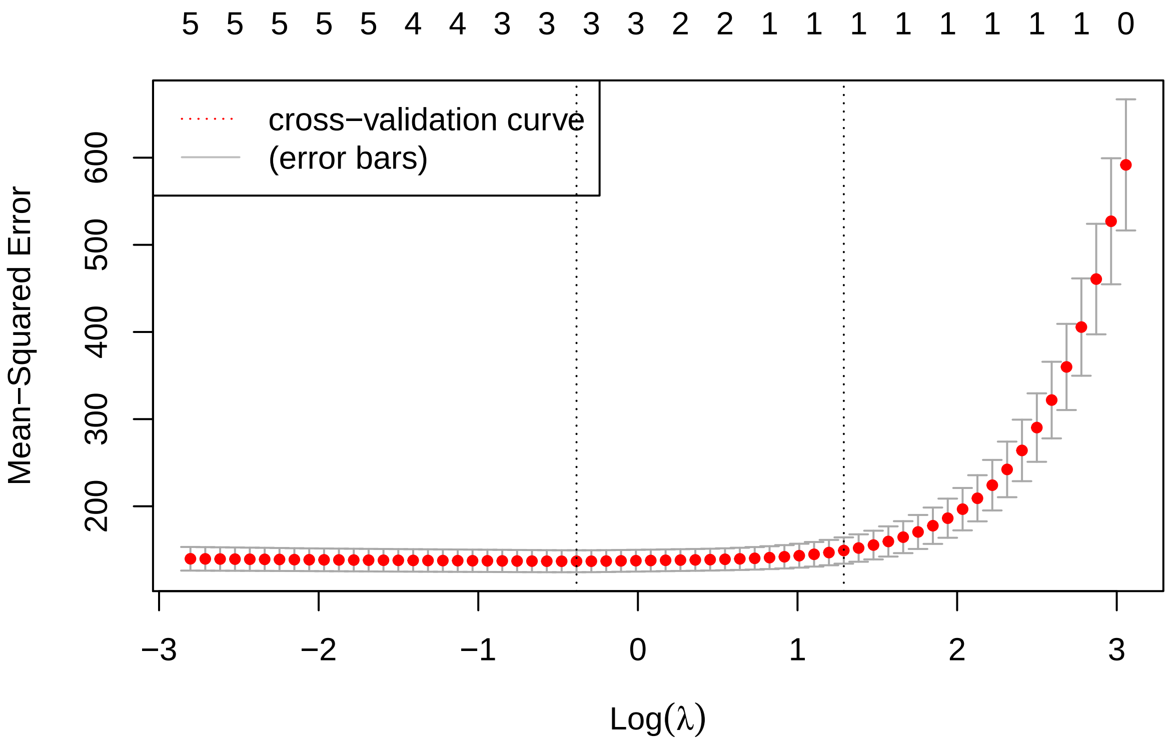
```
## 12  1 0.6737  7.6460
## 13  1 0.6907  6.9670
## 14  1 0.7048  6.3480
## 15  1 0.7165  5.7840
## 16  1 0.7262  5.2700
## 17  1 0.7342  4.8020
## 18  1 0.7409  4.3750
## 19  1 0.7465  3.9870
## 20  1 0.7511  3.6330
## 21  1 0.7549  3.3100
## 22  1 0.7581  3.0160
## 23  1 0.7607  2.7480
## 24  1 0.7629  2.5040
## 25  1 0.7648  2.2810
## 26  1 0.7663  2.0790
## 27  2 0.7675  1.8940
## 28  2 0.7690  1.7260
## 29  2 0.7703  1.5720
## 30  2 0.7713  1.4330
## 31  2 0.7721  1.3050
## 32  3 0.7729  1.1900
## 33  3 0.7741  1.0840
## 34  3 0.7751  0.9875
## 35  3 0.7759  0.8998
## 36  3 0.7766  0.8199
## 37  3 0.7771  0.7470
## 38  3 0.7776  0.6807
## 39  3 0.7780  0.6202
## 40  3 0.7783  0.5651
## 41  3 0.7786  0.5149
## 42  3 0.7788  0.4692
## 43  3 0.7790  0.4275
## 44  4 0.7792  0.3895
## 45  4 0.7794  0.3549
## 46  4 0.7795  0.3234
## 47  4 0.7797  0.2946
## 48  4 0.7798  0.2685
## 49  4 0.7799  0.2446
## 50  4 0.7799  0.2229
## 51  4 0.7800  0.2031
## 52  5 0.7801  0.1850
## 53  5 0.7801  0.1686
## 54  5 0.7802  0.1536
## 55  5 0.7802  0.1400
## 56  5 0.7802  0.1275
## 57  5 0.7803  0.1162
## 58  5 0.7803  0.1059
## 59  5 0.7803  0.0965
## 60  5 0.7803  0.0879
## 61  5 0.7803  0.0801
## 62  5 0.7803  0.0730
## 63  5 0.7803  0.0665
## 64  5 0.7803  0.0606
```

