Homework 3

Naga Kartheek Peddisetty, 50538422

10/15/2023

Question 1) Consider the Diabetes dataset (posted with assignment). Assume the population prior probabilities are estimated using the relative frequencies of the classes in the data.

```r
library(MASS)
library(ggplot2)
setwd("D:/Buffalo/files")
load("D:/Buffalo/files/Diabetes.RData")
dim(Diabetes)  # 145 * 6
```

```
## [1] 145   6
```

```r
str(Diabetes)
```

```
## 'data.frame':    145 obs. of  6 variables:
##  $ relwt  : num  0.81 0.95 0.94 1.04 1 0.76 0.91 1.1 0.99 0.78 ...
##  $ glufast: int  80 97 105 90 90 86 100 85 97 97 ...
##  $ glutest: int  356 289 319 356 323 381 350 301 379 296 ...
##  $ instest: int  124 117 143 199 240 157 221 186 142 131 ...
##  $ sspg   : int  55 76 105 108 143 165 119 105 98 94 ...
##  $ group  : Factor w/ 3 levels "Normal","Chemical_Diabetic",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
head(Diabetes)
```

```
##   relwt glufast glutest instest sspg  group
## 1  0.81      80     356     124   55 Normal
## 2  0.95      97     289     117   76 Normal
## 3  0.94     105     319     143  105 Normal
## 4  1.04      90     356     199  108 Normal
## 5  1.00      90     323     240  143 Normal
## 6  0.76      86     381     157  165 Normal
```

```r
table(Diabetes$group)
```

```
## 
##            Normal Chemical_Diabetic    Overt_Diabetic 
##                76                36                33
```

```r
diabetes <- Diabetes

### Converting int to numeric data

diabetes[1:5] <- data.frame(sapply(diabetes[1:5], as.numeric))
str(diabetes)
```

```
## 'data.frame':    145 obs. of  6 variables:
##  $ relwt  : num  0.81 0.95 0.94 1.04 1 0.76 0.91 1.1 0.99 0.78 ...
##  $ glufast: num  80 97 105 90 90 86 100 85 97 97 ...
##  $ glutest: num  356 289 319 356 323 381 350 301 379 296 ...
##  $ instest: num  124 117 143 199 240 157 221 186 142 131 ...
##  $ sspg   : num  55 76 105 108 143 165 119 105 98 94 ...
##  $ group  : Factor w/ 3 levels "Normal","Chemical_Diabetic",..: 1 1 1 1 1 1 1 1 1 1 ...
```

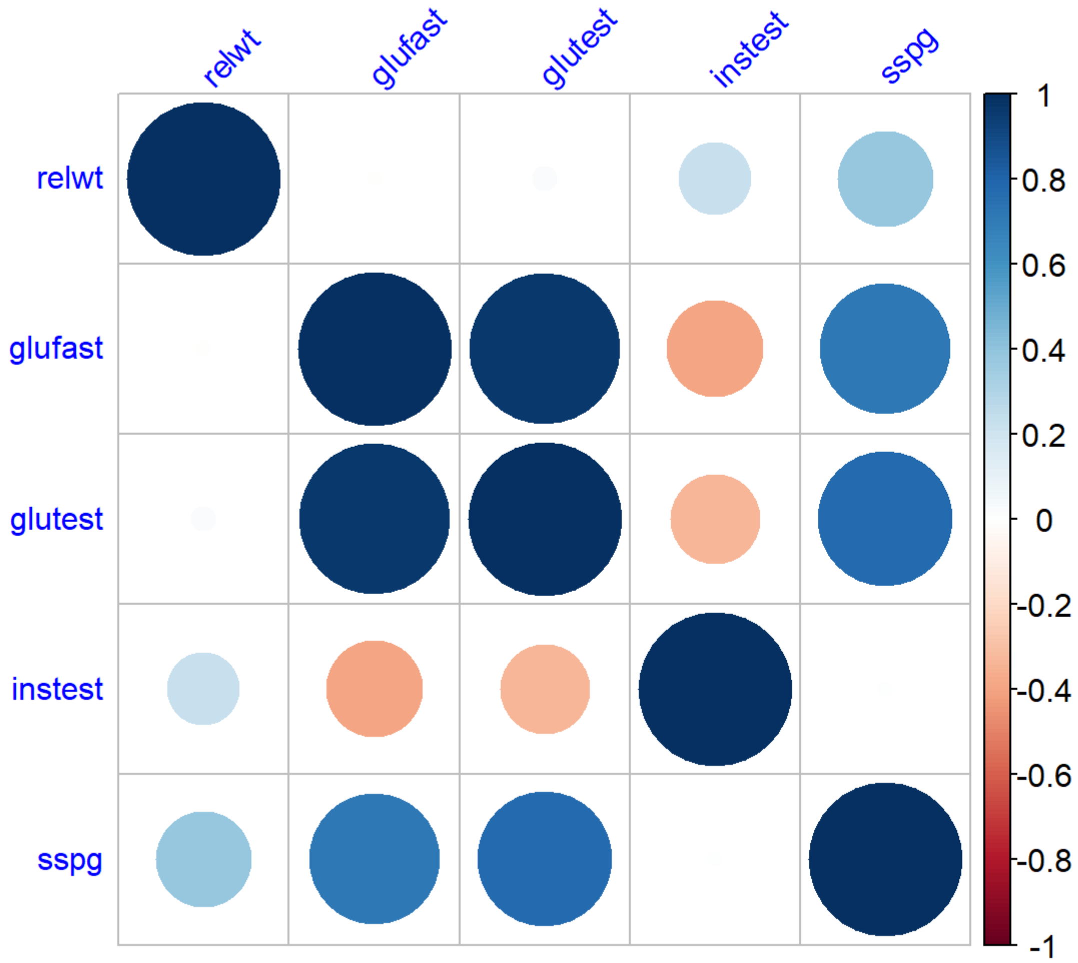
```r
cor(diabetes[1:5])
```

```
##                relwt      glufast    glutest      instest        sspg
## relwt    1.000000000 -0.008813193  0.0239843  0.222237813 0.384319804
## glufast -0.008813193  1.000000000  0.9646281 -0.396234858 0.715480192
## glutest  0.023984304  0.964628091  1.0000000 -0.337020435 0.770942459
## instest  0.222237813 -0.396234858 -0.3370204  1.000000000 0.007914263
## sspg     0.384319804  0.715480192  0.7709425  0.007914263 1.000000000
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
corrplot(cor(diabetes[1:5]), tl.col = "blue", tl.srt= 45, tl.cex=1, cl.cex=1)
```



a) Produce pairwise scatterplots for all five variables, with different symbols or colors representing the three different classes. Do you see any evidence that the classes may have difference covariance matrices? That they may not be multivariate normal?

```
library(psych)
```

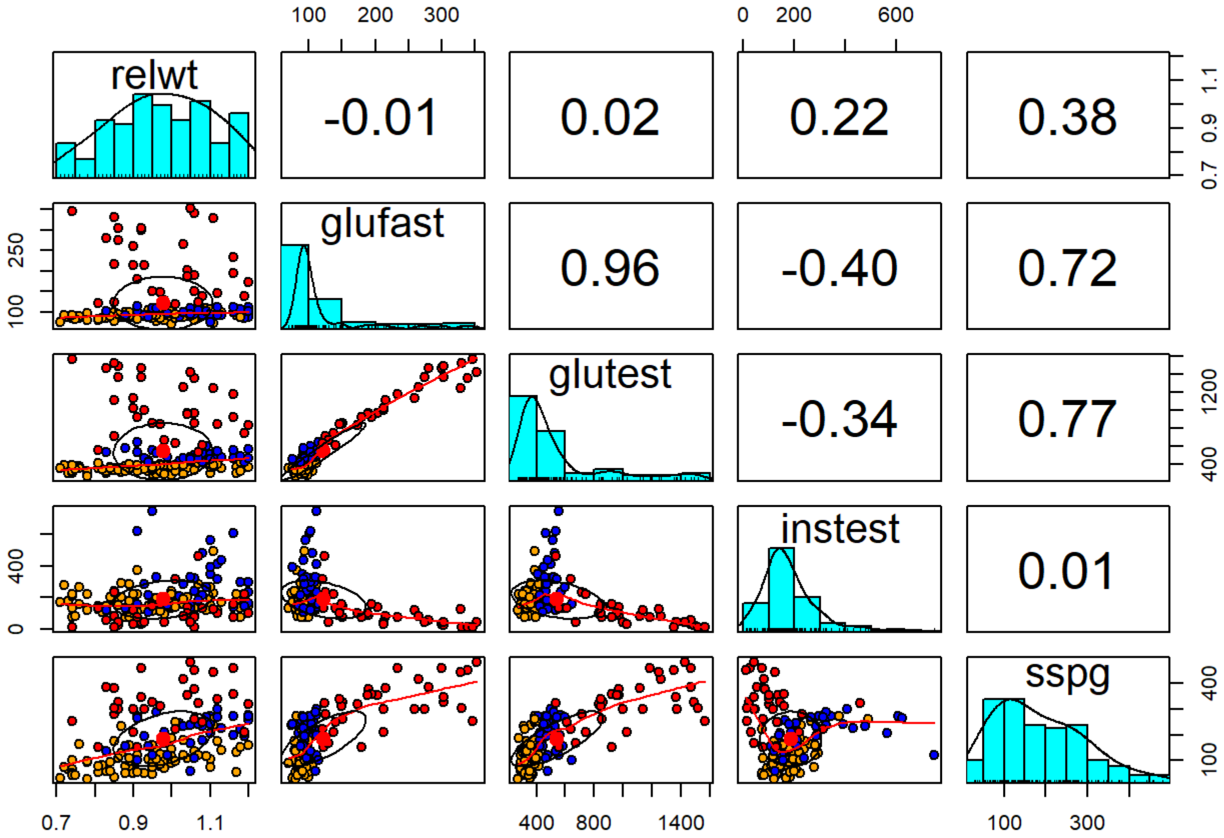
```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

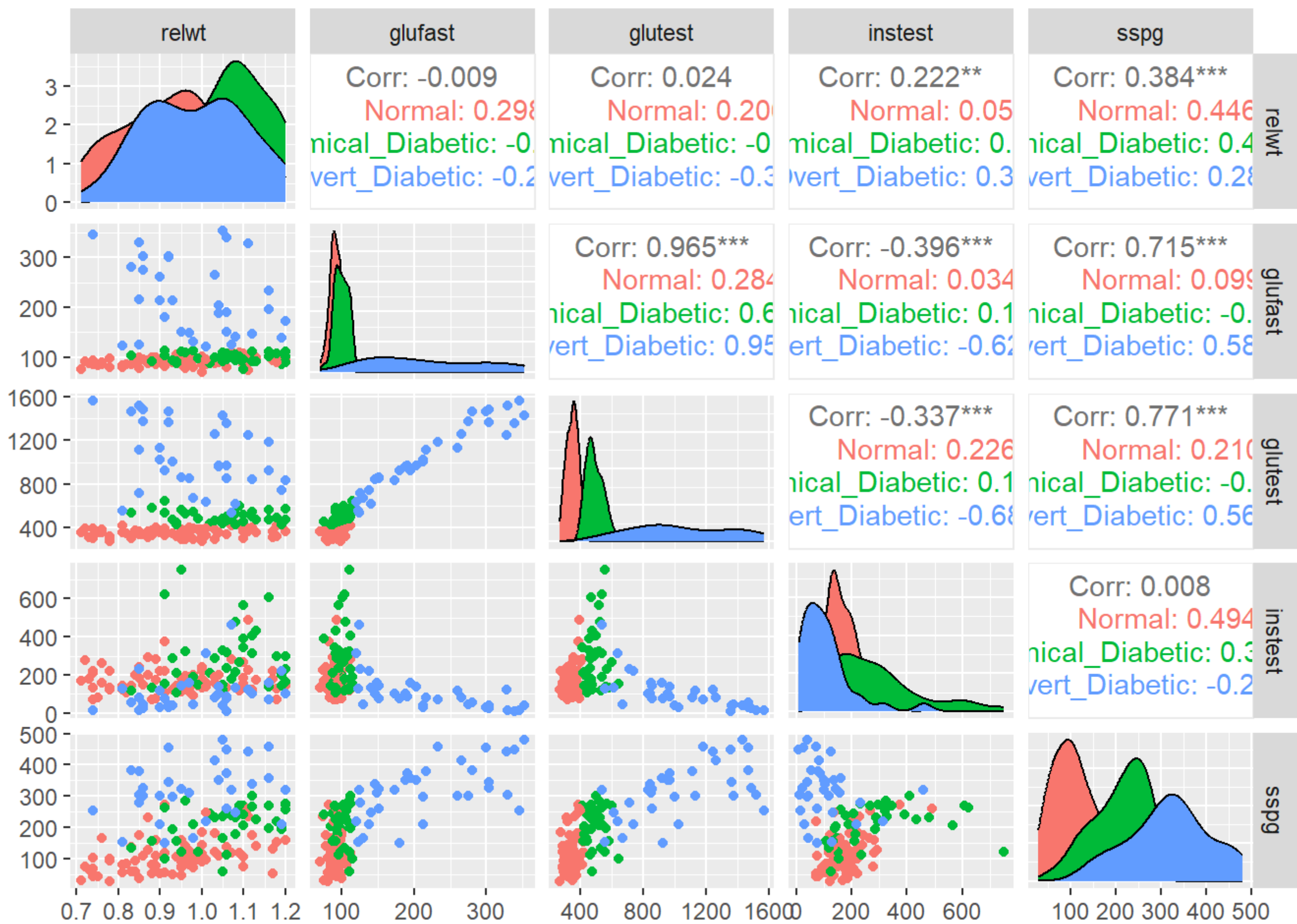
```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
pairs.panels(diabetes[,1:5],
             bg = c("orange", "blue", "red")[diabetes$group],
             pch = 21)
```




