Homework4

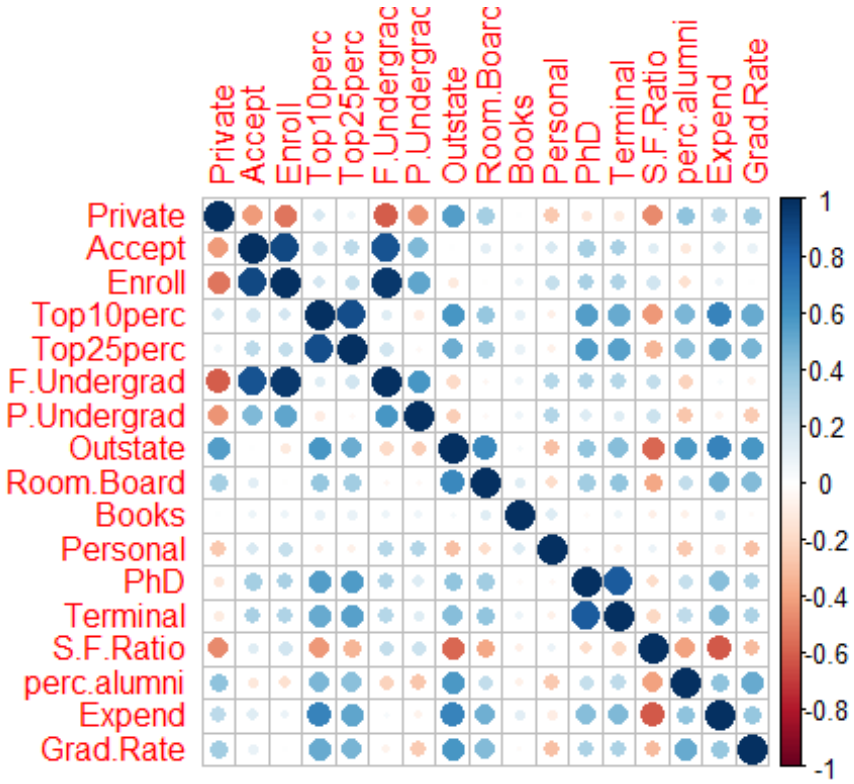
Bhriguvanshi

02/28/2020

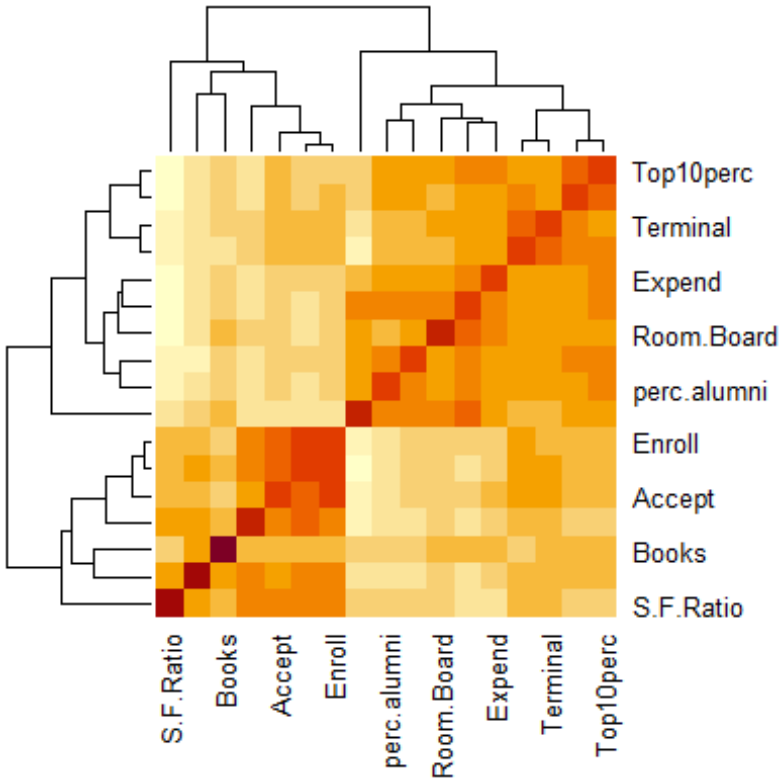
```
knitr::opts_chunk$set(echo = TRUE)

library(glmnet)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-18

library(leaps)
library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.

library(car)

## Loading required package: carData

## Data Loading and Preparation

data <- read.csv("College.csv", header = TRUE)
dim(data)

## [1] 777  19

# Converting Factor into Integer
data$Private <- as.integer(data$Private)

## Using ifelse statement to decode Private schools: 1 for Private, 0 for not
being Private
data$Private <- ifelse(data$Private == 2,1,0)

## Splitting the data into training and testing sets through random sampling
set.seed(1)

n <- nrow(data)
train_index <- sample(1:n,floor(0.7*n),replace = FALSE)

train <- data[train_index,]
dim(train)
```

```
## [1] 543  19

test <- data[-train_index,]
dim(test)

## [1] 234  19

matrix <- train[,-c(1,3)]   ## Remove School Names before forming the correla
tion matrix

## Correlation Matrix
(cormatrix <- cor(matrix))

##                  Private       Accept       Enroll   Top10perc   Top25perc
## Private      1.000000000 -0.425883804 -0.539133555  0.16203981  0.07678566
## Accept      -0.425883804  1.000000000  0.908631173  0.19630530  0.26218634
## Enroll      -0.539133555  0.908631173  1.000000000  0.18411831  0.24187452
## Top10perc    0.162039811  0.196305302  0.184118309  1.00000000  0.88390365
## Top25perc    0.076785658  0.262186337  0.241874523  0.88390365  1.00000000
## F.Undergrad -0.606793503  0.866049736  0.966231155  0.13551489  0.19519880
## P.Undergrad -0.446428923  0.444050750  0.529052295 -0.09745011 -0.03786369
## Outstate     0.551156088  0.008128624 -0.114173167  0.58137270  0.49694531
## Room.Board   0.338366808  0.120093012 -0.006624202  0.38700881  0.34161598
## Books        0.002755891  0.077358600  0.078030634  0.10317945  0.09886938
## Personal    -0.263658194  0.161074156  0.239810262 -0.08496766 -0.07741557
## PhD         -0.134146848  0.339459871  0.320221683  0.55222947  0.56950791
## Terminal    -0.105397611  0.323677103  0.301217627  0.50612208  0.54860212
## S.F.Ratio   -0.471568270  0.134862289  0.199268091 -0.43636778 -0.33809931
## perc.alumni  0.406773734 -0.127547512 -0.162430692  0.45260012  0.41033381
## Expend       0.260772582  0.139650498  0.080575443  0.66367130  0.52088928
## Grad.Rate    0.340859768  0.092117855  0.004032947  0.50013989  0.46566325
##              F.Undergrad P.Undergrad     Outstate   Room.Board        Books
## Private     -0.60679350 -0.44642892  0.551156088  0.338366808  0.002755891
## Accept       0.86604974  0.44405075  0.008128624  0.120093012  0.077358600
## Enroll       0.96623116  0.52905230 -0.114173167 -0.006624202  0.078030634
## Top10perc    0.13551489 -0.09745011  0.581372701  0.387008812  0.103179453
## Top25perc    0.19519880 -0.03786369  0.496945309  0.341615982  0.098869380
## F.Undergrad  1.00000000  0.58705100 -0.192579365 -0.044644539  0.079109164
## P.Undergrad  0.58705100  1.00000000 -0.244150727 -0.030535923  0.063026722
## Outstate    -0.19257936 -0.24415073  1.000000000  0.645784579  0.046568339
## Room.Board  -0.04464454 -0.03053592  0.645784579  1.000000000  0.134180304
## Books        0.07910916  0.06302672  0.046568339  0.134180304  1.000000000
## Personal     0.28447144  0.29500860 -0.295715687 -0.182220937  0.142621502
## PhD          0.30547177  0.14421671  0.399857070  0.345237614 -0.028936805
## Terminal     0.28658933  0.13157412  0.428435985  0.391758781  0.072433659
## S.F.Ratio    0.24058472  0.21300904 -0.574148482 -0.382838603 -0.083240350
## perc.alumni -0.22980921 -0.27173863  0.571182989  0.247379250 -0.077196068
## Expend       0.03182602 -0.06402549  0.663976969  0.486863131  0.121014240
## Grad.Rate   -0.06866374 -0.25587899  0.580077175  0.433340602 -0.026259359
##                Personal         PhD    Terminal   S.F.Ratio perc.alumni
## Private     -0.26365819 -0.13414685 -0.10539761 -0.47156827  0.40677373
```

