Project: College Dataset

Stefano Cattonar, Ines El Gataa, Andrija Nicić, Angelica Rota

2025-02-10

Contents


```r
library(dplyr)
library(corrplot)
library(caret)
```

Introduction

This project aims to analyze the `College` dataset from *An Introduction to Statistical Learning* to explore the factors that influence the number of applications received by colleges, represented by the response variable `Apps`. The dataset provides comprehensive information on various attributes of U.S. colleges, such as tuition costs, room and board expenses, financial aid, and student demographics. By examining these factors, we seek to better understand the trends and drivers behind college application rates.

Preprocessing

Data Examination

```r
College = read.csv("https://www.statlearning.com/s/College.csv", header = TRUE)
help(head)
head(College)
```

```
##                              X Private Apps Accept Enroll Top10perc Top25perc
## 1 Abilene Christian University     Yes 1660   1232    721        23        52
## 2           Adelphi University     Yes 2186   1924    512        16        29
## 3               Adrian College     Yes 1428   1097    336        22        50
## 4          Agnes Scott College     Yes  417    349    137        60        89
## 5    Alaska Pacific University     Yes  193    146     55        16        44
## 6            Albertson College     Yes  587    479    158        38        62
##   F.Undergrad P.Undergrad Outstate Room.Board Books Personal PhD Terminal
## 1        2885         537     7440       3300   450     2200  70       78
## 2        2683        1227    12280       6450   750     1500  29       30
## 3        1036          99    11250       3750   400     1165  53       66
## 4         510          63    12960       5450   450      875  92       97
## 5         249         869     7560       4120   800     1500  76       72
## 6         678          41    13500       3335   500      675  67       73
##   S.F.Ratio perc.alumni Expend Grad.Rate
## 1      18.1          12   7041        60
## 2      12.2          16  10527        56
## 3      12.9          30   8735        54
## 4       7.7          37  19016        59
## 5      11.9           2  10922        15
## 6       9.4          11   9727        55
```

`X`: name of the college

`Private`: A factor with levels No and Yes indicating private or public university

`Accept`: Number of applications accepted

`Enroll`: Number of new students enrolled

`Top10perc`: Percentage new students from top 10% of high school class

`Top25perc`: Percentage new students from top 25% of high school class

`F.Undergrad`: Number of full time undergraduates

`P.Undergrad`: Number of part time undergraduates

`Outstate`: Out-of-state tuition

`Room.Board`: Room and board costs

`Books`: Estimated book costs

`Personal`: Estimated personal spending

`PhD`: Percentage of faculty with Ph.D.'s

`Terminal`: Percentage of faculty with terminal degree

`S.F.Ratio`: Student/faculty ratio

`perc.alumni`: Percentage alumni who donate

`Expend`: Instructional expenditure per student

`Grad.Rate`: Graduation rate

Response variable

`Apps`: Number of applications received

Data Exploration

```
summary(College)
```

```
##       X               Private              Apps           Accept     
##  Length:777         Length:777         Min.   :   81   Min.   :   72  
##  Class :character   Class :character   1st Qu.:  776   1st Qu.:  604  
##  Mode  :character   Mode  :character   Median : 1558   Median : 1110  
##                                        Mean   : 3002   Mean   : 2019  
##                                        3rd Qu.: 3624   3rd Qu.: 2424  
##                                        Max.   :48094   Max.   :26330  
##      Enroll       Top10perc       Top25perc      F.Undergrad   
##  Min.   :  35   Min.   : 1.00   Min.   :  9.0   Min.   :  139  
##  1st Qu.: 242   1st Qu.:15.00   1st Qu.: 41.0   1st Qu.:  992  
##  Median : 434   Median :23.00   Median : 54.0   Median : 1707  
##  Mean   : 780   Mean   :27.56   Mean   : 55.8   Mean   : 3700  
##  3rd Qu.: 902   3rd Qu.:35.00   3rd Qu.: 69.0   3rd Qu.: 4005  
##  Max.   :6392   Max.   :96.00   Max.   :100.0   Max.   :31643  
##   P.Undergrad         Outstate       Room.Board       Books       
##  Min.   :    1.0   Min.   : 2340   Min.   :1780   Min.   :  96.0  
##  1st Qu.:   95.0   1st Qu.: 7320   1st Qu.:3597   1st Qu.: 470.0  
##  Median :  353.0   Median : 9990   Median :4200   Median : 500.0  
##  Mean   :  855.3   Mean   :10441   Mean   :4358   Mean   : 549.4  
##  3rd Qu.:  967.0   3rd Qu.:12925   3rd Qu.:5050   3rd Qu.: 600.0  
##  Max.   :21836.0   Max.   :21700   Max.   :8124   Max.   :2340.0  
##     Personal         PhD            Terminal       S.F.Ratio    
##  Min.   : 250   Min.   :  8.00   Min.   : 24.0   Min.   : 2.50  
##  1st Qu.: 850   1st Qu.: 62.00   1st Qu.: 71.0   1st Qu.:11.50  
##  Median :1200   Median : 75.00   Median : 82.0   Median :13.60  
##  Mean   :1341   Mean   : 72.66   Mean   : 79.7   Mean   :14.09  
##  3rd Qu.:1700   3rd Qu.: 85.00   3rd Qu.: 92.0   3rd Qu.:16.50  
##  Max.   :6800   Max.   :103.00   Max.   :100.0   Max.   :39.80  
##   perc.alumni        Expend        Grad.Rate    
```

```
##  Min.   : 0.00   Min.   : 3186   Min.   : 10.00
##  1st Qu.:13.00   1st Qu.: 6751   1st Qu.: 53.00
##  Median :21.00   Median : 8377   Median : 65.00
##  Mean   :22.74   Mean   : 9660   Mean   : 65.46
##  3rd Qu.:31.00   3rd Qu.:10830   3rd Qu.: 78.00
##  Max.   :64.00   Max.   :56233   Max.   :118.00
```

Observations

1. We can note that the `X` column will yield no usable information.

```r
College <- select(College, -X)
```

2. We have to exclude also `Accept` and `Enroll` to prevent data leakage.

```r
College <- select(College, -c(Accept, Enroll))

summary(College)
```

```
##    Private               Apps          Top10perc       Top25perc
##  Length:777         Min.   :   81   Min.   : 1.00   Min.   :  9.0
##  Class :character   1st Qu.:  776   1st Qu.:15.00   1st Qu.: 41.0
##  Mode  :character   Median : 1558   Median :23.00   Median : 54.0
##                     Mean   : 3002   Mean   :27.56   Mean   : 55.8
##                     3rd Qu.: 3624   3rd Qu.:35.00   3rd Qu.: 69.0
##                     Max.   :48094   Max.   :96.00   Max.   :100.0
##   F.Undergrad     P.Undergrad         Outstate       Room.Board
##  Min.   :  139   Min.   :    1.0   Min.   : 2340   Min.   :1780
##  1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320   1st Qu.:3597
##  Median : 1707   Median :  353.0   Median : 9990   Median :4200
##  Mean   : 3700   Mean   :  855.3   Mean   :10441   Mean   :4358
##  3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925   3rd Qu.:5050
##  Max.   :31643   Max.   :21836.0   Max.   :21700   Max.   :8124
##      Books           Personal         PhD            Terminal
##  Min.   :  96.0   Min.   : 250   Min.   :  8.00   Min.   : 24.0
##  1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00   1st Qu.: 71.0
##  Median : 500.0   Median :1200   Median : 75.00   Median : 82.0
##  Mean   : 549.4   Mean   :1341   Mean   : 72.66   Mean   : 79.7
##  3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00   3rd Qu.: 92.0
##  Max.   :2340.0   Max.   :6800   Max.   :103.00   Max.   :100.0
##    S.F.Ratio      perc.alumni        Expend        Grad.Rate
##  Min.   : 2.50   Min.   : 0.00   Min.   : 3186   Min.   : 10.00
##  1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751   1st Qu.: 53.00
##  Median :13.60   Median :21.00   Median : 8377   Median : 65.00
##  Mean   :14.09   Mean   :22.74   Mean   : 9660   Mean   : 65.46
##  3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830   3rd Qu.: 78.00
##  Max.   :39.80   Max.   :64.00   Max.   :56233   Max.   :118.00
```

3. All the variables does not have negative values.

4. The variables does not have NA, so we don't need to replace the missing values

5. We want to check if we have some duplicates

```
idx <- which(duplicated(College))
idx
```

```
## integer(0)
```

6. The variable `Private` is a categorical one so we convert its values.

```
College$Private <- as.factor(College$Private)

College$Private <- as.numeric(College$Private) - 1  # "Yes" is 1, "No" is 0

College$Private[11:30]
```

```
##  [1] 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 1
```

7. We can note that `Private` is not balanced

```
index_dataset_public <- which(College$Private == 0)

index_dataset_private <- which(College$Private == 1)

private_proportion <- (length(index_dataset_private) / length(index_dataset_public))

table(College$Private)
```

```
##
##   0   1
## 212 565
```

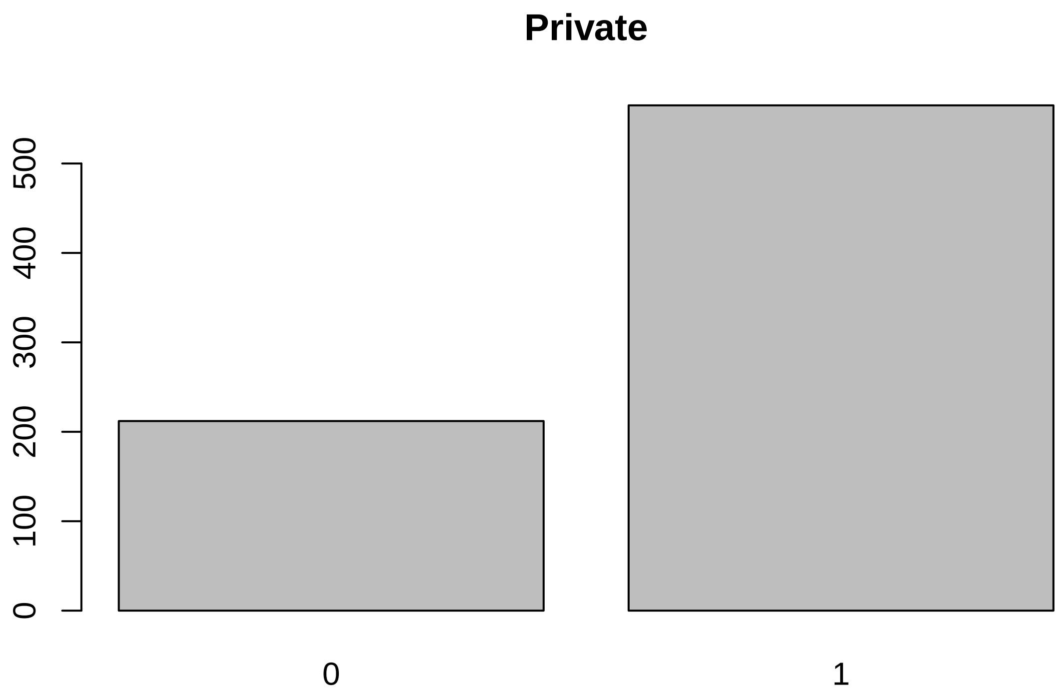
```
private_proportion
```

```
## [1] 2.665094
```

We can see from `private_proportion` that "Yes" is 2.66 times more frequent than "No".

Now we can see the barplot

```
barplot(table(College$Private), main="Private")
```

