Basketball

Nick Bruinsma

12/15/2019

Overview

For the Capstone final project, I have elected to use a dataset of basketball game statistics. Each player is tracked for each game they have played from 2013 to 2018. Stats like points, steals, rebounds, etc are tracked for each game. The application for a machine learnign algorithm for something like this would be to predict player statistics in upcoming games in order to estimate potential daily fantasy sports points to create a team that is more likely to place in the money.We will focus on the points statistic, since it is the most important indicator of daily fantasy sports performance. We will start by revieiwing the data and doing some linear regression in order to better understand which factors are the most improtant to a predictive model

First we load the data

```
###Overview

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------------- tidyverse 1.3.0 --

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

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: lubridate

##
## Attaching package: 'lubridate'

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

```r
if(!require(knncat)) install.packages("knncat", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: knncat
```

```r
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart
```

```r
if(!require(rattle)) install.packages("rattle", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rattle

## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
if(!require(RColorBrewer)) install.packages("RColorBrewer", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: RColorBrewer
```

```r
## Importing packages
library(data.table) # Import big files (CSV or Text)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(readxl) # Import Excel files
library(stringr)
library(lubridate)
library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)

#Import data of nba players for all games between 2013 and 2018
url <- 'https://github.com/NBruinsma/basketballML/raw/master/basic_per_game_player_stats_2013_2018.csv'
dat <- read.csv(url)
#dat <- read.csv('basic_per_game_player_stats_2013_2018.csv')
```

Clean the data for better use with machine learning algorithms

```r
#clean the data

#Rename the home/away column
colnames(dat)[7]  <- "Home"

#Split the Age into Dates and days
dat  <- dat  %>% separate(Age, c("Age", "Days"), sep = "-")



#Split name and unique identifier
dat  <- dat  %>% separate(Player, c("Player", "Unique ID"), sep = "\\\\")


#get year from date
dat <- dat %>% mutate(Year = as.numeric(str_sub(Date,1,4)))
dat$Date <- as.Date(dat$Date)

#Remove any plyers who don't have at least 10 games worth of data
dat <- dat %>% group_by(Player) %>% filter(n() >=10)


#For when we are going beyond linear regression. We can create success or failure criteria based on the
#required to be part of a daily fantasy sports team

#With 9 players, trying to reach a score of at least 200 points we would require an average of 22 point

dat <- dat %>% mutate(Gr22 = case_when(PTS < 22 ~ 'No', PTS >= 22 ~ 'Yes'))
```

Analysis

The plan is to split the data into test and train sets, and then we will conduct a linear regression. Once we have a better idea of the data, we will create a decision tree, to try to predict which players will get higher than 22 points (with nine players, that will give us 198 points total).

First we will split up the data into a test and training set. We have a large sample size, so we can create an effective training set with half of the rows

```r
#Split into test and training sets
# test_set set will be 50% of dat data

set.seed(1)

test_index <- createDataPartition(y = dat$PTS, times = 1, p = 0.5, list = FALSE)
train_set <- dat[-test_index,]
temp <- dat[test_index,]

# Make sure all players in test set are also in train set
test_set <- temp %>%
  semi_join(train_set, by = "Player")

# Add rows removed from test_set set back into train_set set

removed <- anti_join(temp, test_set)
```

```
## Joining, by = c("Rk", "Player", "Unique ID", "Age", "Days", "Pos", "Date", "Tm",
## "Home", "Opp", "Result", "GS", "MP", "FG", "FGA", "FG.", "X2P", "X2PA", "X2P.",
## "X3P", "X3PA", "X3P.", "FT", "FTA", "FT.", "ORB", "DRB", "TRB", "AST", "STL",
## "BLK", "TOV", "PF", "PTS", "GmSc", "Year", "Gr22")
```

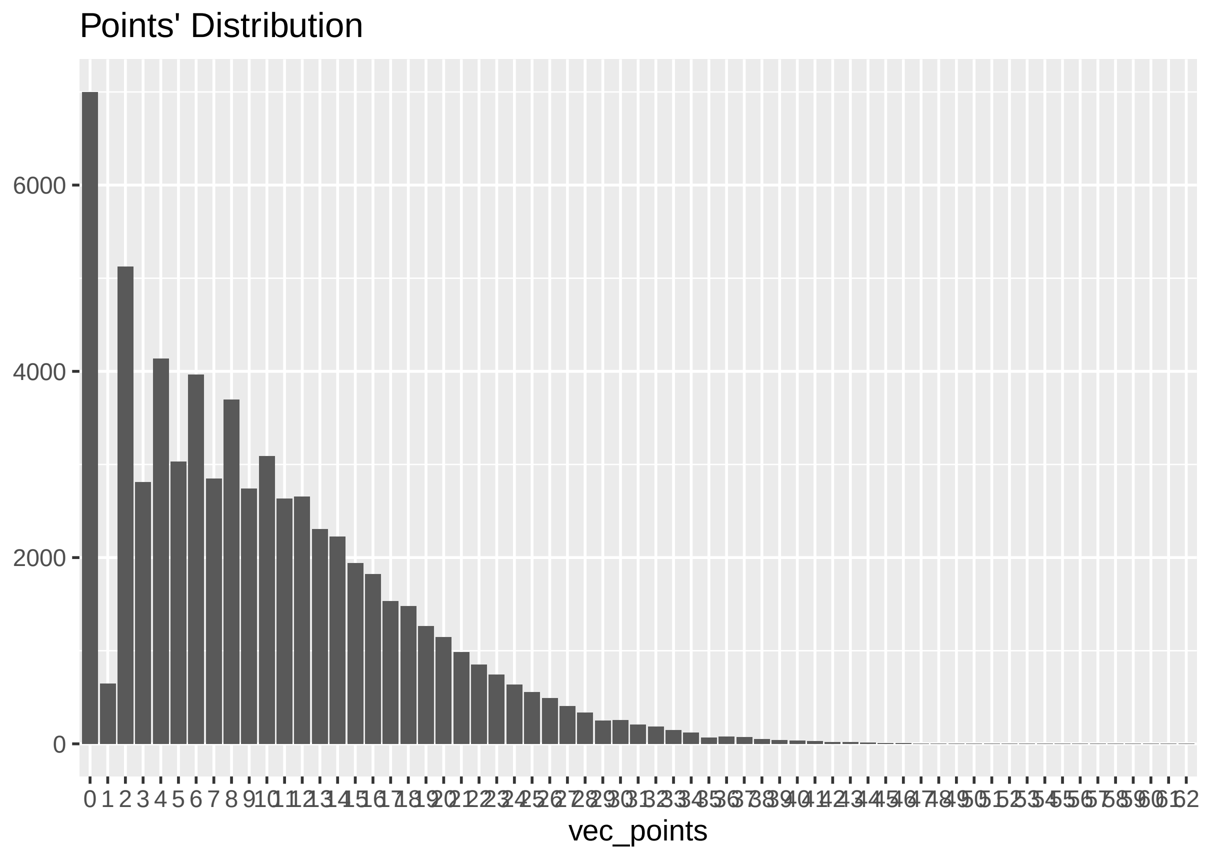
```r
train_set <- rbind(train_set, removed)