```
ggpairs(diabetes[, 1:5], aes(color = diabetes$group))
```



From the above plot it is clear that classes have different covariance matrices. Because, the plot shows the different sizes of clusters or shapes for different classes.

Classes are also multivariate normal(normal distribution can be observed in the above plots).

b) Apply linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). How does the performance of QDA compare to that of LDA in this case?

classes are not multivariate normal; 2 or more variables are to be considered to check for the condition of multivariate normality (-1 mark)

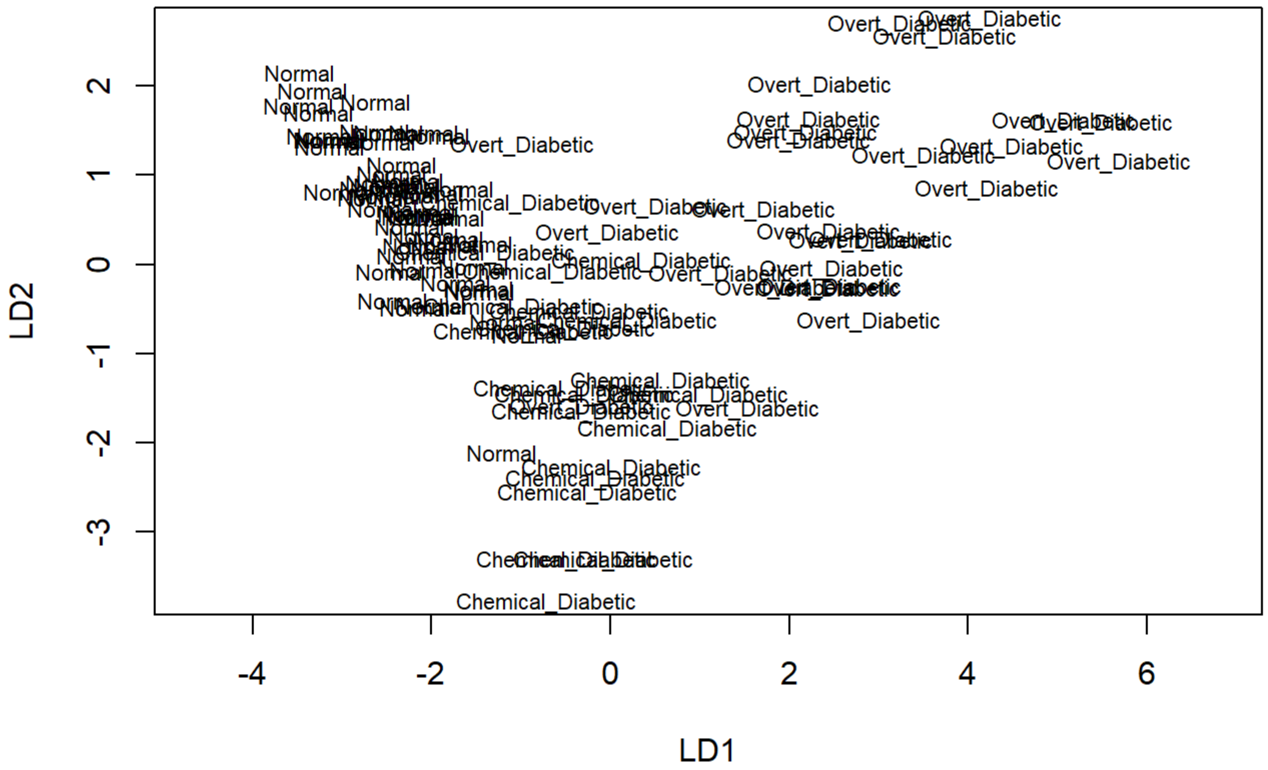
```
### Creating a test and training data

set.seed(123)

indis <- sample(1:nrow(diabetes),size = round(0.7 * nrow(diabetes)))

train_data <- diabetes[indis, ]
test_data <- diabetes[-indis, ]

X_train <- train_data[, -6]
Y_train <- train_data[, 6]

X_test <- test_data[, -6]
Y_test <- test_data[,6]
```

LDA

```
lda.fit <- lda(group ~., data = train_data)
lda.fit
```

```
## Call:
## lda(group ~ ., data = train_data)
##
## Prior probabilities of groups:
##            Normal Chemical_Diabetic    Overt_Diabetic
##         0.5196078         0.2058824         0.2745098
##
## Group means:
##                       relwt  glufast   glutest  instest     sspg
## Normal            0.9364151  91.0000  350.8679 184.3585 124.0566
## Chemical_Diabetic 1.0823810 100.8095  503.1905 323.4286 219.5714
## Overt_Diabetic    0.9775000 214.3571 1029.3214 103.4286 315.6786
##
## Coefficients of linear discriminants:
##                   LD1          LD2
## relwt    1.8137899322 -4.824493227
## glufast -0.0301791937  0.026377497
## glutest  0.0115649319 -0.005711114
## instest -0.0004145498 -0.006564432
## sspg     0.0034478417  0.002125798
##
## Proportion of trace:
##    LD1    LD2
## 0.8603 0.1397
```

```
plot(lda.fit)
```




```
# predictions for the test and training.

test_pred_lda <- predict(lda.fit, newdata = test_data)
names(test_pred_lda)
```

```
## [1] "class"     "posterior" "x"
```

```
test_pred_lda$class
```

```
##  [1] Normal           Normal           Normal           Normal
##  [5] Normal           Normal           Normal           Normal
##  [9] Normal           Normal           Normal           Normal
## [13] Normal           Normal           Normal           Normal
## [17] Normal           Normal           Normal           Normal
## [21] Chemical_Diabetic Chemical_Diabetic Normal           Normal
## [25] Normal           Chemical_Diabetic Normal           Normal
## [29] Chemical_Diabetic Chemical_Diabetic Normal           Chemical_Diabetic
## [33] Chemical_Diabetic Chemical_Diabetic Chemical_Diabetic Normal
## [37] Chemical_Diabetic Normal           Overt_Diabetic    Overt_Diabetic
## [41] Overt_Diabetic    Overt_Diabetic    Chemical_Diabetic
## Levels: Normal Chemical_Diabetic Overt_Diabetic
```

```
train_pred_lda <- predict(lda.fit, newdata = train_data)
train_pred_lda$class
```

```
##   [1] Normal            Normal            Overt_Diabetic    Normal
##   [5] Overt_Diabetic    Overt_Diabetic    Chemical_Diabetic Chemical_Diabetic
##   [9] Overt_Diabetic    Chemical_Diabetic Chemical_Diabetic Normal
##  [13] Chemical_Diabetic Normal            Normal            Normal
##  [17] Overt_Diabetic    Normal            Overt_Diabetic    Normal
##  [21] Normal            Normal            Chemical_Diabetic Chemical_Diabetic
##  [25] Overt_Diabetic    Normal            Normal            Normal
##  [29] Normal            Normal            Normal            Normal
##  [33] Overt_Diabetic    Overt_Diabetic    Overt_Diabetic    Normal
##  [37] Normal            Chemical_Diabetic Normal            Chemical_Diabetic
##  [41] Normal            Normal            Overt_Diabetic    Chemical_Diabetic
##  [45] Normal            Chemical_Diabetic Chemical_Diabetic Chemical_Diabetic
##  [49] Normal            Overt_Diabetic    Normal            Normal
##  [53] Overt_Diabetic    Normal            Overt_Diabetic    Overt_Diabetic
##  [57] Normal            Normal            Chemical_Diabetic Chemical_Diabetic
##  [61] Normal            Normal            Overt_Diabetic    Overt_Diabetic
##  [65] Chemical_Diabetic Normal            Chemical_Diabetic Normal
##  [69] Overt_Diabetic    Normal            Normal            Chemical_Diabetic
##  [73] Normal            Normal            Normal            Normal
##  [77] Normal            Normal            Normal            Normal
##  [81] Overt_Diabetic    Normal            Overt_Diabetic    Overt_Diabetic
##  [85] Normal            Overt_Diabetic    Normal            Normal
##  [89] Overt_Diabetic    Chemical_Diabetic Normal            Normal
##  [93] Normal            Normal            Chemical_Diabetic Normal
##  [97] Normal            Chemical_Diabetic Overt_Diabetic    Normal
## [101] Chemical_Diabetic Normal
## Levels: Normal Chemical_Diabetic Overt_Diabetic
```

```
# compute the error rates

train_error_lda <- (1/length(train_pred_lda$class))*length(which(Y_train != train_pred_lda$cl
ass))
test_error_lda <- (1/length(test_pred_lda$class))*length(which(Y_test != test_pred_lda$clas
s))

train_error_lda
```

```
## [1] 0.09803922
```

```
test_error_lda
```

```
## [1] 0.1395349
```

QDA

```
qda.fit <- qda(group ~., data = train_data)
qda.fit
```

```
## Call:
## qda(group ~ ., data = train_data)
##
## Prior probabilities of groups:
##            Normal Chemical_Diabetic    Overt_Diabetic
##         0.5196078         0.2058824         0.2745098
##
## Group means:
##                       relwt  glufast   glutest  instest     sspg
## Normal            0.9364151  91.0000  350.8679 184.3585 124.0566
## Chemical_Diabetic 1.0823810 100.8095  503.1905 323.4286 219.5714
## Overt_Diabetic    0.9775000 214.3571 1029.3214 103.4286 315.6786
```

```
# predictions for the test and training.

test_pred_qda <- predict(qda.fit, newdata = test_data)
class(test_pred_qda)
```

```
## [1] "list"
```

```
names(test_pred_qda)
```

```
## [1] "class"     "posterior"
```

```
train_pred_qda <- predict(qda.fit, newdata = train_data)
train_pred_qda$class
```

```
##   [1] Normal             Normal             Overt_Diabetic    Normal
##   [5] Overt_Diabetic    Overt_Diabetic    Chemical_Diabetic Chemical_Diabetic
##   [9] Overt_Diabetic    Chemical_Diabetic Chemical_Diabetic Normal
##  [13] Normal             Normal             Normal             Normal
##  [17] Overt_Diabetic    Chemical_Diabetic Overt_Diabetic    Normal
##  [21] Normal             Normal             Chemical_Diabetic Chemical_Diabetic
##  [25] Overt_Diabetic    Normal             Normal             Normal
##  [29] Normal             Normal             Normal             Normal
##  [33] Overt_Diabetic    Overt_Diabetic    Overt_Diabetic    Chemical_Diabetic
##  [37] Normal             Chemical_Diabetic Normal             Chemical_Diabetic
##  [41] Normal             Overt_Diabetic    Overt_Diabetic    Chemical_Diabetic
##  [45] Normal             Chemical_Diabetic Chemical_Diabetic Overt_Diabetic
##  [49] Normal             Overt_Diabetic    Normal             Normal
##  [53] Overt_Diabetic    Normal             Overt_Diabetic    Overt_Diabetic
##  [57] Normal             Normal             Chemical_Diabetic Chemical_Diabetic
##  [61] Normal             Normal             Overt_Diabetic    Overt_Diabetic
##  [65] Overt_Diabetic    Normal             Chemical_Diabetic Chemical_Diabetic
##  [69] Overt_Diabetic    Normal             Normal             Overt_Diabetic
##  [73] Normal             Normal             Normal             Normal
##  [77] Normal             Normal             Normal             Normal
##  [81] Overt_Diabetic    Normal             Overt_Diabetic    Overt_Diabetic
##  [85] Normal             Overt_Diabetic    Normal             Normal
##  [89] Overt_Diabetic    Chemical_Diabetic Normal             Normal
##  [93] Normal             Normal             Chemical_Diabetic Normal
##  [97] Normal             Chemical_Diabetic Overt_Diabetic    Normal
## [101] Chemical_Diabetic Normal
## Levels: Normal Chemical_Diabetic Overt_Diabetic
```

```
# compute the error rates

train_error_qda <- (1/length(train_pred_qda$class))*length(which(Y_train != train_pred_qda$cl
ass))
test_error_qda <- (1/length(test_pred_qda$class))*length(which(Y_test != test_pred_qda$clas
s))
train_error_qda
```

```
## [1] 0.03921569
```

```
test_error_qda
```

```
## [1] 0.09302326
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data <- data.frame(Methods = c("LDA" , "QDA"), Train_Error_Percentage = c(train_error_lda * 1
00,train_error_qda * 100),
                   Test_Error_Percentage = c(test_error_lda * 100,test_error_qda * 100))
arrange(data,Test_Error_Percentage)
```

```
##   Methods Train_Error_Percentage Test_Error_Percentage
## 1     QDA               3.921569              9.302326
## 2     LDA               9.803922             13.953488
```

From the above error rates clearly we can see that QDA is performing better than LDA.

c) Suppose an individual has (glucose test/intolerence = 68, insulin test =122, SSPG = 544. Relative weight = 1.86, fasting plasma glucose = 184). To which class does LDA assign this individual? To which class does QDA?

```
individual <- data.frame(relwt = 1.86, glufast = 184, glutest = 68, instest = 122, sspg = 54
4)
individual
```

```
##   relwt glufast glutest instest sspg
## 1  1.86     184      68     122  544
```

Predicting using LDA model

```
lda_individual <- predict(lda.fit, newdata = individual)
lda_individual$class
```

```
## [1] Normal
## Levels: Normal Chemical_Diabetic Overt_Diabetic
```

Predicting using QDA model

```
qda_individual <- predict(qda.fit, newdata = individual)
qda_individual$class
```

```
## [1] Overt_Diabetic
## Levels: Normal Chemical_Diabetic Overt_Diabetic
```

The LDA Model classifies the individual as “Normal” and the QDA Model classifies the individual as “Overt_Diabetic”

Question 2) The insurance company benchmark data set gives information on customers. Specifically, it contains 86 variables on product-usage data and socio-

demographic data derived from zip area codes. There are 5,822 customers in the training set and another 4,000 in the test set. The data were collected to answer the following questions: Can you predict who will be interested in buying a caravan insurance policy and give an explanation why?

```
library(ISLR2)
```

```
##
## Attaching package: 'ISLR2'
```

```
## The following object is masked from 'package:MASS':
##
##     Boston
```

```
data("Caravan")

setwd("D:/Buffalo/files")

# Loading the data

train_data <- read.table("Caravan_training_ticdata2000.txt")
colnames(train_data) <- colnames(Caravan)
#head(train_data)

X_train <- train_data[,-86]

Y_train <- train_data[ ,86]

X_test <- read.table("Caravan_test_ticeval2000.txt")
colnames(X_test) <- colnames(Caravan[,-86])
#head(X_test)


Y_test <- read.table("Caravan_test_outcome_tictgts2000.txt")
#head(Y_test)
table(Y_test)
```

```
## V1
##    0    1
## 3762  238
```

```
test_data <- data.frame(X_test,Y_test)
colnames(test_data) <- colnames(Caravan)
#head(test_data)
```

```
dim(train_data)
```

```
## [1] 5822   86
```

```
dim(test_data)
```

```
## [1] 4000   86
```

a) Develop a model using the linear model.