The results show that Even though we set $nlambda = 100$, the program stops early if %dev does not change significantly from one lambda to the next.Which happens at low λ values

10-FOLD CROSS-VALIDATION FOR BEST LAMBDA:

The Ridge regression draws a sequence of models implied by λ fitted by coordinate descent; ;but we neede to select the best.So we use the cv.glmnet() function available in the glmnet package.This function does k-fold cross-validation for glmnet, produces a plot, and returns the best λ value

```
Seatbelt.Lasso.cv<- cv.glmnet(x, y, alpha=1, nlambda=100,lambda.min.ratio=0.0001)
#Now we will plot the cross-validation curve produced by plot.cv.glmnet():
plot(Seatbelt.Lasso.cv)
legend("topleft",c("cross-validation curve","(error bars)"),lty=c("dotted", "solid"),
       col=c("red","gray"))
```



We also observe that for very large values of λ, the MSE is very high, and the coefficients are restricted to be too small; and then at some point, it kind of levels off.This seems to indicate that the full model is doing a good job.
There are two vertical dotted-lines at two λ values:
1)The first vertical dotted-line is at the λ value where the mean cross-validated error is minimum
2)The second vertical dotted-line is at a λ value where the cross-validated error is within one standard error of the minimum
We now choose the best λ value based on the 10-fold cross-validation performed above

```r
Seatbelt.Lasso.bestlambda= Seatbelt.Lasso.cv$lambda.min
Seatbelt.Lasso.bestlambda
```

```
## [1] 0.6806771
```

obtaining the coefficients of the Lasso using coef()

The Ridge regression produces a model with all the variables, of which the part with coefficients is closer to zero.Increasing λ forces more coefficients to be close to zero, but almost never exactly equal to zero, unless $\lambda = \infty$. For forecasting this is not a problem, while interpretation can sometimes be problematic.
The Lasso regression penalty term,unlike ridge,forces some coefficients to be exactly equal to zero, if λ is large enough. In practice, Lasso automatically performs a real selection of variables. Lasso regression tries to overcome this aspect.

```r
coef(Seatbelt.Lasso.cv, s = "lambda.min")
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -1.374971e+01
## drivers      7.760291e-02
## front        .
## rear         5.250162e-03
## kms          2.914472e-04
## PetrolPrice  .
## VanKilled    .
```

We see that unlike the ridge regression ,Lasso regression has made a selection of variables alongwith performing shrinkage.A model built with the above coefficients is the best model for our dataset

COMPARISON OF RIDGE AND LASSO REGRESSION:

1)Lasso method is able to make a selection of variables unlike ridge method
2)Lasso implicitly assumes that part of the coefficients are zero or not significant.In our dataset,Lasso method says that coefficients of front,petrolprice,vankilled are not significant
3)Lasso tends to have a higher performance than Ridge in cases where not many predictors have strong relationship with the response variable.And ridge performs better when many predictor variables are tied to the response variable