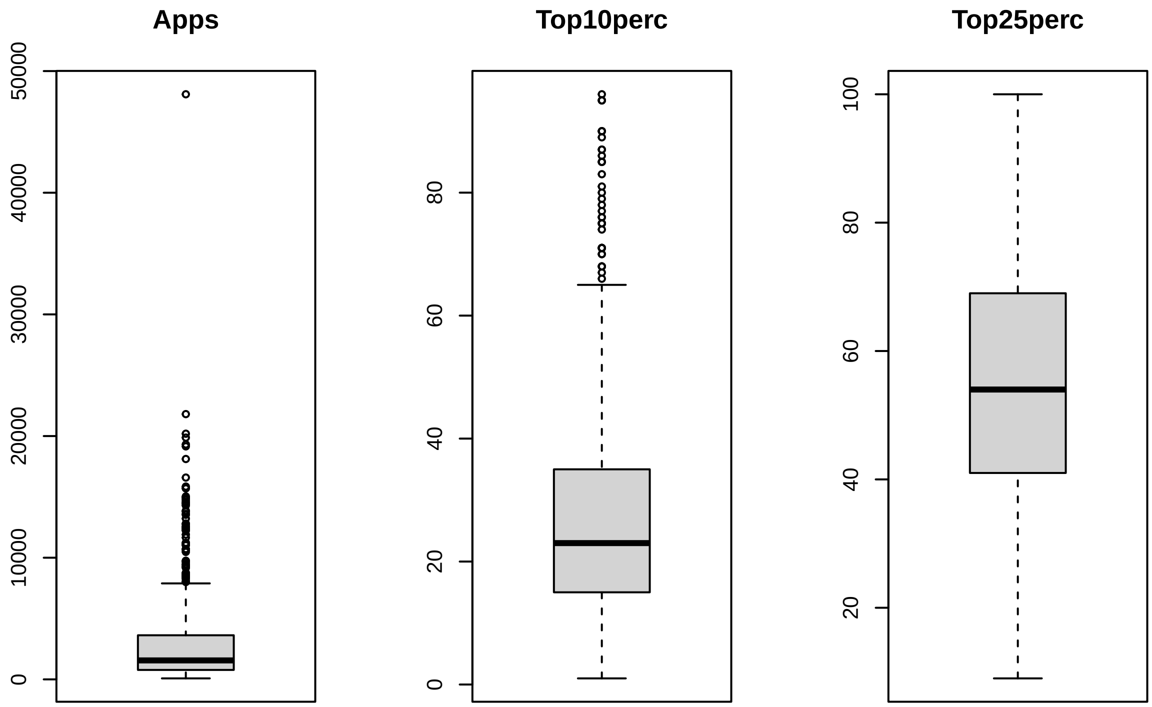


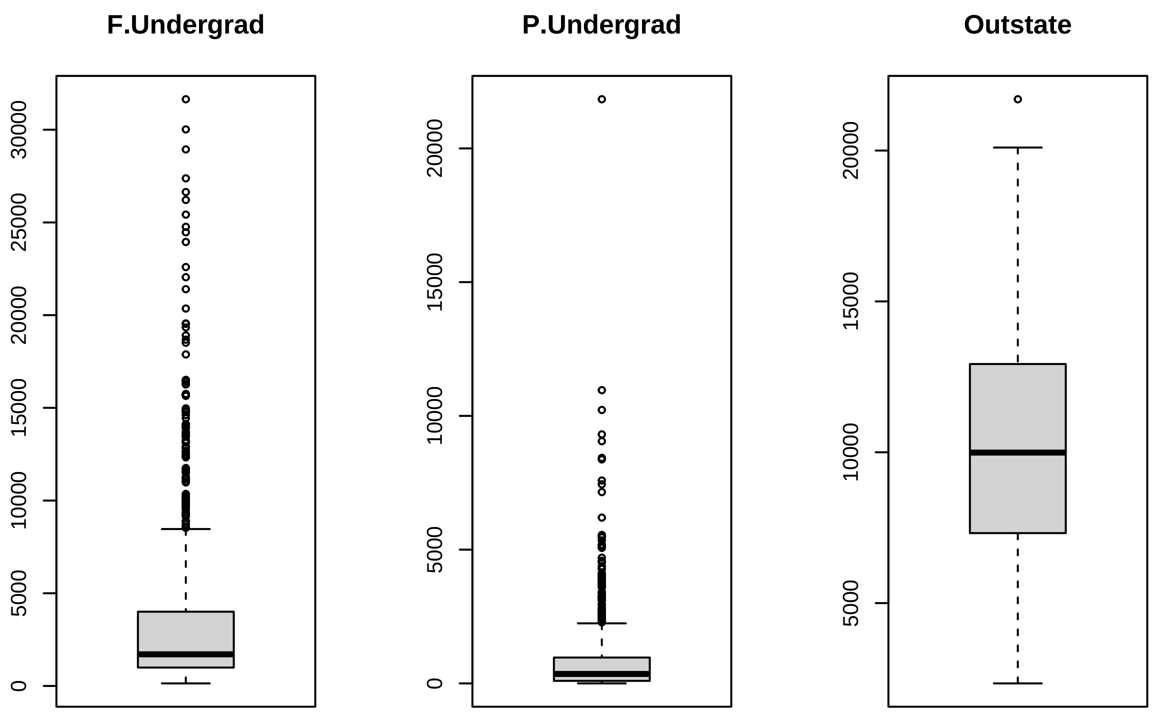
Variable visualization

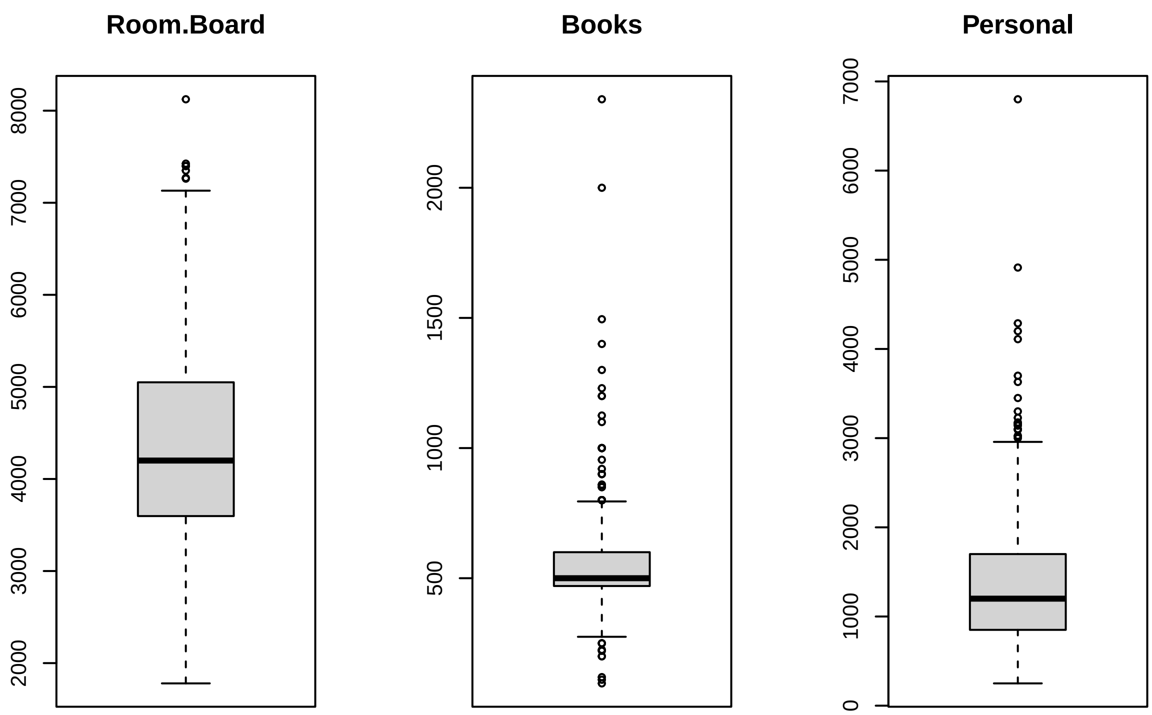
Now we can visualize the variables and their distributions.

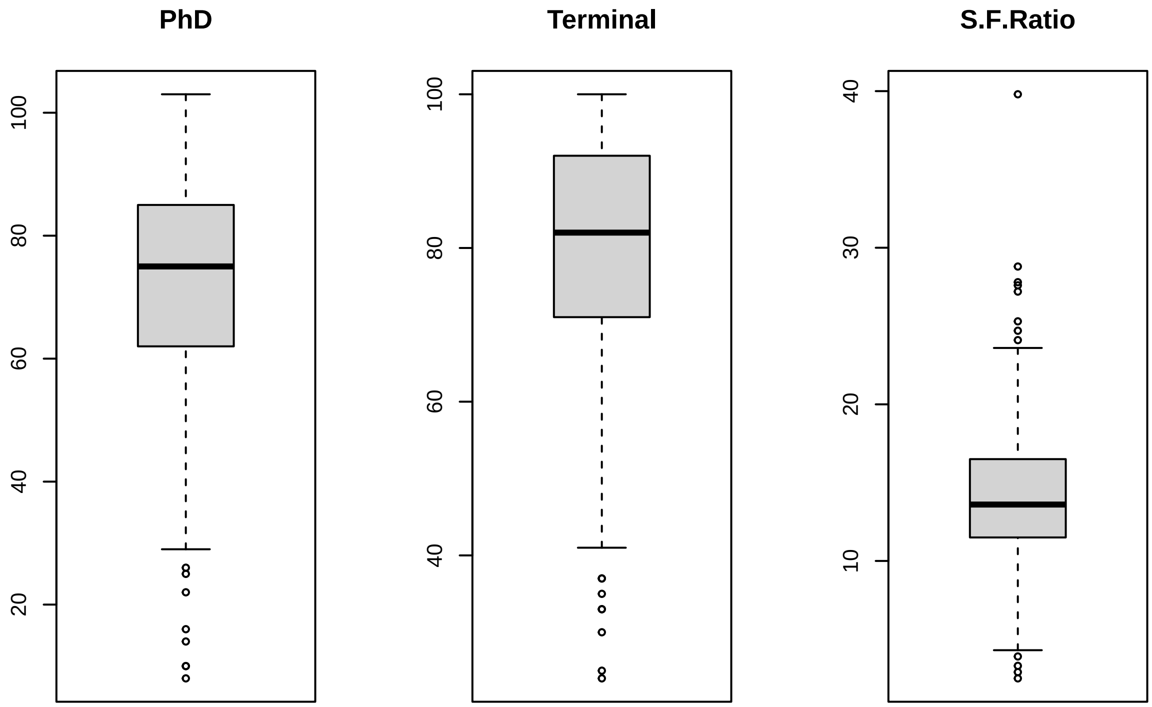
Boxplot, excluded `Private` that is a categorical variable

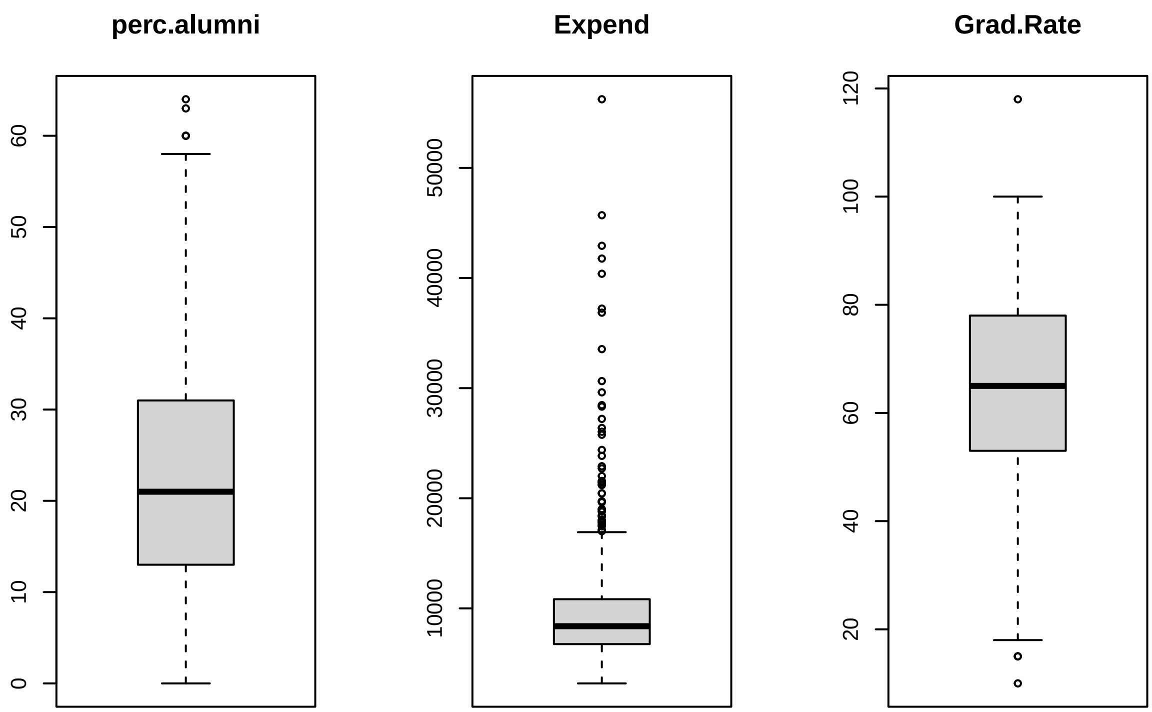
```
par(mfrow=c(1,3))
for (i in 1:length(College)) {
  if ( names(College[i]) != "Private"){
    boxplot(College[,i], main=names(College[i]))
  }
}
```


Apps
50000
40000
30000
20000
10000
0
Top10perc
80
60
40
20
0
Top25perc
100
80
60
40
20


F.Undergrad
P.Undergrad
Outstate
0
5000
10000
15000
20000
25000
30000

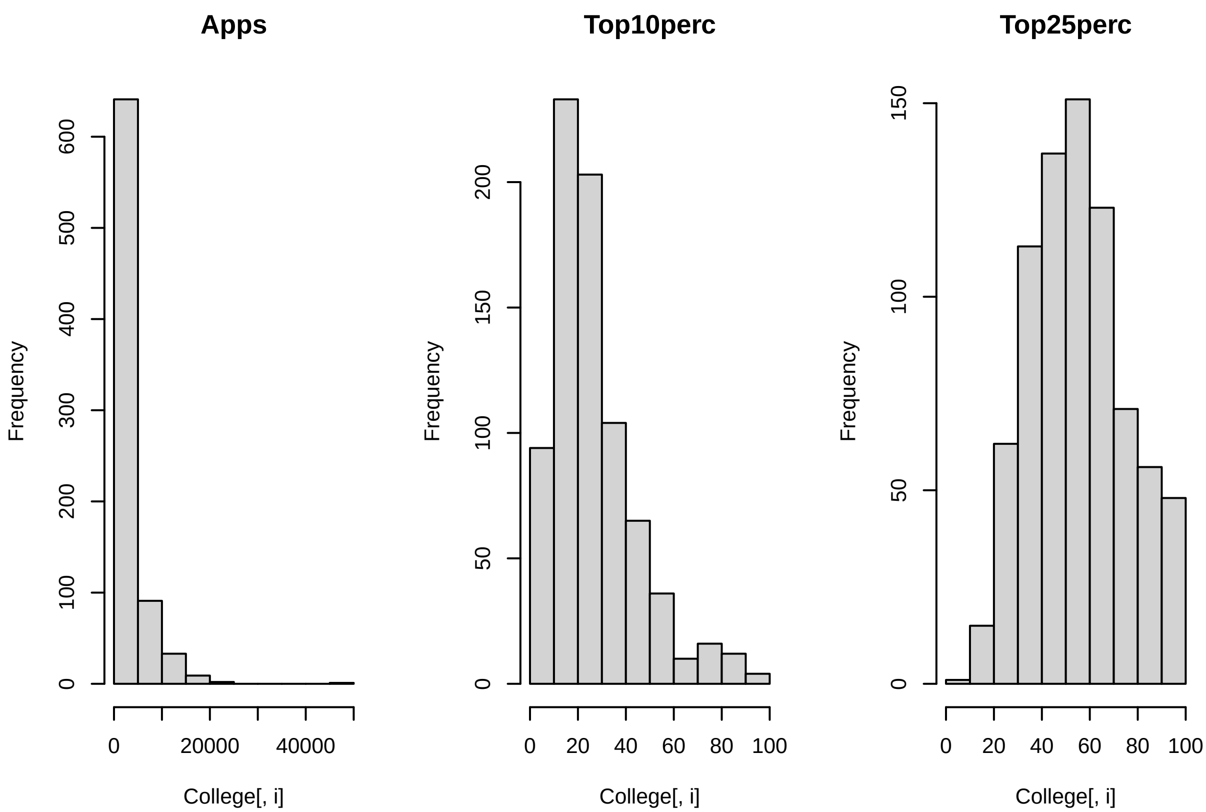

Room.Board
Books
Personal
8000
7000
6000
5000
4000
3000
2000
2000
1500
1000
500
7000
6000
5000
4000
3000
2000
1000
0