```
lm.fit <- lm(Purchase ~. , train_data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Purchase ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67293 -0.08720 -0.04593 -0.00639  1.04628
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.7685381  0.4298406   1.788 0.073835 .
## MOSTYPE      0.0035209  0.0022512   1.564 0.117866
## MAANTHUI    -0.0072642  0.0076739  -0.947 0.343875
## MGEMOMV     -0.0012739  0.0071737  -0.178 0.859055
## MGEMLEEF     0.0107473  0.0049596   2.167 0.030279 *
## MOSHOOFD    -0.0154869  0.0101044  -1.533 0.125405
## MGODRK      -0.0056016  0.0056016  -1.000 0.317353
## MGODPR      -0.0002069  0.0060664  -0.034 0.972795
## MGODOV       0.0003569  0.0054592   0.065 0.947874
## MGODGE      -0.0030237  0.0058038  -0.521 0.602399
## MRELGE       0.0086829  0.0075479   1.150 0.250036
## MRELSA       0.0020367  0.0072008   0.283 0.777310
## MRELOV       0.0055682  0.0076295   0.730 0.465526
## MFALLEEN    -0.0038250  0.0065474  -0.584 0.559107
## MFGEKIND    -0.0050625  0.0066861  -0.757 0.448980
## MFWEKIND    -0.0026253  0.0069795  -0.376 0.706824
## MOPLHOOG     0.0021357  0.0068161   0.313 0.754038
## MOPLMIDD    -0.0048456  0.0071396  -0.679 0.497358
## MOPLLAAG    -0.0113977  0.0073004  -1.561 0.118525
## MBERHOOG     0.0021884  0.0045182   0.484 0.628153
## MBERZELF    -0.0004665  0.0052201  -0.089 0.928796
## MBERBOER    -0.0050974  0.0050426  -1.011 0.312122
## MBERMIDD     0.0041254  0.0044806   0.921 0.357228
## MBERARBG    -0.0006060  0.0044709  -0.136 0.892190
## MBERARBO     0.0019733  0.0044532   0.443 0.657690
## MSKA        -0.0013674  0.0051653  -0.265 0.791225
## MSKB1       -0.0031701  0.0050198  -0.632 0.527724
## MSKB2       -0.0012603  0.0044827  -0.281 0.778603
## MSKC         0.0024879  0.0049115   0.507 0.612502
## MSKD        -0.0008866  0.0047145  -0.188 0.850832
## MHHUUR      -0.0454201  0.0376622  -1.206 0.227872
## MHKOOP      -0.0432242  0.0376290  -1.149 0.250730
## MAUT1        0.0085964  0.0075592   1.137 0.255502
## MAUT2        0.0077871  0.0068554   1.136 0.256038
## MAUT0        0.0047215  0.0072646   0.650 0.515762
## MZFONDS     -0.0561024  0.0444643  -1.262 0.207094
## MZPART      -0.0593733  0.0443897  -1.338 0.181097
## MINKM30      0.0070879  0.0051150   1.386 0.165884
## MINK3045     0.0069414  0.0049276   1.409 0.158986
## MINK4575     0.0049679  0.0050144   0.991 0.321862
## MINK7512     0.0059267  0.0052728   1.124 0.261053
## MINK123M    -0.0098939  0.0069270  -1.428 0.153258
## MINKGEM      0.0063044  0.0045645   1.381 0.167277
## MKOOPKLA     0.0029097  0.0022664   1.284 0.199250
## PWAPART      0.0284931  0.0166017   1.716 0.086166 .
```

```
## PWABEDR     -0.0101533  0.0205121  -0.495 0.620625    
## PWALAND     -0.0201220  0.0390424  -0.515 0.606301    
## PPERSAUT     0.0102787  0.0026346   3.901 9.67e-05 ***
## PBESAUT      0.0014405  0.0148574   0.097 0.922765    
## PMOTSCO     -0.0061279  0.0079415  -0.772 0.440364    
## PVRAAUT     -0.0249190  0.0415892  -0.599 0.549083    
## PAANHANG     0.0588044  0.0557610   1.055 0.291662    
## PTRACTOR     0.0121481  0.0142358   0.853 0.393504    
## PWERKT      -0.0062440  0.0370186  -0.169 0.866060    
## PBROM        0.0078683  0.0152793   0.515 0.606598    
## PLEVEN      -0.0155397  0.0064753  -2.400 0.016433 *  
## PPERSONG     0.0098926  0.0335157   0.295 0.767880    
## PGEZONG      0.1937254  0.0793370   2.442 0.014644 *  
## PWAOREG      0.0647933  0.0256913   2.522 0.011696 *  
## PBRAND       0.0132643  0.0035906   3.694 0.000223 ***
## PZEILPL     -0.1917507  0.1439848  -1.332 0.182998    
## PPLEZIER    -0.0299076  0.0269224  -1.111 0.266666    
## PFIETS      -0.0107777  0.0549693  -0.196 0.844564    
## PINBOED     -0.0441620  0.0307404  -1.437 0.150883    
## PBYSTAND    -0.0184858  0.0288890  -0.640 0.522269    
## AWAPART     -0.0377952  0.0323794  -1.167 0.243154    
## AWABEDR      0.0185448  0.0529740   0.350 0.726296    
## AWALAND      0.0180904  0.1374585   0.132 0.895300    
## APERSAUT     0.0002821  0.0127496   0.022 0.982347    
## ABESAUT     -0.0214816  0.0652955  -0.329 0.742175    
## AMOTSCO      0.0203252  0.0310683   0.654 0.513004    
## AVRAAUT      0.0563675  0.1589388   0.355 0.722866    
## AAANHANG    -0.0804238  0.0944352  -0.852 0.394455    
## ATRACTOR    -0.0395651  0.0353795  -1.118 0.263484    
## AWERKT      -0.0010526  0.0728240  -0.014 0.988468    
## ABROM       -0.0236462  0.0467611  -0.506 0.613101    
## ALEVEN       0.0372344  0.0154024   2.417 0.015661 *  
## APERSONG    -0.0464279  0.0954471  -0.486 0.626684    
## AGEZONG     -0.4050642  0.1898715  -2.133 0.032938 *  
## AWAOREG     -0.2304561  0.1243310  -1.854 0.063852 .  
## ABRAND      -0.0211374  0.0116048  -1.821 0.068593 .  
## AZEILPL      0.4958051  0.2815591   1.761 0.078304 .  
## APLEZIER     0.3633887  0.0885318   4.105 4.11e-05 ***
## AFIETS       0.0416061  0.0408644   1.018 0.308650    
## AINBOED      0.0959436  0.0699079   1.372 0.169983    
## ABYSTAND     0.1312250  0.0983836   1.334 0.182319    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.23 on 5736 degrees of freedom
## Multiple R-squared:  0.0729, Adjusted R-squared:  0.05916 
## F-statistic: 5.306 on 85 and 5736 DF,  p-value: < 2.2e-16
```

```
train_predict_lm <- predict(lm.fit, newdata = X_train)
test_predict_lm <- predict(lm.fit, newdata = X_test)

head(train_predict_lm)
```

```
##          1          2          3          4          5          6
## 0.09738541 0.01345938 0.08354523 0.09075754 0.04307400 0.01475749
```

```
head(test_predict_lm)
```

```
##           1           2           3           4           5           6
## 0.014441132 0.215946829 0.099937482 0.095439888 0.005945841 0.027520016
```

```
# since, the target or response is a qualitative variable with 2 classes(0 & 1) we will conve
rt the outcomes to 0 & 1

train_predict_lm <- ifelse(train_predict_lm > 0.5, 1, 0)
test_predict_lm <- ifelse(test_predict_lm > 0.5, 1, 0)

train_error_lm <- mean(Y_train != train_predict_lm)

test_error_lm <- mean(Y_test != test_predict_lm)

train_error_lm
```

```
## [1] 0.05960151
```

```
test_error_lm
```

```
## [1] 0.05975
```

```
table(Y_test[,],test_predict_lm)
```

```
##    test_predict_lm
##        0    1
##   0 3760    2
##   1  237    1
```

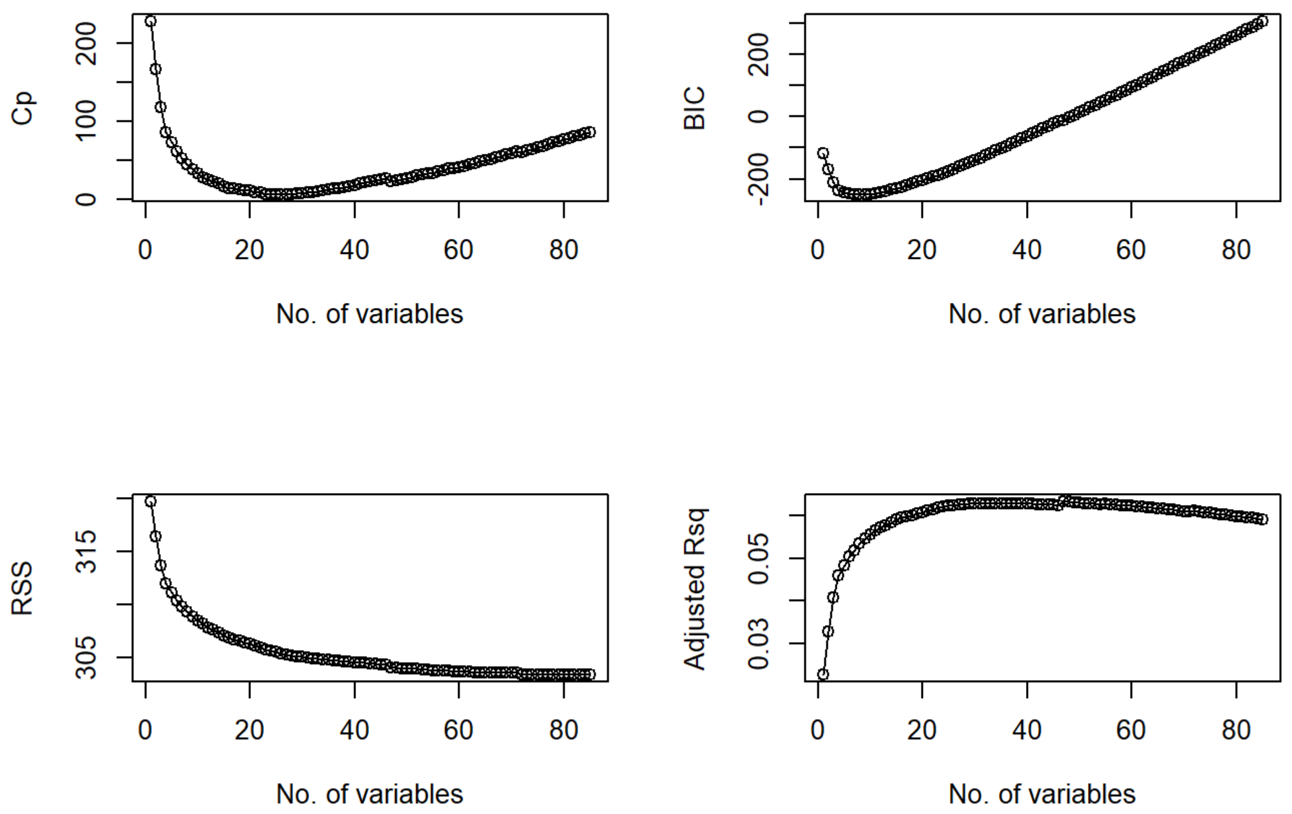
b) Develop a model using Forwards Selection, Backwards Selection, Lasso regression, and Ridge regression.

Forward subset selection:

```
library(leaps)

# Performing forward subset selection on the data

regfit.fwd <- regsubsets(Purchase ~., data = train_data, nbest = 1, nvmax = 85, method = "for
ward")

my_sum_fwd <- summary(regfit.fwd)
names(my_sum_fwd)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

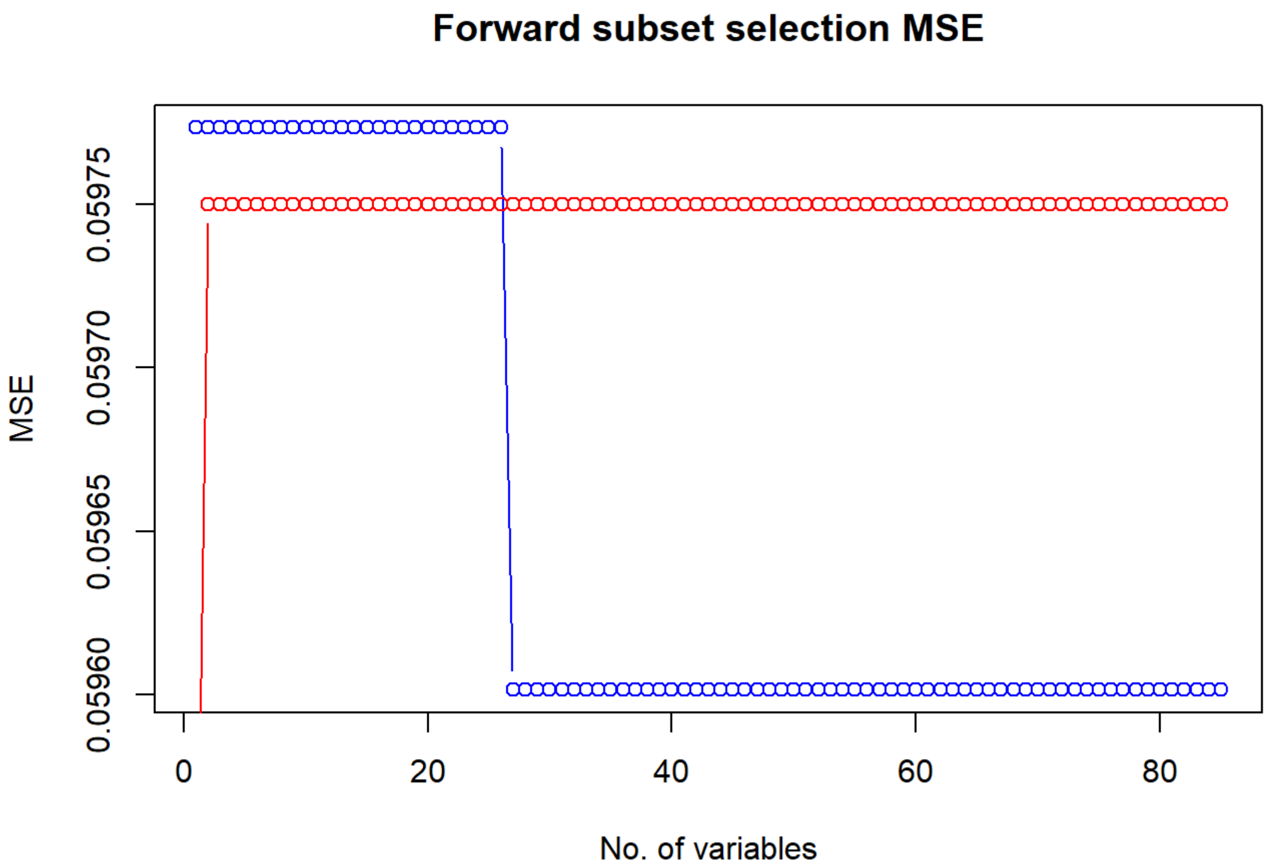
```
# examine the best "p" variables models

### my_sum_fwd$outmat

# plot model selection measures

par(mfrow = c(2,2))
plot(my_sum_fwd$cp, xlab = "No. of variables", ylab = "Cp", type = "o")
plot(my_sum_fwd$bic, xlab = "No. of variables", ylab = "BIC", type = "o")
plot(my_sum_fwd$rss, xlab = "No. of variables", ylab = "RSS", type = "o")
plot(my_sum_fwd$adjr2, xlab = "No. of variables", ylab = "Adjusted Rsq", type = "o")
```




```
# identify the optimal models using model selection measures for forward subset selection

data <- data.frame(Parameters = c("CP","BIC","RSS","Adj Rsquare"), Values = c(which.min(my_su
m_fwd$cp), which.min(my_sum_fwd$bic), which.min(my_sum_fwd$rss), which.max(my_sum_fwd$adjr
2)))
data
```

```
##    Parameters Values
## 1          CP     23
## 2         BIC      8
## 3         RSS     85
## 4 Adj Rsquare     47
```

```r
### Predicting training and test errors

predict.regsubsets = function(object, newdata, id){
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object,id=id)
    xvars=names(coefi)
    mat[,xvars]%*%coefi
}

# create objects to store error.

train_err_store1 <- matrix(rep(NA, 85))
test_err_store1 <- matrix(rep(NA, 85))

for (i in 1:85){
    # make the predictions
    y_hat_train1 <- predict(regfit.fwd, newdata = train_data, id = i)
    y_hat_test1 <- predict(regfit.fwd, newdata = test_data, id = i)

    # converting data to 0 & 1

    y_hat_train1 <- ifelse(y_hat_train1 > 0.5, 1, 0)
    y_hat_test1 <- ifelse(y_hat_test1 > 0.5, 1, 0)

    # compare the prediction with the true
    train_err_store1[i] <- mean(Y_train != y_hat_train1)
    test_err_store1[i] <- mean(Y_test != y_hat_test1)
}

par(mfrow = c(1,1))
plot(train_err_store1, col = "blue", type = "b", xlab = "No. of variables", ylab = "MSE",main
="Forward subset selection MSE")
lines(test_err_store1, col = "red", type = "b")
```