QUESTION:2

PART:1 Load the dataset Swiss from datasets package in R

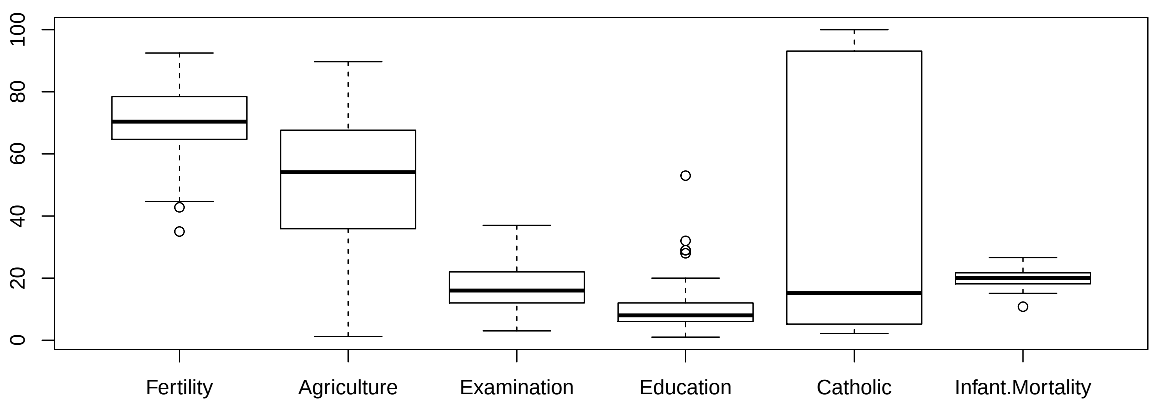
```r
Swiss <- data.frame(swiss)
#Use the str() function to view a compact display of the dataframe
str(Swiss)
```

```
## 'data.frame':    47 obs. of  6 variables:
##  $ Fertility       : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4 82.9 ...
##  $ Agriculture     : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3 45.2 ...
##  $ Examination     : int  15 6 5 12 17 9 16 14 12 16 ...
##  $ Education       : int  12 9 5 7 15 7 7 8 7 13 ...
##  $ Catholic        : num  9.96 84.84 93.4 33.77 5.16 ...
##  $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21 24.4 ...
```

Thus we see that there are 47 observations of the following 6 variables
1)Fertility:index of marital fertility
2)Agriculture: percentage of males involved in agriculture occupation
3)Examination: percentage of draftees receiving the highest mark on army examination
4)Education: percentage of education beyond primary school for draftees
5)Catholic: percentage of catholics
6)Infant.Mortality :live babies in birth that lived less than one year

PART:2 In one single plot, plot the boxplot of each of the given variables

```
boxplot(Swiss)
```



We make the following observations from the boxplot produced above
1)The variable-Catholic has a large amount of variantiona sit covers a wide range of values
2)On the other hand,we observe that the variable-infant.Mortality has very less range as it looks more condenesed plot
3)The variable education shows some potential outliers

PART:3 Produce a grid of scatter plots to analyse the correlation (positive or negative) between all pairs of variables in the dataset