rm( test_index, temp, removed)




#Remove the points and field goal data from the test_set
test_set_full <- test_set

#Remove points, 2 point attempts, 3 point attemts, 3 point attempts, percentages hit for each variable
test_set <- test_set %>% select(-PTS, -X2P, -X3P, -FT, -X2P., -X3P., -FT., -FG., -FGA, Gr22)
```

Next we will explore the data a bit, We will plot some different variables to see if there is a useful relationship between the variable and points to see if it can be used as a predictor We wil check the points distribution and the relationship between points and playing at home or away, player, age, year of game, opponent, etc.

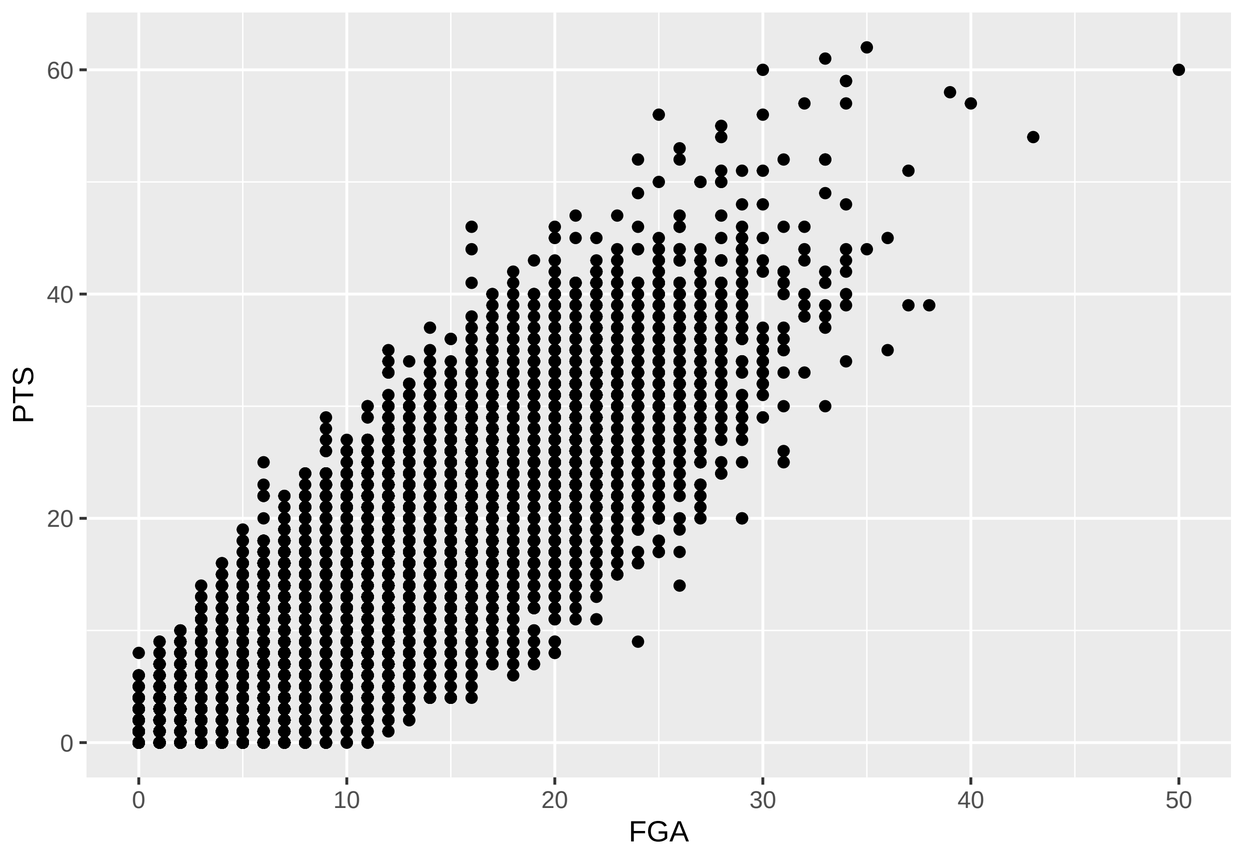
```r
### Analysis