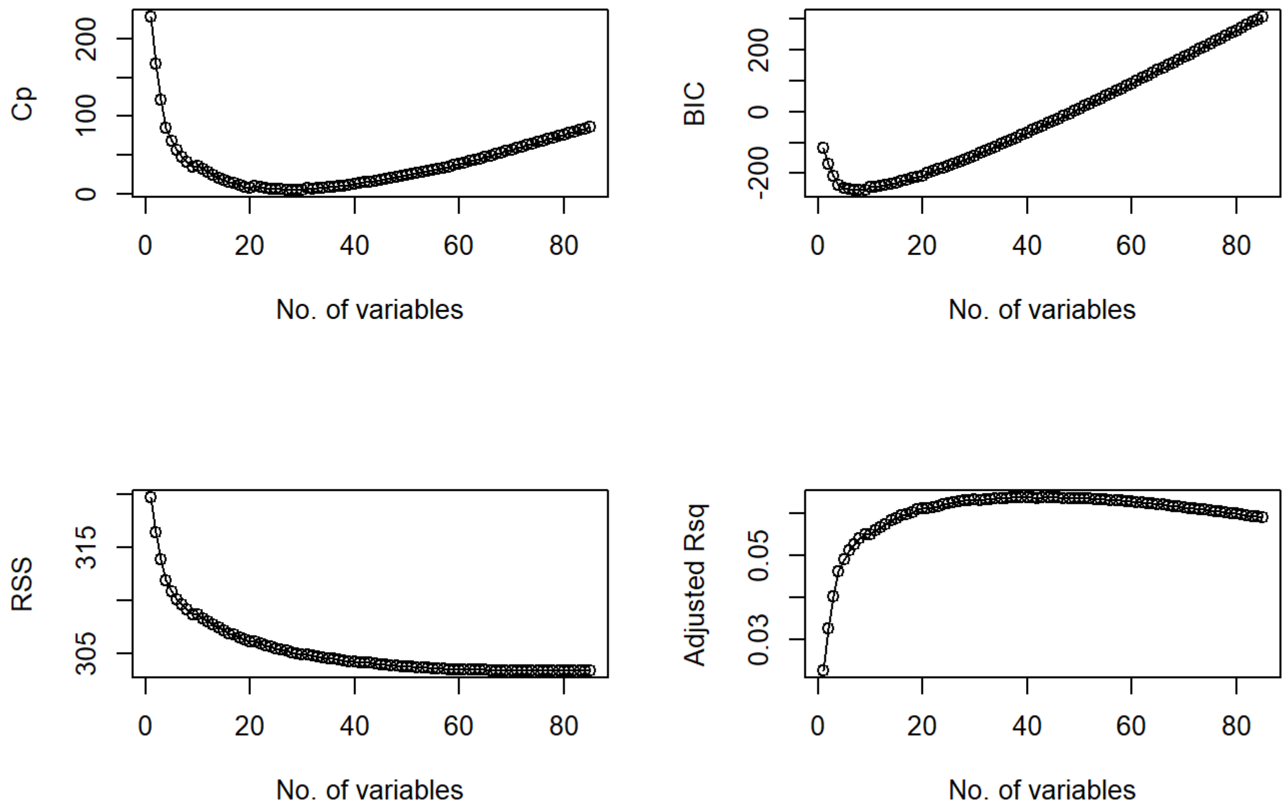


Backward subset selection:

```
regfit.bwd <- regsubsets(Purchase ~., data = train_data, nbest = 1, nvmax = 85, method = "bac
kward")

my_sum_bwd <- summary(regfit.bwd)
names(my_sum_bwd)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

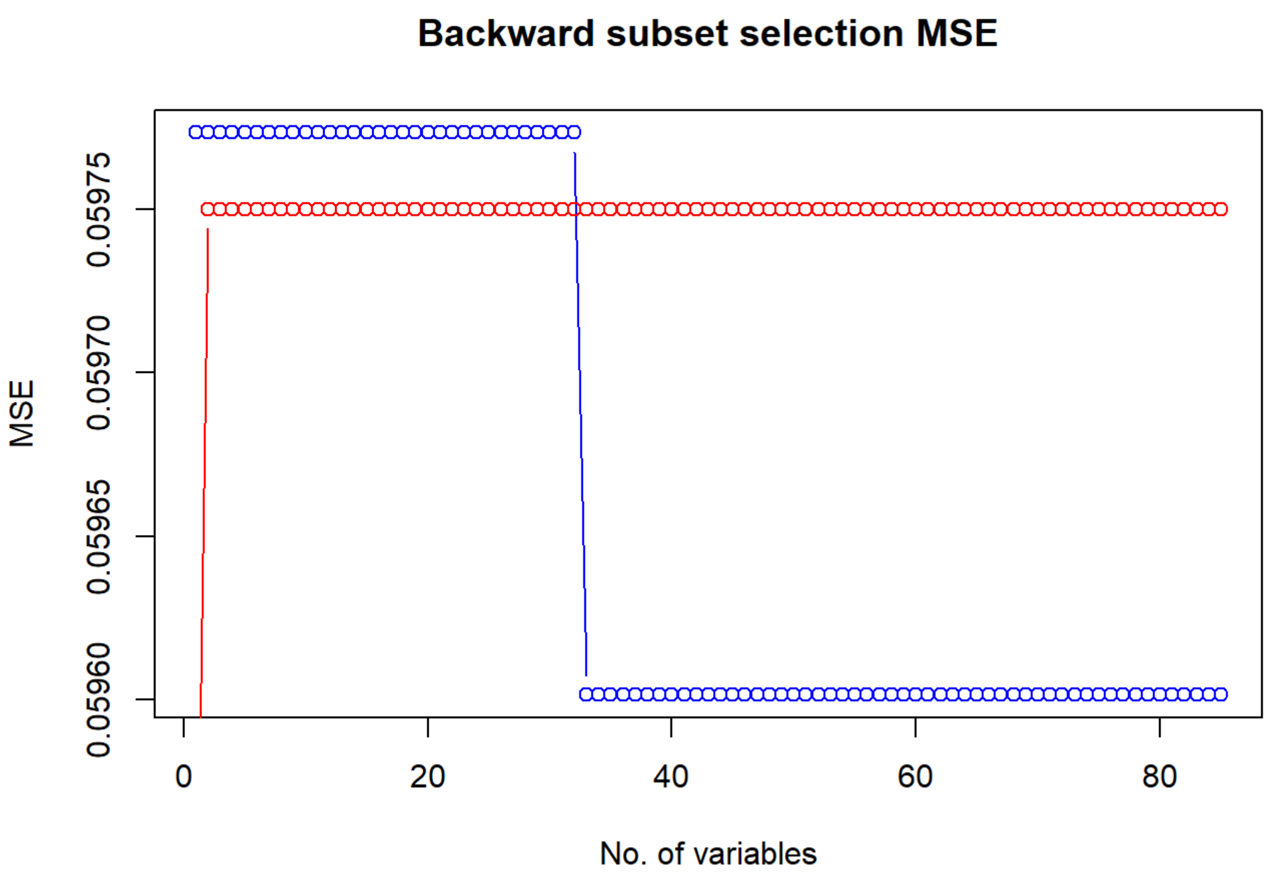
```
# examine the best "p" variables models

#### my_sum_bwd$outmat

# plot model selection measures

par(mfrow = c(2,2))
plot(my_sum_bwd$cp, xlab = "No. of variables", ylab = "Cp", type = "o")
plot(my_sum_bwd$bic, xlab = "No. of variables", ylab = "BIC", type = "o")
plot(my_sum_bwd$rss, xlab = "No. of variables", ylab = "RSS", type = "o")
plot(my_sum_bwd$adjr2, xlab = "No. of variables", ylab = "Adjusted Rsq", type = "o")
```




```
# identify the optimal models using model selection measures for backward subset selection

data <- data.frame(Parameters = c("CP","BIC","RSS","Adj Rsquare"), Values = c(which.min(my_su
m_bwd$cp), which.min(my_sum_bwd$bic), which.min(my_sum_bwd$rss), which.max(my_sum_bwd$adjr
2)))
data
```

```
##    Parameters Values
## 1          CP     29
## 2         BIC      8
## 3         RSS     85
## 4 Adj Rsquare     39
```

```
### Predicting training and test errors

predict.regsubsets = function(object, newdata, id){
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object,id=id)
    xvars=names(coefi)
    mat[,xvars]%*%coefi
}

# create objects to store error.

train_err_store2 <- matrix(rep(NA, 85))
test_err_store2 <- matrix(rep(NA, 85))

for (i in 1:85){
    # make the predictions
    y_hat_train2 <- predict(regfit.bwd, newdata = train_data, id = i)
    y_hat_test2 <- predict(regfit.bwd, newdata = test_data, id = i)

    # converting data to 0 & 1

    y_hat_train2 <- ifelse(y_hat_train2 > 0.5, 1, 0)
    y_hat_test2 <- ifelse(y_hat_test2 > 0.5, 1, 0)

    # compare the prediction with the true
    train_err_store2[i] <- mean(Y_train != y_hat_train2)
    test_err_store2[i] <- mean(Y_test != y_hat_test2)
}

par(mfrow = c(1,1))
plot(train_err_store2, col = "blue", type = "b", xlab = "No. of variables", ylab = "MSE",main
="Backward subset selection MSE")
lines(test_err_store2, col = "red", type = "b")
```



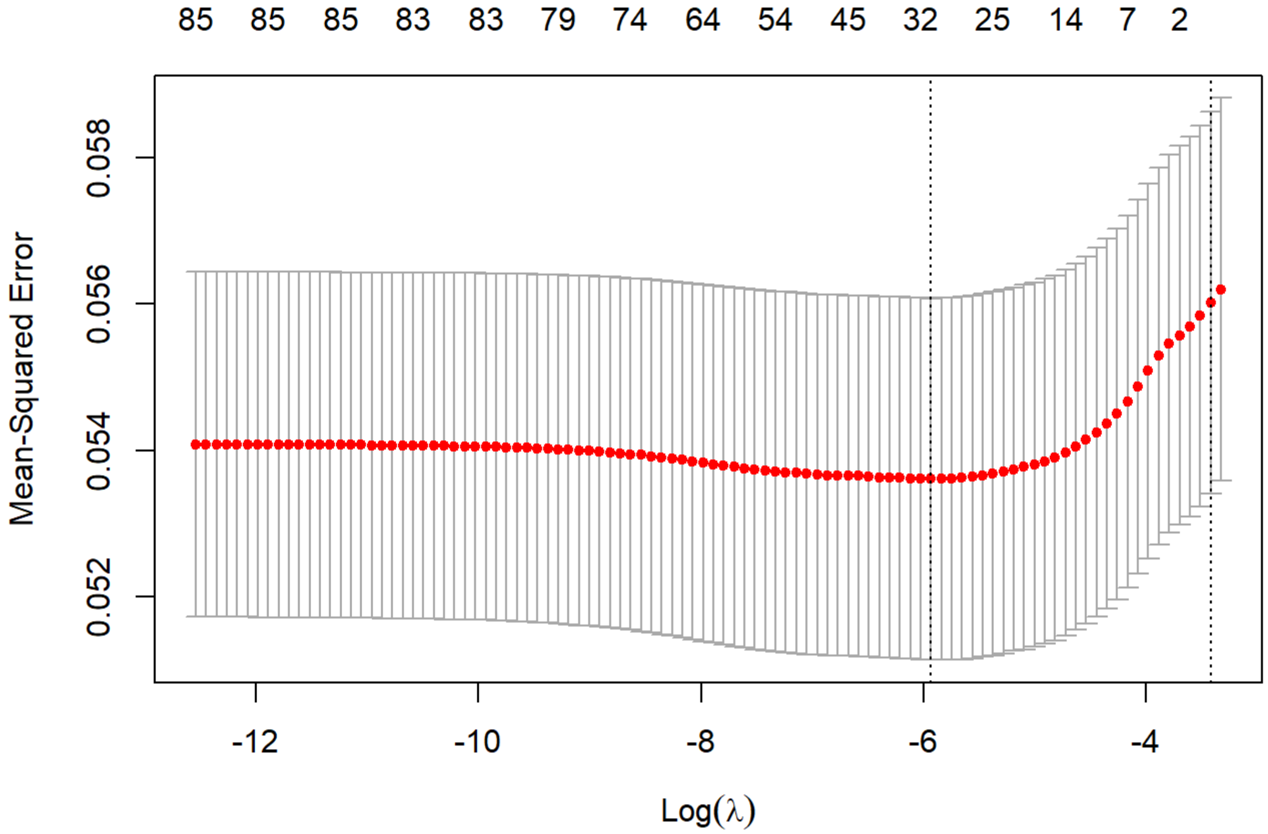
Lasso Regression:

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-8
```

```
set.seed(123)

lasso_model <- cv.glmnet(x = as.matrix(X_train), y = Y_train, alpha = 1, nfolds = 10)
plot(lasso_model)
```




```
best_lambda <- lasso_model$lambda.min
best_lambda
```

```
## [1] 0.002644076
```

```
# predicting the response of the variables using best_lambda on training and test data

train_predict_lr <- predict(lasso_model, s = best_lambda, newx = as.matrix(X_train), type =
'response')
train_predict_lr <- ifelse(train_predict_lr > 0.5, 1, 0)

test_predict_lr <- predict(lasso_model, s = best_lambda, newx = as.matrix(X_test), type = 're
sponse')
test_predict_lr <- ifelse(test_predict_lr > 0.5, 1, 0)

# Calculating train and test errors:
train_error_lr <- mean(Y_train != train_predict_lr)
test_error_lr <- mean(Y_test != test_predict_lr)

train_error_lr
```

```
## [1] 0.05977327
```

```
test_error_lr
```