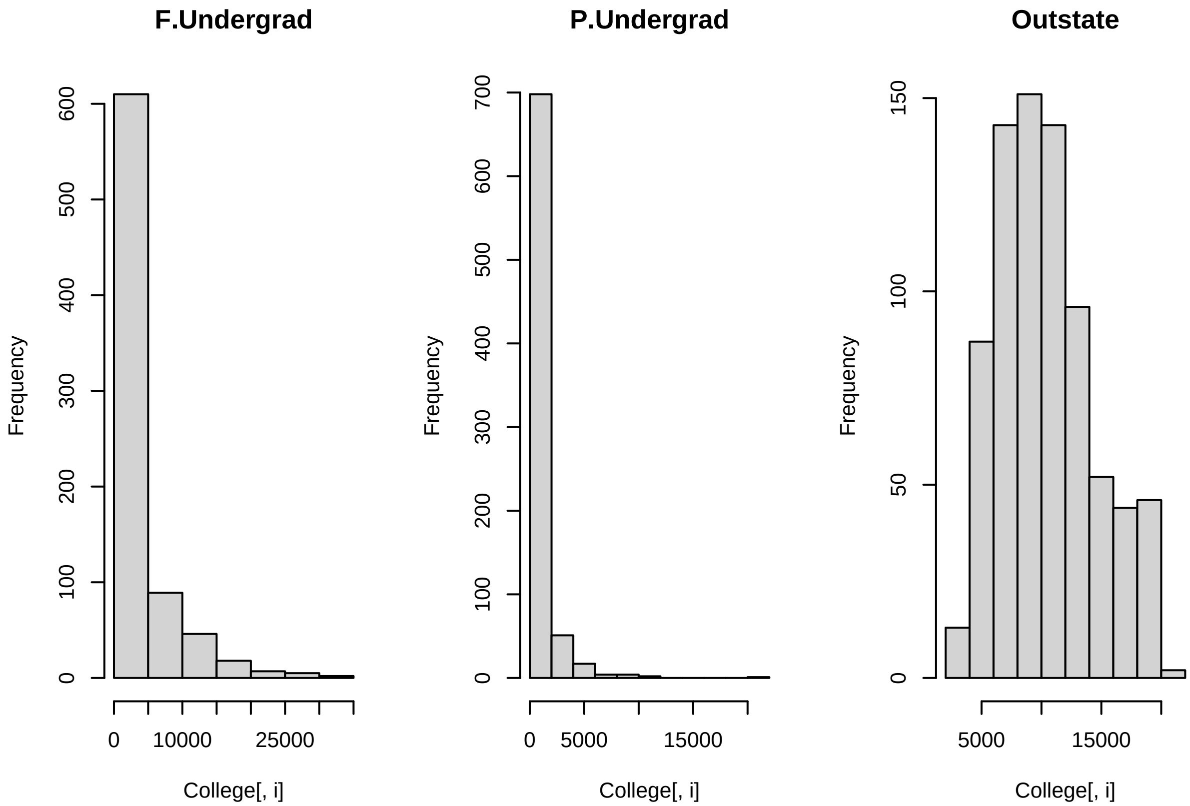

PhD
Terminal
S.F.Ratio
20
40
60
80
100
40
60
80
100
10
20
30
40

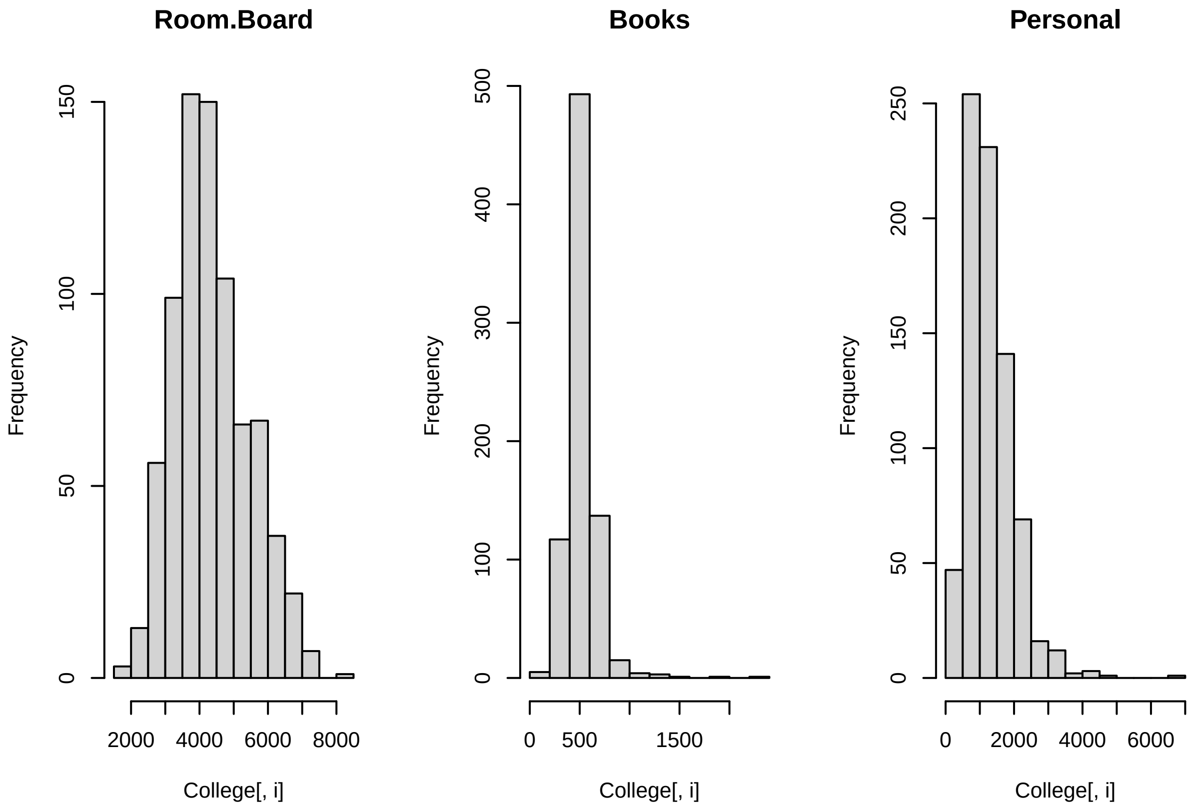


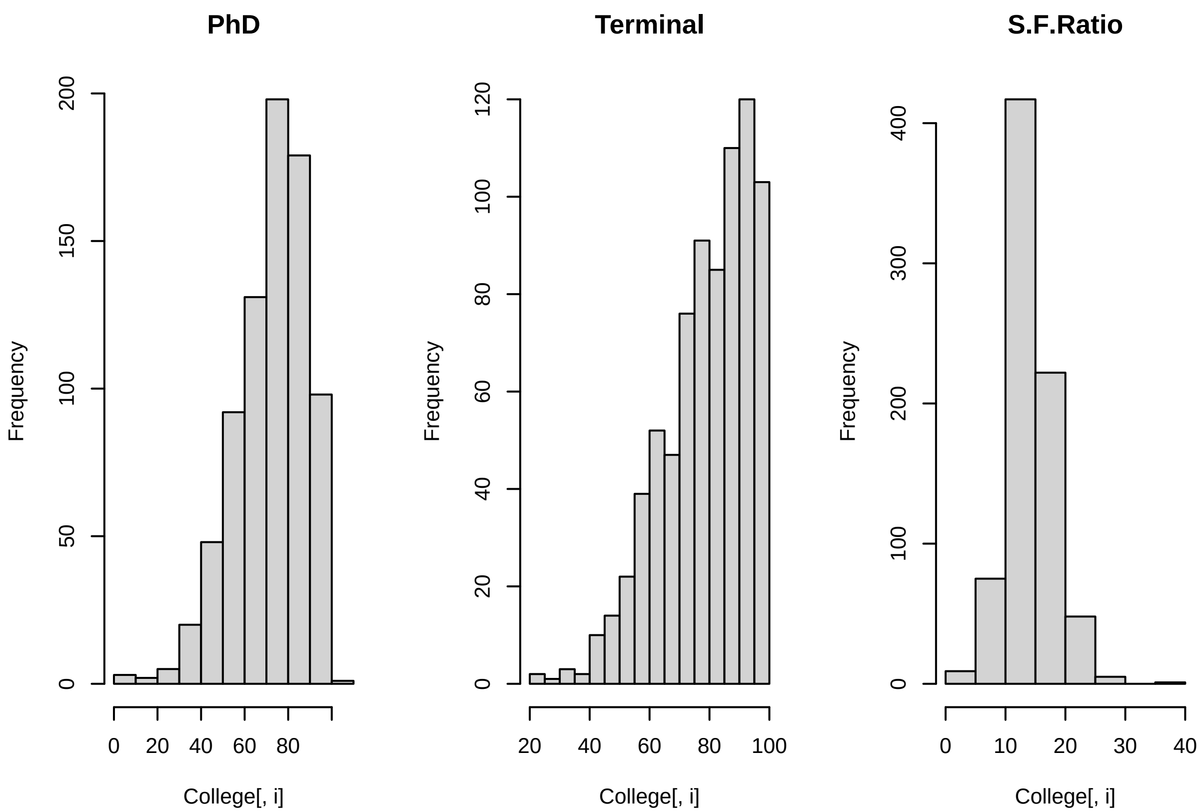
Histogram, excluded `Private` that is a categorical variable

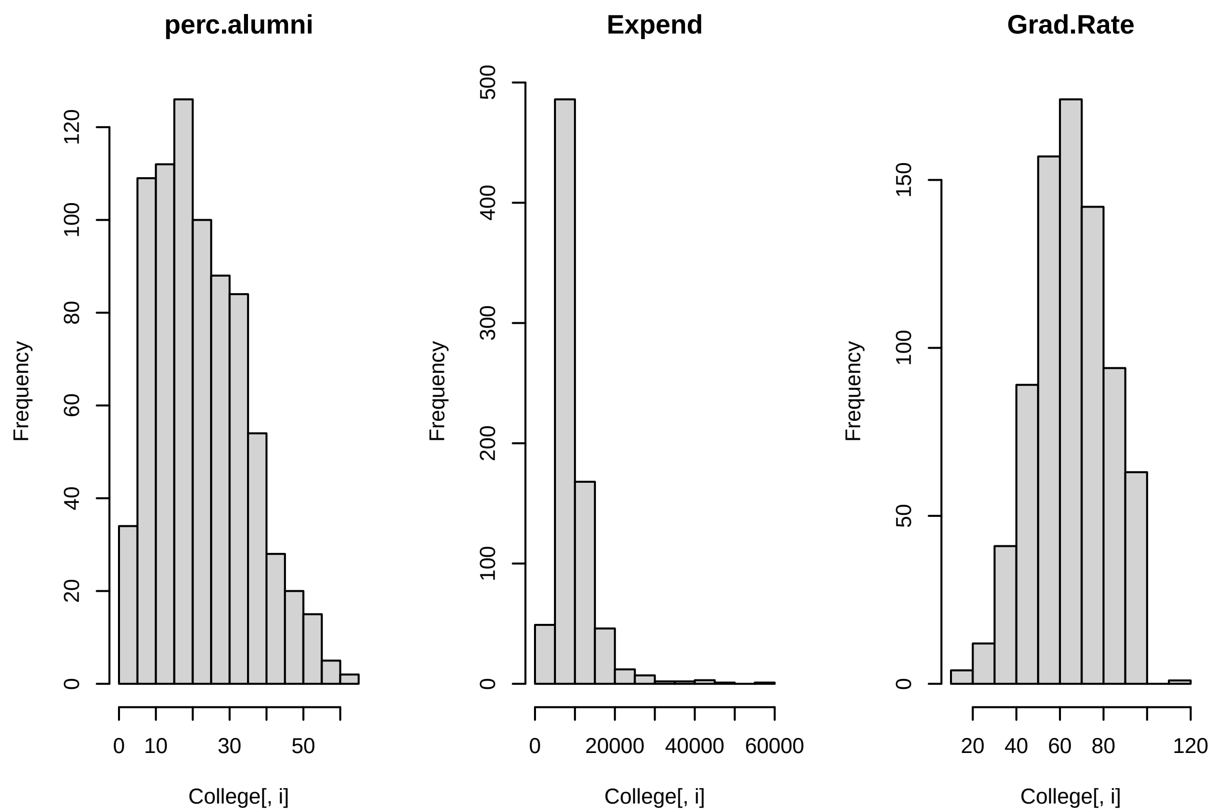
```
par(mfrow=c(1,3))
for (i in 1:length(College)) {
  if ( names(College[i]) != "Private"){
    hist(College[,i], main=names(College[i]))
  }
}
```


Apps
Top10perc
Top25perc
Frequency
College[, i]
0
20000
40000
100
200
300
400
500
600
0
20
40
60
80
100
50
150


F.Undergrad
P.Undergrad
Outstate
Frequency
0
100
200
300
400
500
600
700
0
10000
25000
0
5000
15000
5000
15000
0
50
100
150
College[, i]

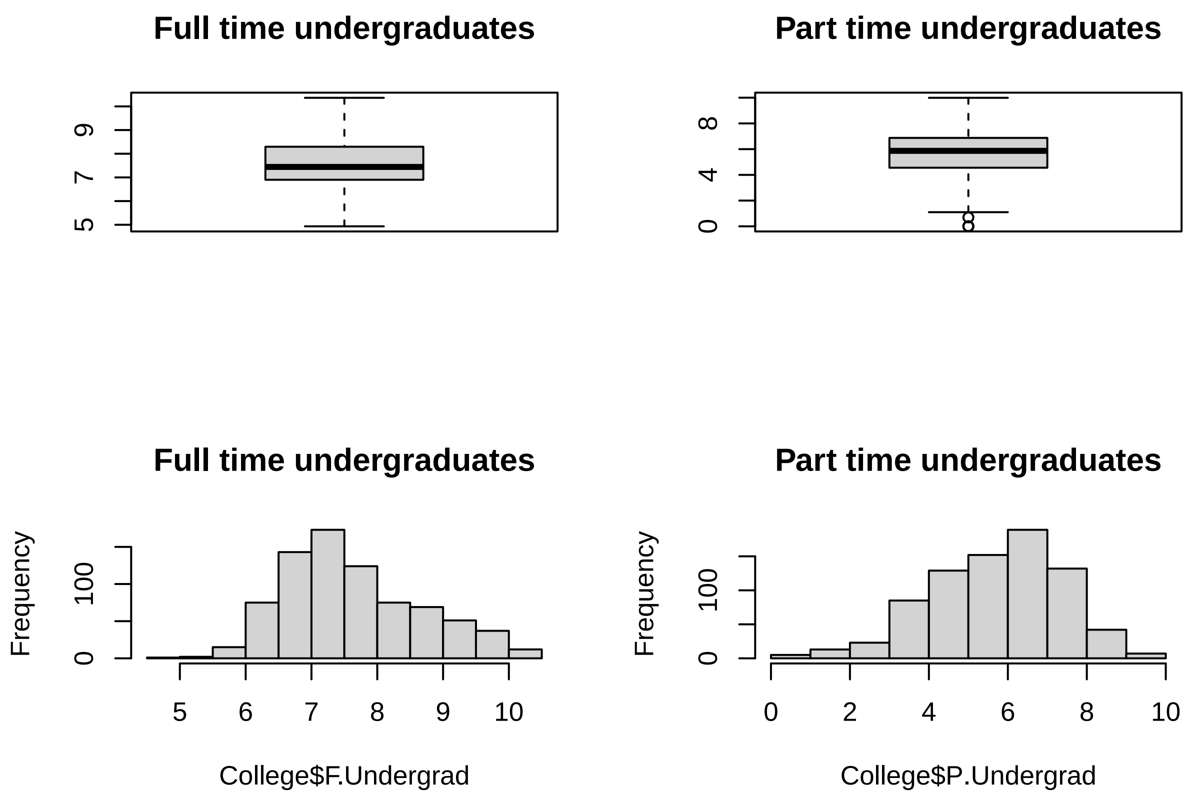

Room.Board
Frequency
0
50
100
150
2000
4000
6000
8000
College[, i]
Books
Frequency
0
100
200
300
400
500
0
500
1500
College[, i]
Personal
Frequency
0
50
100
150
200
250
0
2000
4000
6000
College[, i]


PhD
Terminal
S.F.Ratio
Frequency
College[, i]



We can see that F.Undergrad and P.Undergrad are really skewed so we can take the logarithm.

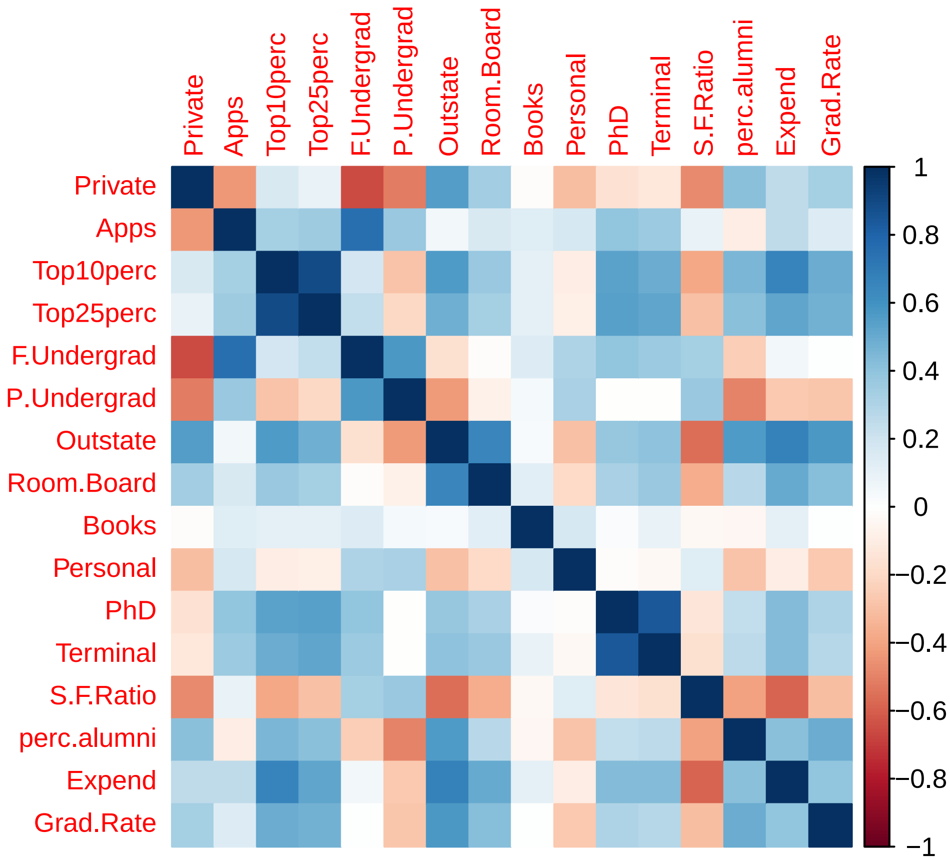
```
College$F.Undergrad <- log(College$F.Undergrad)
College$P.Undergrad <- log(College$P.Undergrad)

par(mfrow = c(2, 2))

boxplot(College$F.Undergrad, main="Full time undergraduates")
boxplot(College$P.Undergrad, main= "Part time undergraduates")

hist(College$F.Undergrad, main="Full time undergraduates")
hist(College$P.Undergrad, main= "Part time undergraduates")
```



Correlation matrix

We plot the correlation matrix to examine the relationships between the predictors and `Apps`

```
correlation_matrix <- cor(College)

corrplot(correlation_matrix, method="color", tl.cex = 0.8)
```

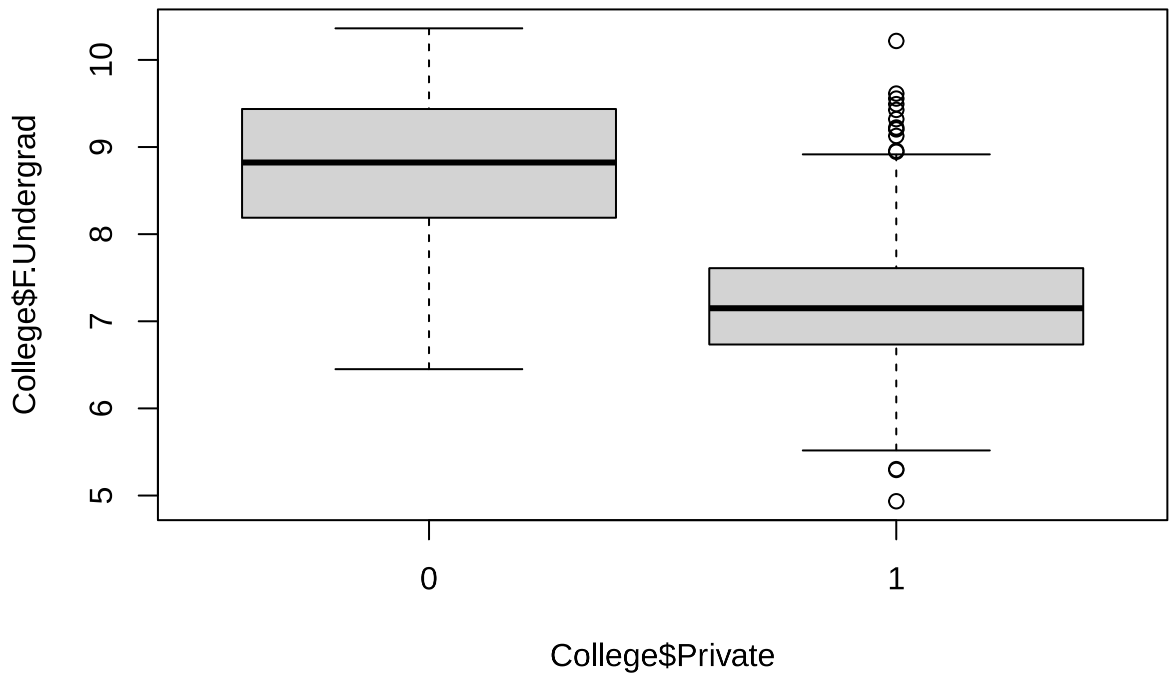


As you can see from the corralation matrix, there are some variables that are correlated with each other. We can see that the variables `Terminal` and `PhD` are really directly heavily correlated. This fenomena is caused by the fact that `PhD` containes a sub group of `Terminal`. So we can exclude `PhD`.