```
## Accept       0.16107416  0.33945987  0.32367710  0.13486229 -0.12754751
## Enroll       0.23981026  0.32022168  0.30121763  0.19926809 -0.16243069
## Top10perc   -0.08496766  0.55222947  0.50612208 -0.43636778  0.45260012
## Top25perc   -0.07741557  0.56950791  0.54860212 -0.33809931  0.41033381
## F.Undergrad  0.28447144  0.30547177  0.28658933  0.24058472 -0.22980921
## P.Undergrad  0.29500860  0.14421671  0.13157412  0.21300904 -0.27173863
## Outstate    -0.29571569  0.39985707  0.42843598 -0.57414848  0.57118299
## Room.Board  -0.18222094  0.34523761  0.39175878 -0.38283860  0.24737925
## Books        0.14262150 -0.02893680  0.07243366 -0.08324035 -0.07719607
## Personal     1.00000000 -0.04149759 -0.07425684  0.08602768 -0.26857134
## PhD         -0.04149759  1.00000000  0.83892036 -0.18796640  0.23726910
## Terminal    -0.07425684  0.83892036  1.00000000 -0.20668443  0.25525322
## S.F.Ratio    0.08602768 -0.18796640 -0.20668443  1.00000000 -0.40928696
## perc.alumni -0.26857134  0.23726910  0.25525322 -0.40928696  1.00000000
## Expend      -0.09607480  0.42961412  0.44337473 -0.61412972  0.40409051
## Grad.Rate   -0.29480993  0.31837645  0.31162713 -0.31011518  0.50777546
##                  Expend    Grad.Rate
## Private      0.26077258  0.340859768
## Accept       0.13965050  0.092117855
## Enroll       0.08057544  0.004032947
## Top10perc    0.66367130  0.500139889
## Top25perc    0.52088928  0.465663254
## F.Undergrad  0.03182602 -0.068663740
## P.Undergrad -0.06402549 -0.255878995
## Outstate     0.66397697  0.580077175
## Room.Board   0.48686313  0.433340602
## Books        0.12101424 -0.026259359
## Personal    -0.09607480 -0.294809931
## PhD          0.42961412  0.318376449
## Terminal     0.44337473  0.311627132
## S.F.Ratio   -0.61412972 -0.310115180
## perc.alumni  0.40409051  0.507775460
## Expend       1.00000000  0.388302876
## Grad.Rate    0.38830288  1.000000000

## Use Corrplot for better Visualization
library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.3

## corrplot 0.84 loaded

## Corrplot with Private
corrplot(cormatrix)
```




```
## HeatMap for the Matrix (without Private)
(heatmap(cormatrix,keep.dendro=TRUE))
```



```
## $rowInd
##  [1] 14 11 10  7  2  6  3  1 15 17  9  8 16 12 13  5  4
##
## $colInd
##  [1] 14 11 10  7  2  6  3  1 15 17  9  8 16 12 13  5  4
##
## $Rowv
## 'dendrogram' with 2 branches and 17 members total, at height 3.688192
##
## $Colv
## 'dendrogram' with 2 branches and 17 members total, at height 3.688192
```

Response:

The heatmap reorder variables and observations using a clustering algorithm as it computes the distance between each pair of rows and columns and try to order them by similarity. The dark color boxes in the heat map portrays the variables of most interest. As the color fades away, the association between the variables becomes weaker. For example, the there is a strong association between combination of following variables:

Strong Association:
F.Undergrad and Enroll
Accept and Enroll
Terminal and PhD
Top10perc and Top25perc
F.Undergrad and Accept

