DATA311 Project

Parsa Rajabi, Chelsey Hvingelby, Mackenzie Salloum, Cameron Chong, Jeff Bulmer

2019-04-04

The following libraries are required in order to run the markdown script properly. They can be installed from the CRAN repositories.

```
install.packages("FNN")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("mvtnorm")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
## Warning: package 'mvtnorm' is not available (for R version 3.4.4)
```

```
install.packages("mclust")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("cluster")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("tree")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("randomForest")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("fpc")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("boot")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("MASS")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("MLMetrics")
```

```
## Installing package into '/home/jeff/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
## Warning: package 'MLMetrics' is not available (for R version 3.4.4)
```

```
## Warning: Perhaps you meant 'MLmetrics' ?
```

```
library(FNN)
library(mvtnorm)
library(mclust)
```

```
## Package 'mclust' version 5.4.3
## Type 'citation("mclust")' for citing this R package in publications.
```

```
library(cluster)
library(fpc)
library(boot)
library(tree)
library(MASS)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(neuralnet)
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

The data in question is 500 observations of graduate admission students for Universities in India. It consists of categorical and continuous variables.

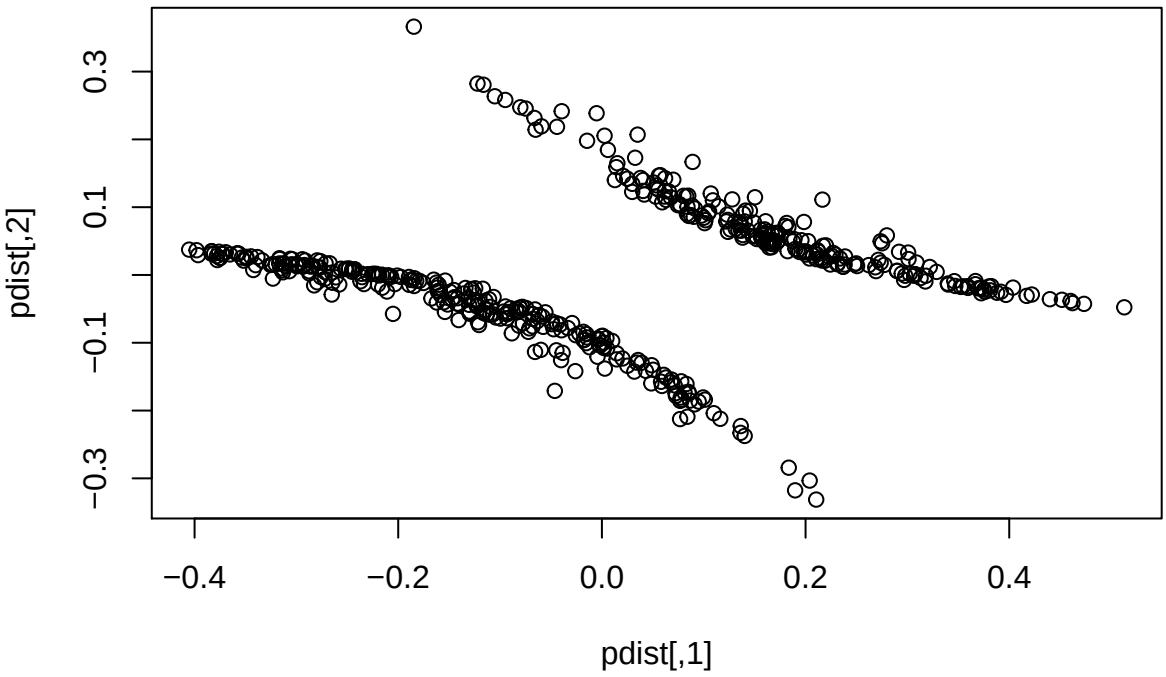
The dataset being explored in this report consists of graduate admissions data for students in India. Data was collected from 500 prospective graduate students, including various scores achieved in the Test of English as a First Language (TOEFL.Score) Graduate Record Examinations (GRE.Score), and scores indicating the strength of each candidates Statement of Purpose (SOP) and Letter of Recommendation (LOR). Other attributes include Undergraduate Cumulative GPA (CGPA), a unique identifier (Serial.No.), and whether or not the prospective student had Research Experience (Research). Finally, each candidate was polled about their confidence of being accepted into graduate school (Chance.of.Admit).

The data must be attached in order to run the analysis. As long as the file is in the same directory as the Rmd file it will run.

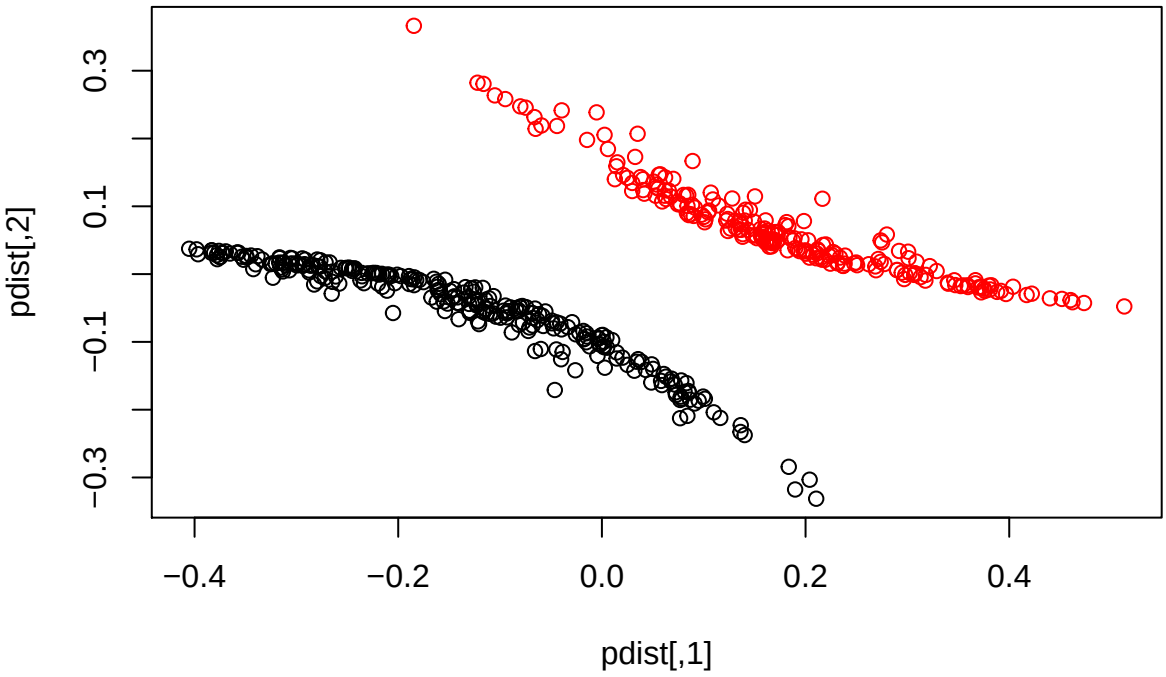
Clustering

We begin by computing the respective pairwise distances in our data, and plotting the output.

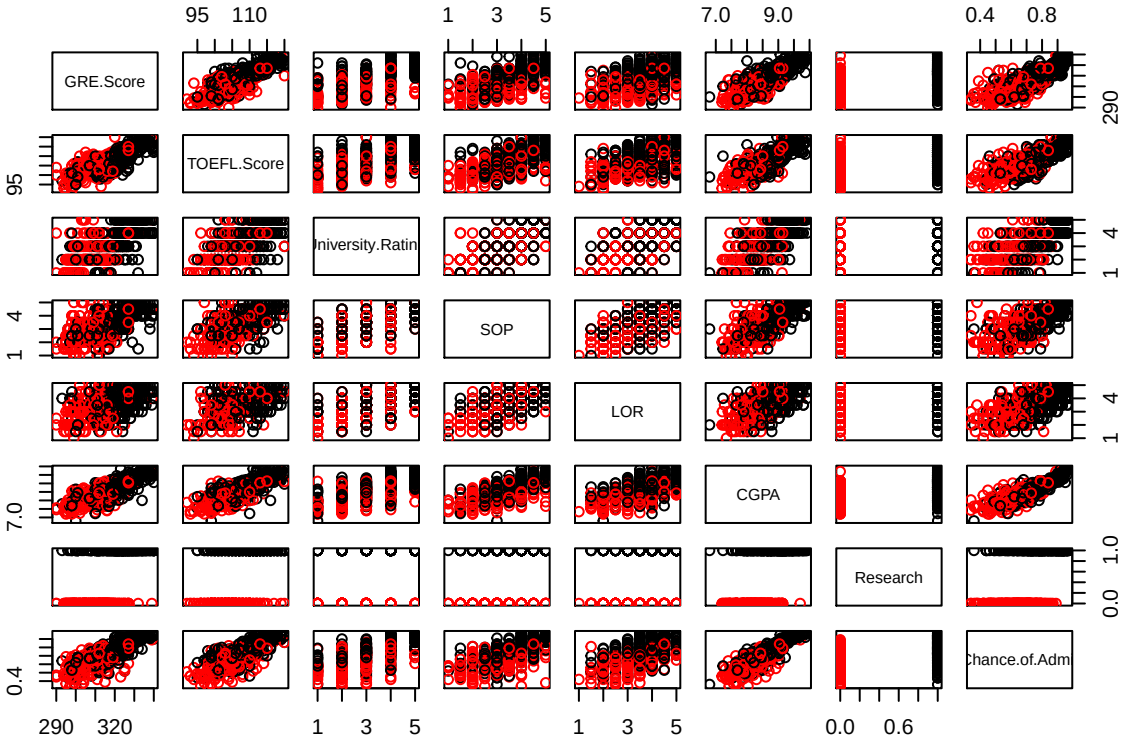
```
## Warning in daisy(admissionsData[, -1], metric = "gower"): binary
## variable(s) 7 treated as interval scaled
```



We quickly see that two clear groups appear. We can isolate these two groups using hierarchical clustering with single-linkage chaining.



We can then use scatterplots to show the entirety of the data, while still keeping the groups intact, to see if we can determine which predictors most affect these clusters.



We notice that, using the single linkage chaining from above, we can predict whether or not a student performs research almost perfectly.

So, by applying Gower's Distance on all predictors and using single-linkage chaining, we have two clear clusters directly coinciding with the presence of a research variable. This tells us that we should use Research as a response variable in models, in addition to Chance of Admit.

We can now perform analyses on the data to attempt to predict a candidate's Chance of Admission, as well as the presence of Research Experience.

Linear Models

PARSAS CODE GOES HERE

Bootstrap

JEFFS CODE HERE

Trees and Random Forest

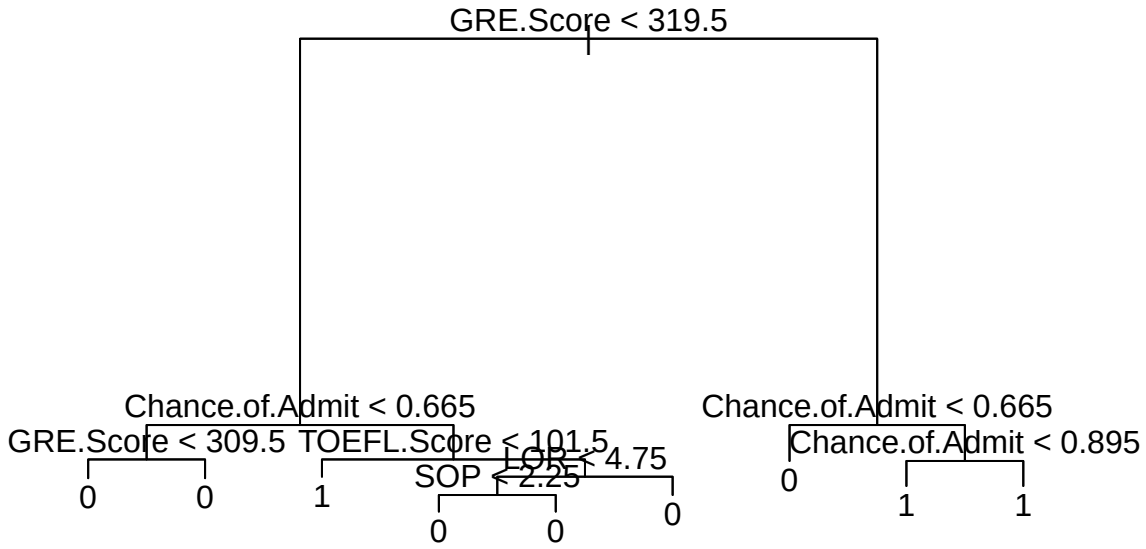
We will apply 70/30 split of training and testing data. There are 500 observations, so we will have 350 training observations and 150 testing points.

