Project EDA

Anna Li, Jess Eng, Seth Billiau

Due: November 15, 2019

A Brief Description of the Dataset

Our dataset was exported from the Sports Reference College Basketball website. Sports Reference College Basketball states that their contemporary data is "is provided by data companies on an ongoing basis." Assuming that this data is reliable, we extracted basic and advanced team statistics for the 330 NCAA Division 1 teams in the 2007-2008, 2009-2010 and 2017-2018 college basketball seasons. We will team statistics from the 2007-2008 (and possibly the 2008-2009) basketball seasons as our training set with the 2008-2009 W-L% as our response variable. This training set will be split into training and validation datasets to be used for model selection. The 2017-2018 dataset (and possibly 2019-2020) will be used as our test set. The accuracy of our model on this dataset will be unknown until the completion of the 2019-2020 season.

Jess, Anna and Seth Project

11/6/19

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ---------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded glmnet 2.0-18
```

```r
library(rpart)
```

```r
rm(list = ls())

teams78 = read.csv('data/teams0708.csv')
teams89 = read.csv('data/teams0809.csv')

RMSE = function(y,yhat){
  SSE = sum((y-yhat)^2)
  return(sqrt(SSE/length(y)))
}
```

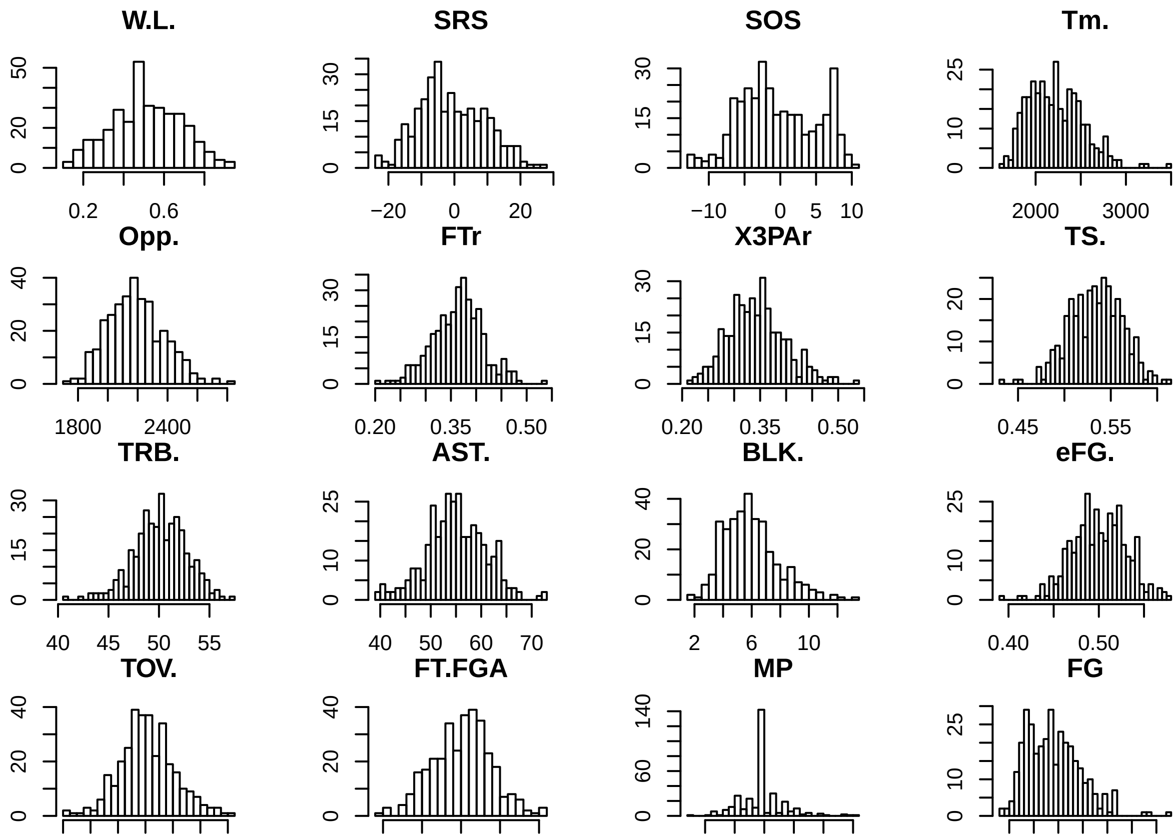
```r
names(teams78)[1] = "Rk"
names(teams89)[1] = "Rk"
```