#Determine point distribution
points_dist <- train_set%>%
  group_by(PTS) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

vec_points <- as.vector(train_set$PTS)
unique(vec_points)
```

```
##  [1] 22 20 19 18 15 13 12 11 10  9  8  6  5  4  3  0 32 28 25 24 23 21 17 16 14
## [26]  7  2  1 42 26 33 27 31 30 34 29 45 37 38 35 39 36 43 41 44 54 40 62 51 48
## [51] 61 50 52 57 49 47 46 55 53 56 59 60 58
```

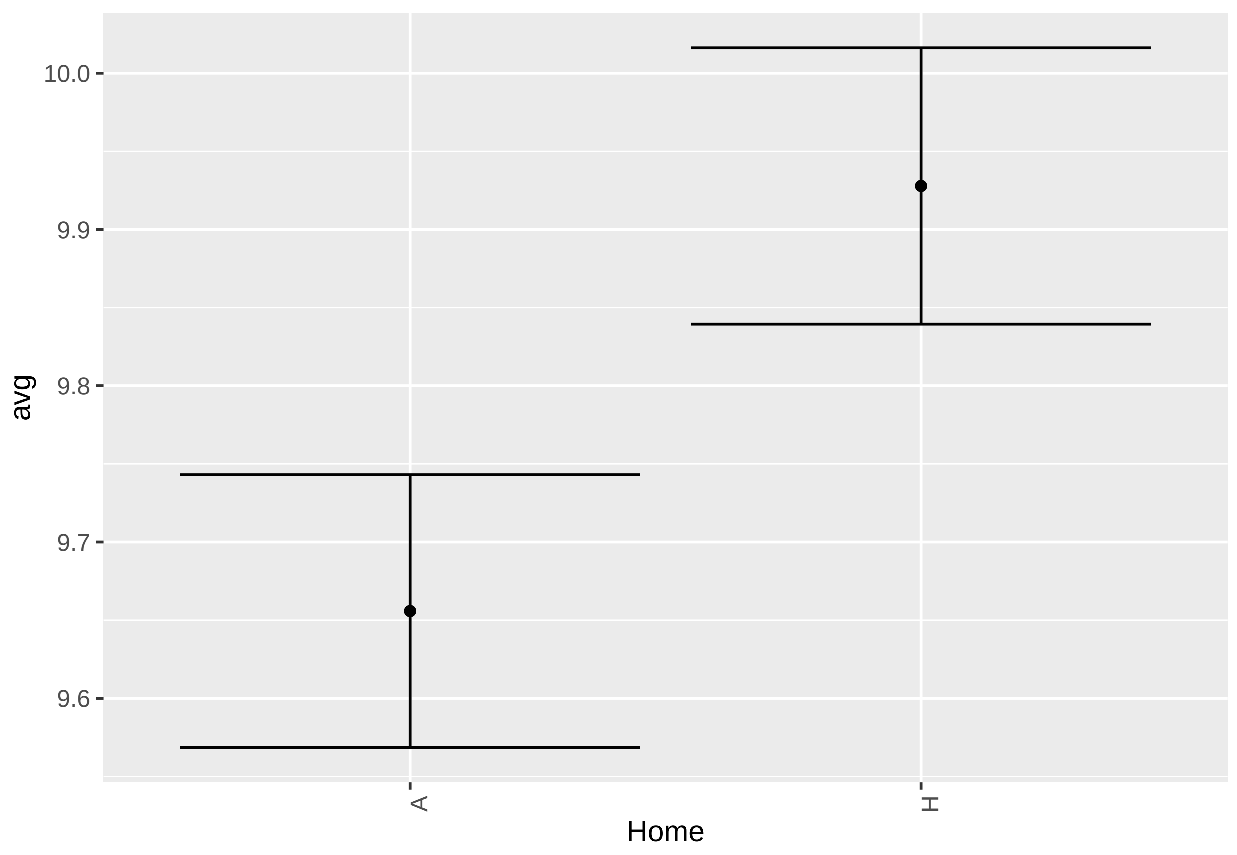
```r
vec_points <- factor(vec_points)
qplot(vec_points) +
  ggtitle("Points' Distribution")
```



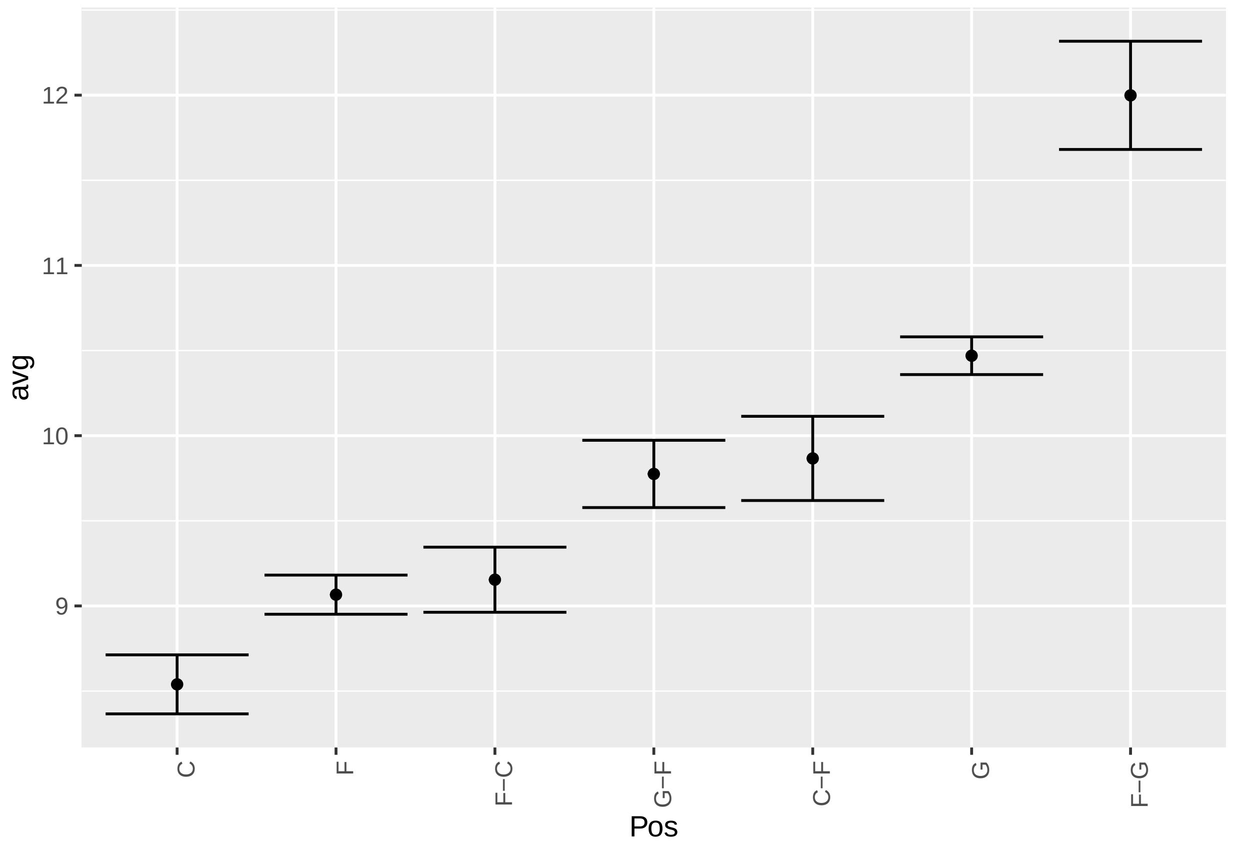
```r
# Field Goal attempts vs points
train_set %>% ggplot(aes(FGA, PTS)) + geom_point()
```



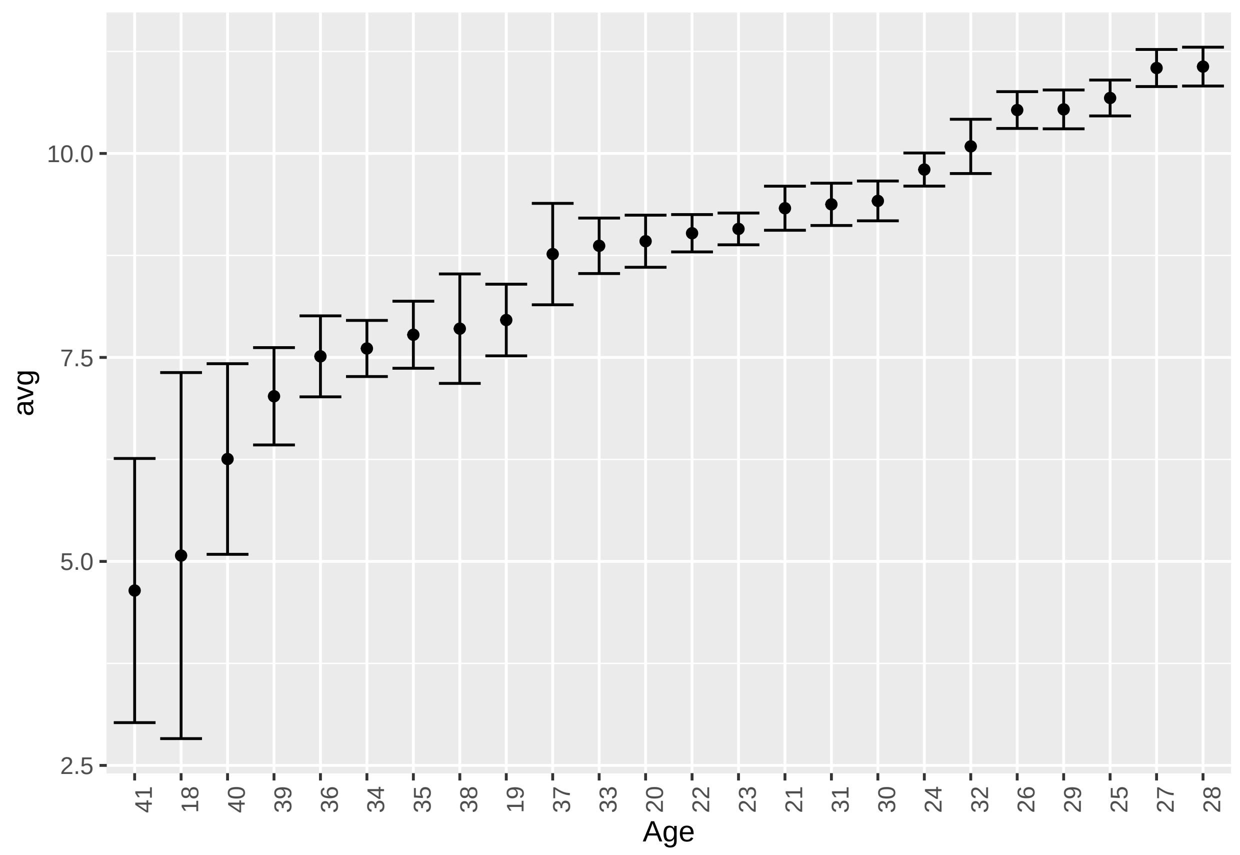
```
#Not surprisingly, more attempts correlates closely with more points - this is not as useful in predict