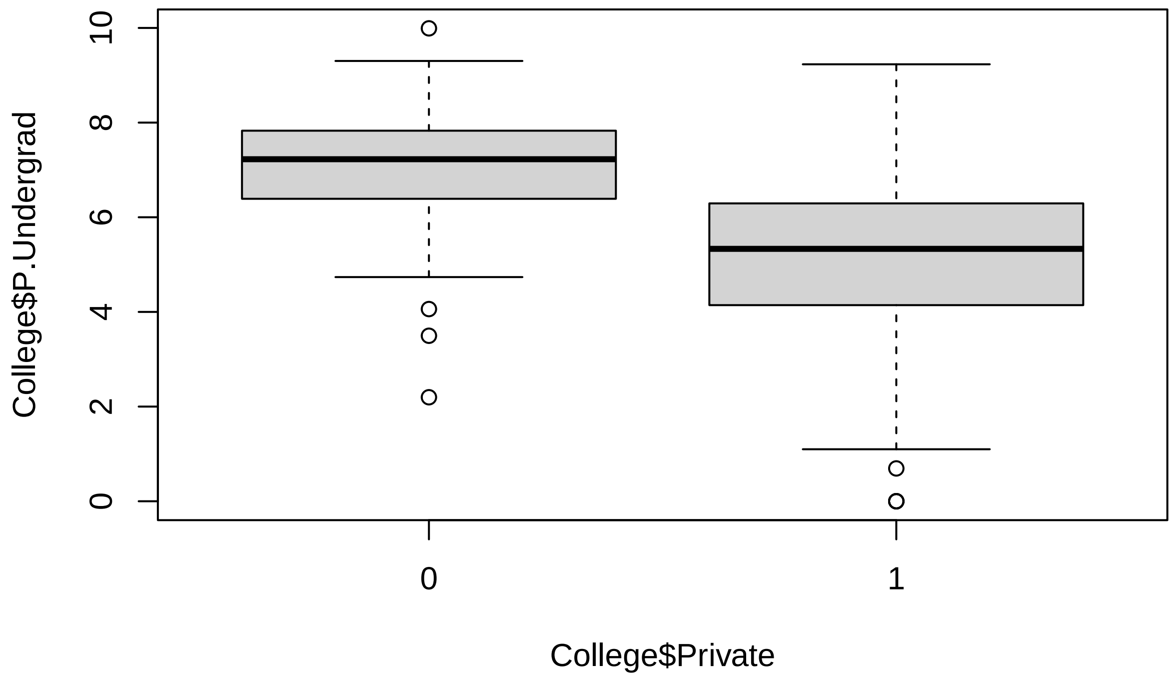
`Top10perc` and `Top25perc` are the same so for the same reason we can exclude `Top10perc`.

Now we can visualize some of the correlated variables

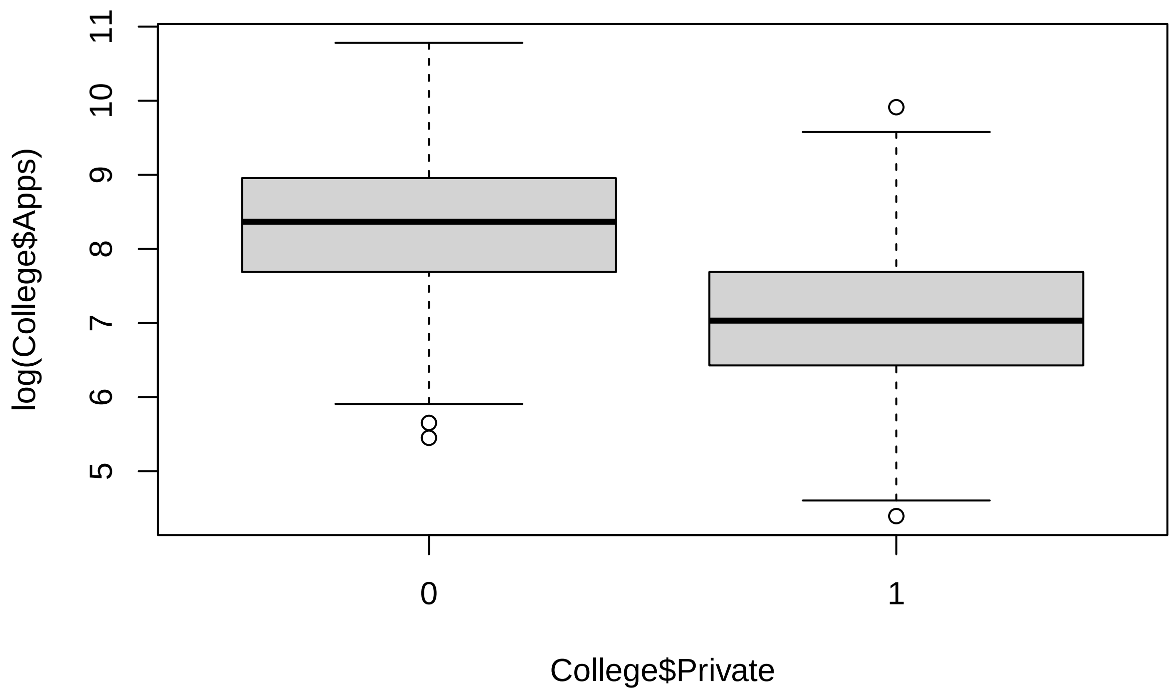
```
boxplot(College$F.Undergrad ~ College$Private)
```




```r
boxplot(College$P.Undergrad ~ College$Private)
```




```r
boxplot(log(College$Apps) ~ College$Private)
```



Data Normalization

Now let's normalize the data so that we have all the data with comparable scales.

```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

college_norm <- College
college_norm[ , 2:ncol(College)] <- lapply(College[ , 2:ncol(College)], normalize)

summary(college_norm)
```

```
##     Private            Apps            Top10perc        Top25perc
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.01448   1st Qu.:0.1474   1st Qu.:0.3516
##  Median :1.0000   Median :0.03076   Median :0.2316   Median :0.4945
##  Mean   :0.7272   Mean   :0.06083   Mean   :0.2796   Mean   :0.5142
##  3rd Qu.:1.0000   3rd Qu.:0.07379   3rd Qu.:0.3579   3rd Qu.:0.6593
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
##   F.Undergrad      P.Undergrad       Outstate        Room.Board
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.3621   1st Qu.:0.4558   1st Qu.:0.2572   1st Qu.:0.2864
##  Median :0.4621   Median :0.5872   Median :0.3951   Median :0.3815
```

```
##  Mean   :0.4976   Mean   :0.5696   Mean   :0.4184   Mean   :0.4063
##  3rd Qu.:0.6192   3rd Qu.:0.6880   3rd Qu.:0.5467   3rd Qu.:0.5154
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      Books           Personal            PhD             Terminal
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1667   1st Qu.:0.0916   1st Qu.:0.5684   1st Qu.:0.6184
##  Median :0.1800   Median :0.1450   Median :0.7053   Median :0.7632
##  Mean   :0.2020   Mean   :0.1665   Mean   :0.6806   Mean   :0.7329
##  3rd Qu.:0.2246   3rd Qu.:0.2214   3rd Qu.:0.8105   3rd Qu.:0.8947
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##    S.F.Ratio        perc.alumni         Expend           Grad.Rate
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.2413   1st Qu.:0.2031   1st Qu.:0.06720   1st Qu.:0.3981
##  Median :0.2976   Median :0.3281   Median :0.09786   Median :0.5093
##  Mean   :0.3107   Mean   :0.3554   Mean   :0.12205   Mean   :0.5135
##  3rd Qu.:0.3753   3rd Qu.:0.4844   3rd Qu.:0.14410   3rd Qu.:0.6296
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
```

GLM

Dataset preparation

Now we start to clean the dataset as told before:

```r
#Copy the dataset
college_cleared <- college_norm

#Remove the target variable and the variables that are not significant
college_cleared$Accept <- NULL
college_cleared$Enroll <- NULL
college_cleared$X <- NULL
college_cleared$Apps <- NULL
college_cleared$Top10perc <- NULL
college_cleared$PhD <- NULL
```

Dataset split

Now we must split the dataset in train set and test set. We will use 80% of the dataset for the training and 20% for the test.

```r
# Set a seed to make it deterministic
set.seed(42)

# Calculate the number of training lines (80% of the dataset)
n_training_lines <- floor(0.8 * nrow(college_cleared))

# Ensure the number of training lines is even
if (n_training_lines %% 2 != 0) {
  n_training_lines <- n_training_lines - 1
}
```

```r
# save index of lines with Private 1 and lines with Private 0
index_dataset_public <- which(college_cleared$Private == 0)

index_dataset_private <- which(college_cleared$Private == 1)

# Calculate the exact percentage of private 1 lines in the dataset
percentage_private <- length(index_dataset_private) / nrow(college_cleared)

# take 80% of lines from index_dataset_public
train_index_dataset_public <- sample(index_dataset_public, floor(n_training_lines * (1 - percentage_pri
train_index_dataset_private <- sample(index_dataset_private, floor(n_training_lines * (percentage_priva

# Cobine the two train_index_dataset_* into train_index
train_index <- c(train_index_dataset_public, train_index_dataset_private)

train_data <- college_cleared[train_index,]
test_data <- college_cleared[-train_index,]
```

Model fitting

Now we can fit the glm on the training set:

```r
glm <- glm(College$Apps[train_index] ~ .,data=train_data, family=poisson)

summary(glm)
```

```
##
## Call:
## glm(formula = College$Apps[train_index] ~ ., family = poisson,
##     data = train_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.092305   0.008438 485.009  < 2e-16 ***
## Private     -0.184864   0.002964 -62.360  < 2e-16 ***
## Top25perc    0.177462   0.004820  36.818  < 2e-16 ***
## F.Undergrad  4.729292   0.007555 625.964  < 2e-16 ***
## P.Undergrad -0.146204   0.007927 -18.443  < 2e-16 ***
## Outstate     0.492628   0.007140  68.997  < 2e-16 ***
## Room.Board   0.748332   0.006285 119.061  < 2e-16 ***
## Books        0.282714   0.010330  27.368  < 2e-16 ***
## Personal    -0.350038   0.008107 -43.179  < 2e-16 ***
## Terminal     0.019375   0.006914   2.802  0.00508 **
## S.F.Ratio    0.351358   0.009843  35.696  < 2e-16 ***
## perc.alumni -0.049954   0.005894  -8.475  < 2e-16 ***
## Expend       0.570555   0.010381  54.959  < 2e-16 ***
## Grad.Rate    0.955605   0.006981 136.879  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
```

```
##     Null deviance: 2120079  on 618  degrees of freedom
## Residual deviance:  221113  on 605  degrees of freedom
## AIC: 226872
##
## Number of Fisher Scoring iterations: 4
```

Model evaluation

Now we can calculate the RMSE for the glm:

```r
RMSE <- function(predicted, actual) {
  sqrt(mean((predicted - actual)^2))
}


rmse_train_set <- RMSE(predict.glm(glm), College$Apps[train_index])
rmse_test_set <- RMSE(predict.glm(glm, newdata = test_data), College$Apps[-train_index])
```

The rmse for the train_set is inside `rmse_train_set`:

```r
rmse_train_set
```

```
## [1] 5009.428
```

The rmse for the test_set is inside `rmse_test_set`:

```r
rmse_test_set
```

```
## [1] 4394.249
```

The RMSE for the test set is lower than the RMSE for the train set, which is a good sign that the model is not overfitting and it is generalizing well.

Random Forest

One hundred trees model fitting

Now, we will try to fit a random forest model on the training set and see if it performs better than the glm:

```r
library(randomForest)
```

```
## randomForest 4.7-1.2
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(randomForestExplainer)
```

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

```r
random_forest <- randomForest(x=train_data, formula=Apps ~ ., y=College$Apps[train_index], data = train_

random_forest
```

```
##
## Call:
##  randomForest(x = train_data, y = College$Apps[train_index], xtest = test_data,      ytest = College$
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 3851069
##                     % Var explained: 76.05
##                        Test set MSE: 1992994
##                     % Var explained: 81.22
```

One hundred trees model evaluation

Now we can calculate the RMSE for the random forest model:

```r
#RMSE train_set
rf_rmse_train_set <- sqrt(random_forest$mse[length(random_forest$mse)])

#RMSE test_set
rf_rmse_test_set <- sqrt(random_forest$test$mse[length(random_forest$test$mse)])
```

The rmse for the train_set is inside `rf_rmse_train_set`:

```r
rf_rmse_train_set
```

```
## [1] 1962.414
```

The rmse for the test_set is inside `rf_rmse_test_set`:

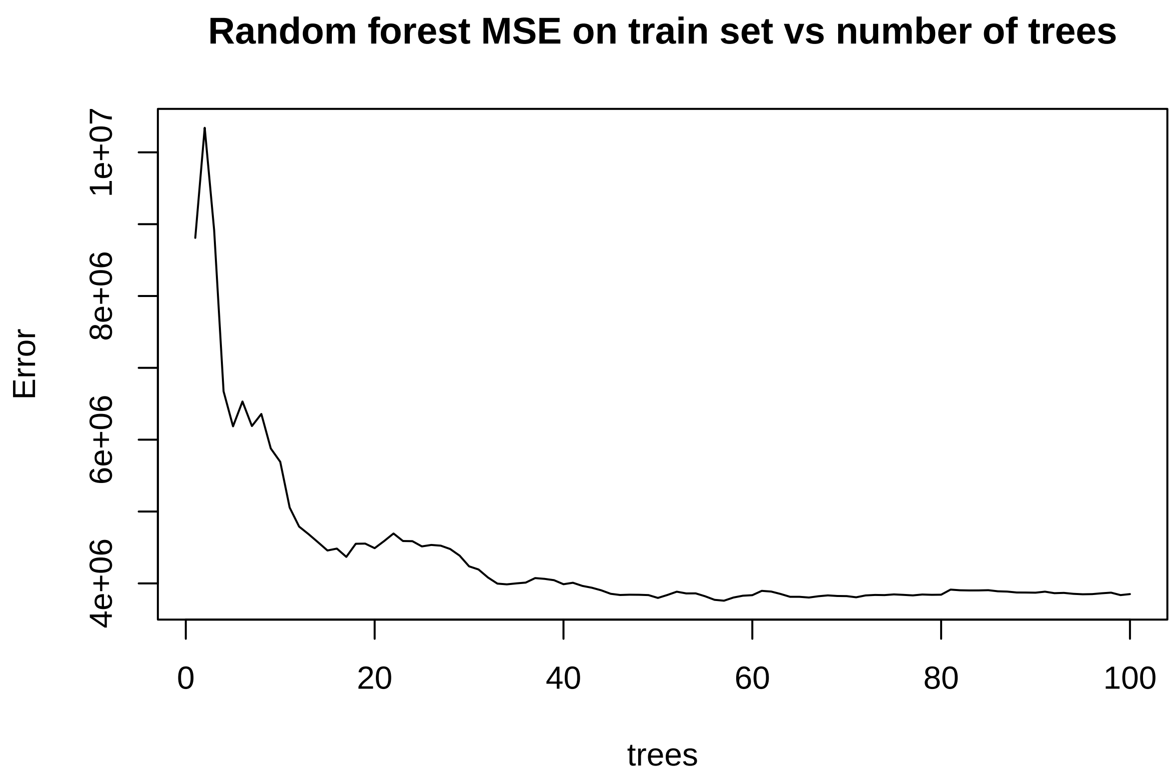
```
rf_rmse_test_set
```

```
## [1] 1411.734
```

The RMSE of the random forest model is lower than the RMSE of the glm model, which means that the random forest model is performing better than the glm model.