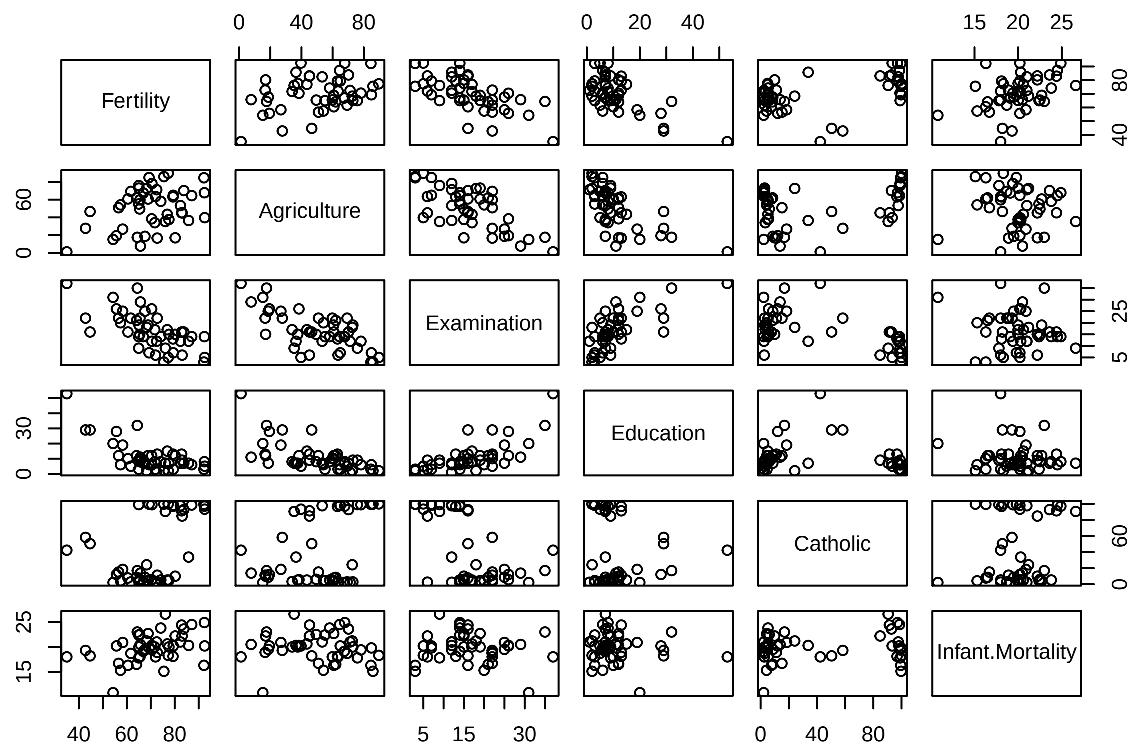
To analyse the correlation b/w variables,we use the cor() function that returns a correlation-coeffieicent matrix between all possible pairs of variables in the dataset

```r
correlation<-cor(Swiss)
correlation
```

```
##                  Fertility Agriculture Examination   Education   Catholic
## Fertility        1.0000000  0.35307918  -0.6458827 -0.66378886  0.4636847
## Agriculture      0.3530792  1.00000000  -0.6865422 -0.63952252  0.4010951
## Examination     -0.6458827 -0.68654221   1.0000000  0.69841530 -0.5727418
## Education       -0.6637889 -0.63952252   0.6984153  1.00000000 -0.1538589
## Catholic         0.4636847  0.40109505  -0.5727418 -0.15385892  1.0000000
## Infant.Mortality  0.4165560 -0.06085861  -0.1140216 -0.09932185  0.1754959
##                 Infant.Mortality
## Fertility             0.41655603
## Agriculture          -0.06085861
## Examination          -0.11402160
## Education            -0.09932185
## Catholic              0.17549591
## Infant.Mortality      1.00000000
```

Scatter plots are a great way to roughly determine if there exist a linear correlation between multiple variables. This is particularly helpful in locating variables that might have mutual correlations, indicating a possible redundancy of data.We now produce a scatterplot to visualise the correaliton b/w variables

```r
plot(Swiss)
```



From the above matrix of plots we make the following observations

1)Infant.Mortality is positively correlated with Fertility
2)Infant.Mortality is positively correlated with Catholic
we observe that the coefficients of these two are positive which agrees with the plot
3)Infant.Mortality is negatively correlated with Examination and Education
we observe that the coefficients of these two are negaitive which agrees with the plot
4)Fertility is positively correlated with being Catholic and with Agriculture
we observe that the coefficients of these two are positive which agrees with the plot
5)Fertility is negatively correlated with Education and Examination.
we observe that the coefficients of these two are negaitive which agrees with the plot

PART:4 building a linear regression model that predicts Infant.Mortality on other variables

SUBSET-SELECTION

```
#loading leaps library
library(leaps)
#using regsubsets() function for subset selection by exhaustive search:
predictorSubset <- regsubsets(Infant.Mortality~., Swiss,nvmax=5)
#summary() function returns a matrix with the best subset of
#predictors for one to five predictor models
brief=summary(predictorSubset)
brief
```

```
## Subset selection object
## Call: regsubsets.formula(Infant.Mortality ~ ., Swiss, nvmax = 5)
## 5 Variables  (and intercept)
##             Forced in Forced out
## Fertility       FALSE      FALSE
## Agriculture     FALSE      FALSE
## Examination     FALSE      FALSE
## Education       FALSE      FALSE
## Catholic        FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: exhaustive
##          Fertility Agriculture Examination Education Catholic
## 1  ( 1 ) "*"       " "         " "         " "       " "
## 2  ( 1 ) "*"       " "         " "         "*"       " "
## 3  ( 1 ) "*"       "*"         " "         "*"       " "
## 4  ( 1 ) "*"       "*"         "*"         "*"       " "
## 5  ( 1 ) "*"       "*"         "*"         "*"       "*"
```