```
admissionsTreeData <- admissionsData[,-1]
trainindex <- sample(1:nrow(admissionsTreeData), 350)
admissionsTrain <- admissionsTreeData[trainindex, ]
admissionsTest <- admissionsTreeData[-trainindex, ]
```

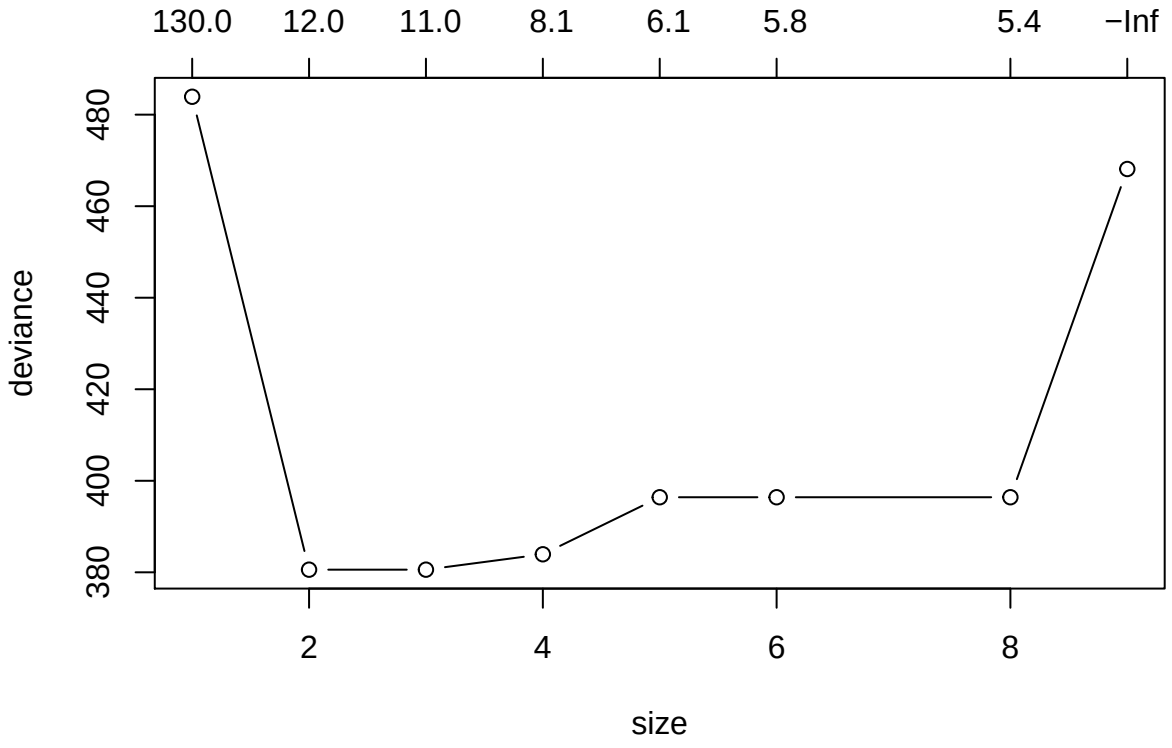
Research Tree

Cross Validation

```
set.seed(1232343124)
researchTree <- tree(as.factor(Research)~., data = admissionsTrain)
plot(researchTree)
text(researchTree, pretty=0)
```




```
researchTreeCV <- cv.tree(researchTree, FUN = prune.tree, K = 5)
plot(researchTreeCV, type = "b")
```



```
which.min(researchTreeCV$dev)
```

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

```
researchTreeCV$dev
```

```
## [1] 468.1265 396.4028 396.4028 396.4028 383.9398 380.5777 380.5777 483.9097
```

```
researchTreeCV$dev
```

```
## [1] 468.1265 396.4028 396.4028 396.4028 383.9398 380.5777 380.5777 483.9097
```

```
researchTreeCV$size
```

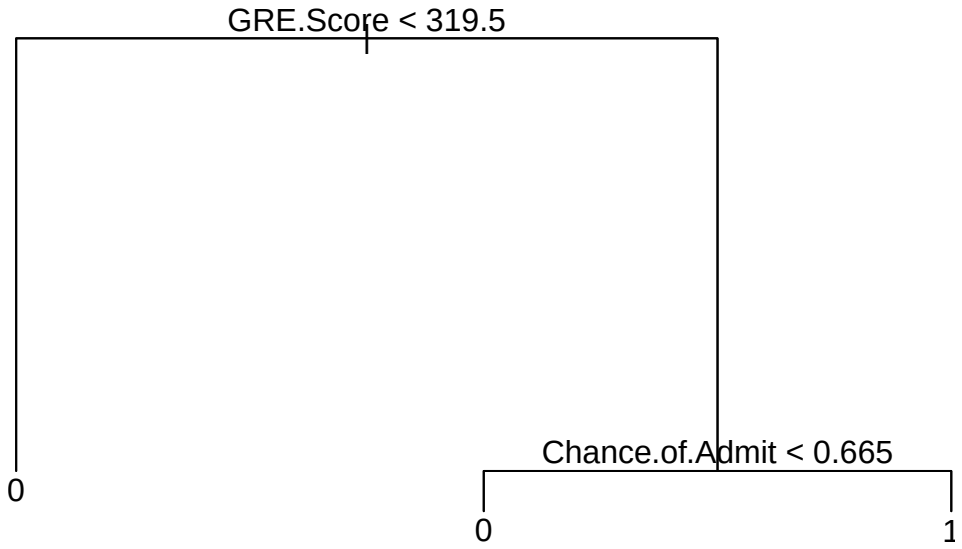
```
## [1] 9 8 6 5 4 3 2 1
```

```
which.min(researchTreeCV$dev)
```

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

Cross Validation Suggests 3 terminal nodes would be best. So we will prune our tree to 3 terminal nodes