# Home average vs away average
train_set %>% group_by(Home) %>% summarize(n = n(), avg = mean(PTS), se = sd(PTS)/sqrt(n()))  %>% mutat
```



```
#Home enjoys a slight advantage over away

#Graph points by position
train_set %>% group_by(Pos) %>% summarize(n = n(), avg = mean(PTS), se = sd(PTS)/sqrt(n()))  %>% mutate
```

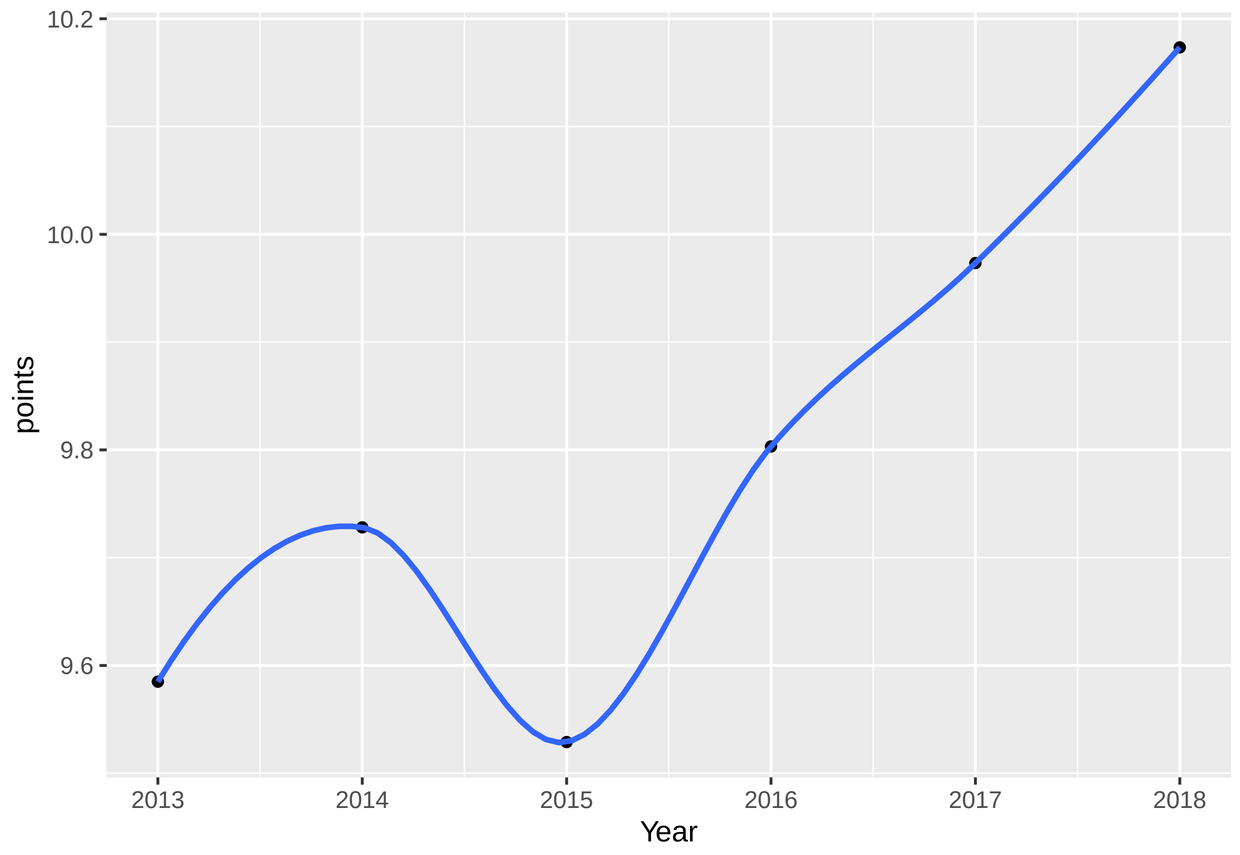


```
#Certain positions are expected to get more points than others

#Graph points by age
train_set %>% group_by(Age) %>% summarize(n = n(), avg = mean(PTS), se = sd(PTS)/sqrt(n()))  %>% mutate
```

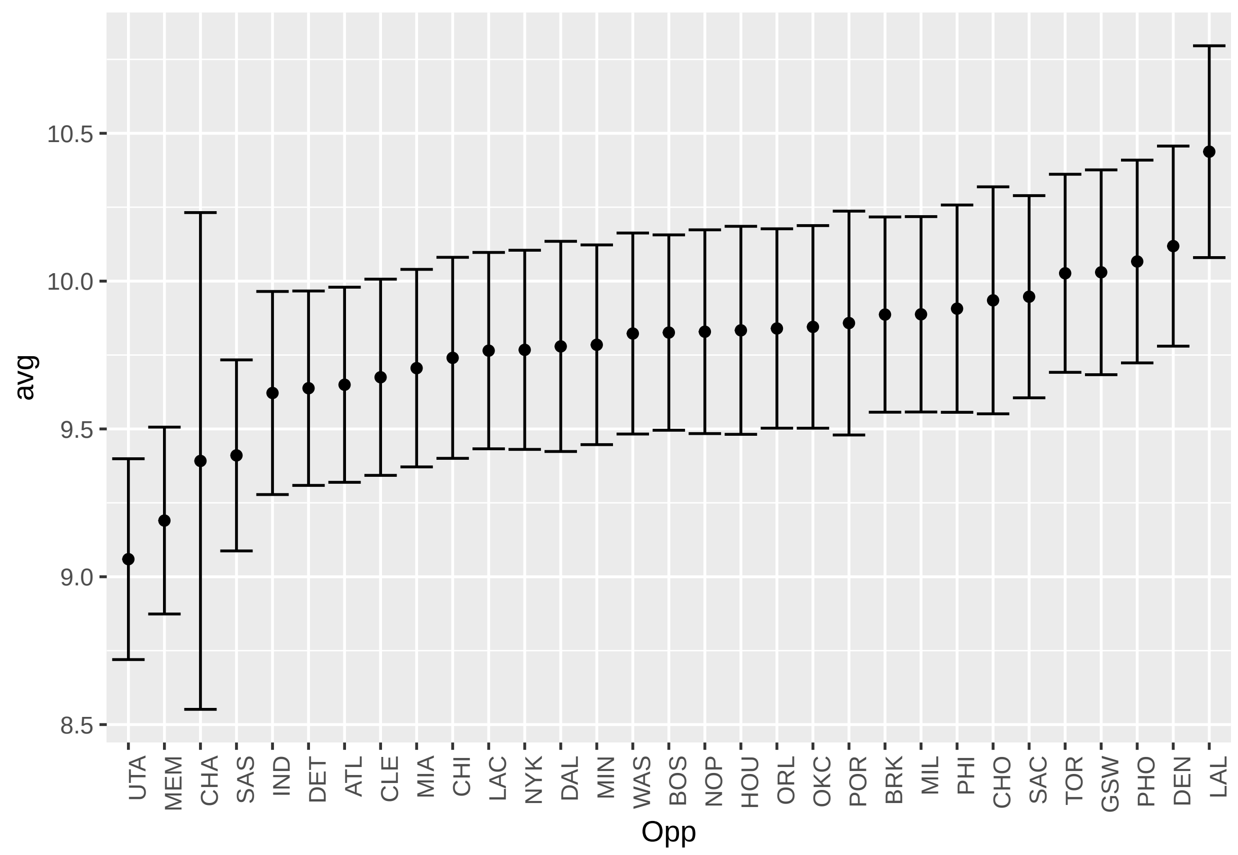