From the above results,we conclude the following
1)the best-model with one predictor var is Fertility
2)the best model with two variables includes Fertility and Education
3)the best model with three variables includes Fertility,Education and Agriculture
4)the best model with 4 variables includes Fertility,Education,Agriculture and Examination
5)the best model with 4 variables includes all of them

Determining the best overall model by computing the adjusted r-squared and CP values. Mention the variables

The adjusted R-squared measures the descriptive power of regression models that include diverse numbers of predictors.We know that as more predictors are added to a model, R-squared increases and never decreases it. Thus, a model with more terms may seem to have a better fit just because it has more terms, while the adjusted R-squared compensates for the addition of variables and only increases if the new term enhances the model. Thus the best model among all 5 would be having the maximum adjusted R-squared.

```
#finding out the best subset among the above 5 subsets using R-squared values<br>
rsqBestSubset <- which.max(brief$adjr2)
cat("best-model with max adjusted R-squares is the one with",rsqBestSubset,"variables")
```

```
## best-model with max adjusted R-squares is the one with 2 variables
```

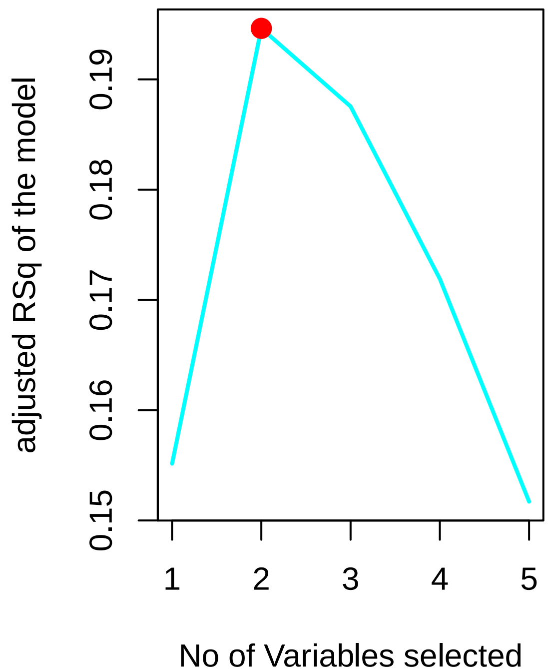
Let us see what analysing models with the Mallow's cp element tells us

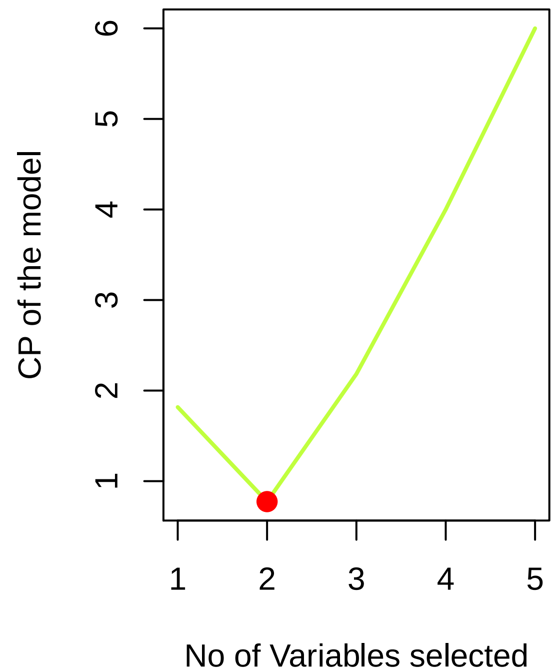
```
cpBestSubset<- which.min(brief$cp)
cat("best-model with min c.p is the one with",cpBestSubset,"variables")
```

```
## best-model with min c.p is the one with 2 variables
```