```
pruneResearchTreeCV <- prune.tree(researchTree, best=3)
plot(pruneResearchTreeCV)
text(pruneResearchTreeCV, pretty = 0)
```

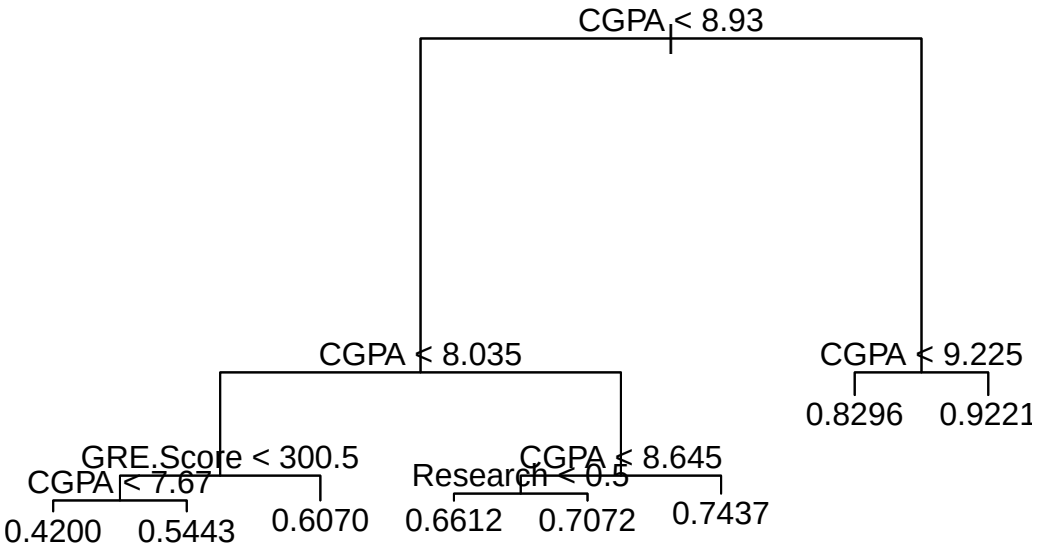



```
summary(pruneResearchTreeCV)
```

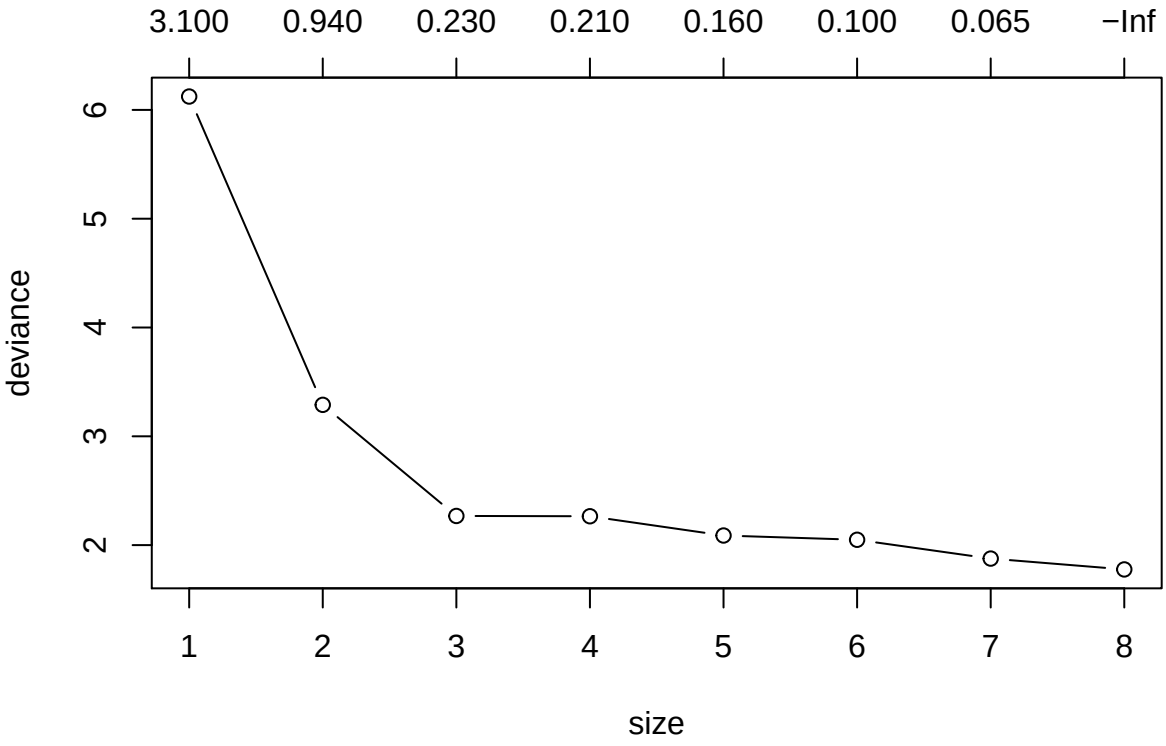
```
##
## Classification tree:
## snip.tree(tree = researchTree, nodes = c(7L, 2L))
## Variables actually used in tree construction:
## [1] "GRE.Score"       "Chance.of.Admit"
## Number of terminal nodes:  3
## Residual mean deviance:  0.993 = 344.6 / 347
## Misclassification error rate: 0.2257 = 79 / 350
```

Chance of Admissions Random Tree

```
set.seed(110101010)
admissionTree <- tree(Chance.of.Admit~., data = admissionsTrain)
plot(admissionTree)
text(admissionTree, pretty=0)
```




```
admissionTreeCV <- cv.tree(admissionTree, FUN = prune.tree, K = 10)
plot(admissionTreeCV, type = "b")
```




```
admissionTreeCV
```

```
## $size
```

```
## [1] 8 7 6 5 4 3 2 1
##
## $dev
## [1] 1.776647 1.876277 2.049129 2.088378 2.265256 2.268093 3.289263 6.122959
##
## $k
## [1]       -Inf 0.06460131 0.09981099 0.16429308 0.20789147 0.22720009
## [7] 0.94434693 3.05414716
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
admissionTreeCV$dev
```

```
## [1] 1.776647 1.876277 2.049129 2.088378 2.265256 2.268093 3.289263 6.122959
```

```
admissionTreeCV$size
```

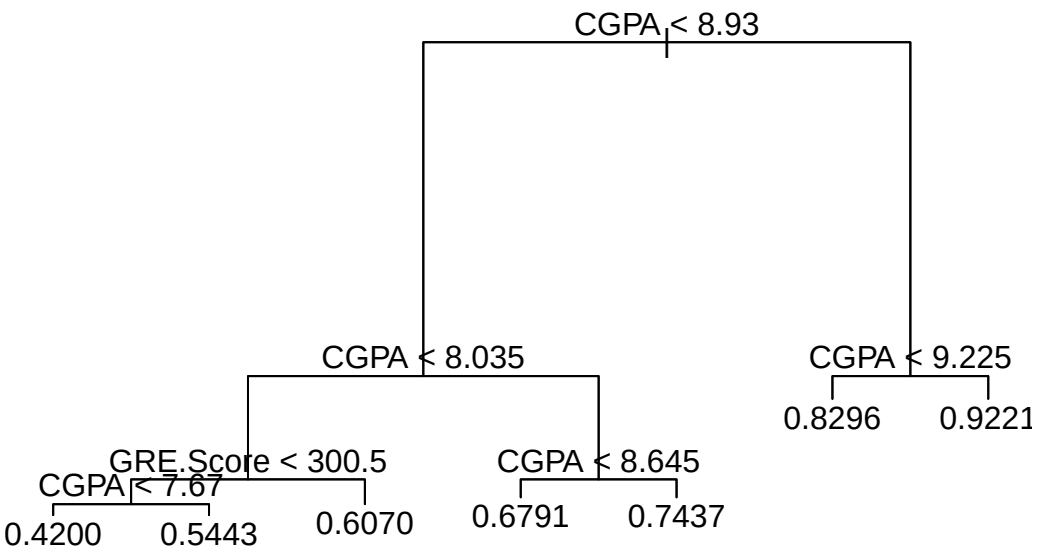
```
## [1] 8 7 6 5 4 3 2 1
```

```
which.min(admissionTreeCV$dev)
```

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

Cross validation suggest 7 nodes would be best, so we will prune the tree using 7 terminal nodes.

```
pruneAdmissionTreeCV <- prune.tree(admissionTree, best=7)
plot(pruneAdmissionTreeCV)
text(pruneAdmissionTreeCV, pretty = 0)
```




```
summary(pruneAdmissionTreeCV)
```

```
##
## Regression tree:
## snip.tree(tree = admissionTree, nodes = 10L)
## Variables actually used in tree construction:
## [1] "CGPA"      "GRE.Score"
```

```
## Number of terminal nodes:  7
## Residual mean deviance:  0.004009 = 1.375 / 343
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.259100 -0.033680  0.007907  0.000000  0.040860  0.173000
```

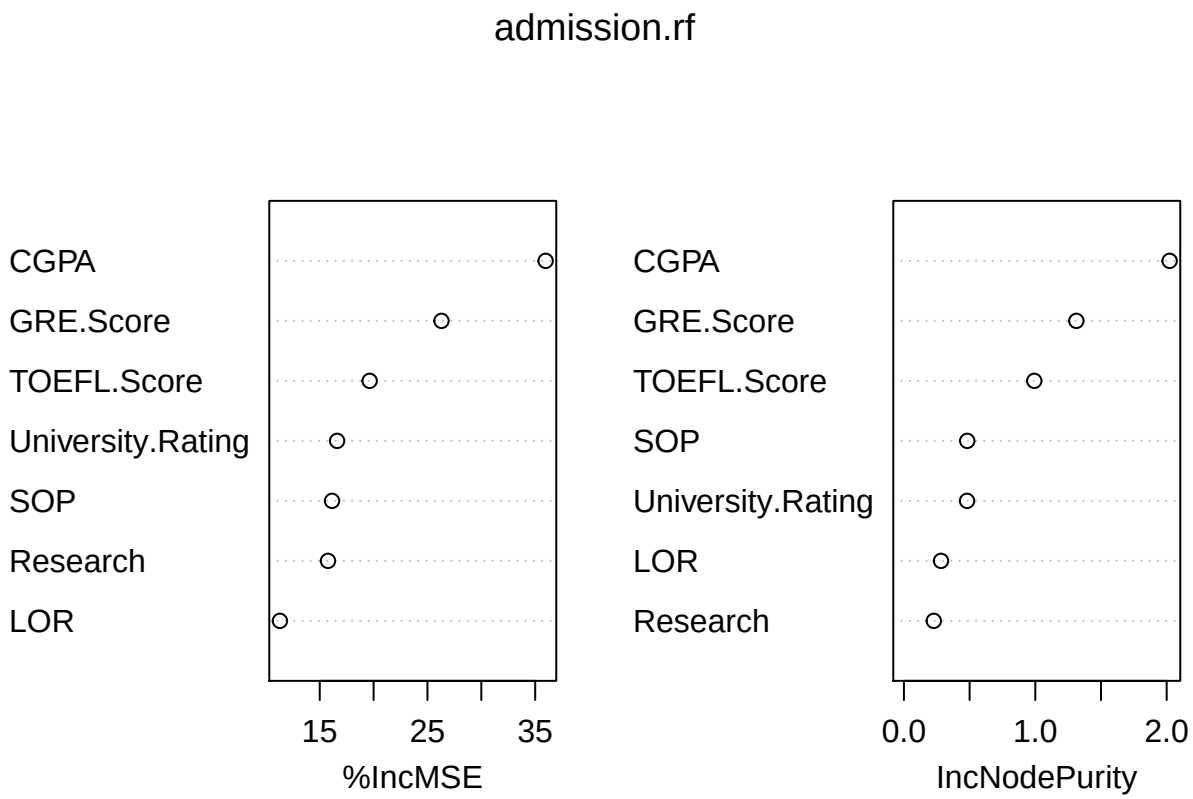
Chance of Admittance Random Forest