As the heat map dies down to light brownish or white color, it depicts that the variables are not strongly correlated or demonstrate very weak association.

```
q1b <- train[,-1]
model <- lm(Apps ~., data = q1b)

## Reporting the summary of the model
summary(model)

##
## Call:
## lm(formula = Apps ~ ., data = q1b)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5816.1  -451.6    -1.0   327.2  7445.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.377e+02  5.076e+02  -1.059  0.28995
## Private     -5.045e+02  1.648e+02  -3.061  0.00232 **
## Accept       1.722e+00  4.763e-02  36.159  < 2e-16 ***
## Enroll      -1.055e+00  2.437e-01  -4.329 1.80e-05 ***
```

```
## Top10perc    5.358e+01  6.440e+00   8.320 7.64e-16 ***
## Top25perc   -1.614e+01  5.092e+00  -3.170  0.00161 **
## F.Undergrad  2.970e-02  4.399e-02   0.675  0.49976
## P.Undergrad  7.162e-02  3.649e-02   1.963  0.05019 .
## Outstate    -8.841e-02  2.176e-02  -4.064 5.57e-05 ***
## Room.Board   1.630e-01  5.577e-02   2.923  0.00362 **
## Books        2.727e-01  2.723e-01   1.001  0.31715
## Personal    -7.316e-03  7.283e-02  -0.100  0.92002
## PhD         -9.676e+00  5.360e+00  -1.805  0.07161 .
## Terminal    -3.781e-01  6.015e+00  -0.063  0.94990
## S.F.Ratio    1.627e+01  1.608e+01   1.012  0.31214
## perc.alumni  2.358e+00  4.853e+00   0.486  0.62722
## Expend       5.986e-02  1.337e-02   4.476 9.34e-06 ***
## Grad.Rate    7.158e+00  3.520e+00   2.034  0.04248 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1022 on 525 degrees of freedom
## Multiple R-squared:  0.9332, Adjusted R-squared:  0.9311
## F-statistic: 431.6 on 17 and 525 DF,  p-value: < 2.2e-16
```

```
## Coefficients of the model(Rounding off to two decimal places)
round(summary(model)$coefficients[-1,1],2)
```

```
##     Private      Accept      Enroll   Top10perc   Top25perc F.Undergrad
##     -504.47        1.72       -1.05       53.58      -16.14        0.03
## P.Undergrad    Outstate  Room.Board       Books    Personal         PhD
##        0.07       -0.09        0.16        0.27       -0.01       -9.68
##    Terminal   S.F.Ratio perc.alumni      Expend   Grad.Rate
##       -0.38       16.27        2.36        0.06        7.16
```

```
## Variance Inflation Factor (VIF)
vif(model)
```

```
##     Private      Accept      Enroll   Top10perc   Top25perc F.Undergrad
##    2.702957    6.744378   23.413244    6.780113    5.360385   20.305669
## P.Undergrad    Outstate  Room.Board       Books    Personal         PhD
##    1.772711    4.197844    2.025009    1.125340    1.284780    3.972529
##    Terminal   S.F.Ratio perc.alumni      Expend   Grad.Rate
##    3.874528    2.025388    1.873289    2.913778    1.917230
```

Response: VIF ranges from 1 and upwards and prints out a numerical value which predicts what percentage the variance or standard error squared is inflated for each coefficient. For example, Books has a VIF of 1.13 which means that variance of coefficients of Books is 13% bigger than what we will expect it to be considering there's no multi-collinearity (no correlation of books with other predictors).

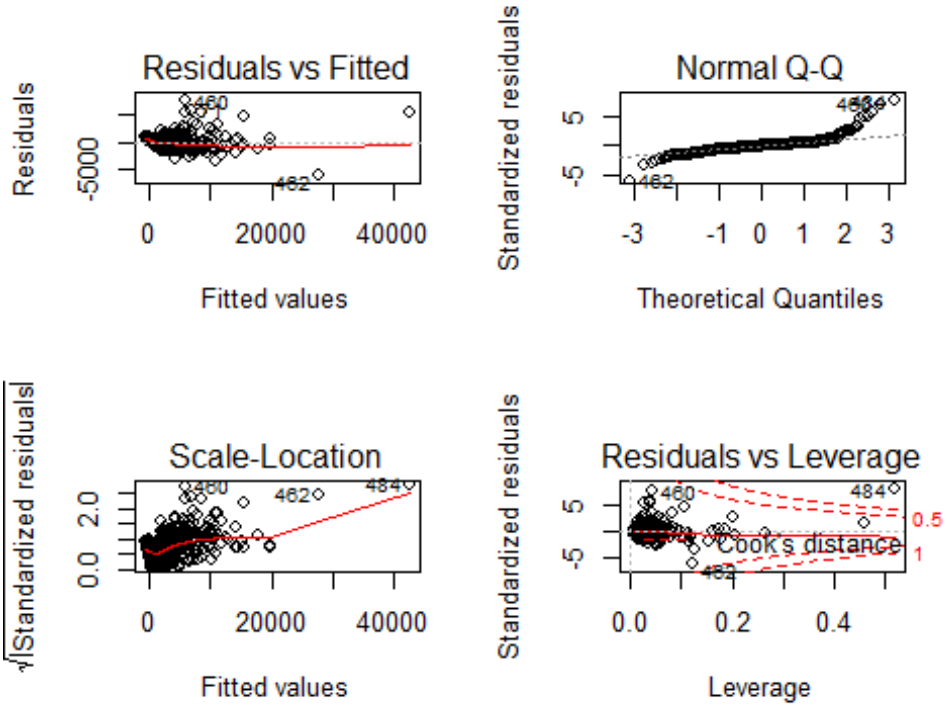
A VIF of Close to **1** means the variable is **not correlated**. Following variables fall in the same class: **Books**, **Personal**

A VIF of between **1 to 5** means the variable is **moderately correlated**. Following variables fall in the same class: **Private**, **P.Undergrad**, **Outstate**, **Room.Board**, **PhD**, **Terminal**, **S.F.Ratio**, **perc.alumni**, **Expend**, **Grad.Rate**

A VIF of greater than **5** means the variable is **highly correlated**. Following variables fall in the same class: **Accept**, **Top10perc**,**Top25perc**,

If VIF is more than **10**, the prediction is not accurate. We have two variables which has a VIF greater than 10 in the model: **Enroll**,**F.Undergrad**

```
## Four Plots to assess the model
par(mfrow=c(2,2))
plot(model)
```



Response:

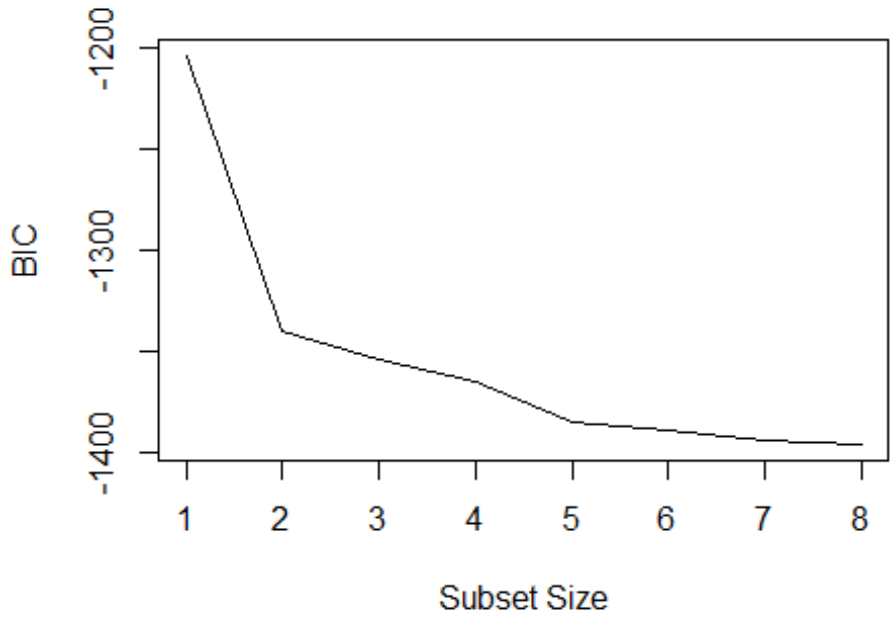
Due to presence of large sample size, normality assumption can be met even though the QQ-plot has some violations at the end or tails. The points shows independece and randomness of error terms. The Residuals vs Fitted plot indicated depicts non-constant variance but there's no regular pattern or trend of increased variance. At least one high influential leverage plot is observed in the final plot; higher both in the horizontal and vertical axis.

```
## Fitting the model on Test Data
prediction <- predict(model, newdata = test[,-c(1,3)])
error <- test$Apps - prediction
(RMSE <- sqrt(mean(error^2)))
```

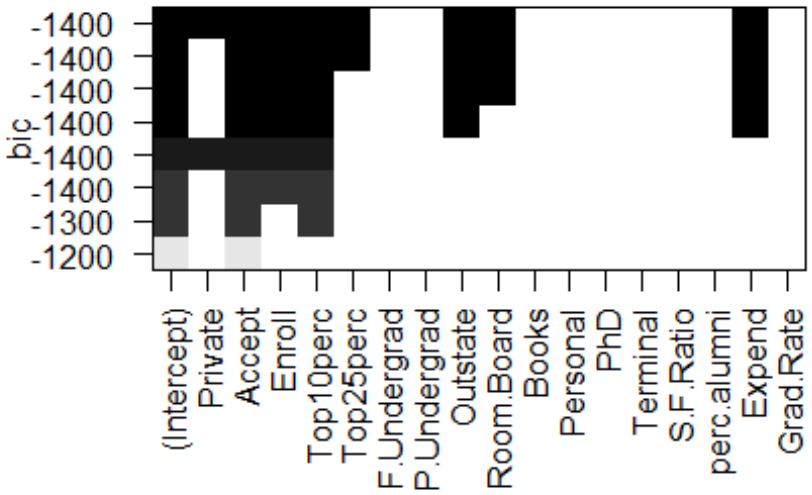
```
## [1] 1123.223
```

Response: The predicted values on test data will be off by approximately 1123. In context of the problem, on an average, the number of applications will be off by 1123, as estimated by the model when fitted on the test data.

```
## Variable Selection Approach
VS <- regsubsets(Apps ~., data = train[,-1])

## Summary for Variable Selection
VS_Summary <- summary(VS)
names(VS_Summary)

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

## Minimum number of predictor variables required through BIC
c("BIC" = which.min(VS_Summary$bic))

## BIC
##   8

## Plot depicting the number of predictors
plot(VS_Summary$bic, xlab="Subset Size", ylab="BIC", type='l')
```




```
## Plot to check the significance of predictors
plot(VS, ylab = "BIC")
```




```
## Extracting the 8 coefficients found from BIC to find their coefficients
coef(VS,8)

##   (Intercept)       Private        Accept        Enroll     Top10perc
## -264.66868986 -430.32683351    1.71788757   -0.84423284   53.02102386
##     Top25perc      Outstate    Room.Board        Expend
##  -17.42700134   -0.09531971    0.17984875    0.05303245

## Linear Model on Train Data
model <- lm(Apps ~ Private + Accept + Enroll + Top10perc + Top25perc + Outsta
te + Room.Board + Expend , data = train[,-1])

## Summary of the Linear Model fitted on test data
summary(model)

##
## Call:
## lm(formula = Apps ~ Private + Accept + Enroll + Top10perc + Top25perc +
##     Outstate + Room.Board + Expend, data = train[, -1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5927.4  -448.6    34.0   322.2  7646.5
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -264.66869  243.21833  -1.088 0.277000
```

```
## Private     -430.32683  148.86485  -2.891 0.004000 **
## Accept         1.71789    0.04726  36.348  < 2e-16 ***
## Enroll        -0.84423    0.13705  -6.160 1.43e-09 ***
## Top10perc     53.02102    6.30219   8.413 3.67e-16 ***
## Top25perc    -17.42700    4.95455  -3.517 0.000473 ***
## Outstate      -0.09532    0.01975  -4.827 1.81e-06 ***
## Room.Board     0.17985    0.05300   3.393 0.000741 ***
## Expend         0.05303    0.01225   4.329 1.79e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1029 on 534 degrees of freedom
## Multiple R-squared:  0.9311, Adjusted R-squared:   0.93
## F-statistic: 901.5 on 8 and 534 DF,  p-value: < 2.2e-16
```

```
## Selected Predictors and their corresponding coefficients
round(summary(model)$coefficients[-1,1],2)
```

```
##    Private     Accept     Enroll  Top10perc  Top25perc   Outstate
##    -430.33       1.72      -0.84      53.02     -17.43      -0.10
## Room.Board     Expend
##       0.18       0.05
```

```
## Finding Prediction and Fitting the model on Test Data
prediction <- predict(model,newdata = test[,-c(1,3)])
error <- test$Apps - prediction

## Root Mean Square Error
(RMSE <- sqrt(mean(error^2)))
```

```
## [1] 1134.934
```