Now we can plot how the MSE of the model changes with the number of trees:

```
plot(random_forest, main = "Random forest MSE on train set vs number of trees")
```



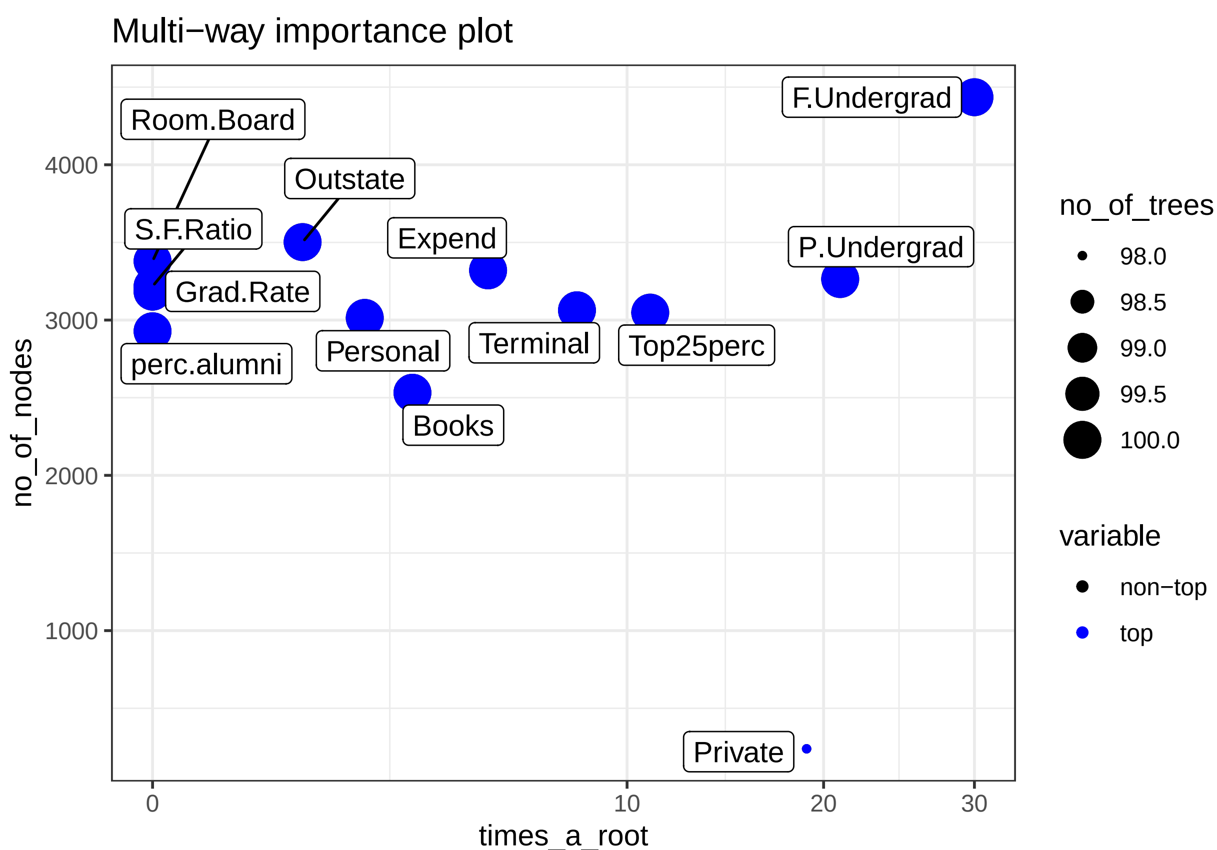
Now explore how the variables are used in the random forest:

```
importance_random_forest <- measure_importance(random_forest)

importance_random_forest
```

```
##       variable mean_min_depth no_of_nodes mse_increase node_purity_increase
## 1        Books         3.7400        2532    -270617.3            234728137
## 2       Expend         2.3900        3320     968186.6            595865266
## 3  F.Undergrad         1.1600        4435   13947069.3           3853122354
## 4    Grad.Rate         2.7700        3183    2870676.7            620419170
## 5     Outstate         2.6800        3502    1364694.6            494687655
## 6  P.Undergrad         2.1800        3264    2763740.6           1111092839
## 7  perc.alumni         3.9300        2928     147399.1            189797185
```

```
## 8      Personal          3.8400          3014     186349.0              202430936
## 9       Private          4.2166           239    3283327.3              562522412
## 10   Room.Board          3.2100          3379     460229.4              283365283
## 11    S.F.Ratio          3.2400          3217     151082.2              332505229
## 12     Terminal          2.6700          3063     416483.8              498339608
## 13    Top25perc          2.1400          3048     960212.7              944531722
##     no_of_trees times_a_root       p_value
## 1           100            3  1.000000e+00
## 2           100            5  3.420597e-09
## 3           100           30 1.329068e-142
## 4           100            0  5.648074e-04
## 5           100            1  3.814932e-20
## 6           100           21  9.631230e-07
## 7           100            0  9.406634e-01
## 8           100            2  4.689851e-01
## 9            98           19  1.000000e+00
## 10          100            0  3.031443e-12
## 11          100            0  5.028945e-05
## 12          100            8  1.574935e-01
## 13          100           11  2.352934e-01
```

```r
plot_multi_way_importance(importance_random_forest, y_measure = "no_of_nodes", x_measure = "times_a_roo
```



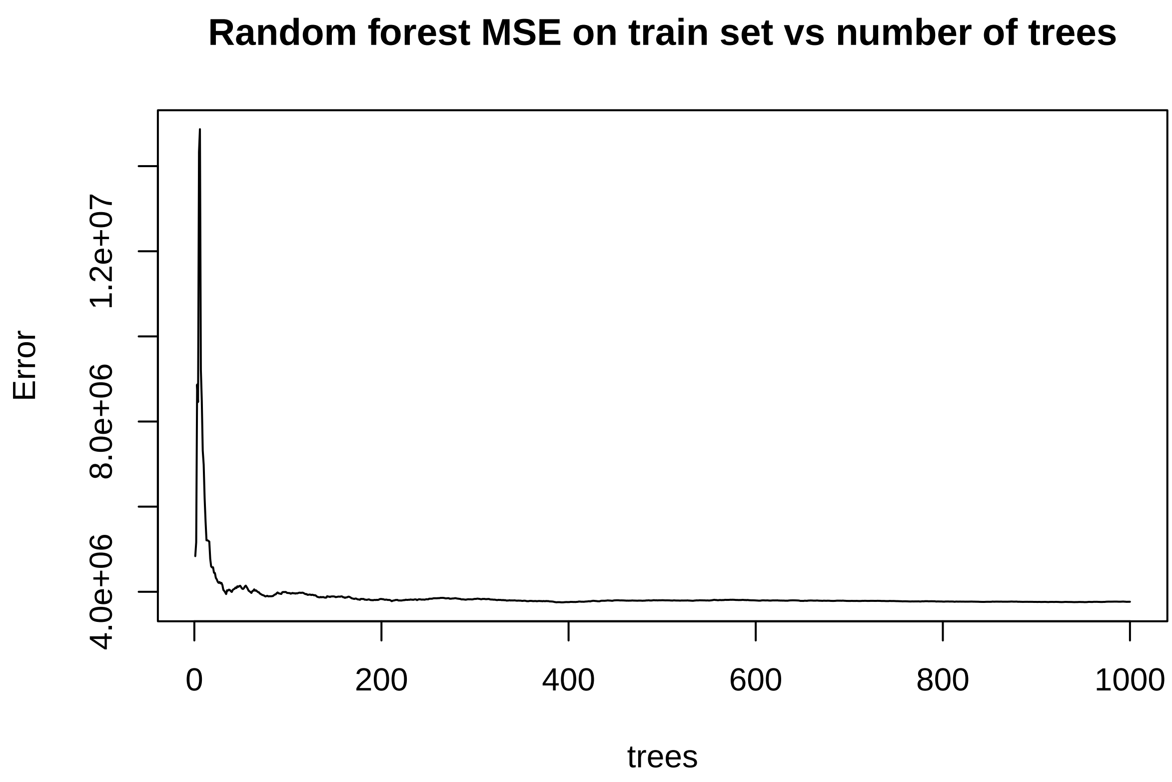
One thousand trees model fitting

And if we increse the number of trees from 100 to 1000:

```
random_forest_1000 <- randomForest(x=train_data, formula=Apps ~ ., y=College$Apps[train_index], data = t

random_forest_1000
```

```
##
## Call:
##  randomForest(x = train_data, y = College$Apps[train_index], xtest = test_data,      ytest = College$
##                Type of random forest: regression
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 3766066
##                     % Var explained: 76.58
##                        Test set MSE: 1739743
##                     % Var explained: 83.61
```

```
plot(random_forest_1000, main = "Random forest MSE on train set vs number of trees")
```




```
#RMSE train_set
rf_rmse_train_set_1000 <- sqrt(random_forest_1000$mse[length(random_forest_1000$mse)])
```

```
#RMSE test_set
rf_rmse_test_set_1000 <- sqrt(random_forest_1000$test$mse[length(random_forest_1000$test$mse)])
```

One thousand trees model evaluation

Analyze Root Mean Square Error for the random forest model with 1000 trees:

```
rf_rmse_train_set_1000
```

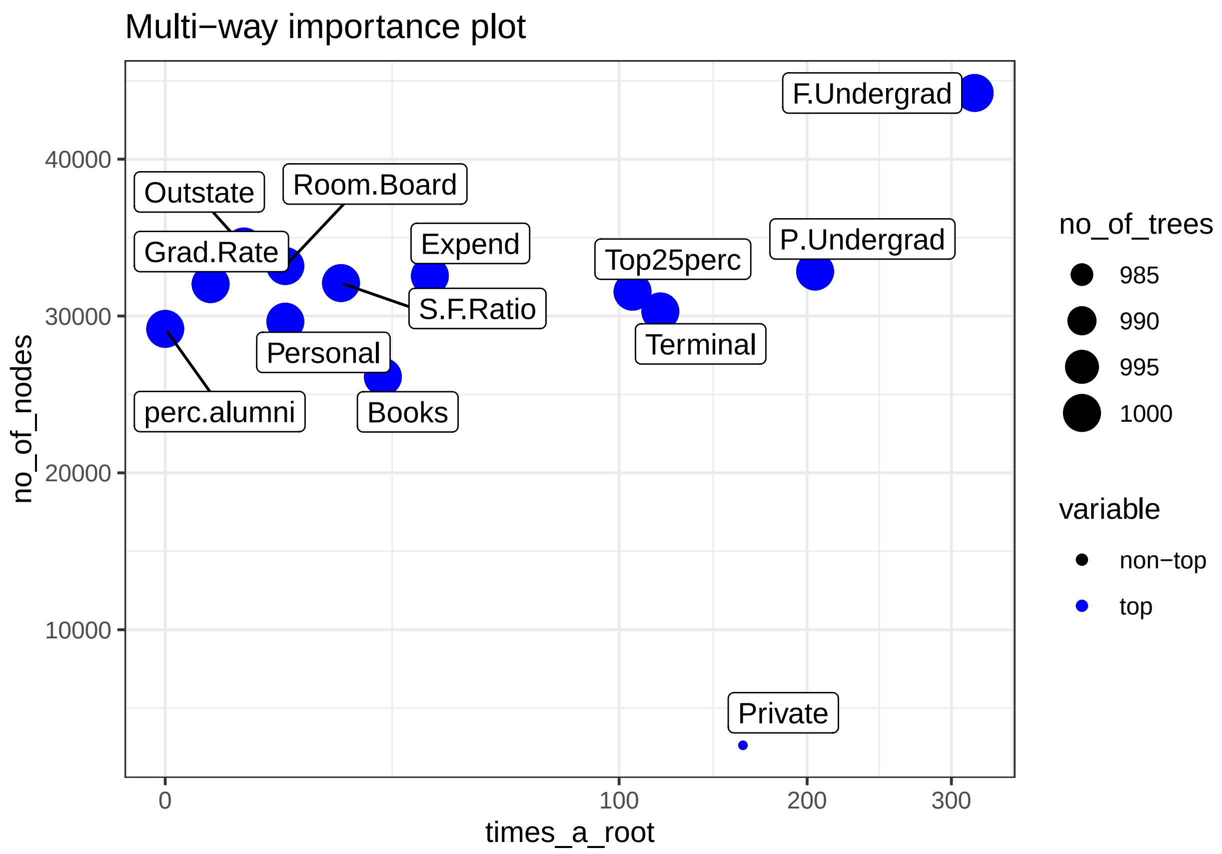
```
## [1] 1940.636
```

```
rf_rmse_test_set_1000
```

```
## [1] 1318.993
```

And now how the variables are used:

```
importance_random_forest_1000 <- measure_importance(random_forest_1000)
plot_multi_way_importance(importance_random_forest_1000, y_measure = "no_of_nodes", x_measure = "times_a
```



Ten thousand trees model fitting

How said in the original paper, the random forest never overfits, so we can try to increase the number of trees to 10000:

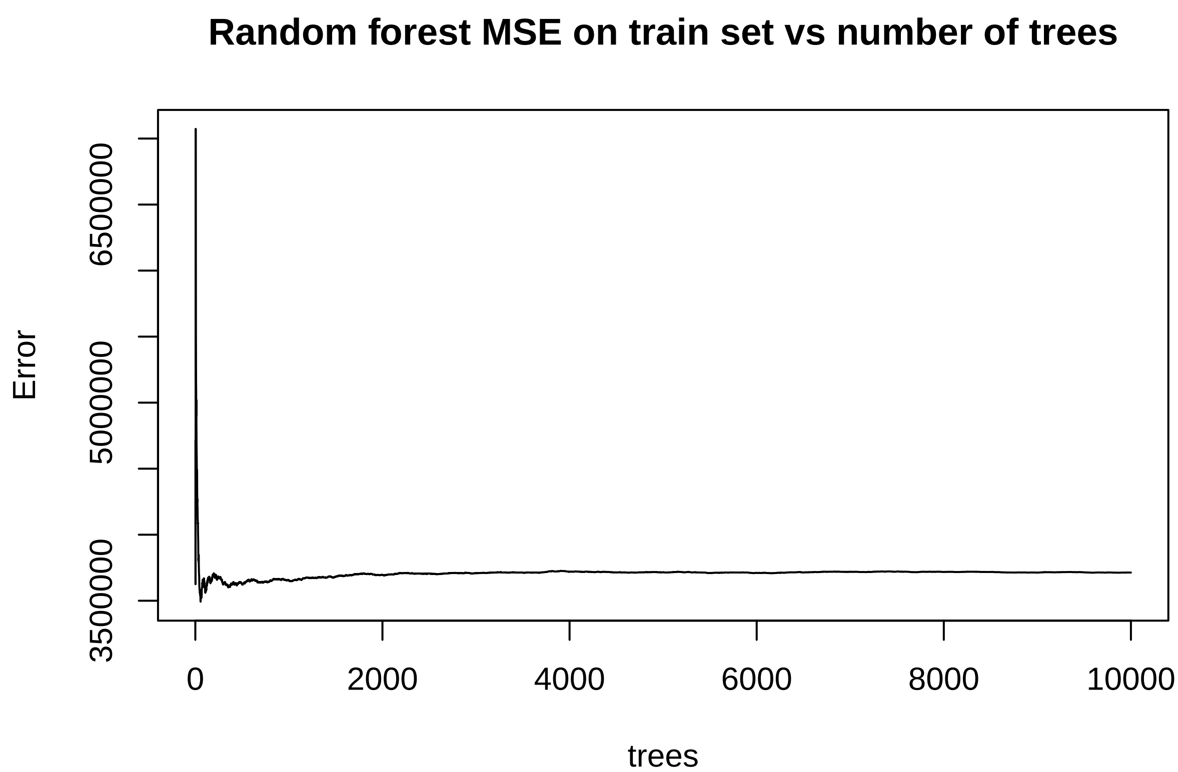
```
random_forest_10000 <- randomForest(x=train_data, formula=Apps ~ ., y=College$Apps[train_index], data =

random_forest_10000
```

```
##
## Call:
##  randomForest(x = train_data, y = College$Apps[train_index], xtest = test_data,      ytest = College$
##                Type of random forest: regression
##                      Number of trees: 10000
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 3713422
##                     % Var explained: 76.91
##                        Test set MSE: 1782928
##                     % Var explained: 83.2
```

Ten thousand trees model evaluation

Now see how it performs:

```
plot(random_forest_10000, main = "Random forest MSE on train set vs number of trees")
```




```
#RMSE train_set
rf_rmse_train_set_10000 <- sqrt(random_forest_10000$mse[length(random_forest_10000$mse)])

#RMSE test_set
rf_rmse_test_set_10000 <- sqrt(random_forest_10000$test$mse[length(random_forest_10000$test$mse)])

rf_rmse_train_set_10000
```

```
## [1] 1927.024
```

```
rf_rmse_test_set_10000
```