```
set.seed(1000101010)
admission.rf <- randomForest(Chance.of.Admit~., data = admissionsTrain, importance = TRUE)
admission.rf
```

```
##
## Call:
##  randomForest(formula = Chance.of.Admit ~ ., data = admissionsTrain,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.003615576
##                     % Var explained: 79.16
```

Since Random Forest uses out-of-bag which is similar to cross validation so no cross validation was performed. We can look at the importance of the variables.

```
varImpPlot(admission.rf)
```



As seen from the Importance Plot the most important variables are CGPA, GRE Score and TOEFL scores

when using chance of admission as a response variable.

Neural Network

We can use a neural network to attempt prediction as well. While this model is more complex so we lose inference, we gain a lot in terms of predictions. We will also remove serial number as a variable.

```
set.seed(420)
# Neural Net

graduateAdmissions.numeric <- admissionsData[, -c(1)] # Remove serial number
graduateAdmissions.normalizedNumeric <- apply(graduateAdmissions.numeric, 2, function(v) (v-min(v)) / (
graduateAdmissions.normalizedNumeric <- data.frame(graduateAdmissions.normalizedNumeric)
graduateAdmissions.normalizedNumeric$Research <- as.factor(graduateAdmissions.normalizedNumeric$Researc
```

Using a 70/30 ratio for splitting our data into training/testing sets, we can cross-validate to improve model performance estimation.

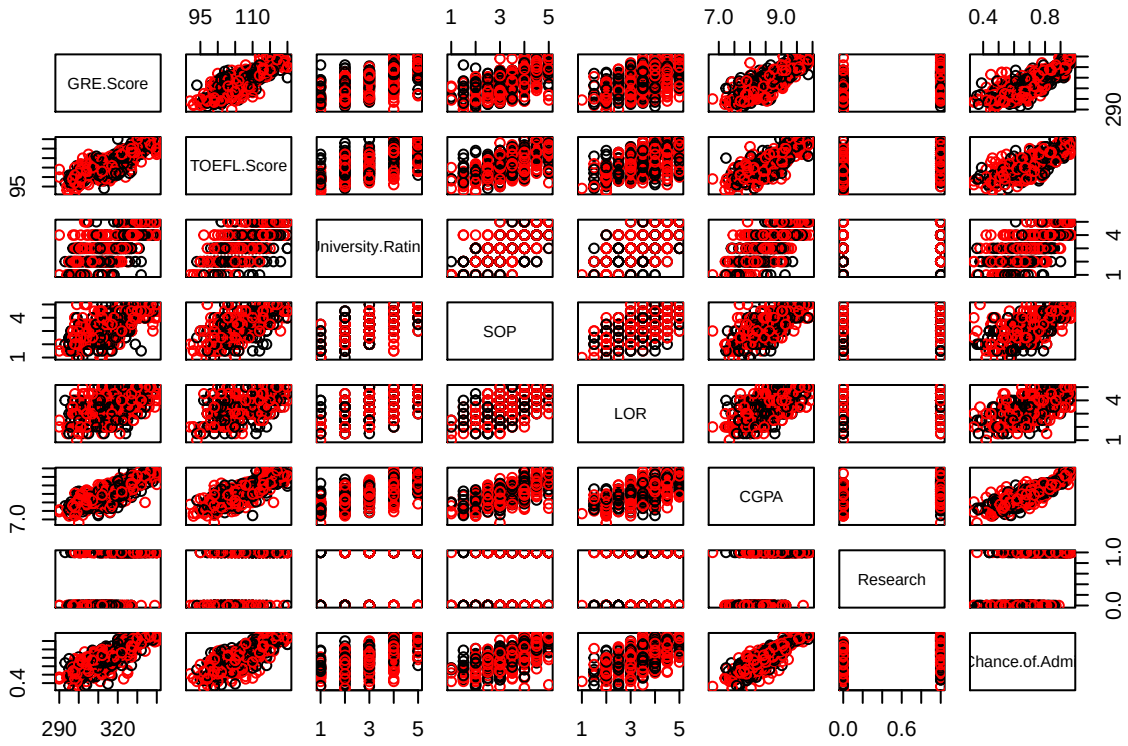
```
# Training/Testing Set
ind <- sample(1:nrow(graduateAdmissions.normalizedNumeric), 350)
train <- graduateAdmissions.normalizedNumeric[ind,]
test <- graduateAdmissions.normalizedNumeric[-ind,]

graduateAdmissions.nnWithTrain <- neuralnet(Research ~ ., data = train, hidden = 5, linear.output = FALS
predicted = data.frame(round(predict(graduateAdmissions.nnWithTrain, test, type = "class"), 0))$X2

table(predicted, actual = test$Research)
```

```
##           actual
## predicted  0  1
##         0 47 21
##         1 20 62
```

```
plot(graduateAdmissions.numeric, col = predicted + 1)
```




```
researchAccuracy <- Accuracy(test$Research, predicted)
researchSensitivity <- Sensitivity(test$Research, predicted)
researchF1 <- F1_Score(test$Research, predicted)
```

```
graduateAdmissions.nnWithTrainAdmit <- neuralnet(Chance.of.Admit ~ GRE.Score + TOEFL.Score + University
predictedChanceOfAdmit = predict(graduateAdmissions.nnWithTrainAdmit, test)
plot(graduateAdmissions.nnWithTrainAdmit)

graduateAdmissions.nnWithTrainAdmit.MSE <- sum((predict(graduateAdmissions.nnWithTrainAdmit, data.frame
```

Turns out the cross-validated MSE of the neural network is very low. So we can predict with great accuracy how a student feels about their particular Grad application.

```
scoreTable <- cbind( (41) / 150, researchF1, researchSensitivity)
colnames(scoreTable)<-c("Misclass", "F1 Score", "Sensitivity")
rownames(scoreTable)<-c("Neural Network (5)")
round(scoreTable,3)
```

```
##                    Misclass F1 Score Sensitivity
## Neural Network (5)     0.273    0.696       0.701
```

```
yHat <- predict(graduateAdmissions.nnWithTrainAdmit, test)
yMax <- max(admissionsData$Chance.of.Admit)
yMin <- min(admissionsData$Chance.of.Admit)

yHatDenormalzed <- yHat * (yMax - yMin) + yMin
yDenormalized <- admissionsData$Chance.of.Admit[-ind]

avgDiff <- sum(abs(yHatDenormalzed - yDenormalized)) / length(yDenormalized)


scoreTable <- cbind( graduateAdmissions.nnWithTrainAdmit.MSE, avgDiff)
```

```
colnames(scoreTable)<-c("MSE", "Avg difference")
rownames(scoreTable)<-c("Neural Network (5)")
round(scoreTable,5)
```

```
##                       MSE Avg difference
## Neural Network (5) 0.0103        0.04378
```

PCA

With Response Variable Chance.of.Admit

The variable we are interested in predicting, Chance.of.Admit, is the 9th variable.

Run PCA on the data and remove the response variable (chance of admit) and the unique identifier (serial number)