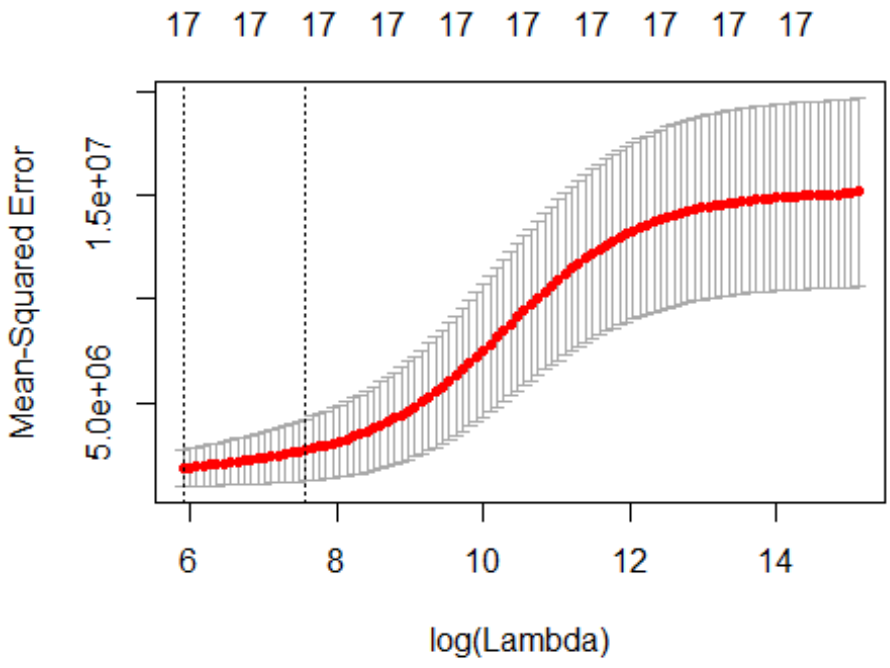
Response: Using the BIC criterion and fitting the model on test data, we can estimate that on an average the number of application will be off by approximately 1135.

Using the BIC criterion on training data, we have a residual standard error of 1029 which goes up when fitted on test data.

The BIC criterion selects the most important predictor variables to fit in the model. When comparing the coefficients of the BIC model to the normal model, the coefficients are relatively in the same range excluding private. The eight variables selected by the BIC model explains the most variation in the model. THe BIC helps to penalize the complexity of the model, where the complexity refers to the number of parameters(essentially predictor variables) in the model.

The original model and the model under BIC criterion when fitted on training data have around the same R-square and RMSE values. However, the RMSE value in the BIC model goes up slightly when we fit the data on test set, comparing it with the initial model in q1(b).

```r
## Fitting the model on training data after little data preparation
YY <- as.numeric(data[,3])
XX <- as.matrix(data[,-c(1,3)])

X <- XX[train_index,]
Y <- YY[train_index]

X_test <- XX[-c(train_index),]
Y_test <- YY[-c(train_index)]

## glmnet with Ridge on Training Data and Cross-Validation
cvfitr <- cv.glmnet(x = X,y = Y,family = "gaussian", alpha = 0)

## Plot for Lambda and Mean Sqaure Error
par(mfrow=c(1,1))
plot(cvfitr)
```




```r
r.min <- cvfitr$lambda.min  ## Smallest lambda will give less error
cvfitr$lambda.1se

## [1] 1961.39

betas <- as.matrix(cbind(coef(lm(Y~X)), coef(cvfitr,s="lambda.min"), coef(cvf
itr,s="lambda.1se")))
colnames(betas) <- c("OLS", "Ridge.lambda.min","Ridge.lambda.1se")
```

```
## Coefficients of the best Ridge Model
(betas[-1,])
```

```
##                        OLS Ridge.lambda.min Ridge.lambda.1se
## Private     -5.044736e+02    -422.02172455    -4.440618e+02
## Accept       1.722400e+00       1.09747712     6.065489e-01
## Enroll      -1.054935e+00       0.39050300     8.028431e-01
## Top10perc    5.358118e+01      26.56457927     1.468427e+01
## Top25perc   -1.614280e+01       0.03881481     7.355209e+00
## F.Undergrad  2.970536e-02       0.05880539     1.311036e-01
## P.Undergrad  7.161603e-02       0.03575870     7.731183e-02
## Outstate    -8.841304e-02      -0.03383476     5.666193e-03
## Room.Board   1.630283e-01       0.21655395     1.956340e-01
## Books        2.726531e-01       0.30106826     3.663957e-01
## Personal    -7.316383e-03      -0.03031505     2.399494e-03
## PhD         -9.675691e+00      -3.99648934     2.082741e+00
## Terminal    -3.781472e-01      -4.12481627     1.103110e+00
## S.F.Ratio    1.626753e+01      15.54305293     1.275722e+01
## perc.alumni  2.358318e+00      -4.71214502    -6.738386e+00
## Expend       5.986427e-02       0.06414701     4.904279e-02
## Grad.Rate    7.158264e+00       9.93898601     1.094295e+01
```

```
(round(betas[-1,2],2))
```

```
##     Private      Accept      Enroll   Top10perc   Top25perc F.Undergrad
##     -422.02        1.10        0.39       26.56        0.04         0.06
## P.Undergrad    Outstate  Room.Board       Books    Personal          PhD
##        0.04       -0.03        0.22        0.30       -0.03        -4.00
##    Terminal   S.F.Ratio perc.alumni      Expend   Grad.Rate
##       -4.12       15.54       -4.71        0.06        9.94
```