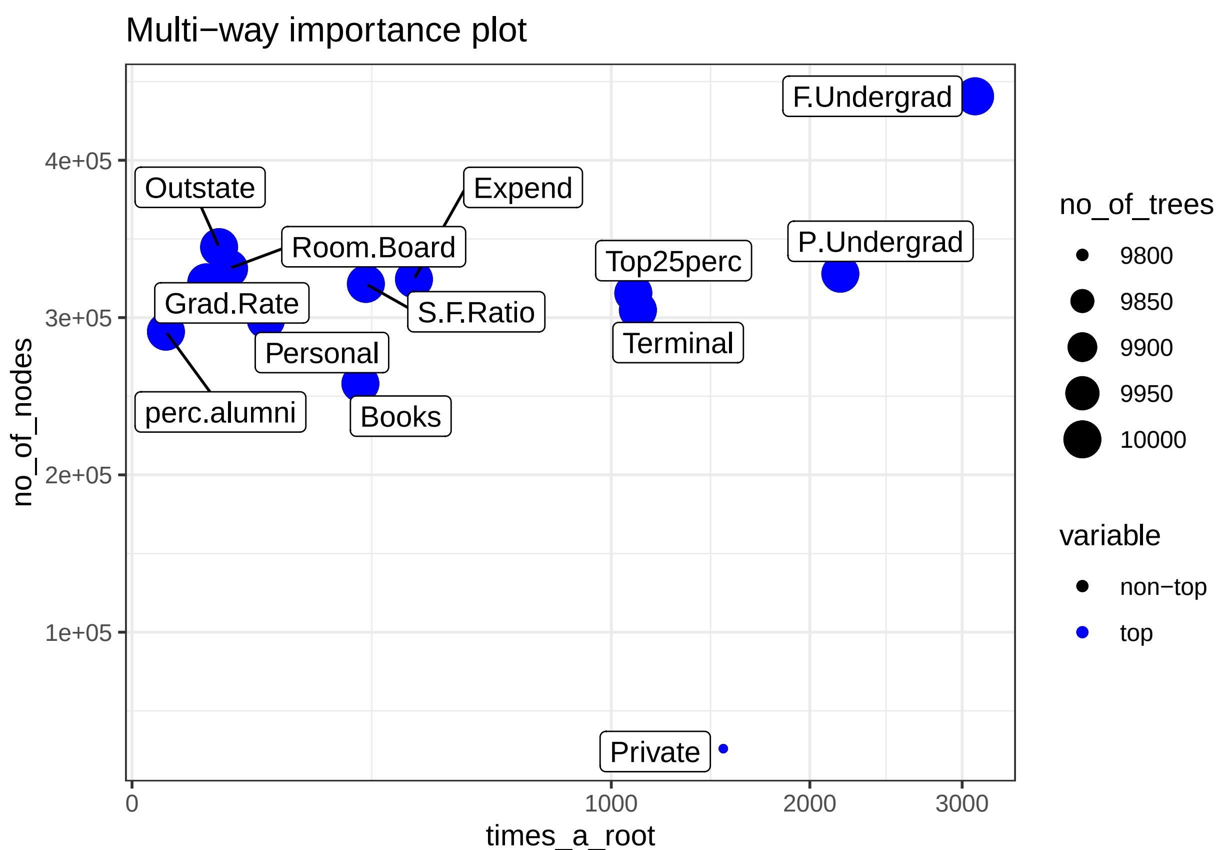
```
## [1] 1335.263
```

```
importance_random_forest_10000 <- measure_importance(random_forest_10000)

importance_random_forest_10000
```

```
##       variable mean_min_depth no_of_nodes mse_increase node_purity_increase
## 1        Books       3.774000      258049    -257220.2            214542716
## 2       Expend       2.611100      324372    1342183.3            606995063
## 3  F.Undergrad       1.126900      440661   14269744.0           3982724134
## 4    Grad.Rate       2.726600      322383    2969593.7            690952767
## 5     Outstate       2.657500      344772    1095401.2            501927836
```

```
## 6  P.Undergrad       2.208300      327868    3184373.1            1083674887
## 7  perc.alumni       3.880100      291053     407994.9             183954868
## 8     Personal       3.716300      298400      48999.6             221915378
## 9      Private       4.411572       25966    2326405.7             464815615
## 10  Room.Board       3.187900      331161     496765.5             327263613
## 11   S.F.Ratio       3.222300      321410     522618.1             340410084
## 12    Terminal       2.706400      304717     478133.4             498010888
## 13   Top25perc       2.269200      315546     994797.3             839793111
##    no_of_trees times_a_root       p_value
## 1        10000          227  1.000000e+00
## 2        10000          346  0.000000e+00
## 3        10000         3094  0.000000e+00
## 4        10000           24  0.000000e+00
## 5        10000           33  0.000000e+00
## 6        10000         2184  0.000000e+00
## 7        10000            5  1.000000e+00
## 8        10000           78  9.999643e-01
## 9         9798         1522  1.000000e+00
## 10       10000           41  0.000000e+00
## 11       10000          238  0.000000e+00
## 12       10000         1114  5.739285e-16
## 13       10000         1094 2.099544e-177
```

```
plot_multi_way_importance(importance_random_forest_10000, y_measure = "no_of_nodes", x_measure = "times_
```



We can see that the random forest with 10000 trees perform sligthly better the the previous one.

```r
summary(glm)
```

Lets go back to the results for the glm model

```
##
## Call:
## glm(formula = College$Apps[train_index] ~ ., family = poisson,
##     data = train_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.092305   0.008438 485.009  < 2e-16 ***
## Private     -0.184864   0.002964 -62.360  < 2e-16 ***
## Top25perc    0.177462   0.004820  36.818  < 2e-16 ***
## F.Undergrad  4.729292   0.007555 625.964  < 2e-16 ***
## P.Undergrad -0.146204   0.007927 -18.443  < 2e-16 ***
## Outstate     0.492628   0.007140  68.997  < 2e-16 ***
## Room.Board   0.748332   0.006285 119.061  < 2e-16 ***
## Books        0.282714   0.010330  27.368  < 2e-16 ***
## Personal    -0.350038   0.008107 -43.179  < 2e-16 ***
## Terminal     0.019375   0.006914   2.802  0.00508 **
## S.F.Ratio    0.351358   0.009843  35.696  < 2e-16 ***
## perc.alumni -0.049954   0.005894  -8.475  < 2e-16 ***
## Expend       0.570555   0.010381  54.959  < 2e-16 ***
## Grad.Rate    0.955605   0.006981 136.879  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2120079  on 618  degrees of freedom
## Residual deviance:  221113  on 605  degrees of freedom
## AIC: 226872
##
## Number of Fisher Scoring iterations: 4
```

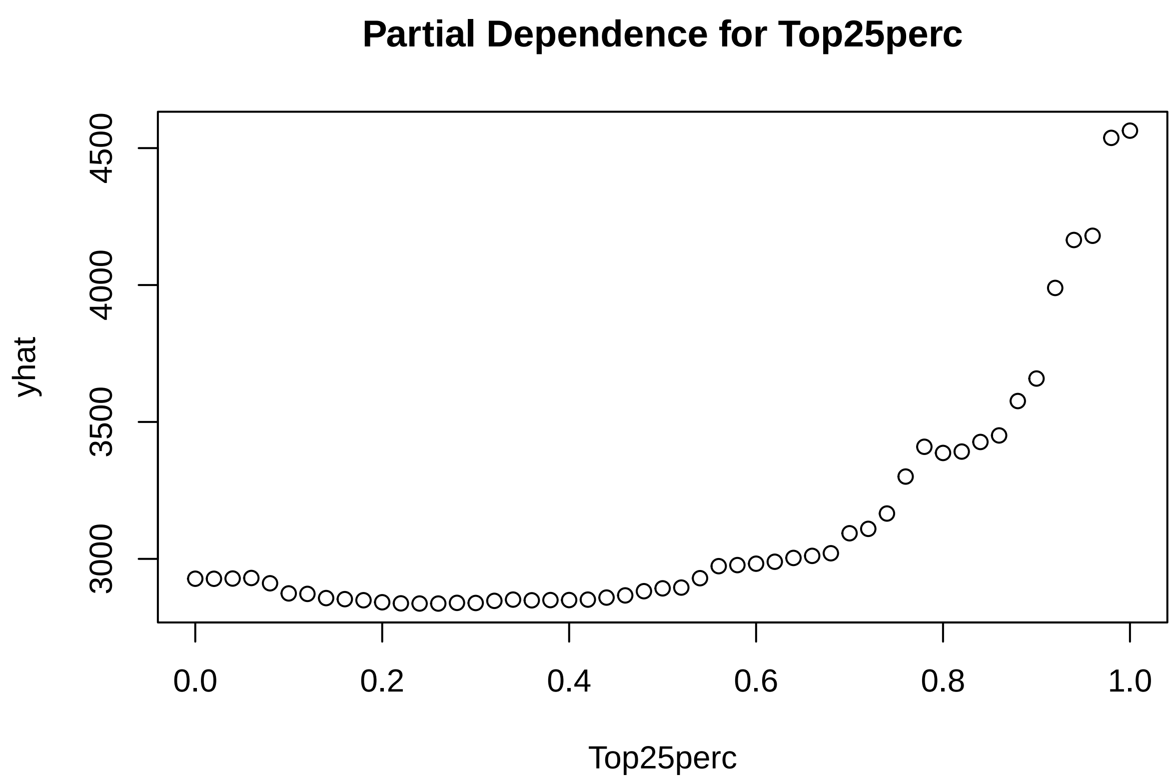
There are a couple of takeaways here. First of all the the residual deviance is much lower than the null deviance, so the glm model explains a lot of variance. However the residual deviance od 221,113 is still quite large. Compared to the degrees of freedom this could be due to potential model misspecification, which could be due to non-linearity.

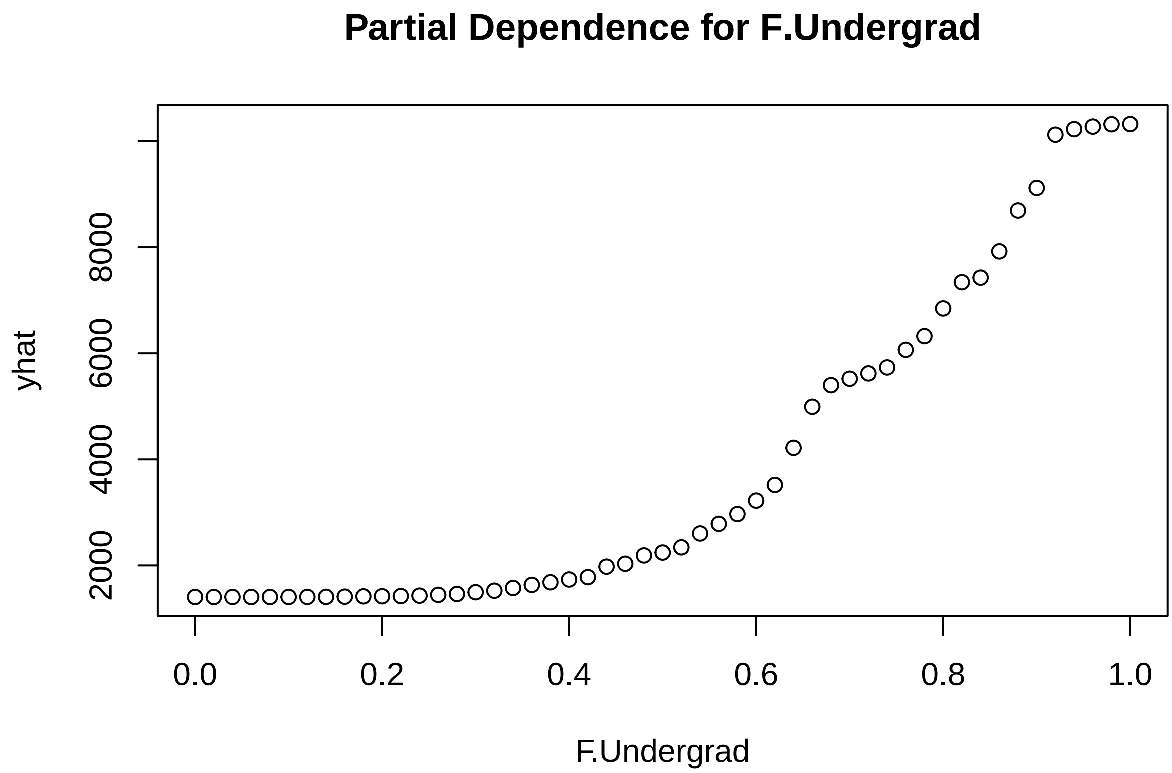
```r
library(pdp)

names <- colnames(train_data)

names <- names[-1]

names
```

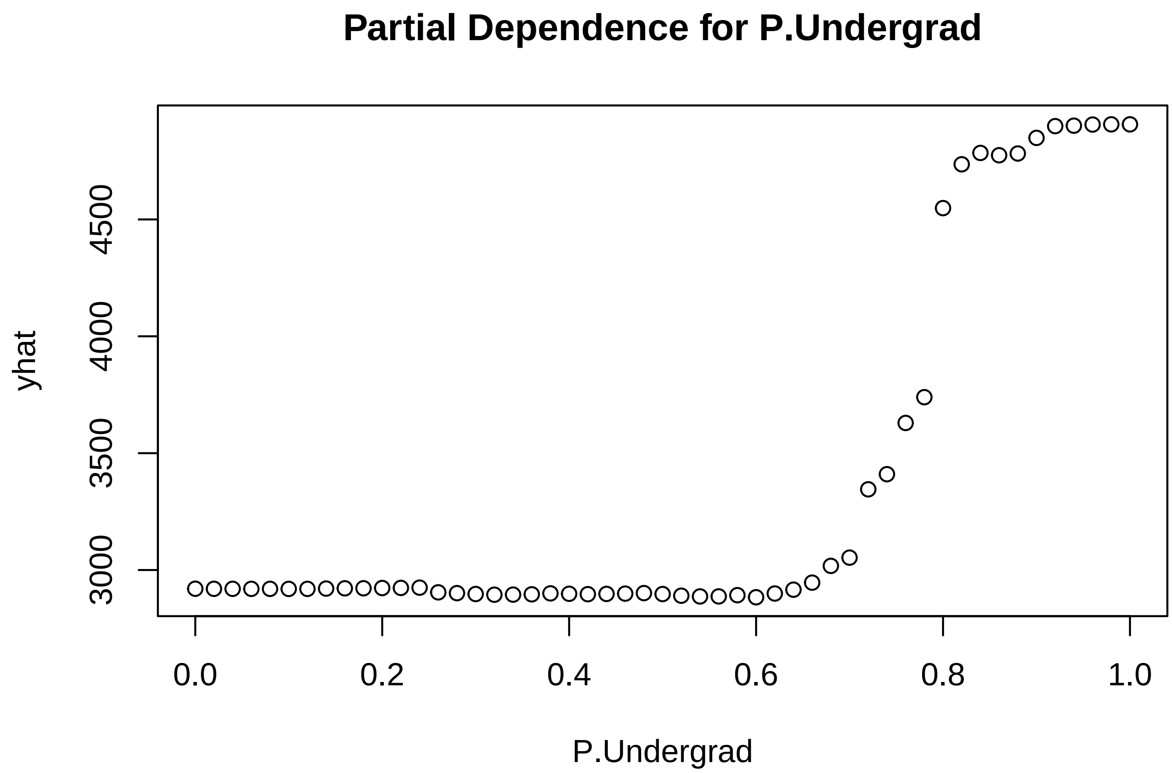
Now when we look at back at the multi way importance plot we see that the F.Undergrad and P.Undergrad are the most important features. Lets see the partial dependence plot for these two features.

```
##  [1] "Top25perc"   "F.Undergrad" "P.Undergrad" "Outstate"    "Room.Board"
##  [6] "Books"       "Personal"    "Terminal"    "S.F.Ratio"   "perc.alumni"
## [11] "Expend"      "Grad.Rate"
```