```
#As players get older they tend to score more points, and be more consistant (smaller standard deviatio

#Graph points by year
train_set  %>% group_by(Year) %>% summarize(points = mean(PTS)) %>% ggplot(aes(Year, points)) + geom_poi
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : Chernobyl! trL>n 6

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : Chernobyl! trL>n 6

## Warning in sqrt(sum.squares/one.delta): NaNs produced

## Warning in stats::qt(level/2 + 0.5, pred$df): NaNs produced
```



```
#It appears that average points increase each year

#Graph points by opponent
train_set %>% group_by(Opp) %>% summarize(n = n(), avg = mean(PTS), se = sd(PTS)/sqrt(n()))  %>% mutate
```



We will put together a baseline for out RMSE using just the average.

```
#Initialize RMSE function for comparing actual points scored to predicted rating
RMSE <- function(actual_rating, predicted_rating){ sqrt(mean((actual_rating - predicted_rating)^2)) }
RMSE_results <- tibble()

#Just the mean as a baseline for RMSE
mu <- mean(train_set$PTS)
RMSE_base <- RMSE(train_set$PTS, mu)
rmse_results <- data_frame(method = "Using mean only", RMSE = RMSE_base)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

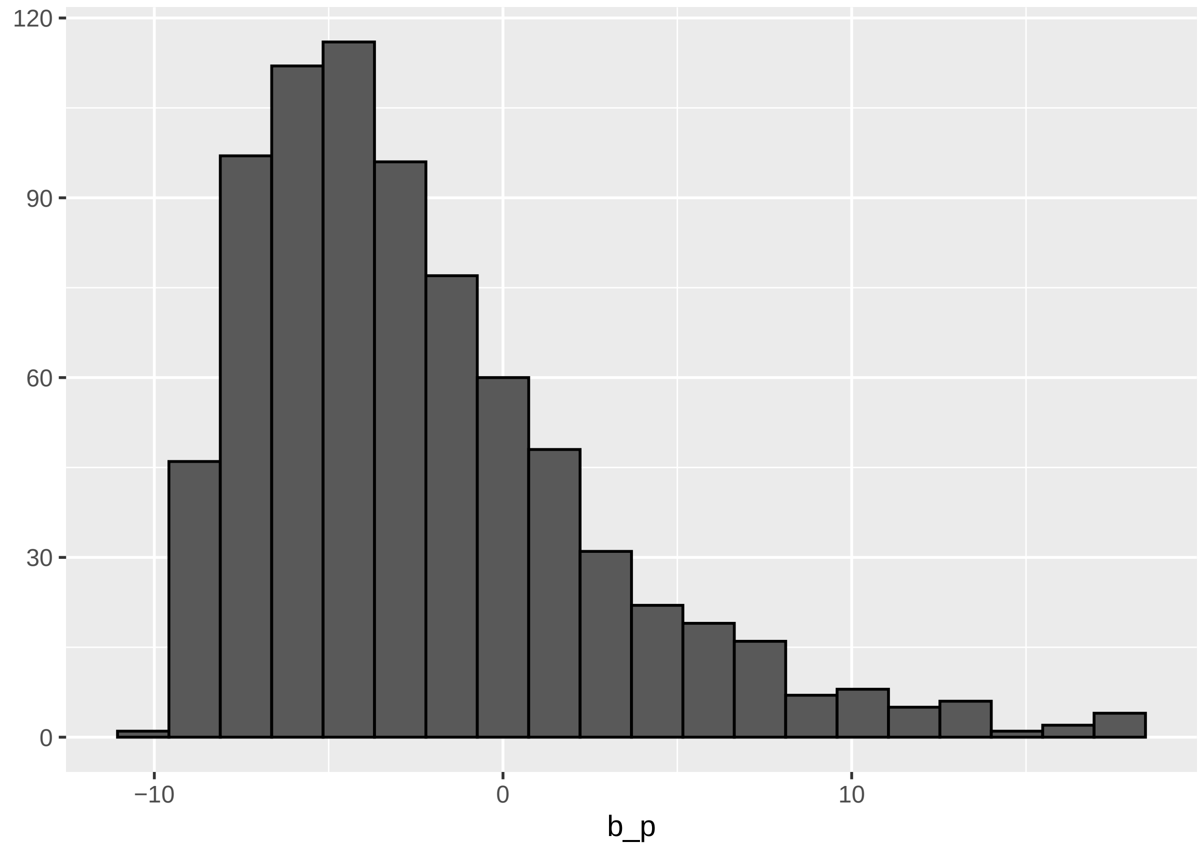
```
options(pillar.sigfig = 9)

rmse_results
```

```
## # A tibble: 1 x 2
##   method                RMSE
##   <chr>                <dbl>
## 1 Using mean only 7.90216151
```

```
#With just the mu, we not that the RMSE is 7.9, which is a significant number of points off on average.
```