We observe that both the adjusted R-squared and C.P return the same result.
Let us produce plots to visualise how adjusted R-squared and C.P varies with no of predictors in the model

```
par(mfrow=c(1,2))
plot(brief$adjr2, xlab="No of Variables selected",ylab="adjusted RSq of the model", type="l",
     col="cyan",lwd=2)
points(rsqBestSubset, brief$adjr2[rsqBestSubset],col="red", cex =2, pch =20)
plot(brief$cp, xlab="No of Variables selected", ylab="CP of the model",
     type="l",col="olivedrab1",lwd=2)
points(cpBestSubset, brief$cp[cpBestSubset],col="red", cex =2, pch =20)
```





The best overall model is the one with 2-variables.The best model predicts Infant.Mortality with the two predictor variables-Fertility and Education

Extracting the coefficients of the best model that was selected using the adjusted R-squared and CP metrics

```r
coef(predictorSubset,2)
```

```
## (Intercept)   Fertility   Education
##  8.63757624  0.14615350  0.09594897
```

QUESTION:3
PART:1 Load the dataset Default from ISLR package in R

```r
#loading the ISLR and MASS library
library(ISLR)
library(MASS)
#attahcing default dataset
attach(Default)
#loading the dataset and saving it in a dataframe:
df<-Default
#Using the str() function to view a compact display of dataset
str(df)
```

```
## 'data.frame':    10000 obs. of  4 variables:
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 1 ...
##  $ balance: num  730 817 1074 529 786 ...
##  $ income : num  44362 12106 31767 35704 38463 ...
```

```r
#setting a random seed
set.seed(100)
```

As we can see there are 4 variables in the dataset. among which 2 of them are numerical and two of them are factor.

PART:2Fit a logistic regression model that uses income and balance to predict default
Let us fit such a model and look at the summary of the fit

```r
logitreg<-glm(default~income+balance,family = binomial,data=Default)
summary(logitreg)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial,
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

Now we use the above model to predict default and look at the test error of theis model.Note ,we have not split the dataset yet.We will do so in the next step when performing cross-validation

```
#predicting
pd<-predict(logitreg,Default$default,type="response")
# classifying the individual to the default category if
#the posterior probability is greater than 0.5
pd.class<-ifelse(pd>0.5,"Yes","No")
round(mean(Default$default!=pd.class),4)
```

```
## [1] 0.0263
```

The training error rate for the above model is 0.0263.But we have to split the dataset and look at the testing error rate!!

PART:3 Using the validation set approach to estimate the test error of this model

We explore the use of the validation set approach in order to estimate the test error rates that result from fitting various linear models on the Default data set.It is generally a good idea to set a random seed when performing an analysis such as cross-validation that contains an element of randomness, so that the results obtained can be reproduced precisely at a later time.

Splitting the sample set into a training set and a validation set

```
subset<- sample(dim(Default)[1], dim(Default)[1] / 2)
default.train<-Default[subset,]
default.test<-Default[-subset,]
```

Fitting a multiple logistic regression model using only the training observations

We then use the subset option in glm() to fit a model to predict default with predictors income and balance using only the observations in the training set

```
Default.lr <- glm(default ~ income + balance, data =default.train, family = "binomial")
summary(Default.lr)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = default.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7388  -0.1361  -0.0543  -0.0192   3.5104
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.174e+01  6.250e-01 -18.781  < 2e-16 ***
```

```
## income       2.068e-05  6.954e-06   2.974  0.00294 **
## balance      5.782e-03  3.240e-04  17.844  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1517.14  on 4999  degrees of freedom
## Residual deviance:  791.56  on 4997  degrees of freedom
## AIC: 797.56
##
## Number of Fisher Scoring iterations: 8
```

Obtaining a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5

```r
#We now use the predict() function to estimate default status
#for all  observations in testing set
Default.predict <- predict(Default.lr, newdata = default.test, type = "response")
#creating a container for each observations in testing dataset to
#store the posterior probabilty of default for that observation and intiaising all with "NO"
Default.classify <- rep("No", length(Default.predict))
#assigning a "yes" for those observations where the posterior porbablitity >0.5
Default.classify[Default.predict > 0.5] <- "Yes"
#tabulating the results
table(default.test$default,Default.classify,dnn=c("Actual","Predicted"))
```

```
##       Predicted
## Actual   No  Yes
##    No  4822   20
##    Yes  112   46
```

Computing the validation set error
Validation set error is the fraction of the observations in the validation set that are misclassified.