```
set.seed(43849)
pca.admin <- prcomp(as.matrix(admissionsData[,-c(1,9)]), scale = TRUE)
summary(pca.admin)
```

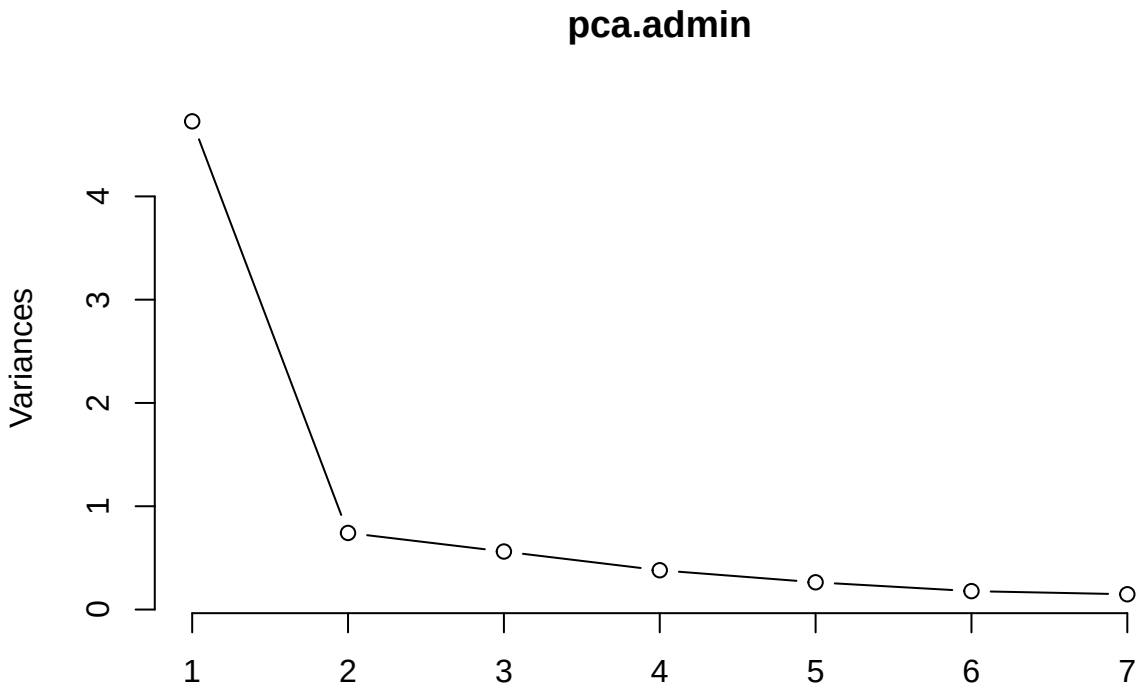
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     2.1740 0.8612 0.74942 0.61674 0.51349 0.42223
## Proportion of Variance 0.6752 0.1060 0.08023 0.05434 0.03767 0.02547
## Cumulative Proportion  0.6752 0.7812 0.86139 0.91573 0.95340 0.97886
##                            PC7
## Standard deviation     0.38464
## Proportion of Variance 0.02114
## Cumulative Proportion  1.00000
```

To choose the number of principal components to keep, we can either use the Kaiser criterian, cumulative proportion/percent of variance, or a scree plot.

Using the Kaiser criterian, we keep all principal components with a standard deviation greater than 1 (since the data is scaled). Hence the Kaiser criterian is telling us to keep only the first principal component.

I will now compare this with a scree plot.

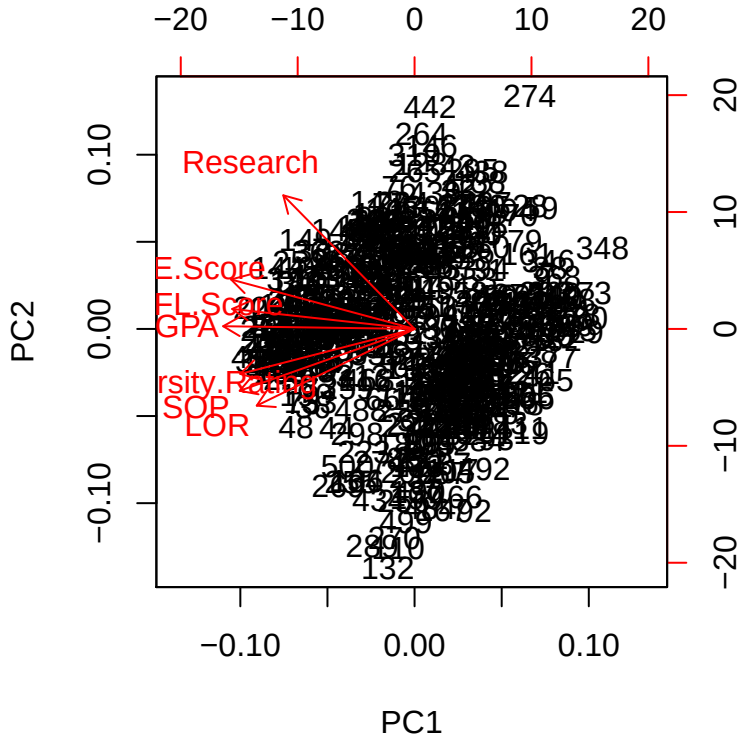
```
plot(pca.admin, type="lines")
```



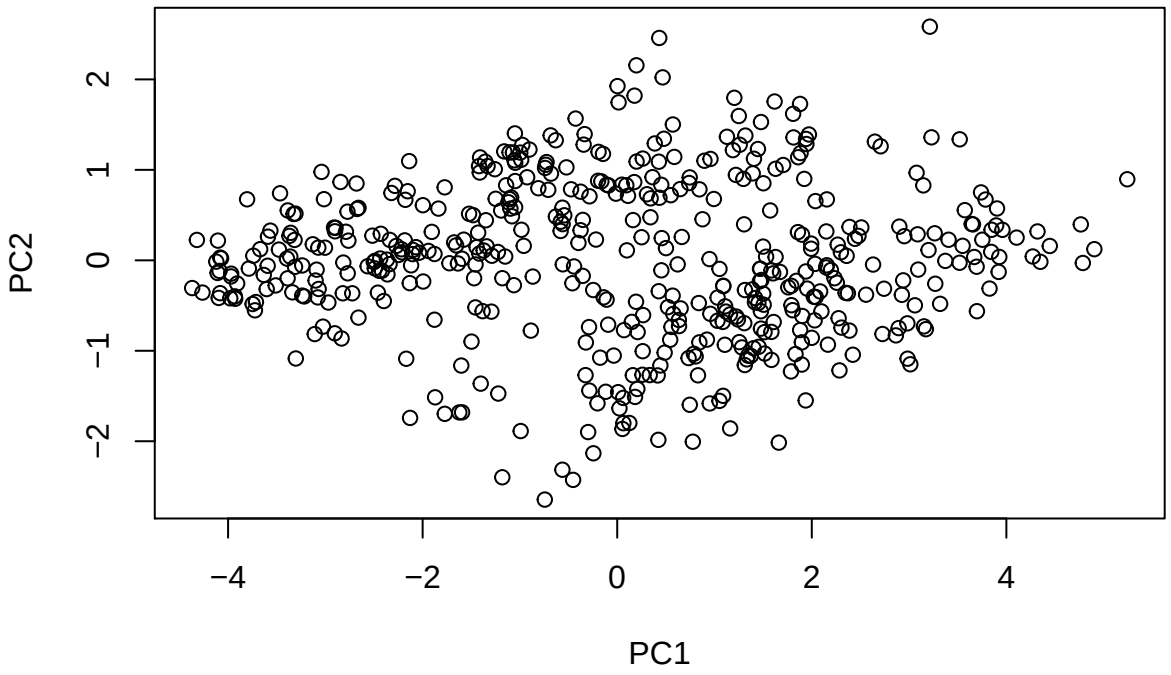
The above scree plot plots the monotonically decreasing eigenvalues and the location of an 'elbow' or plateau indicates the number of principal components. The scree plot suggests probably 2 principal components.

The first two principal components that will be retained explain 78% of the variation in the data. We can now view the data projected onto the components using a biplot.

```
biplot(pca.admin)
```

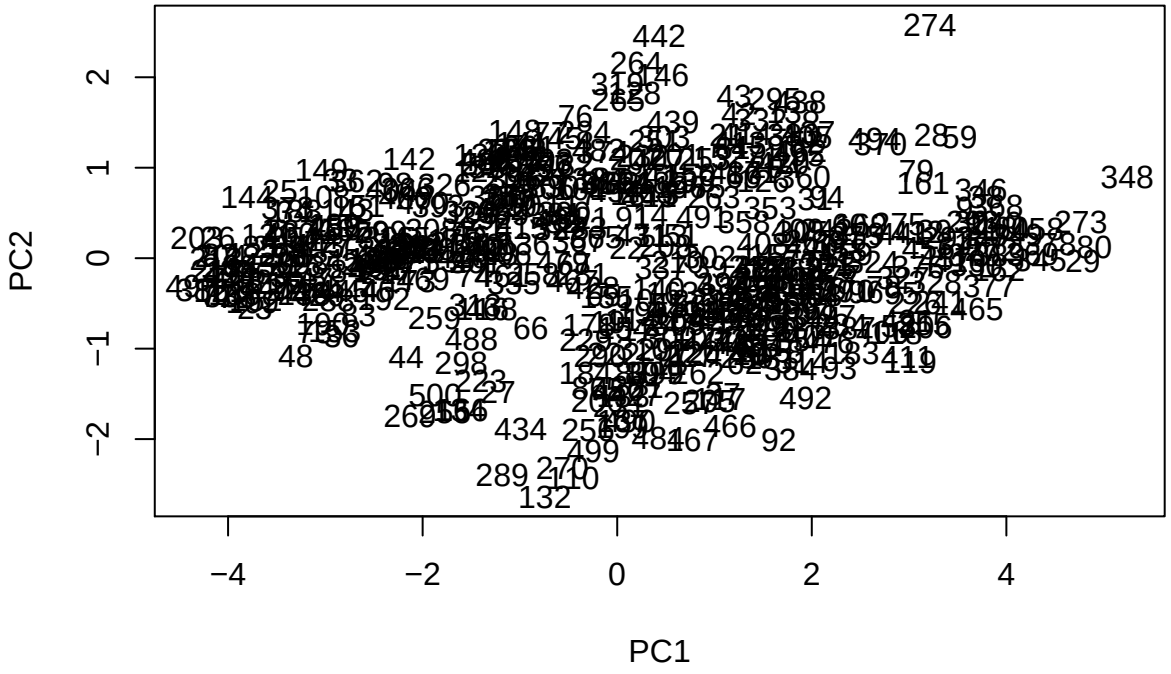



```
plot(pca.admin$x[,1:2])
```



We can put data labels on the biplot by observation number

```
plot(pca.admin$x[,1:2], type = "n")
text(pca.admin$x[,1:2], labels = 1:nrow(admissionsData))
```



It looks like there are two groups in the above principal component plots.