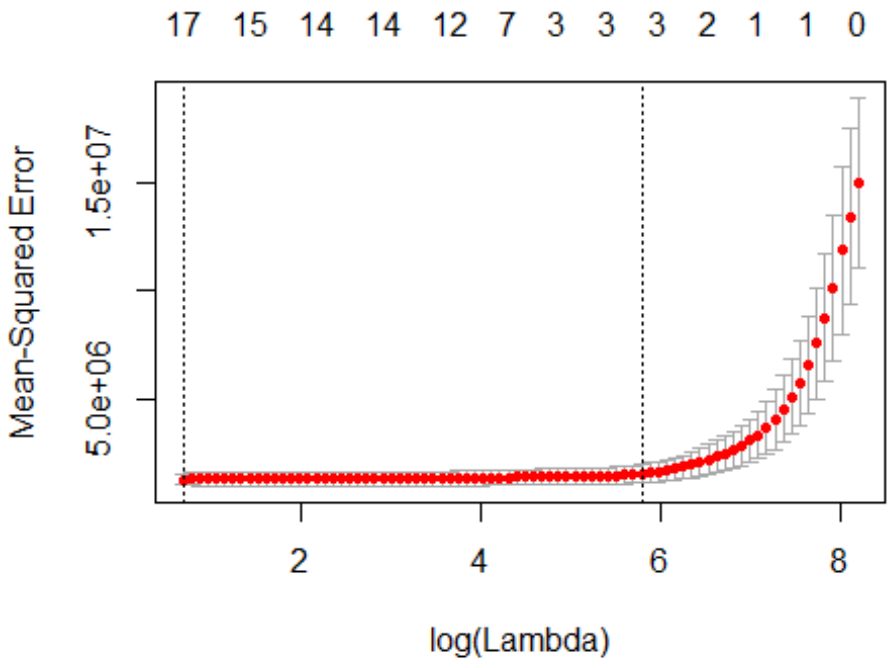
```
## Finding Prediction Values and RMSE by fitting on test data
predictedr <- predict(cvfitr, newx = X_test, s = r.min)

# Root Mean Square Error
(RMSE <- sqrt(mean((predictedr-Y_test)^2)))
```

```
## [1] 1058.789
```

Response: Using the Ridge Model criterion and fitting the model on test data, we can estimate that on an average the number of applications will be off by approximately 1059.

The coefficients obtained from the ridge model changes as majority of the coefficients decrease and are condensed from the original model. The coefficients which increase from the original model have a very small or minimal rise. The model uses a shrinkage penalty which is lambda value times the sum of squares of the coefficients. The coefficients which are very large are penalized. As a result, when lamdba value gets larger, the bias does not change but the variance drops. Unlike the BIC criterion, the ridge model uses all the variables and does not drop any of them

```r
## Fitting the model on training data after little data preparation
YY <- as.numeric(data[,3])
XX <- as.matrix(data[,-c(1,3)])

X <- XX[train_index,]
Y <- YY[train_index]

X_test <- XX[-c(train_index),]
Y_test <- YY[-c(train_index)]

# glmnet with Lasso on Training Data and Cross-Validation
cvfitl <- cv.glmnet(x = X, y = Y, family = "gaussian", alpha = 1)

## Plot for Lambda Value
par(mfrow=c(1,1))
plot(cvfitl)
```




```r
l.min <- cvfitl$lambda.min  ## lambda.min for lasso
cvfitl$lambda.1se

## [1] 327.1795

## Best Model
betas <- as.matrix(cbind(coef(lm(Y~X)), coef(cvfitl,s="lambda.min"),coef(cvfi
tl,s="lambda.1se")))
colnames(betas) <- c("OLS", "Lasso.lambda.min","Lasso.lambda.1se")
```

```
## Coefficients of the Model
(betas[-1,])
```

```
##                        OLS Lasso.lambda.min Lasso.lambda.1se
## Private     -5.044736e+02    -5.022385e+02      0.000000000
## Accept       1.722400e+00     1.707138e+00      1.371587144
## Enroll      -1.054935e+00    -9.239800e-01      0.000000000
## Top10perc    5.358118e+01     5.224804e+01     17.980108449
## Top25perc   -1.614280e+01    -1.514711e+01      0.000000000
## F.Undergrad  2.970536e-02     1.134031e-02      0.000000000
## P.Undergrad  7.161603e-02     7.100787e-02      0.000000000
## Outstate    -8.841304e-02    -8.599271e-02      0.000000000
## Room.Board   1.630283e-01     1.609185e-01      0.000000000
## Books        2.726531e-01     2.575968e-01      0.000000000
## Personal    -7.316383e-03    -1.937043e-03      0.000000000
## PhD         -9.675691e+00    -9.469811e+00      0.000000000
## Terminal    -3.781472e-01    -1.776324e-01      0.000000000
## S.F.Ratio    1.626753e+01     1.509291e+01      0.000000000
## perc.alumni  2.358318e+00     1.575469e+00      0.000000000
## Expend       5.986427e-02     5.924996e-02      0.008015089
## Grad.Rate    7.158264e+00     6.948461e+00      0.000000000
```

```
(round(betas[-1,2],2))
```

```
##     Private      Accept      Enroll   Top10perc   Top25perc F.Undergrad
##     -502.24        1.71       -0.92       52.25      -15.15        0.01
## P.Undergrad    Outstate  Room.Board       Books    Personal         PhD
##        0.07       -0.09        0.16        0.26        0.00       -9.47
##    Terminal   S.F.Ratio perc.alumni      Expend   Grad.Rate
##       -0.18       15.09        1.58        0.06        6.95
```

```
## Predicted values for lasso and Root Mean Square Error
predictedl <- predict(cvfitl, newx = X_test, s = l.min )

## Root Mean Square Error
(RMSE <- sqrt(mean((predictedl-Y_test)^2)))
```

```
## [1] 1120.004
```