```r
teams78_drop = subset(teams78, select = -c(Rk, TeamW, TeamL, ConfW, ConfL, HomeW, HomeL, AwayW, AwayL, S
teams89_drop = subset(teams89, select = -c(Rk, TeamW, TeamL, ConfW, ConfL, HomeW, HomeL, AwayW, AwayL, S
```

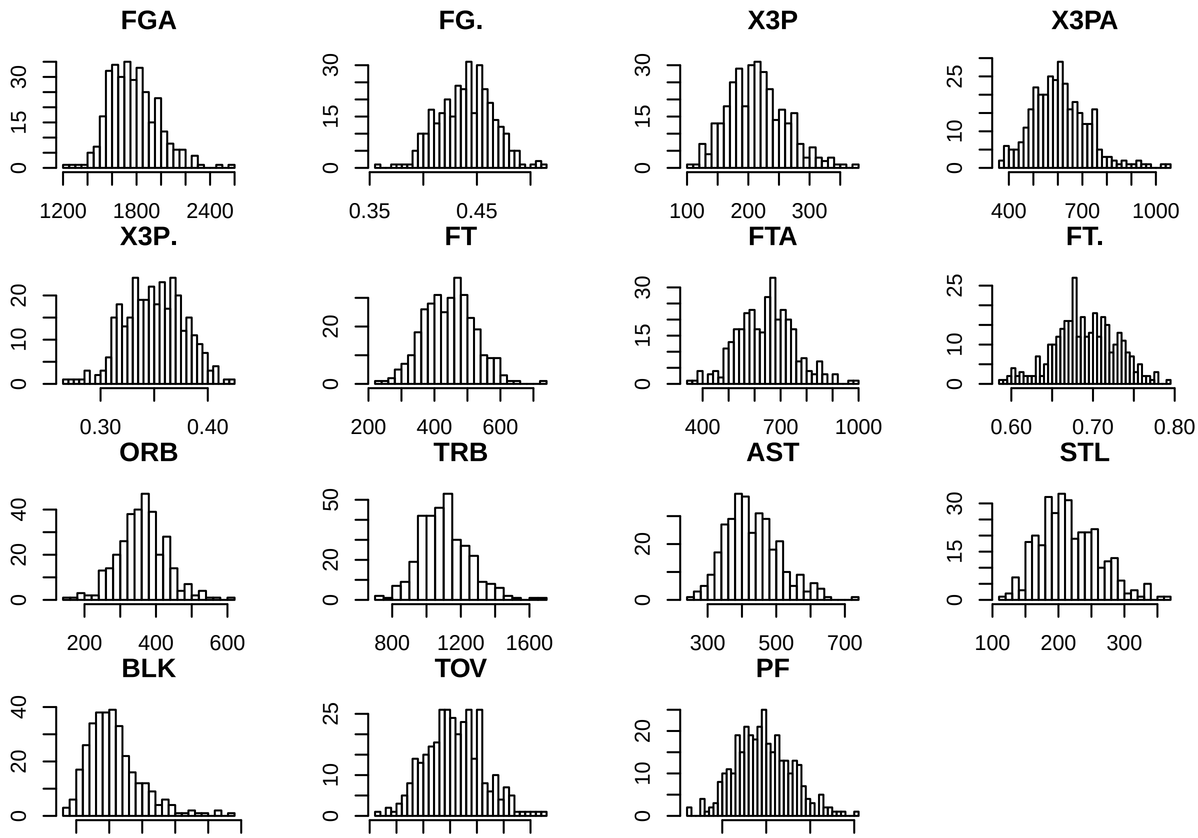
```r
# inpute the data with the column means
for(i in 1:ncol(teams78_drop)){
  teams78_drop[is.na(teams78_drop[,i]), i] <- mean(teams78_drop[,i], na.rm = TRUE)
}

for(i in 1:ncol(teams89_drop)){
  teams89_drop[is.na(teams89_drop[,i]), i] <- mean(teams89_drop[,i], na.rm = TRUE)
}
```

```r
#explore data
summary(teams78_drop)
summary(teams89_drop)
```

```r
par(mfrow =c(4, 4))
par(mar=c(1,2,3,3))
columns = colnames(teams78_drop)[c(1:16)]
for(column in columns){
  hist(teams78_drop[,column], main=column, xlab=column, breaks=30)
}
```




```
par(mfrow =c(4, 4))
columns = colnames(teams78_drop)[c(17:31)]
for(column in columns){
  hist(teams78_drop[,column], main=column, xlab=column, breaks=30)
}
```




```
#linear model with main effects of all included predictors
lm1 = lm(W.L. ~., data=teams78_drop)
lm1rmse_train = RMSE(teams78_drop$W.L, predict(lm1, teams78_drop))
```

```
## Warning in predict.lm(lm1, teams78_drop): prediction from a rank-deficient
## fit may be misleading
```

```
lm1rmse_test = RMSE(teams89_drop$W.L, predict(lm1, teams89_drop))
```

```
## Warning in predict.lm(lm1, teams89_drop): prediction from a rank-deficient
## fit may be misleading
```

```
cat('lm1 train RMSE: ', lm1rmse_train, '\n', 'lm1 test RMSE:  ', lm1rmse_test, sep='')
```