Take a look at the component loadings (eigenvectors) which provide the coefficients of the original variables, rounded to 2 decimal places.

```
round(pca.admin$rotation[,1:2], 2)
```

```
##                      PC1   PC2
```

```
## GRE.Score         -0.40  0.27
## TOEFL.Score       -0.40  0.11
## University.Rating -0.38 -0.25
## SOP               -0.38 -0.34
## LOR               -0.35 -0.43
## CGPA              -0.42  0.02
## Research          -0.29  0.74
```

These are the coefficients of the original variables. The magnitudes are pretty similar for the first component, perhaps with the exception of research. They are also all containing the same sign. This is a little difficult to interpret, but most likely indicates that the first principal component is equally weighting all predictor variables, with the exception of research.

In the second component, the highest magnitude is the research aspect, along with the letter of recommendation. Perhaps this component indicates previous experience a student has. A reference letter most likely comes from someone you have worked with, conducted research with, volunteered with, or TA'd for. Therefore a good reference letter coupled with research experience could be indicative of research and other activities in both academic and non-academic settings.

We can now look at the four students who scored highest on PC1:

```
admissionsData[order(pca.admin$x[,1], decreasing = TRUE)[1:4],1:9]
```

```
##     Serial.No. GRE.Score TOEFL.Score University.Rating SOP LOR CGPA
## 348        348       299          94                 1 1.0 1.0 7.34
## 80          80       294          93                 1 1.5 2.0 7.36
## 29          29       295          93                 1 2.0 2.0 7.20
## 273        273       294          95                 1 1.5 1.5 7.64
##     Research Chance.of.Admit
## 348        0            0.42
## 80         0            0.46
## 29         0            0.46
## 273        0            0.49
```

It is noted that the four students who performed highest on PC1 all had a low belief of their chance of admit. None of them had research, and all had a similar cumulative GPA. In addition, the universities where all rated low (1 to be exact) and the students had similar GRE and TOEFL scores (well below the average). These students in general seem to be ones who are not performing scoring very well across all predictors.

And the four students who scored highest on PC2:

```
admissionsData[order(pca.admin$x[,2], decreasing = TRUE)[1:4], 1:9]
```

```
##     Serial.No. GRE.Score TOEFL.Score University.Rating SOP LOR CGPA
## 274        274       312          99                 1 1.0 1.5 8.01
## 442        442       332         112                 1 1.5 3.0 8.66
## 264        264       324         111                 3 2.5 1.5 8.79
## 146        146       320         113                 2 2.0 2.5 8.64
##     Research Chance.of.Admit
## 274        1            0.52
## 442        1            0.79
## 264        1            0.70
## 146        1            0.81
```

Notice that the four students who performed highest on PC2 all have research experience. In general, these students are scoring better than the students in principal component 1 across the board.

With Response Variable Research

The variable we are interested in predicting, Chance.of.Admit, is the 8th variable.

Run PCA on the data and remove the response variable (research) and the unique identifier (serial number)

```
set.seed(43849)
pca.admin2 <- prcomp(as.matrix(admissionsData[,-c(1,8)]), scale = TRUE)
summary(pca.admin2)
```

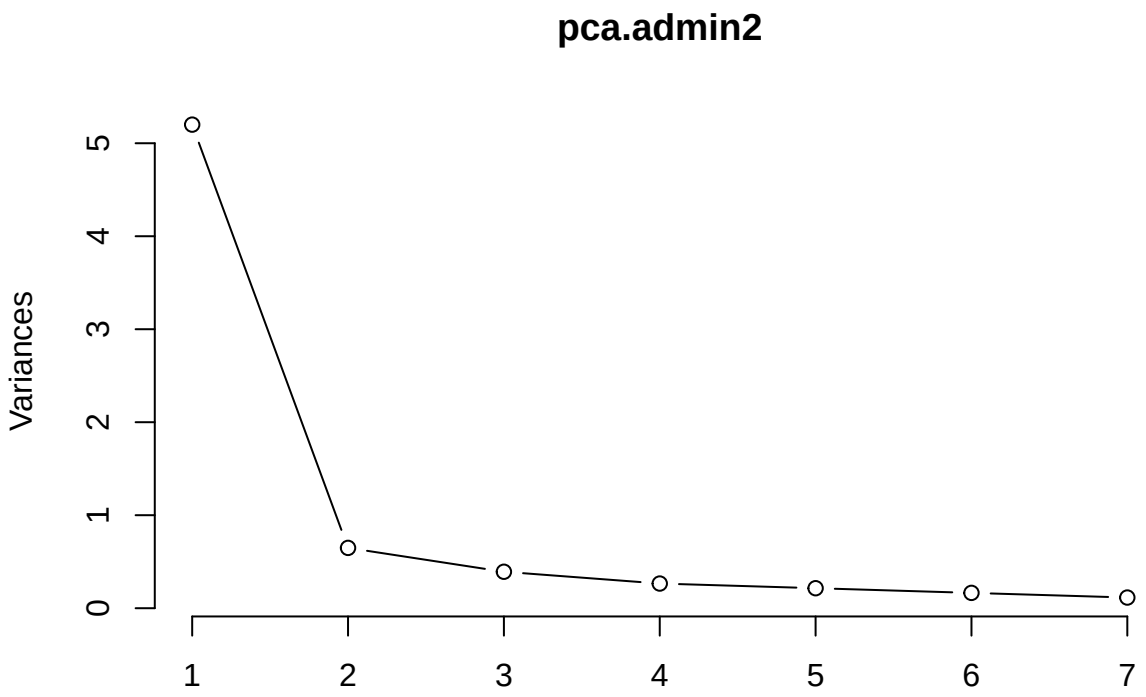
```
## Importance of components:
##                           PC1     PC2     PC3    PC4     PC5     PC6
## Standard deviation     2.2803 0.80529 0.62599 0.5150 0.46369 0.40586
## Proportion of Variance 0.7429 0.09264 0.05598 0.0379 0.03071 0.02353
## Cumulative Proportion  0.7429 0.83549 0.89147 0.9294 0.96008 0.98361
##                           PC7
## Standard deviation     0.33868
## Proportion of Variance 0.01639
## Cumulative Proportion  1.00000
```

To choose the number of principal components to keep, we can either use the Kaiser criterian, cumulative proportion/percent of variance, or a scree plot.

Using the Kaiser criterian, we keep all principal components with a standard deviation greater than 1 (since the data is scaled). Hence the Kaiser criterian is telling us to keep the first principal component.

I will now compare this with a scree plot.

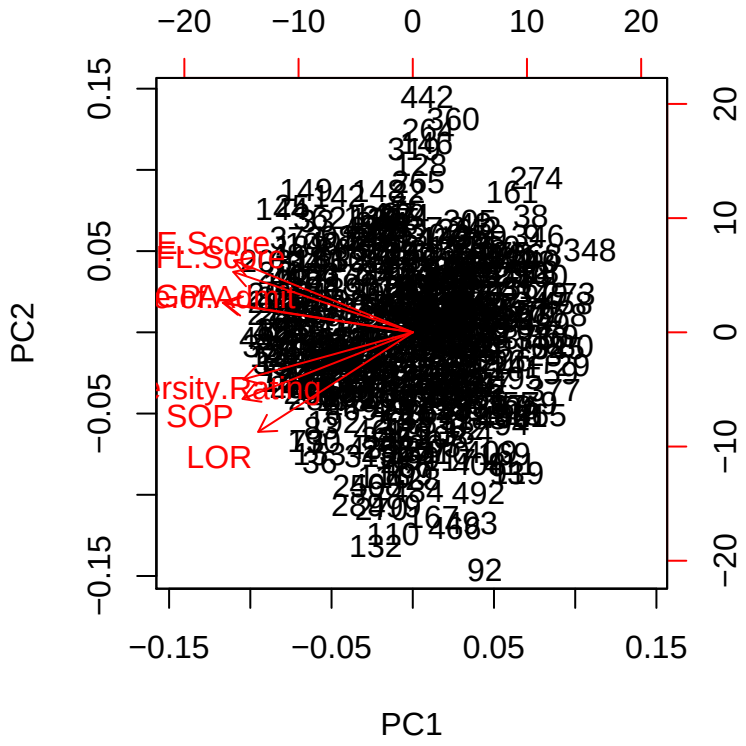
```
plot(pca.admin2, type="lines")
```



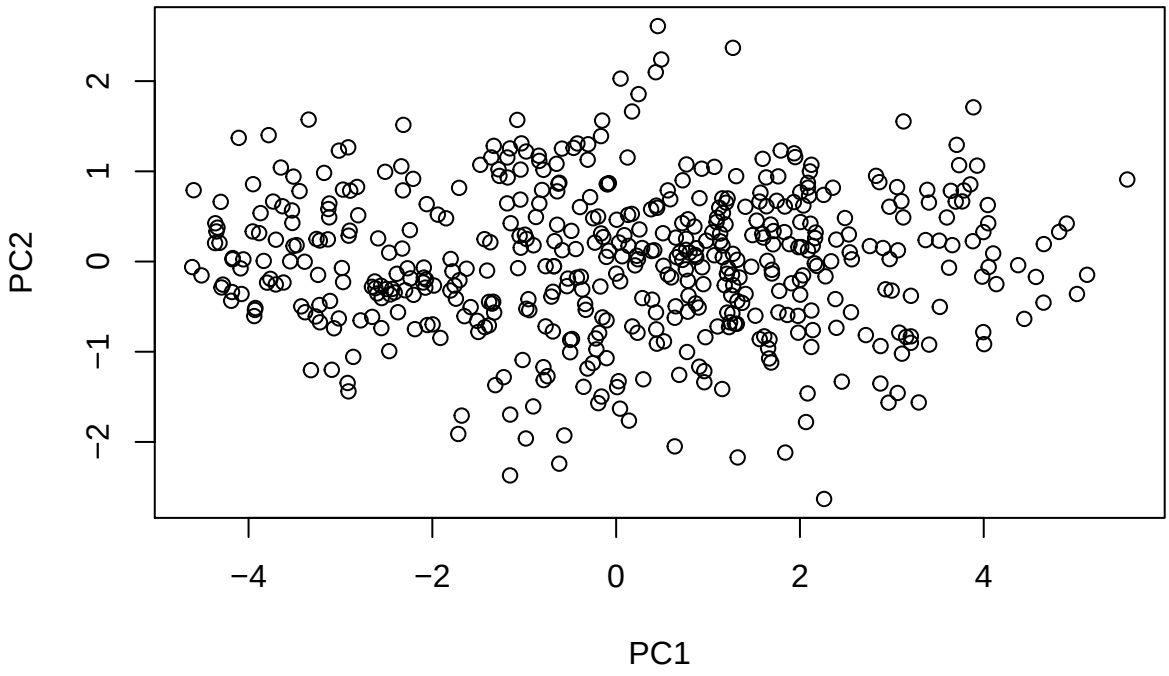
The above scree plot plots the monotonically decreasing eigenvalues and the location of an 'elbow' or plateau indicates the number of principal components. The scree plot suggests probably 2 principal components.