We will start using the different variables to see if we can reduce the RMSE, and figure out if they are useful predictors. We can start with players. This is likely to be the strongest indicator. Other games the player has played will strongly inform how well the player will perform in future games.

```
#Start with a prediction based on the player themselves
player_ave <- train_set %>%
  group_by(Player) %>%
  summarize(b_p = mean(PTS - mu))
player_ave %>% qplot(b_p, geom ="histogram", bins = 20, data = ., color = I("black"))
```



```
predicted_player_ave <- test_set %>%
  left_join(player_ave, by='Player') %>%
  mutate(pred = mu + b_p)
rmse_player <- RMSE(test_set_full$PTS,predicted_player_ave$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Player History Model",
                                     RMSE = rmse_player ))
rmse_results
```

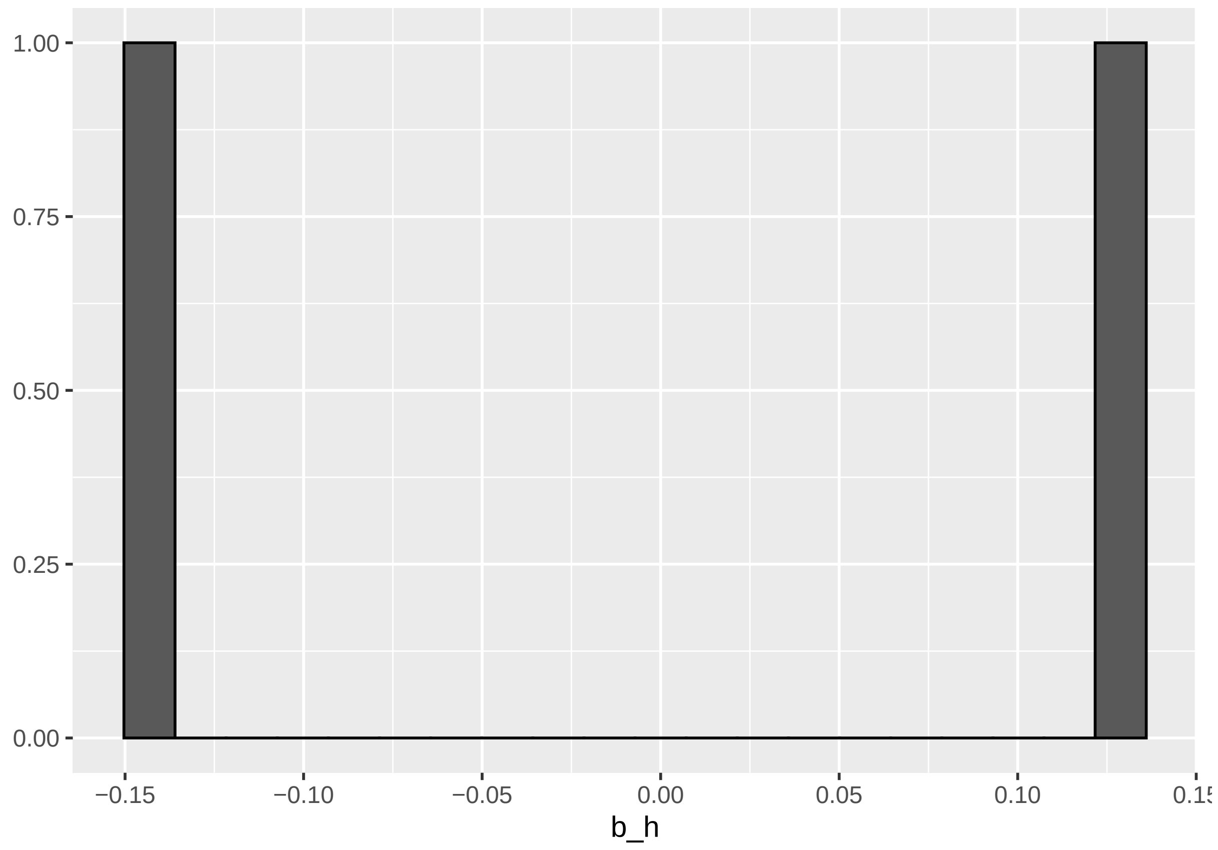
```
## # A tibble: 2 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Using mean only      7.90216151
## 2 Player History Model 5.97132342
```

```
#Drops the RMSE to 5.97 - which is a large improvement, but we can do better
```

Not surprisingly, Player is a strong predictor

Next we will check how effective the home court advantage is

```
#Add Home/Away Effect
home_ave <- train_set %>%
  group_by(Home) %>%
  summarize(b_h = mean(PTS - mu))
home_ave %>% qplot(b_h, geom ="histogram", bins = 20, data = ., color = I("black"))
```