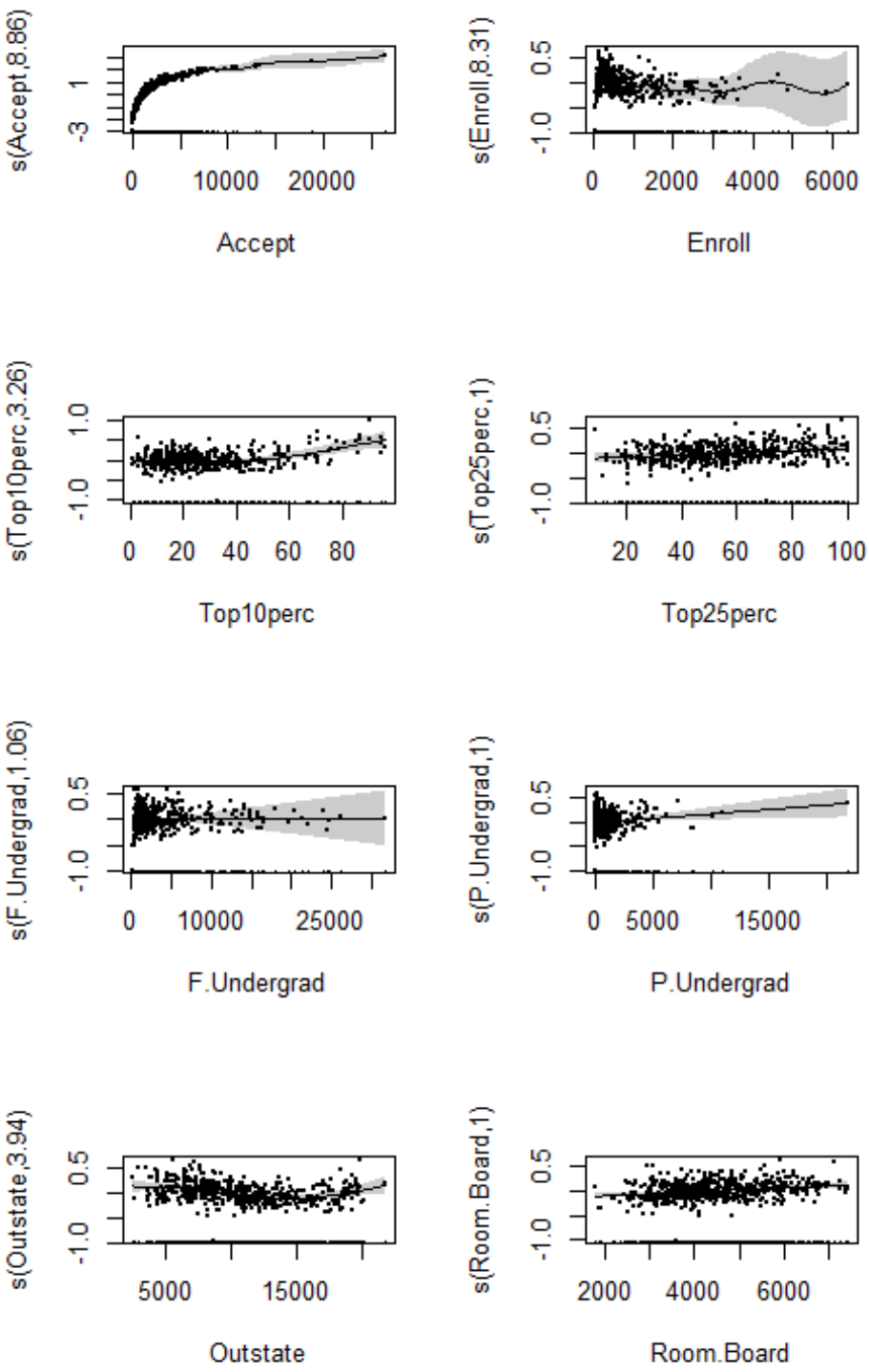
Response: Using the Lasso Model criterion and fitting the model on test data, we can estimate that on an average the number of applications will be off by approximately 1120.

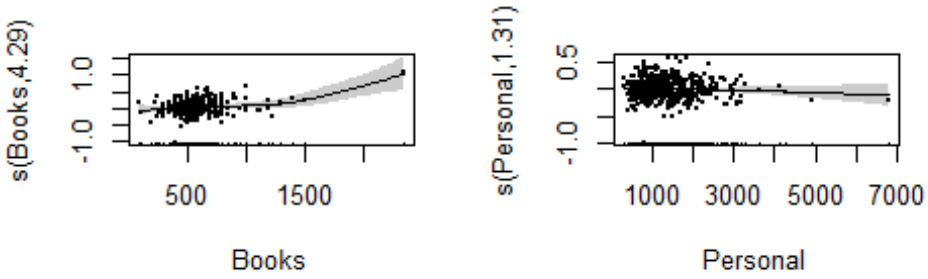
The coefficients obtained from the lasso model also are relatively smaller than the original model. The penalty of the coefficients here is the sum of absolute values of the coefficients. The lasso model tries to shirnk the coefficients estimates towards zero. In some cases, it has the effect of setting the variable equal to zero, which also solves the purpose of variable selection as the predictors with zero coefficients will be eliminated. When the lamdba estimate is small, the model is essentially the least squares estimates. However, as the lambda value increases shrinkage occurs which can help to throw out coefficients when it equals to zero. A lot of coefficients in the lasso model above have coefficients which can

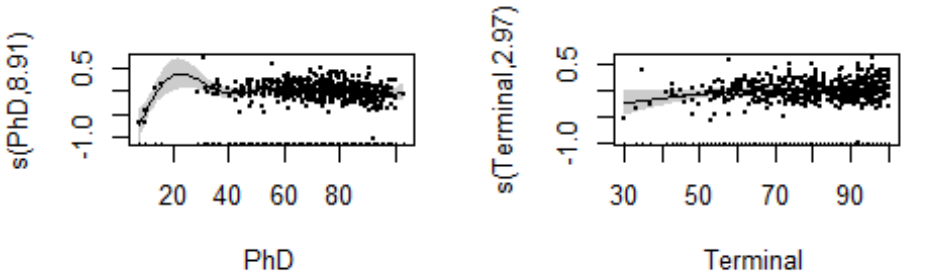
effectively be taken out or possess a negligible effect in the response variable if included in the model.