```r
mean(Default.classify != default.test$default)
```

```
## [1] 0.0264
```

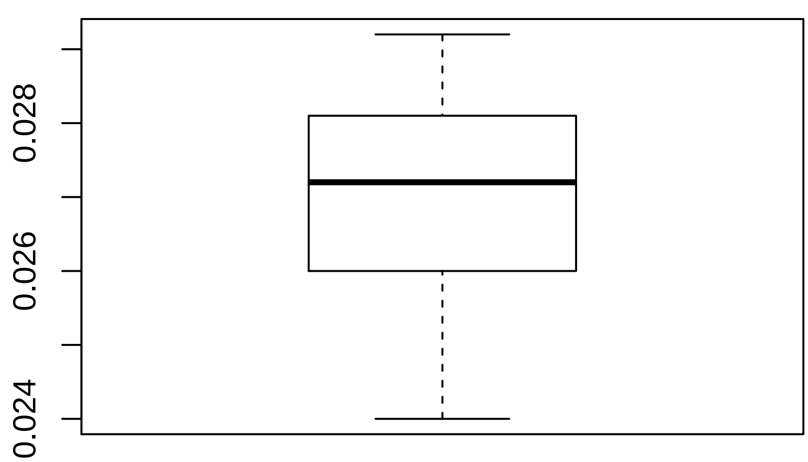
Thus,we observe that for a seed value of 100,the test error rate for the logistic regression model id **0.0264 or 2.64%** by using the validation set approach
But it is important to note that y if we choose a different training set ,then we will obtain somewhat different test error on the validation set for the same model.Therefore,we repeat the above process many times using different splits on dataset and observe the results obtained each time.

```r
#container for storing the test error rates for 15 different splits
testing.error=rep(15)
set.seed(10)
for(i in 1:15){
  subset<- sample(dim(Default)[1], dim(Default)[1] / 2)
  default.train<-Default[subset,]
  default.test<-Default[-subset,]
  Default.lr <- glm(default ~ income + balance, data =default.train, family = "binomial")
  Default.predict <- predict(Default.lr, newdata = default.test, type = "response")
  Default.classify <- rep("No", length(Default.predict))
  Default.classify[Default.predict > 0.5] <- "Yes"
  testing.error[i]<-mean(Default.classify != default.test$default)
}
testing.error
```

```
##  [1] 0.0288 0.0272 0.0288 0.0260 0.0284 0.0240 0.0278 0.0262 0.0244 0.0292
## [11] 0.0260 0.0260 0.0262 0.0276 0.0276
```

From the above test errors returned by the validation set approach,we can clearly see that the test error depends precisely on which observations goes into the testing set and which goes into the training set.

```r
boxplot(testing.error)
```



PART:4 predicting default using income, balance, and a dummy variable for student and estimating the test error for this model using the validation set approach

```r
#setting seed
set.seed(82)
#splitting dataset into train and test
subset<- sample(dim(Default)[1], dim(Default)[1] / 2)
default.train<-Default[subset,]
default.test<-Default[-subset,]
```

Fitting a multiple logistic regression model using only the training observations
We then use the subset option in glm() to fit a model to predict default with predictors income , balance and student using only the observations in the training set

```r
Default.lr.dummy <- glm(default ~ income + balance+student, data =default.train,
                        family = "binomial")
summary(Default.lr.dummy)
```