```
## lm1 train RMSE: 0.05182295
## lm1 test RMSE:  0.05492276
```

```
#backward sequential variable selection regression model
lm2 = step(lm1, direction = "backward", k=2, trace = 0)
lm2rmse_train = RMSE(teams78_drop$W.L, predict(lm2, teams78_drop))
lm2rmse_test = RMSE(teams89_drop$W.L, predict(lm2, teams89_drop))
cat('lm2 train RMSE: ', lm2rmse_train, '\n', 'lm2 test RMSE:  ', lm2rmse_test, sep='')
```

```
## lm2 train RMSE: 0.05214415
## lm2 test RMSE:  0.05404892
```

```
#forward sequential variable selection regression model
lm3 = step(lm1, scope = list(upper = formula(lm1)), direction = "forward", trace=0)
lm3rmse_train = RMSE(teams78_drop$W.L, predict(lm3, teams78_drop))
```

```
## Warning in predict.lm(lm3, teams78_drop): prediction from a rank-deficient
## fit may be misleading
```

```
lm3rmse_test = RMSE(teams89_drop$W.L, predict(lm3, teams89_drop))
```

```
## Warning in predict.lm(lm3, teams89_drop): prediction from a rank-deficient
## fit may be misleading
```

```
cat('lm3 train RMSE: ', lm3rmse_train, '\n', 'lm3 test RMSE:  ', lm3rmse_test, sep='')
```

```
## lm3 train RMSE: 0.05182295
## lm3 test RMSE:  0.05492276
```

```
#well-tuned Ridge regression model
set.seed(2)
X_train = model.matrix(formula(lm1), data=teams78_drop)[,-1]
X_test = model.matrix(formula(lm1), data=teams89_drop)[,-1]
ridges = cv.glmnet(X_train, teams78_drop$W.L, alpha=0, lambda=10^seq(-4,4,0.1), nfolds=5)
best_lambda = ridges$lambda.min
ridge1 = glmnet(X_train, teams78_drop$W.L, alpha=0, lambda=best_lambda)
ridgermse_train = RMSE(teams78_drop$W.L, predict(ridge1, X_train))
ridgermse_test = RMSE(teams89_drop$W.L, predict(ridge1, X_test))
cat('ridge train RMSE: ', ridgermse_train, '\n', 'ridge test RMSE:  ', ridgermse_test, sep='')
```

```
## ridge train RMSE: 0.05234369
## ridge test RMSE:  0.05412679
```

```
#well-tuned Lasso regression model
lassos = cv.glmnet(X_train, teams78_drop$W.L, alpha=0, lambda=10^seq(-4,4,0.1), nfolds=5)
best_lambda = lassos$lambda.min
lasso1 = glmnet(X_train, teams78_drop$W.L, alpha=1, lambda=best_lambda)
lassormse_train = RMSE(teams78_drop$W.L, predict(lasso1, X_train))
lassormse_test = RMSE(teams89_drop$W.L, predict(lasso1, X_test))
cat('lasso train RMSE: ', lassormse_train, '\n', 'lasso test RMSE:  ', lassormse_test, sep='')
```

```
## lasso train RMSE: 0.05390289
## lasso test RMSE:  0.0546133
```

```
#regresion tree
tree1 = rpart(W.L. ~ . , data=teams78_drop, control=list(minsplit=1,cp=0.00001,maxdepth=20))
tree1RMSE_train = RMSE(teams78_drop$W.L, predict(tree1, teams78_drop))
tree1RMSE_test = RMSE(teams89_drop$W.L., predict(tree1, teams89_drop))
cat('tree1 train RMSE: ', tree1RMSE_train, '\n', 'tree1 test RMSE:  ', tree1RMSE_test, sep='')
```

```
## tree1 train RMSE: 0.05522243
## tree1 test RMSE:  0.09639657
```

```
#tuned pruned regression tree
cps = 10^(seq(-10, 2,.1))
rmses = rep(NA,length(cps))
for(i in 1:length(cps)){
  cp=cps[i]
  temptree = prune(tree1,cp)
  rmses[i] = RMSE(teams89_drop$W.L., predict(temptree,new=teams89_drop))
}

tree2 = rpart(W.L. ~ ., data=teams78_drop, control=list(minsplit=1,cp=cps[which.min(which.min(rmses))],m
tree2RMSE_train = RMSE(teams78_drop$W.L, predict(tree2, teams78_drop))
```

```r
tree2RMSE_test = RMSE(teams89_drop$W.L., predict(tree2, teams89_drop))
cat('tree2 train RMSE: ', tree2RMSE_train, '\n', 'tree2 test RMSE:  ', tree2RMSE_test, sep='')
```

```
## tree2 train RMSE: 0.05522243
## tree2 test RMSE:  0.09639657
```