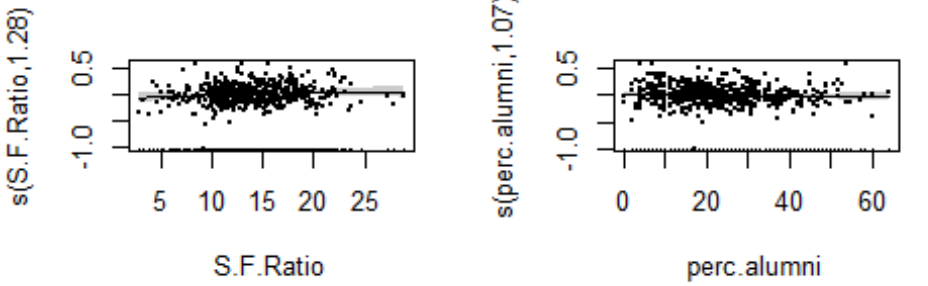
```
gam1 <- gam(log(Apps) ~  Private + s(Accept) + s(Enroll) + s(Top10perc)+ s(To
p25perc) + s(F.Undergrad) + s(P.Undergrad) + s(Outstate)+ s(Room.Board) + s(B
ooks) + s(Personal) + s(PhD) + s(Terminal)+ s(S.F.Ratio) + s(perc.alumni) + s
(Expend) + s(Grad.Rate), family = "gaussian" ,data = train)

summary(gam1)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(Apps) ~ Private + s(Accept) + s(Enroll) + s(Top10perc) +
##     s(Top25perc) + s(F.Undergrad) + s(P.Undergrad) + s(Outstate) +
##     s(Room.Board) + s(Books) + s(Personal) + s(PhD) + s(Terminal) +
##     s(S.F.Ratio) + s(perc.alumni) + s(Expend) + s(Grad.Rate)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.44342    0.02714 274.224   <2e-16 ***
## Private     -0.03675    0.03496  -1.051    0.294
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df       F  p-value
## s(Accept)       8.865  8.990 117.884  < 2e-16 ***
## s(Enroll)       8.308  8.859   2.452  0.00773 **
## s(Top10perc)    3.257  4.118  10.137 5.40e-08 ***
## s(Top25perc)    1.000  1.000   3.561  0.05975 .
## s(F.Undergrad) 1.058  1.112   0.005  0.92660
## s(P.Undergrad) 1.000  1.000   6.899  0.00889 **
## s(Outstate)     3.942  4.922   7.558 9.77e-07 ***
## s(Room.Board)   1.000  1.000  14.470  0.00016 ***
## s(Books)        4.288  5.301   5.742 2.81e-05 ***
## s(Personal)     1.310  1.560   0.926  0.49104
## s(PhD)          8.907  8.990   3.913 9.29e-05 ***
## s(Terminal)     2.967  3.817   1.572  0.15267
## s(S.F.Ratio)    1.283  1.516   1.173  0.23439
## s(perc.alumni) 1.072  1.139   0.809  0.33991
## s(Expend)       1.793  2.279   3.542  0.02491 *
## s(Grad.Rate)    8.437  8.897   4.945 2.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.97   Deviance explained = 97.3%
## GCV = 0.038871  Scale est. = 0.034541  n = 543
```

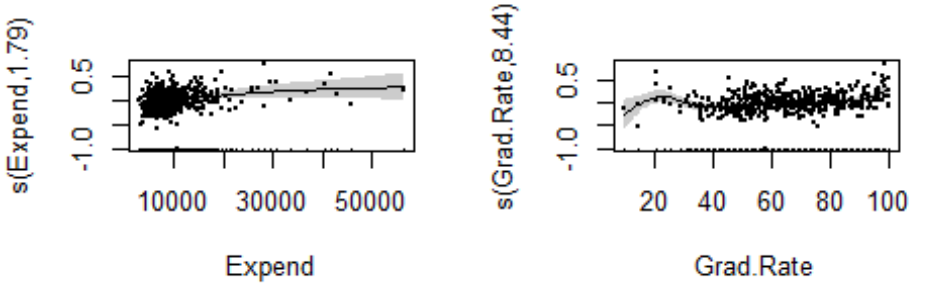
```r
## Partial Effect Plots from GAMs
plot(gam1, scale = 0, se = 2, shade = TRUE, resid = TRUE, pages = 4, pch = 19
, cex = 0.25)
```












```r
## Predictions
predictions <- predict(gam1, newdata = test[,-c(1,3)])
(RMSE <- sqrt(mean((predictions-Y_test)^2)))   ## Y_test is defined in the pr
evious models
```

```
## [1] 4914.225
```

Response: The edf column from the model summary shows the linear or non-linear relationship of the predictor variables with the response variable. So for example, variables such as “Top25perc”, “F.Undergrad”, “P.Undergrad”, “Personal”, “Room.Board”,“S.F.Ratio”,“perc.alumni” have edf value of 1 which depicts that they have linear relationship with the response variable. On the contrary, other variables which have higher edf values demonstrate non-linear relationship with the response variable (Apps).

There is no conclusive evidence that the GAM model is better than the initial model because there’s issues with fitting of the data. The partial plot depict the spread of the points for different variables and not all the predictor variables are evenly spread.

For example the variable **accept**, **enroll**, **Top10perc** follow a certain pattern where data values aare clustered towards the left and follow to trend as it moves along the X-axis.

Similary predictor variables - **Top25perc** and **Room.Board** have a good fit as the points are spread out across the line and it does not follow a certain pattern.

Other variables such as **Terminal** or **PhD** have points cluttered vertically which portrays the variant effect of the variable to predict the response variable.

The partial plots helps to estimate the fitting of data points as well the effect of predictor variable to the response variable. The partial plots are also important because it helps to see the trend of the variables as GAM models are not interpreted as a linear model (one unit change in the predictor variable would change the response by the coefficient of the predictor variable). Therefore, partial plots help is recognizing non-linear relationship of the predictors with the response.

##Response 1i

The RMSE of the different models computed above vary slightly but the numbers are not off by a lot. While BIC does not give the lowest RMSE, I think it is the best model since it tries to remove multi-collinearity among the predictor variables in the models and picks out the 8 best variables which explains approximates the best variation in the data. Additionally, the BIC model is parsimonious and simpler since we have less coefficients to interpret and estimate. From a management’s perspective, cosidering the time and money involved in estimating paramters, BIC model should be really efficient. The RMSE for the Ridge model is the lowest but it has a lot more coefficients to interpret.

For accurately predicting the results it is important to split the data into two sets to see and compare the results from both the sets. Additionally, comparing different models (with each pros and cons) can help to make an appropriate decision depending on the situation.

The variales of interest would be: Private, Accept, Enroll, Top10perc, Top25perc, Outstate, Room.Board, Expend. The model with these variables gives a R-square of 93% when tested on the training set; which explains the variation modeled by these variables in order to estimate the number of applications. It also has a low RMSE when fitted on training data; comparing to the original OLS model.