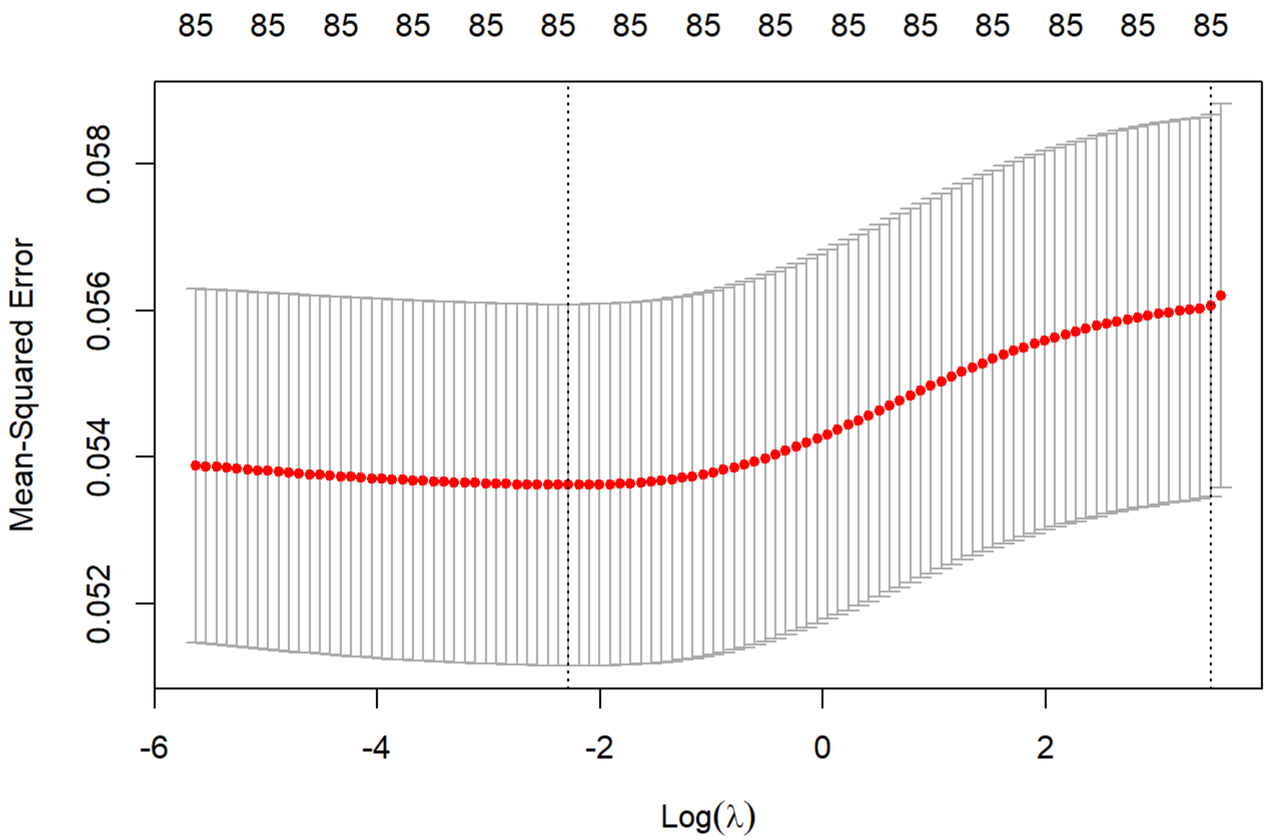
```
## [1] 0.05975
```

```
table(Y_test[,],test_predict_lr)
```

```
##    test_predict_lr
##        0    1
##   0 3760    2
##   1  237    1
```

Ridge Regression:

```
library(glmnet)

set.seed(123)

ridge_model <- cv.glmnet(x = as.matrix(X_train), y = Y_train, alpha = 0, nfolds = 10)
plot(ridge_model)
```




```
best_lambda <- ridge_model$lambda.min
best_lambda
```

```
## [1] 0.1018902
```

```r
# predicting the response of the variables using best_lambda on training and test data

train_predict_rr <- predict(ridge_model, s = best_lambda, newx = as.matrix(X_train), type =
'response')
train_predict_rr <- ifelse(train_predict_lr > 0.5, 1, 0)

test_predict_rr <- predict(ridge_model, s = best_lambda, newx = as.matrix(X_test), type = 're
sponse')
test_predict_rr <- ifelse(test_predict_lr > 0.5, 1, 0)

# Calculating train and test errors:
train_error_rr <- mean(Y_train != train_predict_rr)
test_error_rr <- mean(Y_test != test_predict_rr)

train_error_rr
```

```
## [1] 0.05977327
```

```r
test_error_rr
```

```
## [1] 0.05975
```

```r
table(Y_test[,], test_predict_rr)
```

```
##    test_predict_rr
##        0    1
##   0 3760    2
##   1  237    1
```

c) Develop a model using logistic regression.

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
set.seed(123)
glm.fit <- glm(Purchase ~. , data = train_data, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Purchase ~ ., family = "binomial", data = train_data)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.542e+02  1.116e+04   0.023  0.98183
## MOSTYPE      6.580e-02  4.624e-02   1.423  0.15468
## MAANTHUI    -1.832e-01  1.927e-01  -0.951  0.34157
## MGEMOMV     -2.696e-02  1.399e-01  -0.193  0.84723
## MGEMLEEF     2.096e-01  1.016e-01   2.063  0.03911 *
## MOSHOOFD    -2.767e-01  2.076e-01  -1.333  0.18247
## MGODRK      -1.142e-01  1.069e-01  -1.068  0.28535
## MGODPR      -1.910e-02  1.177e-01  -0.162  0.87112
## MGODOV      -1.618e-02  1.055e-01  -0.153  0.87818
## MGODGE      -6.817e-02  1.113e-01  -0.612  0.54024
## MRELGE       2.310e-01  1.566e-01   1.475  0.14031
## MRELSA       8.509e-02  1.466e-01   0.580  0.56169
## MRELOV       1.467e-01  1.562e-01   0.939  0.34759
## MFALLEEN    -8.291e-02  1.311e-01  -0.633  0.52702
## MFGEKIND    -1.154e-01  1.337e-01  -0.863  0.38813
## MFWEKIND    -8.140e-02  1.417e-01  -0.575  0.56561
## MOPLHOOG     9.717e-04  1.311e-01   0.007  0.99408
## MOPLMIDD    -9.077e-02  1.365e-01  -0.665  0.50605
## MOPLLAAG    -1.994e-01  1.376e-01  -1.449  0.14740
## MBERHOOG     8.883e-02  9.349e-02   0.950  0.34204
## MBERZELF     3.918e-02  9.897e-02   0.396  0.69219
## MBERBOER    -1.169e-01  1.104e-01  -1.059  0.28951
## MBERMIDD     1.353e-01  9.191e-02   1.472  0.14106
## MBERARBG     3.976e-02  9.067e-02   0.438  0.66104
## MBERARBO     9.954e-02  9.143e-02   1.089  0.27628
## MSKA         2.690e-02  1.035e-01   0.260  0.79502
## MSKB1       -8.801e-03  1.011e-01  -0.087  0.93064
## MSKB2        1.200e-02  9.081e-02   0.132  0.89485
## MSKC         9.016e-02  9.958e-02   0.905  0.36527
## MSKD        -2.468e-02  9.724e-02  -0.254  0.79967
## MHHUUR      -1.472e+01  8.140e+02  -0.018  0.98557
## MHKOOP      -1.469e+01  8.140e+02  -0.018  0.98561
## MAUT1        1.819e-01  1.514e-01   1.202  0.22953
## MAUT2        1.507e-01  1.371e-01   1.099  0.27162
## MAUT0        9.325e-02  1.436e-01   0.649  0.51603
## MZFONDS     -1.445e+01  9.359e+02  -0.015  0.98768
## MZPART      -1.451e+01  9.359e+02  -0.016  0.98763
## MINKM30      1.181e-01  1.006e-01   1.174  0.24039
## MINK3045     1.366e-01  9.650e-02   1.415  0.15694
## MINK4575     1.009e-01  9.667e-02   1.043  0.29678
## MINK7512     1.144e-01  1.027e-01   1.114  0.26513
## MINK123M    -1.607e-01  1.449e-01  -1.109  0.26738
## MINKGEM      9.214e-02  9.945e-02   0.927  0.35417
## MKOOPKLA     6.856e-02  4.642e-02   1.477  0.13966
## PWAPART      5.954e-01  3.901e-01   1.526  0.12693
## PWABEDR     -2.757e-01  4.635e-01  -0.595  0.55196
## PWALAND     -4.405e-01  1.035e+00  -0.425  0.67052
## PPERSAUT     2.306e-01  4.199e-02   5.491 4.01e-08 ***
## PBESAUT      1.215e+01  4.029e+02   0.030  0.97595
```

```
## PMOTSCO     -8.101e-02  1.147e-01  -0.706  0.48006   
## PVRAAUT     -2.106e+00  2.557e+03  -0.001  0.99934   
## PAANHANG     1.014e+00  9.371e-01   1.082  0.27917   
## PTRACTOR     7.229e-01  4.278e-01   1.690  0.09107 . 
## PWERKT      -5.525e+00  4.805e+03  -0.001  0.99908   
## PBROM        2.170e-01  4.865e-01   0.446  0.65559   
## PLEVEN      -2.382e-01  1.170e-01  -2.036  0.04173 * 
## PPERSONG    -4.523e-01  2.094e+00  -0.216  0.82901   
## PGEZONG      1.444e+00  1.029e+00   1.404  0.16033   
## PWAOREG      8.239e-01  5.943e-01   1.386  0.16565   
## PBRAND       2.401e-01  7.714e-02   3.113  0.00185 **
## PZEILPL     -8.658e+00  3.261e+03  -0.003  0.99788   
## PPLEZIER    -1.886e-01  3.259e-01  -0.579  0.56289   
## PFIETS       3.664e-01  8.325e-01   0.440  0.65985   
## PINBOED     -1.068e+00  8.764e-01  -1.219  0.22301   
## PBYSTAND    -1.676e-01  3.321e-01  -0.505  0.61373   
## AWAPART     -9.293e-01  7.802e-01  -1.191  0.23364   
## AWABEDR      4.197e-01  1.082e+00   0.388  0.69824   
## AWALAND      2.762e-01  3.528e+00   0.078  0.93758   
## APERSAUT    -3.902e-02  1.772e-01  -0.220  0.82566   
## ABESAUT     -7.298e+01  2.417e+03  -0.030  0.97591   
## AMOTSCO      2.418e-01  3.772e-01   0.641  0.52142   
## AVRAAUT     -4.490e+00  1.078e+04   0.000  0.99967   
## AAANHANG    -1.351e+00  1.687e+00  -0.801  0.42322   
## ATRACTOR    -2.376e+00  1.524e+00  -1.559  0.11899   
## AWERKT      -8.749e-01  9.682e+03   0.000  0.99993   
## ABROM       -1.060e+00  1.549e+00  -0.684  0.49367   
## ALEVEN       4.789e-01  2.245e-01   2.133  0.03291 * 
## APERSONG     3.997e-01  4.329e+00   0.092  0.92644   
## AGEZONG     -3.163e+00  2.706e+00  -1.169  0.24247   
## AWAOREG     -3.212e+00  3.433e+00  -0.936  0.34939   
## ABRAND      -4.118e-01  2.787e-01  -1.477  0.13956   
## AZEILPL      1.047e+01  3.261e+03   0.003  0.99744   
## APLEZIER     2.516e+00  1.010e+00   2.490  0.01276 * 
## AFIETS       2.318e-01  5.699e-01   0.407  0.68420   
## AINBOED      1.947e+00  1.412e+00   1.378  0.16812   
## ABYSTAND     1.078e+00  1.103e+00   0.977  0.32870   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 2635.5  on 5821  degrees of freedom
## Residual deviance: 2243.5  on 5736  degrees of freedom
## AIC: 2415.5
## 
## Number of Fisher Scoring iterations: 17
```

```
names(glm.fit)
```

```
##  [1] "coefficients"      "residuals"         "fitted.values"
##  [4] "effects"           "R"                 "rank"
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"
## [13] "iter"              "weights"           "prior.weights"
## [16] "df.residual"       "df.null"           "y"
## [19] "converged"         "boundary"          "model"
## [22] "call"              "formula"           "terms"
## [25] "data"              "offset"            "control"
## [28] "method"            "contrasts"         "xlevels"
```

```
# Predictions

glm.probs.train <- predict(glm.fit, newdata = train_data, type = "response")
y_hat_train <- ifelse(glm.probs.train > 0.5, 1, 0)

glm.probs.test <- predict(glm.fit, newdata = test_data, type = "response")
y_hat_test <- ifelse(glm.probs.test > 0.5, 1, 0)

# Calculate the error rates

train_err <- mean(y_hat_train != Y_train)
train_err
```

```
## [1] 0.05994504
```

```
test_err <- mean(y_hat_test != Y_test)
test_err
```

```
## [1] 0.06025
```

```
# Confusion matrix

conf <- confusionMatrix(as.factor(y_hat_test),as.factor(Y_test[ , ]))
conf$table
```

```
##           Reference
## Prediction    0    1
##          0 3756  235
##          1    6    3
```

The MSE of all the methods forward subset selection,backward subset selection, Lasso Regression, Ridge Regression and logistic regression are very close.

Lasso and Ridge have similar test accuracy.

And our models didn’t performed well in predicting Yes(1’s), The reason might be number of 1’s are very less in the given data.

Question 3) In this exercise, we will predict the number of applications received using the other variables in the College data set.