The first two principal components that will be retained explain 84% of the variation in the data. We can now view the data projected onto the components using a biplot.

```
biplot(pca.admin2)
```

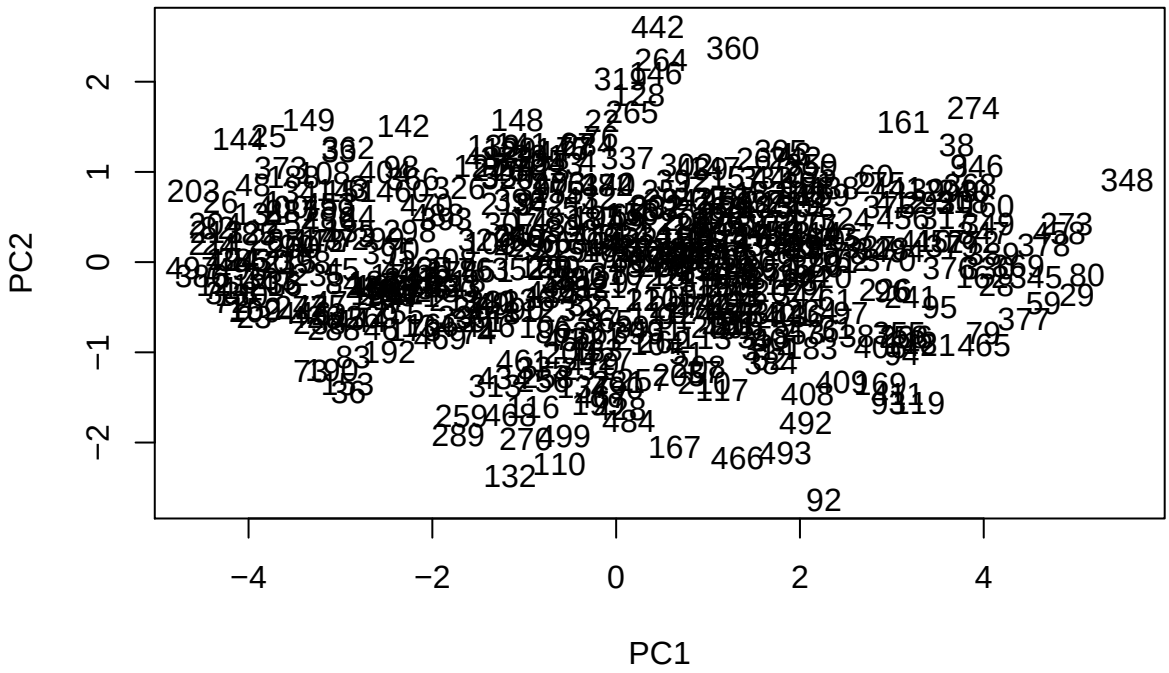



```
plot(pca.admin2$x[,1:2])
```



We can put data labels on the biplot by observation number

```
plot(pca.admin2$x[,1:2], type = "n")
text(pca.admin2$x[,1:2], labels = 1:nrow(admissionsData))
```



It looks like there are two groups in the above principal component plots.

Take a look at the component loadings (eigenvectors) which provide the coefficients of the original variables, rounded to 2 decimal places.

```
round(pca.admin2$rotation[,1:2], 2)
```

```
##                     PC1   PC2
## GRE.Score         -0.38  0.44
## TOEFL.Score       -0.39  0.37
## University.Rating -0.36 -0.29
## SOP               -0.37 -0.40
## LOR               -0.33 -0.61
## CGPA              -0.41  0.18
## Chance.of.Admit   -0.40  0.17
```

These are the coefficients of the original variables. The magnitudes are extremely similar for the first component. They are also all containing the same sign. This is a little difficult to interpret again, but most likely indicates that the first principal component is equally weighting all predictor variables.

In the second component, the highest magnitude is the lettor of recommendation which has a negative sign. Other variables with the same sign include the SOP score and the university rating. Variables of opposite sign with higher magnitude include GRE Score, TOEFL Score, as well as CGPA and Chance of Admit having a lower magnitude. Students who score high on this principal component, likely scored high on their standardized tests.

We can now look at the four students who scored highest on PC1:

```
admissionsData[order(pca.admin2$x[,1], decreasing = TRUE)[1:4],1:9]
```

```
##     Serial.No. GRE.Score TOEFL.Score University.Rating SOP LOR CGPA
## 348        348       299          94                 1 1.0 1.0 7.34
## 80          80       294          93                 1 1.5 2.0 7.36
## 29          29       295          93                 1 2.0 2.0 7.20
## 273        273       294          95                 1 1.5 1.5 7.64
##     Research Chance.of.Admit
```

```
## 348        0             0.42
## 80         0             0.46
## 29         0             0.46
## 273        0             0.49
```

The top four students in this first principal component are the same as the first four students in the previous PC1 (compared using Serial.No.). Even when looking at the loadings, this principal component is very similar to the principal component in the previous section.

And the four students who scored highest on PC2:

```
admissionsData[order(pca.admin2$x[,2], decreasing = TRUE)[1:4], 1:9]
```

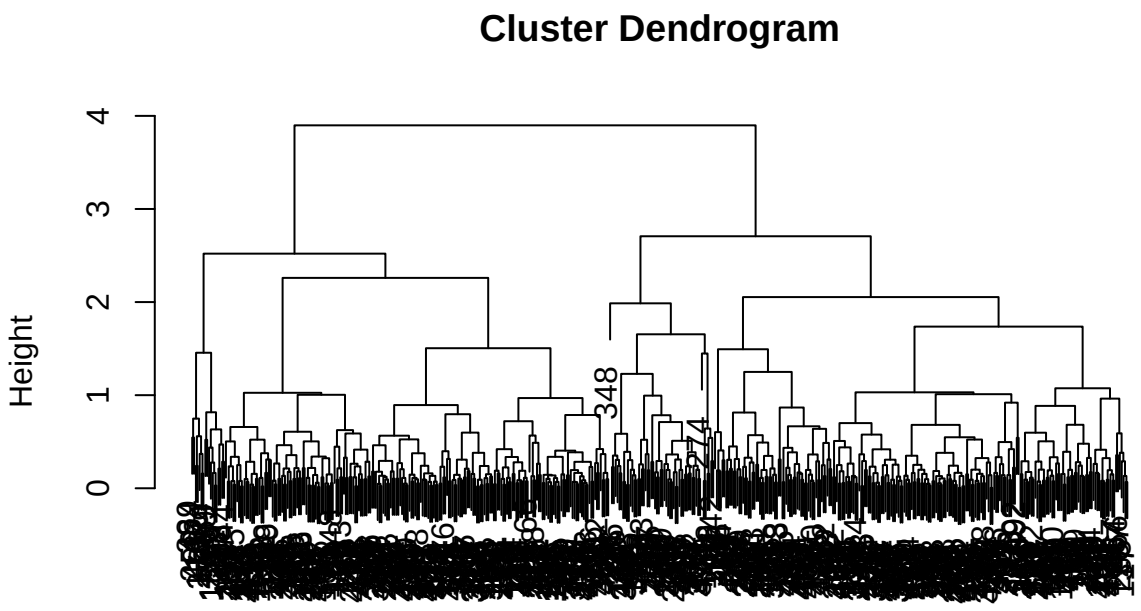
```
##     Serial.No. GRE.Score TOEFL.Score University.Rating SOP LOR CGPA
## 442        442       332         112                 1 1.5 3.0 8.66
## 360        360       321         107                 2 2.0 1.5 8.44
## 264        264       324         111                 3 2.5 1.5 8.79
## 146        146       320         113                 2 2.0 2.5 8.64
##     Research Chance.of.Admit
## 442        1             0.79
## 360        0             0.81
## 264        1             0.70
## 146        1             0.81
```

As hypothesized above, the first four students in PC2 are scoring higher on their standardized tests (GRE.Score and TOEFL.Score). These students are performing the at, or above average on these standardized tests. However, they all have a below average score on SOP, and LOR. The CGPA of the students scoring high on PC2 hovers fairly close to the mean. This proves the initial hypothesis that standardized testing is most important for PC2.