```
##
## Call:
## glm(formula = default ~ income + balance + student, family = "binomial",
##     data = default.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0081  -0.1574  -0.0659  -0.0260   3.6304
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.007e+01  6.385e-01 -15.768   <2e-16 ***
## income      -2.152e-07  1.123e-05  -0.019   0.9847
## balance      5.349e-03  2.991e-04  17.882   <2e-16 ***
## studentYes  -7.326e-01  3.234e-01  -2.265   0.0235 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1490.52  on 4999  degrees of freedom
## Residual deviance:  842.74  on 4996  degrees of freedom
## AIC: 850.74
##
## Number of Fisher Scoring iterations: 8
```

Obtaining a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5

```r
#We now use the predict() function to estimate
#default status for all  observations in testing set
Default.predict.dummy <- predict(Default.lr.dummy, newdata = default.test,
                                 type = "response")
#creating a container for each observations in testing dataset to
#store the posterior probabilty of default for that observation,intiaising all with "NO"
Default.classify.dummy <- rep("No", length(Default.predict.dummy))
#assigning a "yes" for those observations where the posterior porbablitity >0.5
Default.classify.dummy[Default.predict.dummy > 0.5] <- "Yes"
#tabulating the results
table(default.test$default,Default.classify.dummy,dnn=c("Actual","Predicted"))
```

```
##       Predicted
## Actual   No  Yes
##    No  4825   13
##    Yes  111   51
```

Computing the validation set error
Validation set error is the fraction of the observations in the validation set that are misclassified.

```r
mean(Default.classify.dummy != default.test$default)
```

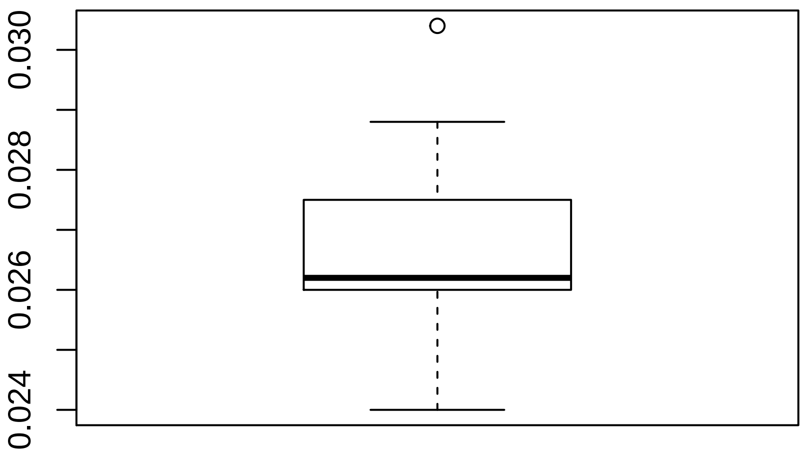
```
## [1] 0.0248
```

Thus,we observe that for the above partition,the test error rate for the logistic regression model id **0.0248 or 2.48%** by using the validation set approach.We see the test error has not reduced much.
We repeat the above process many times using different splits on dataset and observe the results obtained each time and see of there is a reduction observed anytime

```r
#container for storing the test error rates for 15 different splits
testing.error.dummy=rep(15)
set.seed(10)
for(i in 1:15){
  subset<- sample(dim(Default)[1], dim(Default)[1] / 2)
  default.train<-Default[subset,]
  default.test<-Default[-subset,]
  Default.lr.dummy <- glm(default ~ income + balance+student,
                          data =default.train,family = "binomial")
  Default.predict.dummy <- predict(Default.lr.dummy, newdata = default.test,
                                   type = "response")
  Default.classify.dummy <- rep("No", length(Default.predict.dummy))
  Default.classify.dummy[Default.predict.dummy > 0.5] <- "Yes"
  testing.error.dummy[i]<-mean(Default.classify.dummy != default.test$default)
}
testing.error.dummy
```

```
## [1] 0.0286 0.0262 0.0276 0.0260 0.0288 0.0240 0.0272 0.0262 0.0252 0.0304
## [11] 0.0260 0.0262 0.0254 0.0274 0.0274
```

```r
boxplot(testing.error.dummy)
```



From the above test errors returned by the validation set approach by including the dummy variable student in the model,we can clearly see that the test error has not reduced