```
library(caret)
library(ISLR2)
library(glmnet)
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
data("College")
dim(College)  ## 777 * 18
```

```
## [1] 777  18
```

```
head(College)
```

```
##                              Private Apps Accept Enroll Top10perc Top25perc
## Abilene Christian University     Yes 1660   1232    721        23        52
## Adelphi University               Yes 2186   1924    512        16        29
## Adrian College                   Yes 1428   1097    336        22        50
## Agnes Scott College              Yes  417    349    137        60        89
## Alaska Pacific University        Yes  193    146     55        16        44
## Albertson College                Yes  587    479    158        38        62
##                              F.Undergrad P.Undergrad Outstate Room.Board Books
## Abilene Christian University        2885         537     7440       3300   450
## Adelphi University                  2683        1227    12280       6450   750
## Adrian College                      1036          99    11250       3750   400
## Agnes Scott College                  510          63    12960       5450   450
## Alaska Pacific University            249         869     7560       4120   800
## Albertson College                    678          41    13500       3335   500
##                              Personal PhD Terminal S.F.Ratio perc.alumni Expend
## Abilene Christian University     2200  70       78      18.1          12   7041
## Adelphi University               1500  29       30      12.2          16  10527
## Adrian College                   1165  53       66      12.9          30   8735
## Agnes Scott College               875  92       97       7.7          37  19016
## Alaska Pacific University        1500  76       72      11.9           2  10922
## Albertson College                 675  67       73       9.4          11   9727
##                              Grad.Rate
## Abilene Christian University        60
## Adelphi University                  56
## Adrian College                      54
## Agnes Scott College                 59
## Alaska Pacific University           15
## Albertson College                   55
```

```
### Private variable is a factor changing it to 1(Yes) and 0(No)
College$Private <- ifelse(College$Private == 'Yes', 1 , 0)
```

a) Split the data set into a training set and a test set.

```
set.seed(123)
train_indis <- sample(c(1:length(College[,1])), size = round(2/3*length(College[,1])), replac
e = FALSE)

train_data <- College[train_indis, ]
test_data <- College[-train_indis, ]

X_train <- train_data[, -2]
Y_train <- train_data[, 2]

X_test <- test_data[, -2]
Y_test <- test_data[, 2]
```

b) Fit a linear model using least squares on the training set, and report the test error obtained.

```
set.seed(123)
lm.fit <- lm(Apps ~. , data = train_data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3098.1  -435.7   -32.6   326.9  6524.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -320.63000  483.82540  -0.663 0.507830
## Private     -631.06608  166.38884  -3.793 0.000167 ***
## Accept         1.22765    0.05907  20.782  < 2e-16 ***
## Enroll         0.07342    0.22242   0.330 0.741483
## Top10perc     45.28449    6.30692   7.180 2.54e-12 ***
## Top25perc    -12.88783    5.12008  -2.517 0.012144 *
## F.Undergrad    0.02496    0.04024   0.620 0.535386
## P.Undergrad    0.03394    0.03505   0.968 0.333304
## Outstate      -0.06350    0.02155  -2.947 0.003361 **
## Room.Board     0.20100    0.05392   3.728 0.000215 ***
## Books          0.16346    0.27890   0.586 0.558084
## Personal      -0.03987    0.07418  -0.537 0.591204
## PhD           -6.76818    5.36695  -1.261 0.207866
## Terminal      -5.29390    5.82889  -0.908 0.364201
## S.F.Ratio     -0.13458   14.77294  -0.009 0.992735
## perc.alumni   -7.16431    4.68079  -1.531 0.126506
## Expend         0.08032    0.01338   6.005 3.69e-09 ***
## Grad.Rate      9.82319    3.37117   2.914 0.003730 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 980.1 on 500 degrees of freedom
## Multiple R-squared:  0.918,  Adjusted R-squared:  0.9153
## F-statistic: 329.5 on 17 and 500 DF,  p-value: < 2.2e-16
```

```
ols_predict <- predict(lm.fit, newdata = test_data)
ols_mse <- mean((Y_test - ols_predict)^2)
ols_mse
```

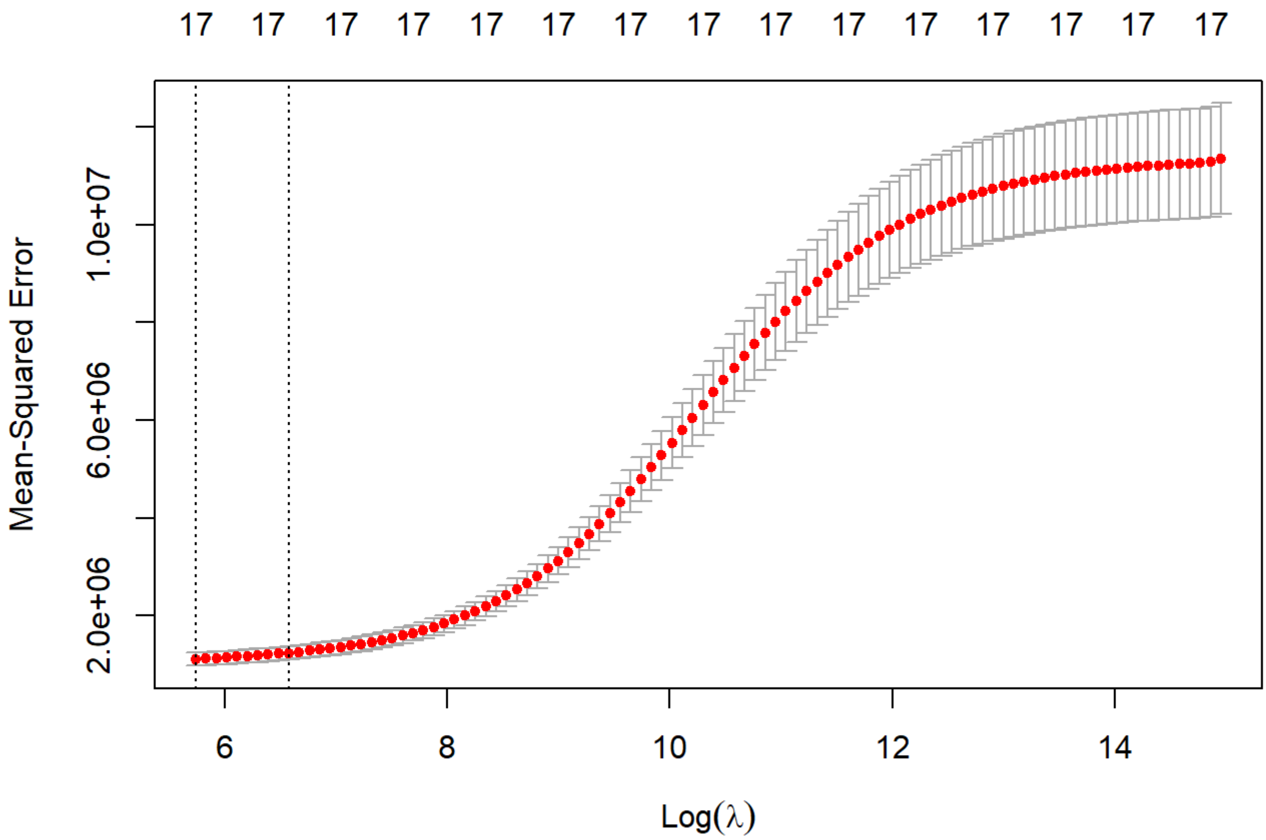
```
## [1] 1684049
```

c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
set.seed(123)
ridge_cv <- cv.glmnet(as.matrix(X_train), Y_train, alpha = 0,nfolds = 10)
sel <- ridge_cv$lambda.min
sel
```

```
## [1] 311.779
```

```
plot(ridge_cv)
```




```
ridge_predict <- predict(ridge_cv, s = sel, newx= as.matrix(X_test))
ridge_mse <- (1/length(Y_test))*(sum((ridge_predict - Y_test)^2))
ridge_mse
```

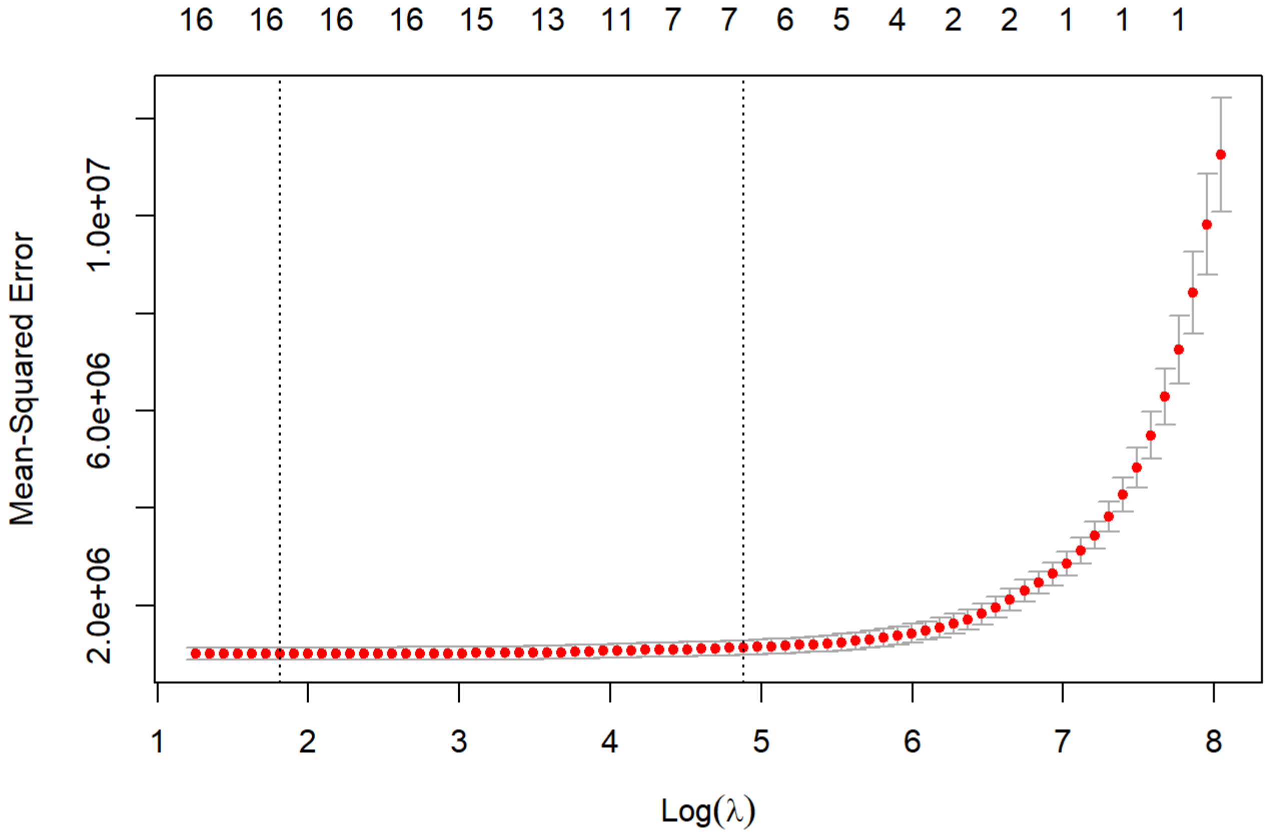
```
## [1] 2787195
```

d) Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
set.seed(123)
lasso_cv <- cv.glmnet(as.matrix(X_train), Y_train, alpha = 1, nfolds = 10)
sel <- lasso_cv$lambda.min
sel
```

```
## [1] 6.120348
```

```
plot(lasso_cv)
```




```
lasso_coef <- predict(lasso_cv, s = sel,type = "coefficients")

lasso_predict <- predict(lasso_cv, s = sel, newx= as.matrix(X_test))
lasso_mse <- (1/length(Y_test))*(sum((lasso_predict - Y_test)^2))
lasso_mse
```

```
## [1] 1685197
```

```
lasso_coef
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -407.16052726
## Private     -612.48955542
## Accept         1.22519081
## Enroll         0.07885531
## Top10perc     41.58235489
## Top25perc     -9.81499052
## F.Undergrad    0.02513096
## P.Undergrad    0.02776503
## Outstate      -0.05735665
## Room.Board     0.18884357
## Books          0.11837682
## Personal      -0.02333079
## PhD           -5.99235815
## Terminal      -5.16787774
## S.F.Ratio      .
## perc.alumni   -6.79379825
## Expend         0.07920785
## Grad.Rate      8.84826047
```