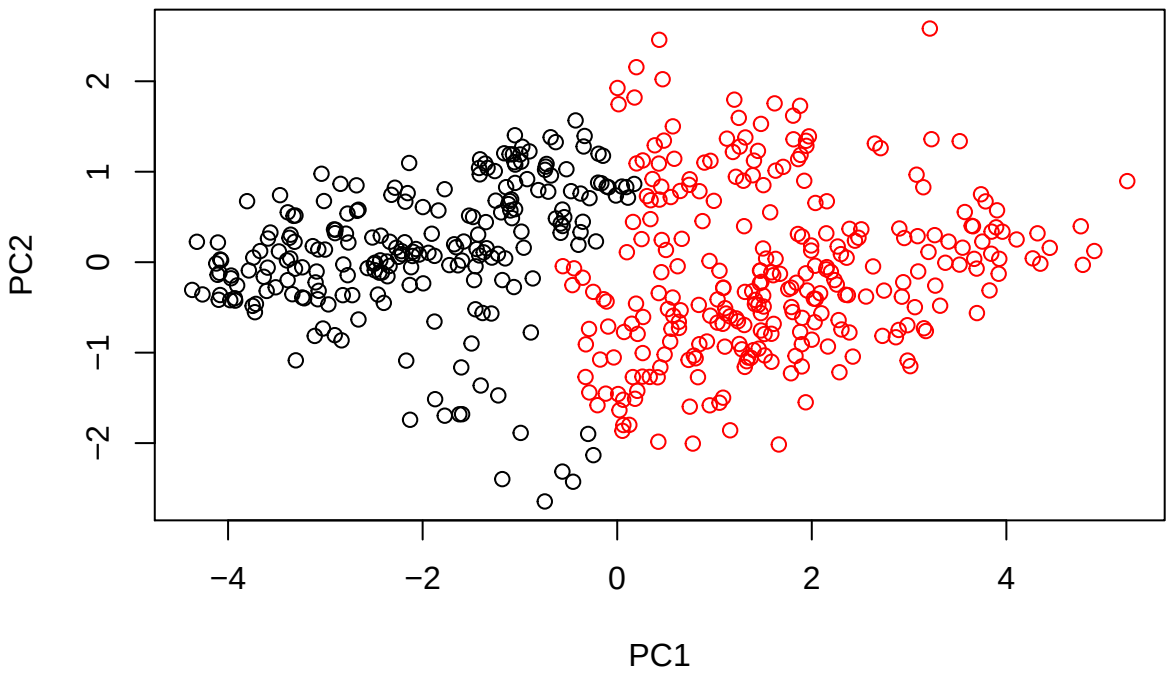
Trying to Cluster on the First Two Principal Components

It appeared that in the first PCA analysis, with the predictor chance.of.Admit removed, there were two groups in the remaining principal component plots. Here we will perform hierarchical clustering to try to find these groups.

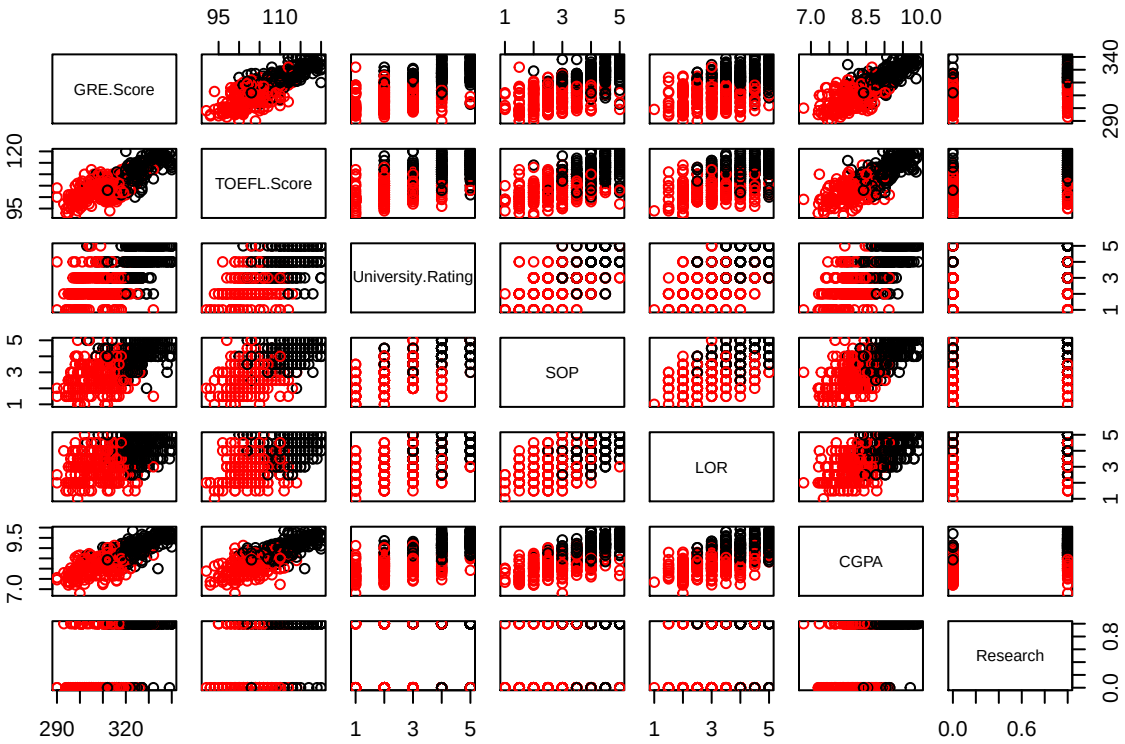
```
set.seed(574847)
clusts <- hclust(dist(pca.admin$x[,1:2]), method="average")
plot(clusts)
```




```
plot(pca.admin$x[,1:2], col=cutree(clusts,2))
```




```
pairs(admissionsData[,-c(1,9)], col=cutree(clusts,2))
```



These are not quiet the groups we noticed by eye. Let's try clustering with a mixture model.

```
#install.packages("mclust")
#install.packages("teigen")
library(mclust)
library(teigen)

mPCA <- Mclust(dist(pca.admin$x[,1:2]), G=1:5, scale = TRUE)
summary(mPCA)

mPCA2 <- Mclust(dist(pca.admin$x[,1:2]), G = 2)
summary(mPCA2)

#could not get the margins to plot the below
#plot(mPCA2)

plot(pca.admin$x[,1:2], col=mPCA2$classification)


set.seed(2521)

#The below takes a long time and does not converge so I am commenting it out, but this is the code that
tPCA <- teigen(as.matrix(dist(pca.admin$x[,1:2])), Gs=1:9, models="all", scale= FALSE, verbose = TRUE)
```

Discriminant Analysis

Discriminant analysis proved to not be a very fitting model as the boundary is not a linear or quadatric boundary.

```r
graduateAdmissions.csv <- read.csv("Admission_Predict_Ver1.1.csv")
admissionsData <- data.frame(graduateAdmissions.csv)
graduateAdmissions.catData <- admissionsData[, -c(1)]
graduateAdmissions.catData$Chance.of.Admit <- factor(admissionsData$Chance.of.Admit > 0.5)
ind <- sample(1:nrow(graduateAdmissions.catData), 350)
train <- graduateAdmissions.catData[ind,]
test <- graduateAdmissions.catData[-ind,]

# Predict Chance of Admit
graduateAdmission.lda <- lda(Chance.of.Admit ~ ., data = train, cv = TRUE)
# Perform LDA
plda <- predict(graduateAdmission.lda, test)

# Classification Table
table(test$Chance.of.Admit, plda$class)
```

```
##
##         FALSE TRUE
##   FALSE     4    3
##   TRUE      1  142
```

```r
# Classification Metrics
LogLoss(plda$posterior[,2], as.numeric(test$Chance.of.Admit) - 1)
```

```
## [1] 0.104894
```

```r
Accuracy(test$Chance.of.Admit, plda$class)
```

```
## [1] 0.9733333
```

```r
Sensitivity(test$Chance.of.Admit, plda$class)
```

```
## [1] 0.5714286
```

```r
# Perform QDA on Chance of admission
graduateAdmission.qda <- qda(Chance.of.Admit ~ ., data = train, cv = TRUE)

# Run predictions
pqda <- predict(graduateAdmission.qda, test)
# Classification table for chance of admission
table(test$Chance.of.Admit, pqda$class )
```

```
##
##         FALSE TRUE
##   FALSE     3    4
##   TRUE      3  140
```

```r
# Metrics
LogLoss(pqda$posterior[,2], as.numeric(test$Chance.of.Admit) - 1)
```

```
## [1] 0.1502773
```

```r
Accuracy(test$Chance.of.Admit, pqda$class)
```

```
## [1] 0.9533333
```

```r
Sensitivity(test$Chance.of.Admit, pqda$class)
```

```
## [1] 0.4285714
```

```r
F1_Score(test$Chance.of.Admit, pqda$class)
```

```
## [1] 0.4615385
```

```r
# Predict Research

# Perform LDA on Research
graduateAdmission.lda <- lda(Research ~ ., data = train, cv = TRUE)
plda <- predict(graduateAdmission.lda, test)

# Classificationtablee
table(test$Research,plda$class)
```

```
##
##      0  1
##   0 35 25
##   1 21 69
```

```r
# Metrics
LogLoss(plda$posterior[,2], test$Research)
```

```
## [1] 0.5920771
```

```r
Accuracy(test$Research,plda$class)
```

```
## [1] 0.6933333
```

```r
Sensitivity(test$Research,plda$class)
```

```
## [1] 0.5833333
```

```r
# Perform QDA on research
graduateAdmission.qda <- qda(Research ~ ., data = train, cv = TRUE)

# Predicr research with QDA
pqda <- predict(graduateAdmission.qda, test)

# Class table
table(test$Research, pqda$class)
```

```
##
##      0  1
##   0 24 36
##   1 15 75
```

```r
# Metrics
LogLoss(pqda$posterior[,2], test$Research)
```

```
## [1] 0.7923724
```

```r
Accuracy(test$Research, pqda$class)
```

```
## [1] 0.66
```

```r
Sensitivity(test$Research, pqda$class)
```

```
## [1] 0.4
```

```r
F1_Score(test$Research, pqda$class)
```

```
## [1] 0.4848485
```