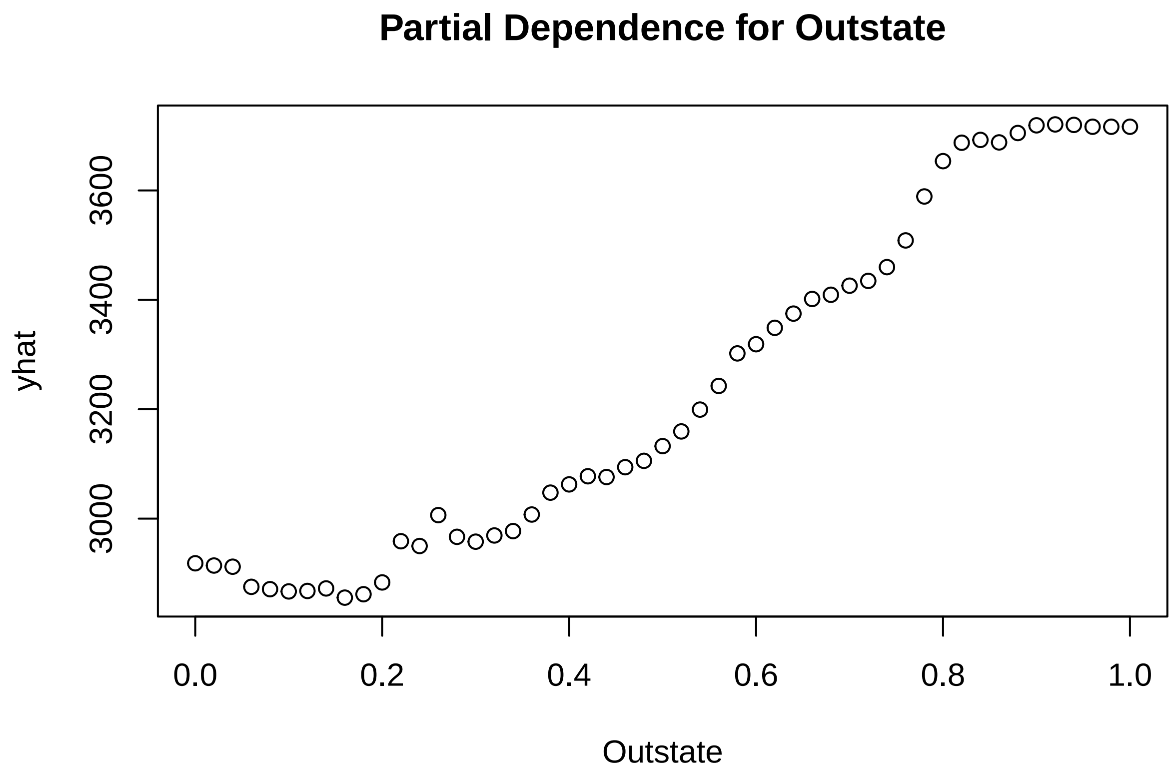
```r
# Partial dependence plot for F.Undergrad
for (pred in names) {
  pdp_obj <- partial(random_forest_1000, pred.var = pred, train = train_data)
  plot(pdp_obj, main = paste("Partial Dependence for", pred))
}
```

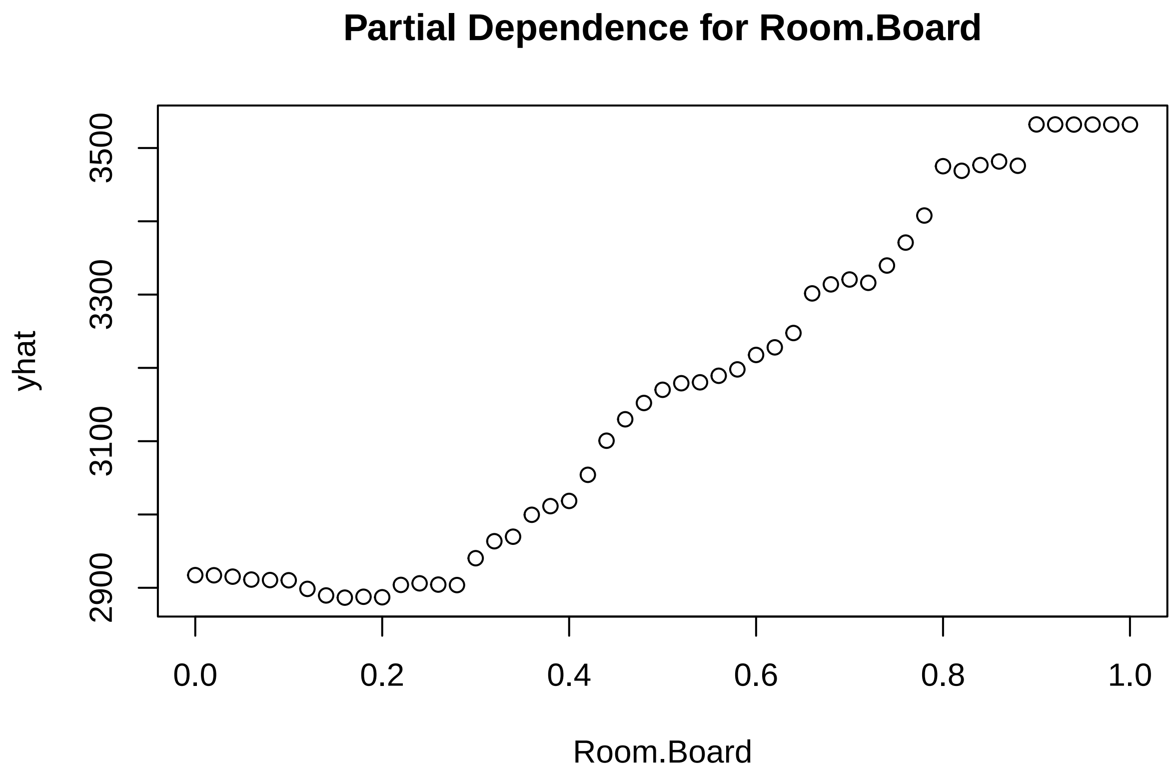


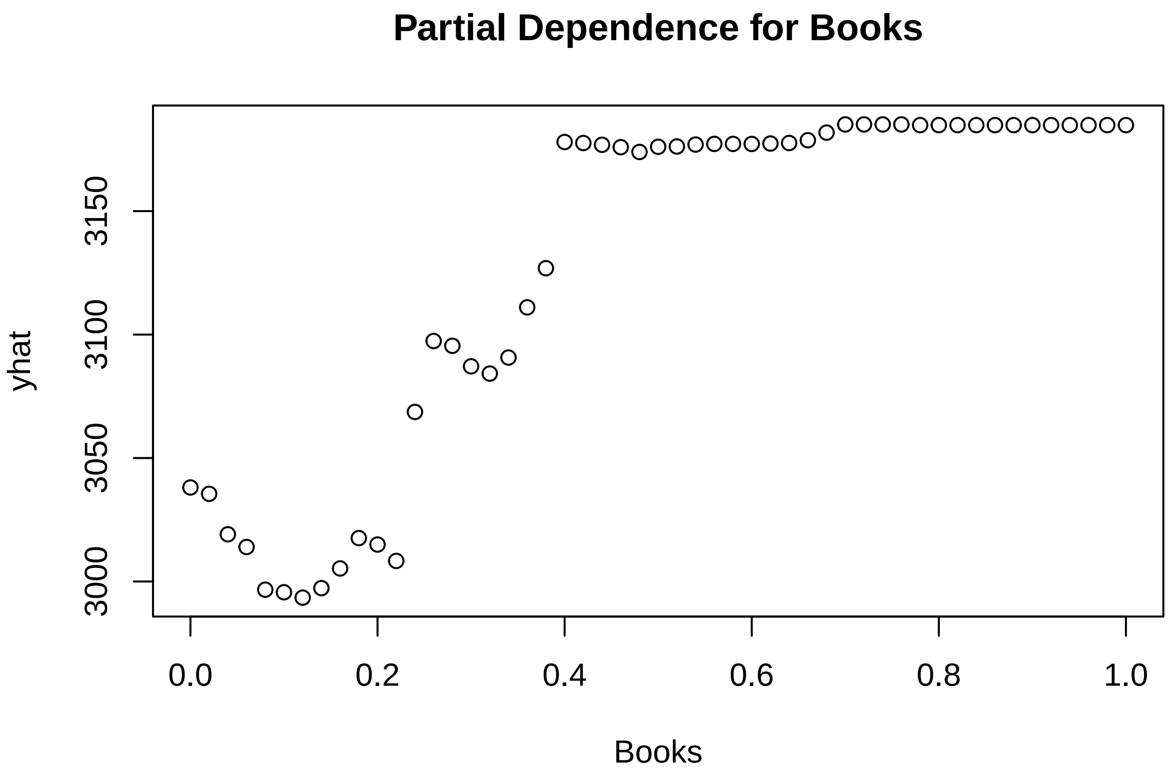

Partial Dependence for F.Undergrad
yhat
2000
4000
6000
8000
0.0
0.2
0.4
0.6
0.8
1.0
F.Undergrad



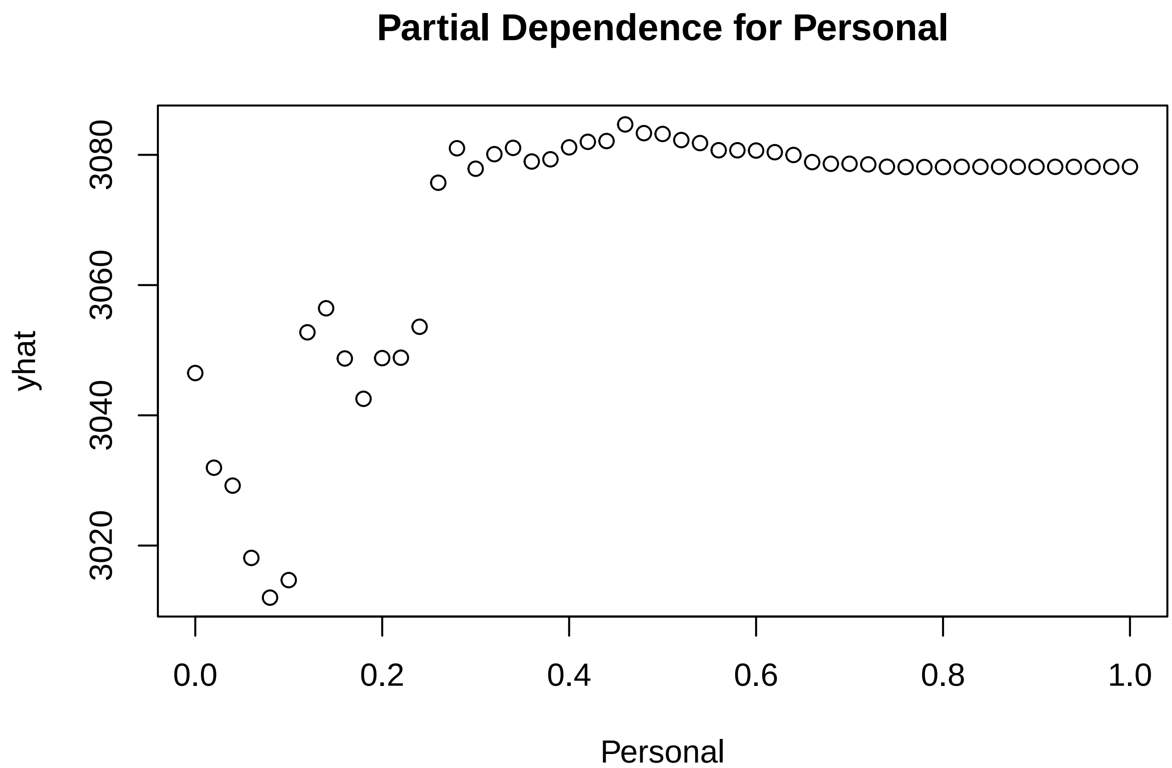
Partial Dependence for P.Undergrad
yhat
3000
3500
4000
4500
0.0
0.2
0.4
0.6
0.8
1.0
P.Undergrad

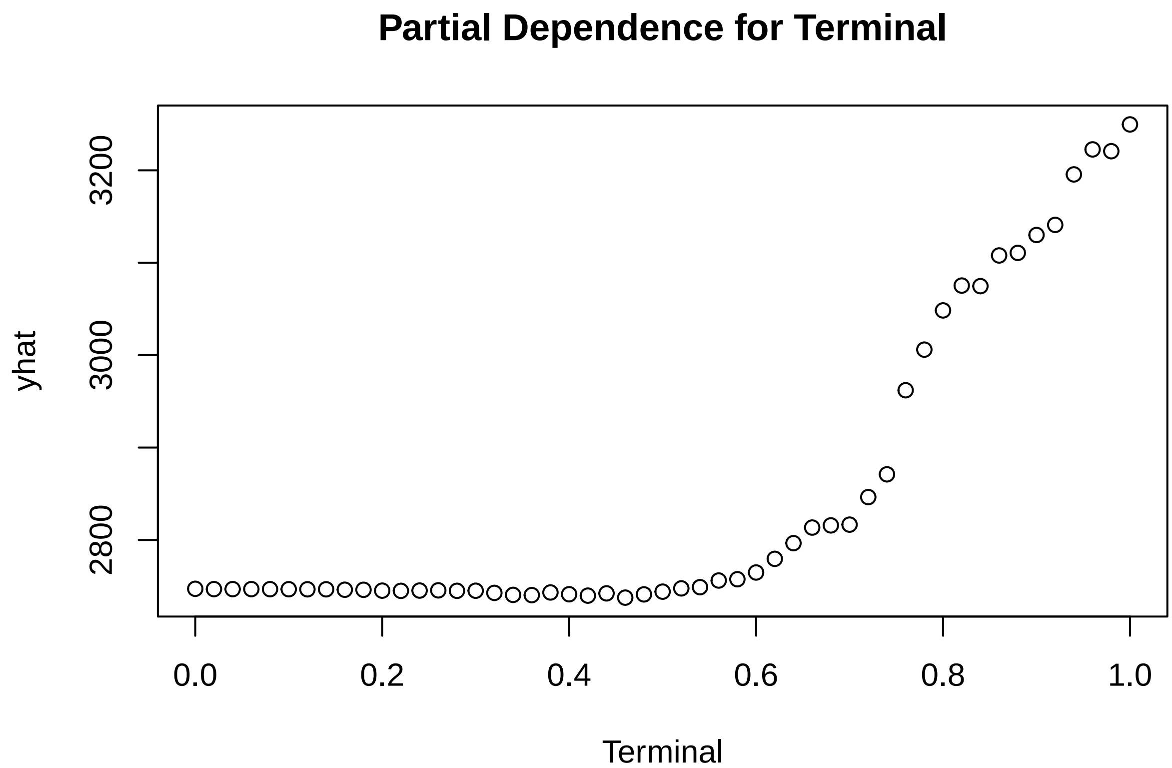

Partial Dependence for Outstate
yhat
3000
3200
3400
3600
0.0
0.2
0.4
0.6
0.8
1.0
Outstate

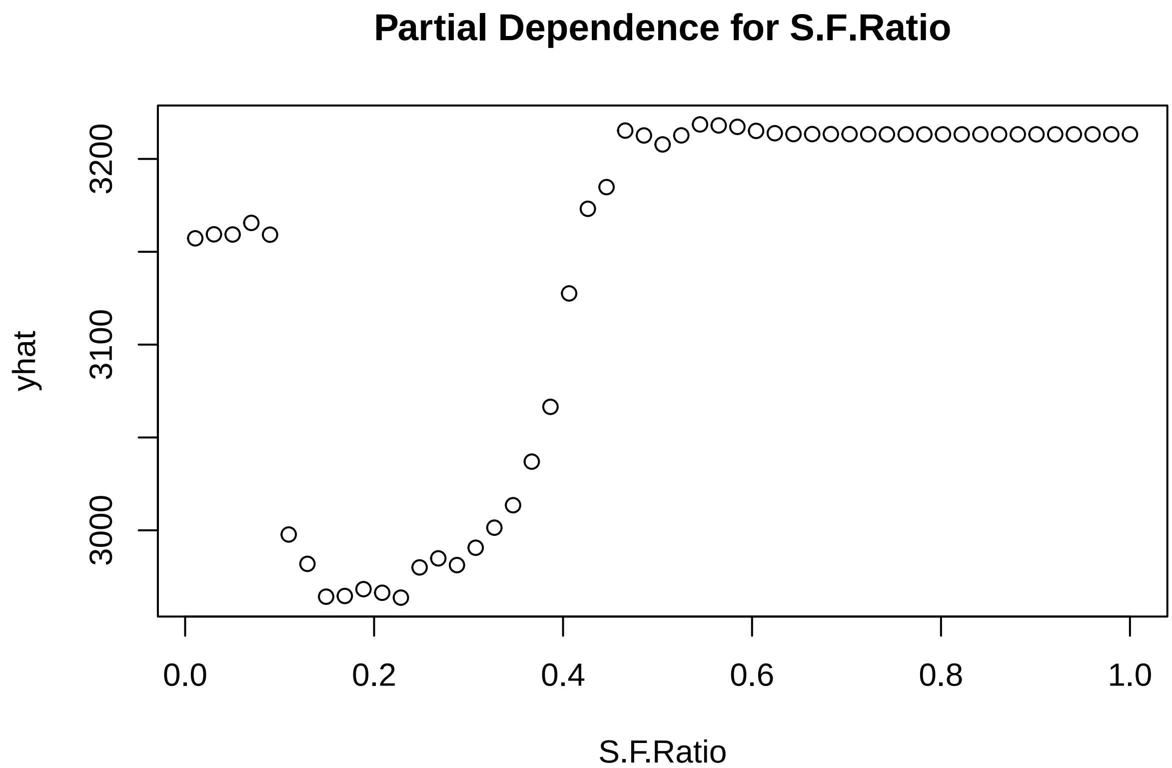

Partial Dependence for Room.Board
yhat
2900
3100
3300
3500
0.0
0.2
0.4
0.6
0.8
1.0
Room.Board