```
number_of_nonzero_coefficients <- sum(lasso_coef[-1] != 0)  ## Exclude intercept
number_of_nonzero_coefficients
```

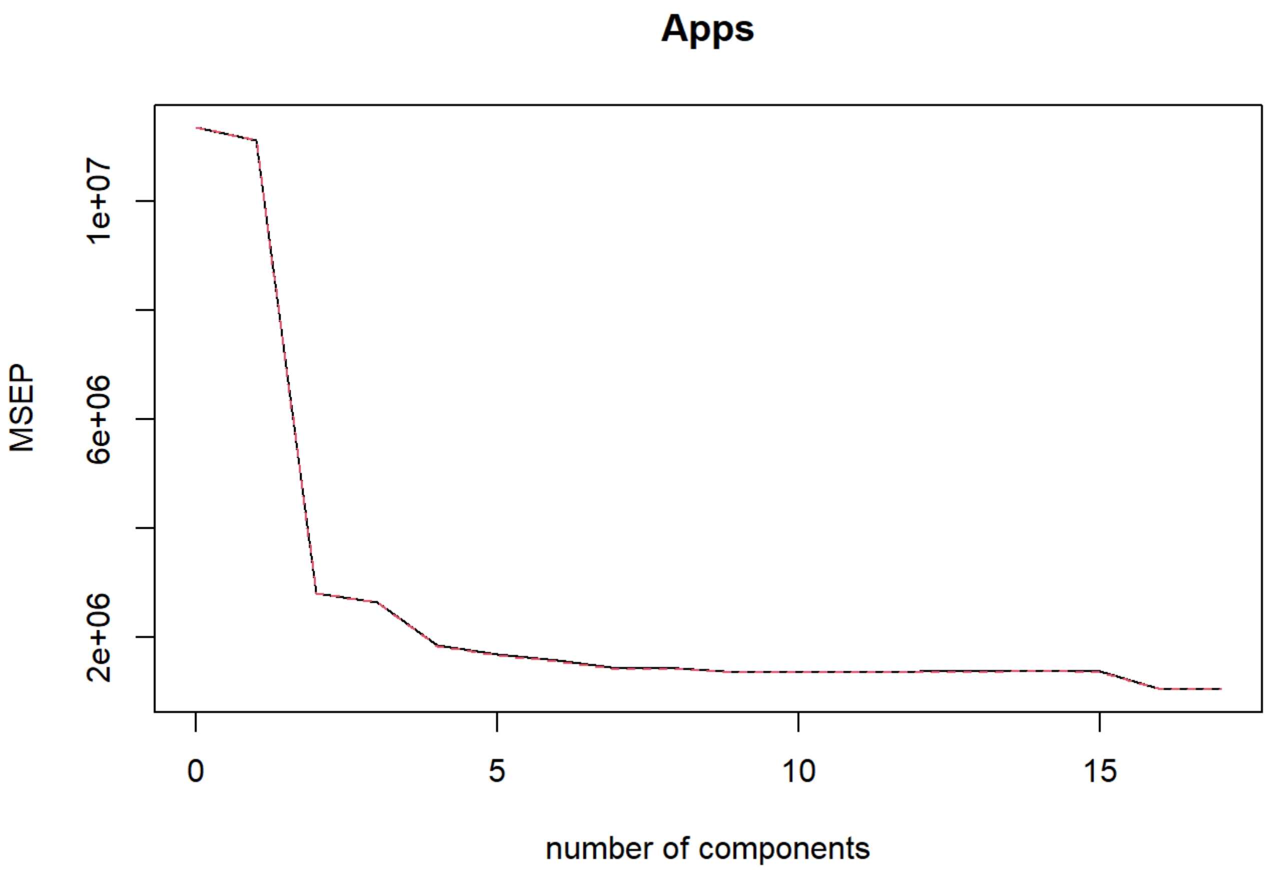
```
## [1] 16
```

e) Fit a PCR model on the training set, with M chosen by cross validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
set.seed(123)
pcr.fit <- pcr(Apps ~ ., data = train_data , scale = TRUE , validation = "CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 518 17
##  Y dimension: 518 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3370     3336     1680     1631     1363     1303     1257
## adjCV         3370     3336     1678     1630     1357     1299     1253
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1202     1201     1169      1169      1168      1174      1176
## adjCV     1195     1196     1166      1167      1165      1171      1173
##        14 comps  15 comps  16 comps  17 comps
## CV         1176      1176      1029      1029
## adjCV      1173      1173      1025      1025
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      31.765    57.84    64.68    70.19    75.49    80.39    84.01    87.40
## Apps    3.386    75.80    77.45    84.75    86.02    86.91    88.03    88.22
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.57     93.02     95.07     96.93     98.02     98.88     99.40
## Apps    88.84     88.89     88.94     88.98     89.03     89.03     89.23
##       16 comps  17 comps
## X        99.82     100.0
## Apps     91.74      91.8
```

```
validationplot (pcr.fit , val.type = "MSEP")
```




```
### From the graph we can clearly see that cross validation selected M = P = 16

### Evaluating the test MSE

pcr_predict <- predict(pcr.fit,X_test, ncomp=16)
pcr_mse <- mean((pcr_predict - Y_test)^2)
pcr_mse
```

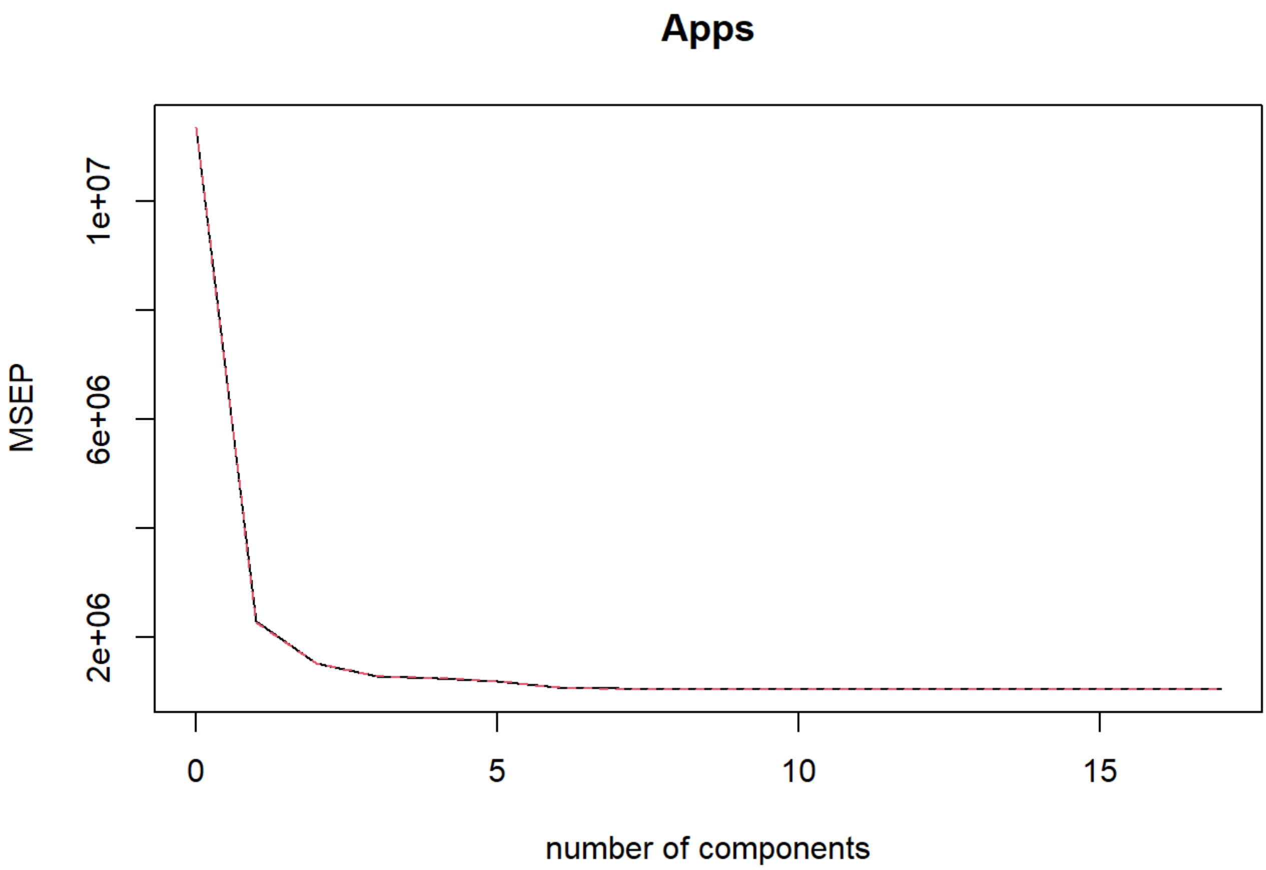
```
## [1] 1785303
```

f) Fit a PLS model on the training set, with M chosen by cross validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
set.seed(123)
pls.fit <- plsr(Apps ~ ., data = train_data , scale = TRUE , validation = "CV")
summary (pls.fit)
```

```
## Data:    X dimension: 518 17
##  Y dimension: 518 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3370     1513     1233     1138     1121     1099     1045
## adjCV         3370     1511     1236     1136     1117     1092     1040
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1031     1028     1029      1030      1027      1028      1028
## adjCV     1027     1025     1026      1026      1024      1025      1025
##        14 comps  15 comps  16 comps  17 comps
## CV          1029      1029      1029      1029
## adjCV       1025      1025      1025      1025
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       26.30    42.01    63.26    67.75    71.41    74.08    77.53    80.83
## Apps    80.53    86.92    89.34    90.16    91.05    91.71    91.77    91.79
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       83.35     86.14     89.53     91.21     93.22     94.67     97.06
## Apps    91.79     91.80     91.80     91.80     91.80     91.80     91.80
##       16 comps  17 comps
## X        99.11     100.0
## Apps     91.80      91.8
```

```
validationplot (pls.fit , val.type = "MSEP")
```




```
### Cross-validation selected M = 11 as the number of principal components to minimize the ou
t-of-sample MSE.
pls_predict <- predict(pls.fit,X_test,ncomp=11)
pls_mse <- mean((pls_predict - Y_test)^2)
pls_mse
```

```
## [1] 1703326
```

g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

$$R^2 = 1 - \frac{\text{Residual sum of squares (RSS)}}{\text{total sum of squares (TSS)}},$$
$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}.$$

```r
library(dplyr)

TSS <- sum((Y_test - mean(Y_test))^2)

data <- data.frame(method = c("OLS", "Ridge", "Lasso", "PCR", "PLS"),
           test_MSE = c(ols_mse, ridge_mse, lasso_mse, pcr_mse, pls_mse),
           test_R2 = c(1 - sum((Y_test - ols_predict)^2) / TSS,
                       1 - sum((Y_test - ridge_predict)^2) / TSS,
                       1 - sum((Y_test - lasso_predict)^2) / TSS,
                       1 - sum((Y_test - pcr_predict)^2) / TSS,
                       1 - sum((Y_test - pls_predict)^2) / TSS))
arrange(data,desc(test_R2))
```

```
##   method test_MSE   test_R2
## 1    OLS  1684049 0.9240638
## 2  Lasso  1685197 0.9240120
## 3    PLS  1703326 0.9231946
## 4    PCR  1785303 0.9194981
## 5  Ridge  2787195 0.8743213
```

Each of the models performed well, with test R2 values above 0.91 except Ridge Regression. There is really a minimal performance difference between 4 methods.

The only model that could potentially have a meaningful performance difference was ridge regression (which performed the worst).

Question 4) We will now try to predict per capita crime rate in the Boston data set.

```
library(leaps)
library(glmnet)
```

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

```
## Loaded glmnet 4.1-8
```

```
library(MASS)
data("Boston")
dim(Boston)   # 506 * 14
```

```
## [1] 506  14
```

```
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
summary(Boston)
```

```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00  
```

(a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

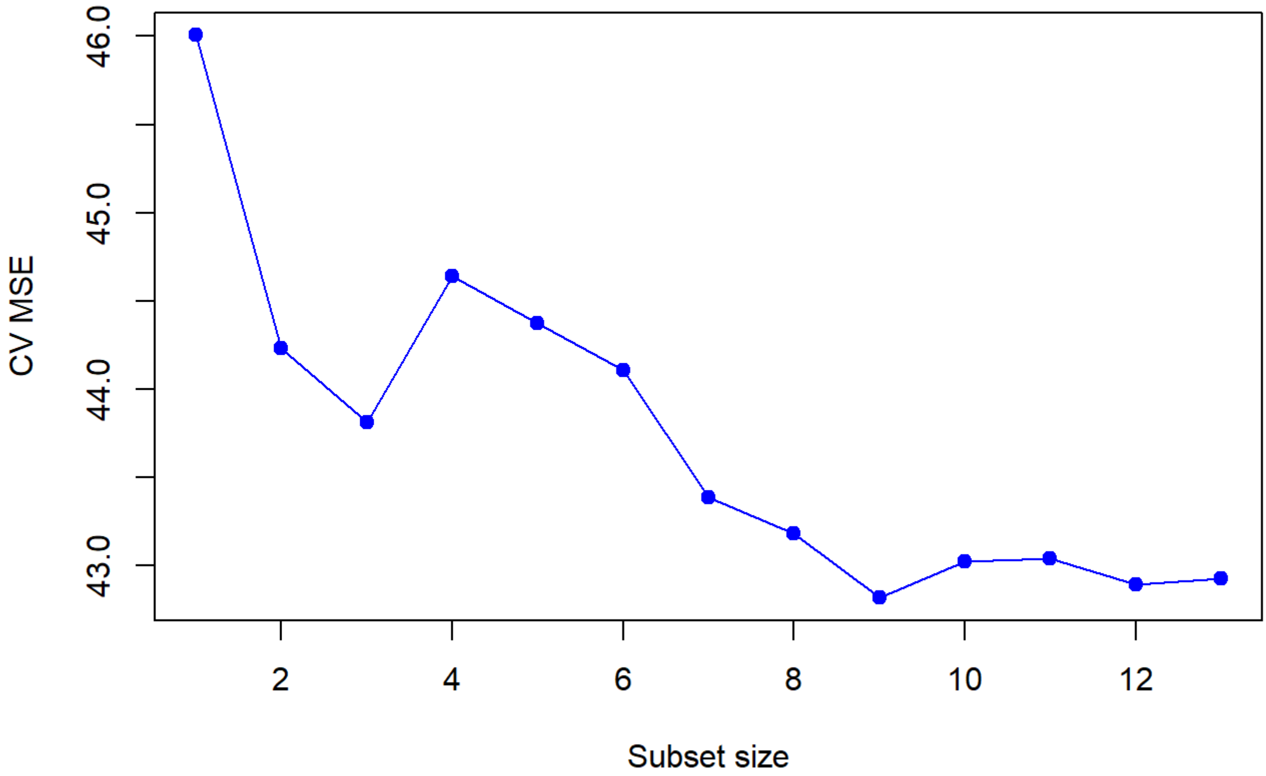
```r
###  best subset selection:

predict.regsubsets <- function(object, newdata, id){
    form <- as.formula(object$call[[2]])
    mat <- model.matrix(form, newdata)
    coefi <- coef(object,id=id)
    xvars <- names(coefi)
    mat[,xvars] %*% coefi
}

## For K=10 folds

k <- 10
n <- nrow (Boston)
set.seed (1)
folds <- sample(rep(1:k, length = n))
cv.errors <- matrix (NA, k, 13,  dimnames = list(NULL, paste(1:13)))

for (i in 1:k) {
  best.fit <- regsubsets(crim ~ ., data = Boston[folds != i, ], nvmax = 13)

  for (j in 1:13) {
    pred <- predict(best.fit , Boston[folds == i, ], id = j)
    cv.errors[i, j] <- mean((Boston$crim[folds == i] - pred)^2)
  }
}

mse.cv <- apply (cv.errors , 2, mean)
mse.cv
```

```
##        1        2        3        4        5        6        7        8
## 46.00617 44.22854 43.80757 44.63674 44.37501 44.10329 43.38296 43.18012
##        9       10       11       12       13
## 42.81453 43.01895 43.03912 42.88730 42.92625
```

```r
paste("CV MSE: " , mse.cv[which.min(mse.cv)])
```

```
## [1] "CV MSE:  42.8145280588506"
```

```r
par (mfrow = c(1, 1))
plot (mse.cv , type = "o",pch = 19, col = 'blue', xlab = 'Subset size', ylab = 'CV MSE')
```

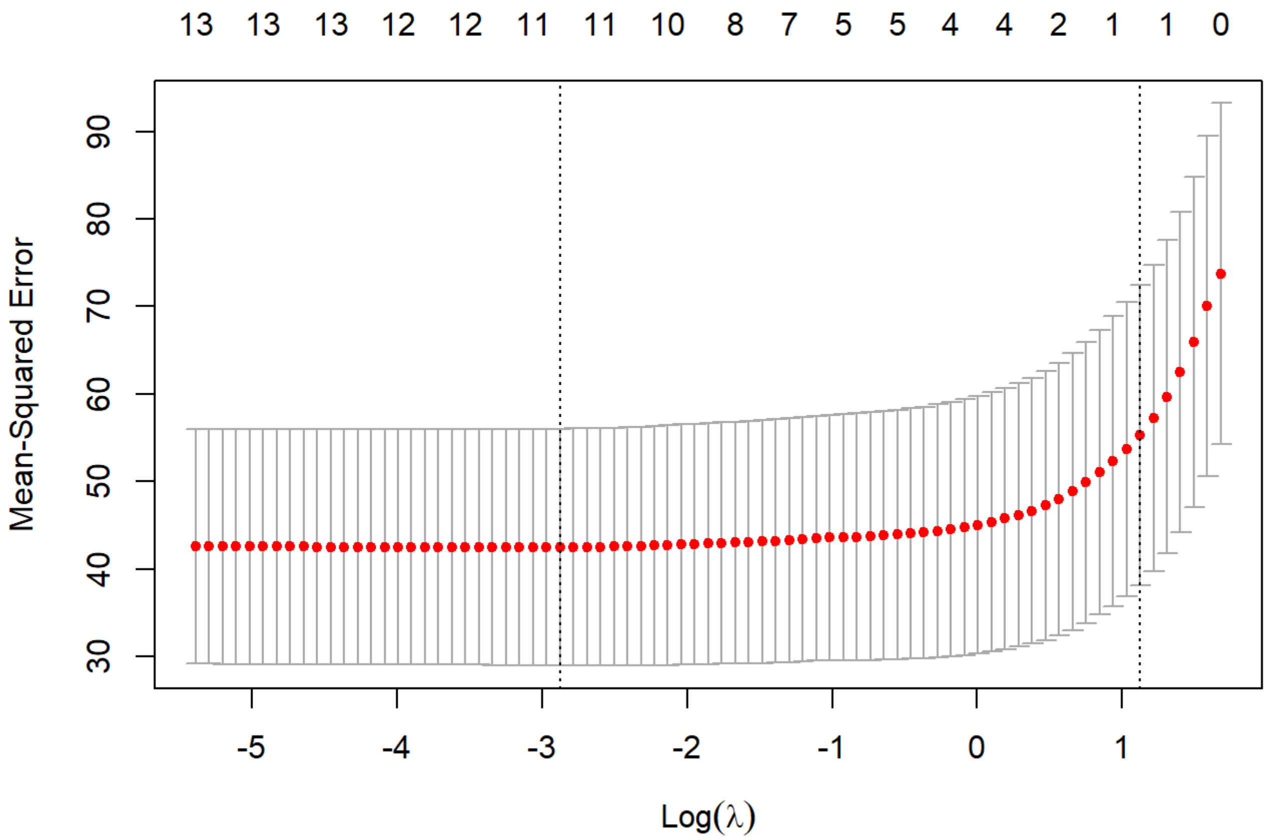



```
which.min(mse.cv)
```

```
## 9
## 9
```

```
### Best subset model selecting 9 parameters or variables.
```

```
### Lasso regression:

set.seed(1)

x <- model.matrix(crim ~ . - 1, data = Boston)
y <- Boston$crim
cv.lasso <- cv.glmnet(x, y, alpha = 1, type.measure = "mse")
plot(cv.lasso)
```



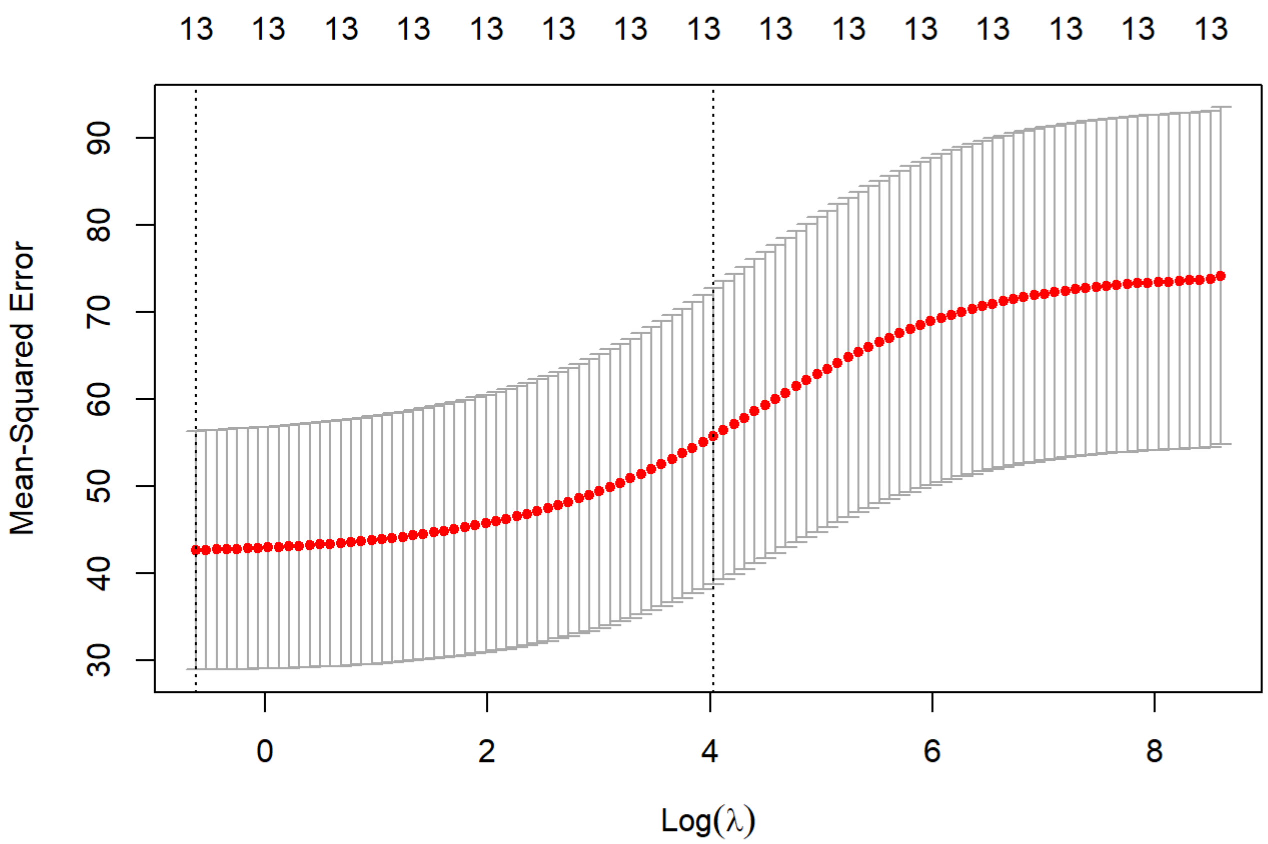

```
coef(cv.lasso)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 1.0894283
## zn          .
## indus       .
## chas        .
## nox         .
## rm          .
## age         .
## dis         .
## rad         0.2643196
## tax         .
## ptratio     .
## black       .
## lstat       .
## medv        .
```

```
cv.lasso$cvm[cv.lasso$lambda == cv.lasso$lambda.1se]
```

```
## [1] 55.3338
```

```
### Ridge regression:

set.seed(1)

x <- model.matrix(crim ~ . - 1, data = Boston)
y <- Boston$crim
cv.ridge <- cv.glmnet(x, y, alpha = 0, type.measure = "mse")
plot(cv.ridge)
```




```
coef(cv.ridge)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  1.017516864
## zn          -0.002805664
## indus        0.034405928
## chas        -0.225250602
## nox          2.249887499
## rm          -0.162546004
## age          0.007343331
## dis         -0.114928730
## rad          0.059813844
## tax          0.002659110
## ptratio      0.086423005
## black       -0.003342067
## lstat        0.044495213
## medv        -0.029124577
```

```
cv.ridge$cvm[cv.ridge$lambda == cv.ridge$lambda.1se]
```

```
## [1] 55.80448
```

```
### Principal Component Regression

library(pls)
```

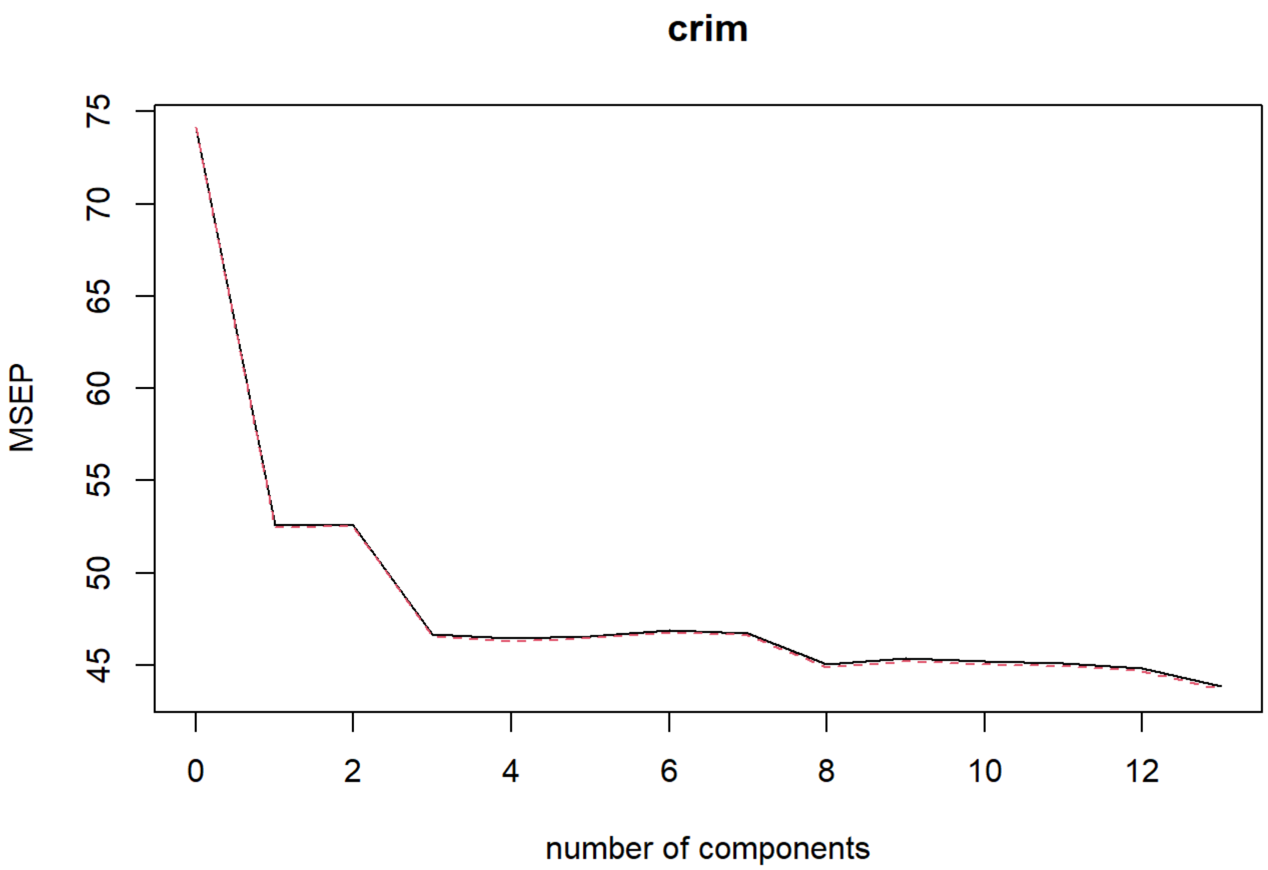
```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed (1)

pcr.fit <- pcr(crim ~ ., data = Boston , scale = TRUE , validation = "CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 506 13
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8.61    7.250    7.253    6.833    6.815    6.826    6.847
## adjCV         8.61    7.245    7.247    6.825    6.803    6.818    6.838
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.837    6.710    6.735     6.723     6.714     6.696     6.624
## adjCV    6.827    6.698    6.724     6.710     6.702     6.682     6.609
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       47.70    60.36    69.67    76.45    82.99    88.00    91.14    93.45
## crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14    42.47
##       9 comps  10 comps  11 comps  12 comps  13 comps
## X       95.40     97.04     98.46     99.52     100.0
## crim    42.55     42.78     43.04     44.13      45.4
```

```
validationplot(pcr.fit , val.type = "MSEP")
```




```
### pcr() function reports the root mean squared error. 13 component analysis has the lowest RMSE = 6.624
```

b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross validation, or some other reasonable alternative, as opposed to using training error.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data <- data.frame(Methods = c("Best subset selection","Lasso Regression","Ridge Regressio
n","Principal Component Reg"),
                   MSE = c(mse.cv[which.min(mse.cv)],cv.lasso$cvm[cv.lasso$lambda == cv.lasso
$lambda.1se], cv.ridge$cvm[cv.ridge$lambda == cv.ridge$lambda.1se], 43.877376),
                   RMSE = c(sqrt(mse.cv[which.min(mse.cv)]), sqrt(cv.lasso$cvm[cv.lasso$lambd
a == cv.lasso$lambda.1se]) ,sqrt(cv.ridge$cvm[cv.ridge$lambda == cv.ridge$lambda.1se]),6.624)
)
arrange(data,RMSE)
```

```
##                   Methods      MSE     RMSE
## 1   Best subset selection 42.81453 6.543281
## 2 Principal Component Reg 43.87738 6.624000
## 3        Lasso Regression 55.33380 7.438669
## 4        Ridge Regression 55.80448 7.470240
```

As i evaluated the performance of the models using cross-validation in part ‘a’ of this question, Best subset selection model performed better compared to other models with MSE = 42.81 and RMSE = 6.54

c) Does your chosen model involve all of the features in the data set? Why or why not?

```
### The model best subset selection chooses 9 features of the data set. Because, we are getti
ng the minimum MSE = 42.81 at subset size = 9.

which.min(mse.cv)
```

```
## 9
## 9
```

```
coef(best.fit, id = 9)
```

```
##   (Intercept)            zn           nox            rm           dis
##  12.021667175   0.039587625 -11.310819327   0.831219883  -0.773294339
##           rad       ptratio         black         lstat          medv
##   0.525470938  -0.295718930  -0.006769892   0.179943832  -0.187327063
```

```
### As far as - adding additional features will reduce the training RSS, but causes the cross
-validation MSE to increase.
```