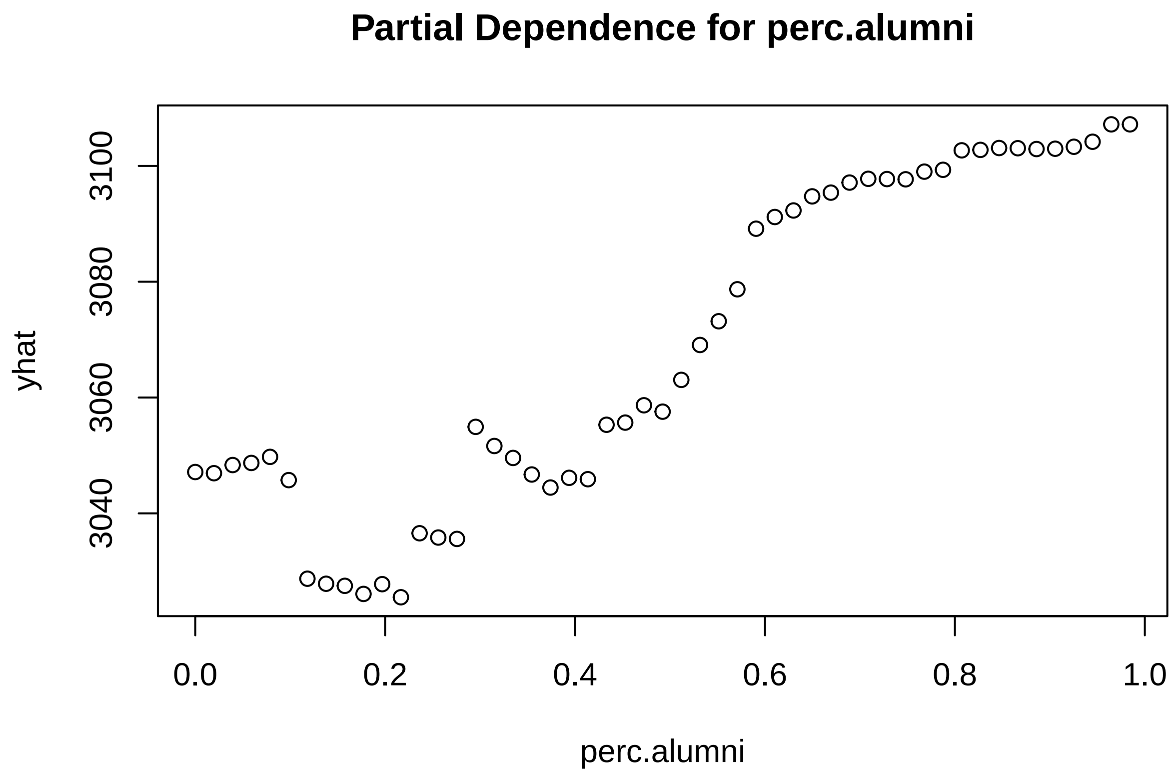


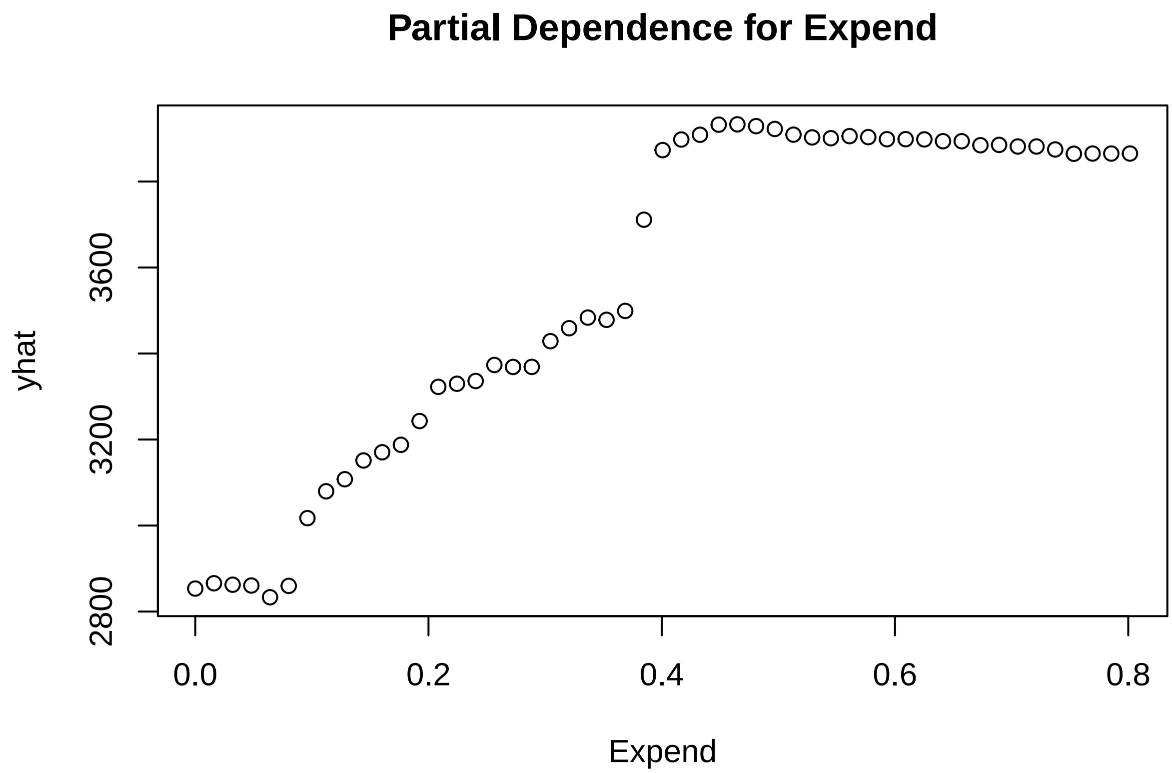
Partial Dependence for Books
yhat
3000
3050
3100
3150
0.0
0.2
0.4
0.6
0.8
1.0
Books

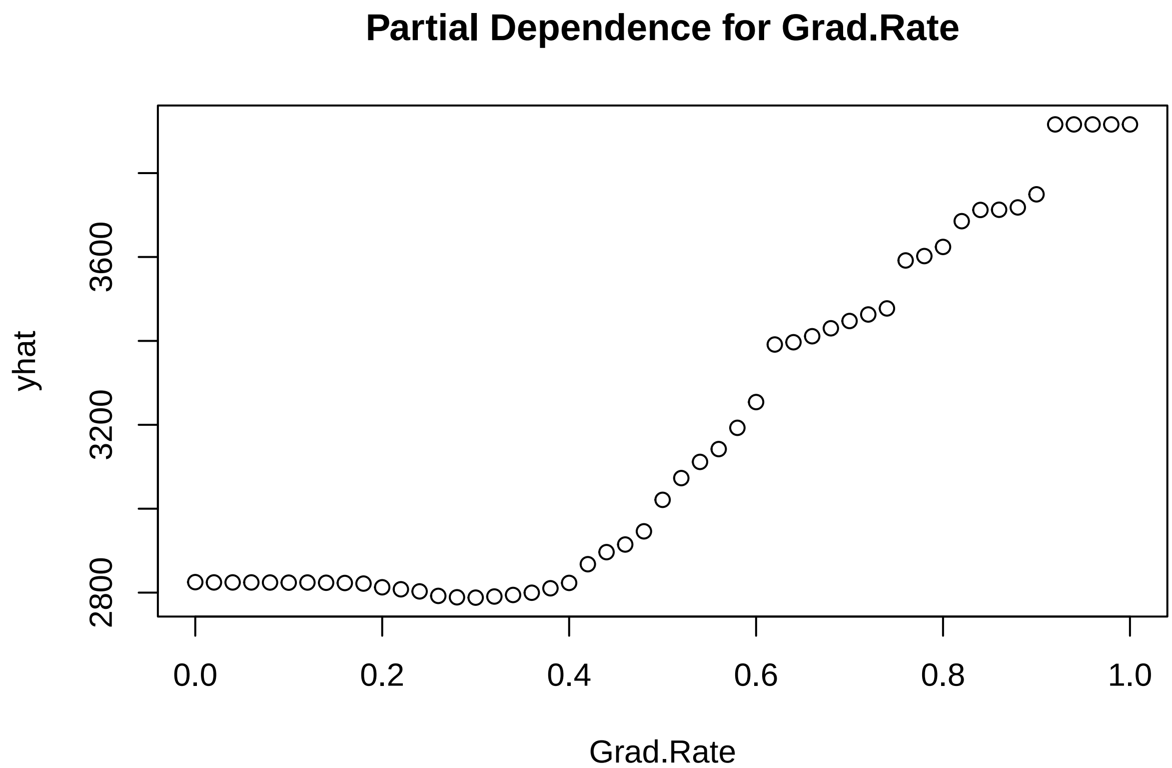

Partial Dependence for Personal
yhat
3020
3040
3060
3080
0.0
0.2
0.4
0.6
0.8
1.0
Personal


Partial Dependence for Terminal
yhat
3200
3000
2800
0.0
0.2
0.4
0.6
0.8
1.0
Terminal


Partial Dependence for S.F.Ratio
yhat
3000
3100
3200
0.0
0.2
0.4
0.6
0.8
1.0
S.F.Ratio


Partial Dependence for perc.alumni
yhat
3040
3060
3080
3100
0.0
0.2
0.4
0.6
0.8
1.0
perc.alumni


Partial Dependence for Expend
yhat
2800
3200
3600
0.0
0.2
0.4
0.6
0.8
Expend



Most of the features show us that they have a non-linear relatiohship with the response variable.

These two points regarding the GLM and the Random forest, suggest that a GAM could be a very strong alternative due to non-linearity between the featuers and the response variable.

```
library(mgcv)
```

Now lets fit a GAM on a training set

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```r
# Fit GAM model
gam_model <- gam(College$Apps[train_index]  ~ F.Undergrad + P.Undergrad +
                                Grad.Rate + Outstate +
                                Room.Board + Expend +
                                Books + Personal +
                                S.F.Ratio + perc.alumni +
                                Top25perc + Terminal,
                 family = poisson, data = train_data)

summary(gam_model)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## College$Apps[train_index] ~ F.Undergrad + P.Undergrad + Grad.Rate +
##     Outstate + Room.Board + Expend + Books + Personal + S.F.Ratio +
##     perc.alumni + Top25perc + Terminal
##
## Parametric coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.895692   0.007828  497.66   <2e-16 ***
## F.Undergrad  4.927114   0.006799  724.68   <2e-16 ***
## P.Undergrad -0.156095   0.007896  -19.77   <2e-16 ***
## Grad.Rate    0.957661   0.006934  138.10   <2e-16 ***
## Outstate     0.362536   0.006824   53.12   <2e-16 ***
## Room.Board   0.650742   0.006084  106.97   <2e-16 ***
## Expend       0.567961   0.010371   54.76   <2e-16 ***
## Books        0.219771   0.010278   21.38   <2e-16 ***
## Personal    -0.334710   0.008041  -41.62   <2e-16 ***
## S.F.Ratio    0.398119   0.009808   40.59   <2e-16 ***
## perc.alumni -0.124358   0.005744  -21.65   <2e-16 ***
## Top25perc    0.174991   0.004836   36.18   <2e-16 ***
## Terminal     0.140012   0.006645   21.07   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.821   Deviance explained = 89.4%
## UBRE =  362.6  Scale est. = 1         n = 619
```

```r
gam_preds_test <- predict(gam_model, test_data, type = "response")

rmse_gam_test <- sqrt(mean((College$Apps[-train_index] - gam_preds_test)^2))

rmse_gam_test
```

And compare it to the glm

```
## [1] 1280.016
```

```r
gam_preds_train <- predict(gam_model, train_data, type = "response")

rmse_gam_train <- sqrt(mean((College$Apps[train_index] - gam_preds_train)^2))

rmse_gam_train
```

```
## [1] 1678.38
```

We can also see a slight improvement in RMSE compared to the random forest models.

```r
library(mgcv)
# Fit GAM model
gam_model_2 <- gam(College$Apps[train_index]  ~ s(F.Undergrad, k=3) + s(P.Undergrad, k=3) +
                               Grad.Rate + Outstate +
                               Room.Board + s(Expend, k=3) +
                               s(Books, k=3) + s(Personal, k=3) +
                               s(S.F.Ratio, k=3) + s(perc.alumni, k=3) +
                               s(Top25perc, k=3) + s(Terminal, k=3),
                 family = poisson, data = train_data, select= TRUE)

summary(gam_model_2)
```

Now lets fine-tune the model a bit to see if we can get even better results

```
##
## Family: poisson
## Link function: log
##
## Formula:
## College$Apps[train_index] ~ s(F.Undergrad, k = 3) + s(P.Undergrad,
##     k = 3) + Grad.Rate + Outstate + Room.Board + s(Expend, k = 3) +
##     s(Books, k = 3) + s(Personal, k = 3) + s(S.F.Ratio, k = 3) +
##     s(perc.alumni, k = 3) + s(Top25perc, k = 3) + s(Terminal,
##     k = 3)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 6.569381   0.004297 1528.76   <2e-16 ***
## Grad.Rate   1.037421   0.007355  141.04   <2e-16 ***
## Outstate    0.383205   0.007291   52.56   <2e-16 ***
## Room.Board  0.537761   0.006299   85.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df   Chi.sq p-value
## s(F.Undergrad) 1.9965      2 441509.2  <2e-16 ***
## s(P.Undergrad) 1.9821      2    992.8  <2e-16 ***
## s(Expend)      1.9914      2   1853.7  <2e-16 ***
```

```
## s(Books)       1.9957      2   1358.7  <2e-16 ***
## s(Personal)    1.9966      2   1988.3  <2e-16 ***
## s(S.F.Ratio)   1.9878      2   3599.0  <2e-16 ***
## s(perc.alumni) 1.9833      2    474.9  <2e-16 ***
## s(Top25perc)   1.9970      2   7404.2  <2e-16 ***
## s(Terminal)    0.9967      2    274.6  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.825   Deviance explained =   90%
## UBRE = 342.49  Scale est. = 1         n = 619
```

```
gam_preds_test_2 <- predict(gam_model_2, test_data, type = "response")

rmse_gam_test_2 <- sqrt(mean((College$Apps[-train_index] - gam_preds_test_2)^2))

rmse_gam_test_2
```

And lets see the RMSE

```
## [1] 1308.551
```

```
gam_preds_train_2 <- predict(gam_model_2, train_data, type = "response")

rmse_gam_train_2 <- sqrt(mean((College$Apps[train_index] - gam_preds_train_2)^2))

rmse_gam_train_2
```

```
## [1] 1649.383
```

```
AIC(gam_model,gam_model_2)
```

Comparing the models

```
##                   df      AIC
## gam_model   13.00000 230796.0
## gam_model_2 20.92708 218353.6
```

Comparing predictions of the GAM vs Random forest

```
plt_num <- length(College$Apps[-train_index])-1

plot(0:plt_num, gam_preds_test, col = "blue", pch = 16,
     main = "GAM vs. Random Forest Predictions",
```

```r
     xlab = "School Index", ylab = "Applications")


points(0:plt_num, predict(random_forest_1000, test_data), col = "red", pch = 16)
points(0:plt_num, College$Apps[-train_index], col = "black", pch = 16)

for (i in 0:plt_num) {
  abline(v = i, col = "gray", lwd = 0.5, lty = 2)  # Dashed thin vertical lines
}


legend("topleft",inset=c(0,-0.1), legend = c("Actual Value", "Random Forest","GAM"),
       col = c("black", "red", "blue"), pch = c(16, 16, 16), bty = "n", horiz=TRUE,xpd=TRUE)
```