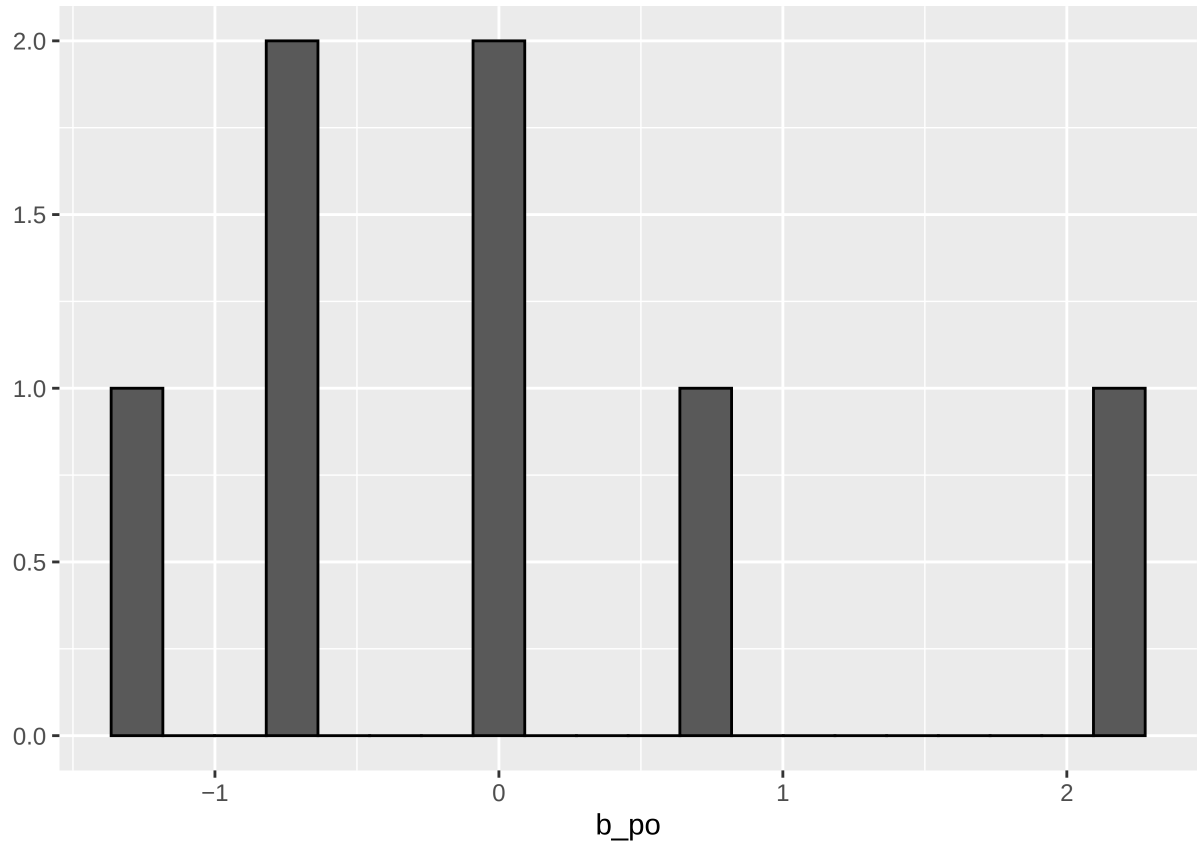
```
predicted_home_ave <- test_set %>%
  left_join(player_ave, by='Player') %>%
  left_join(home_ave, by='Home') %>%
  mutate(pred = mu + b_p + b_h)
rmse_home <- RMSE(test_set_full$PTS,predicted_home_ave$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Add Home Model",
                                     RMSE = rmse_home ))
rmse_results
```

```
## # A tibble: 3 x 2
##   method                  RMSE
##   <chr>                  <dbl>
## 1 Using mean only      7.90216151
## 2 Player History Model 5.97132342
## 3 Add Home Model       5.97118446
```

The player playing home or away makes a slight difference to projected points

Next we can check to see how normalizing for the players position affects the algorithm

```r
#Add Position
pos_ave <- train_set %>%
  group_by(Pos) %>%
  summarize(b_po = mean(PTS - mu))
pos_ave %>% qplot(b_po, geom ="histogram", bins = 20, data = ., color = I("black"))
```



```r
predicted_pos_ave <- test_set %>%
  left_join(player_ave, by='Player') %>%
  left_join(home_ave, by='Home') %>%
  left_join(pos_ave, by='Pos') %>%
  mutate(pred = mu + b_p + b_h + b_po)
rmse_pos <- RMSE(test_set_full$PTS,predicted_pos_ave$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Add Position Model",
                                     RMSE = rmse_pos ))
```

```
rmse_results
```

```
## # A tibble: 4 x 2
##   method                     RMSE
##   <chr>                     <dbl>
## 1 Using mean only      7.90216151
## 2 Player History Model 5.97132342
## 3 Add Home Model       5.97118446
## 4 Add Position Model   6.03393443
```

```
#Note that adding in the position average actually made the model worse - it could be because we are al
#including position by virtue of including the player, and therefore double counting
```

We would expect the RMSE to decrease, because position has a correlation to points. But since we are already using the player, which will intrinsically include position - adding in position as well actually makes our model worse. Basically doubling up on position as a variable.

Just to be sure we are using the best predictor, we can check what the model looks like with position instead of player

```
#Let's remove the player model and see if the position is a more accurate determinate
predicted_pos_ave <- test_set %>%
  left_join(home_ave, by='Home') %>%
  left_join(pos_ave, by='Pos') %>%
  mutate(pred = mu + b_h + b_po)
rmse_pos <- RMSE(test_set_full$PTS,predicted_pos_ave$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Position without Player",
                                     RMSE = rmse_pos ))

rmse_results
```

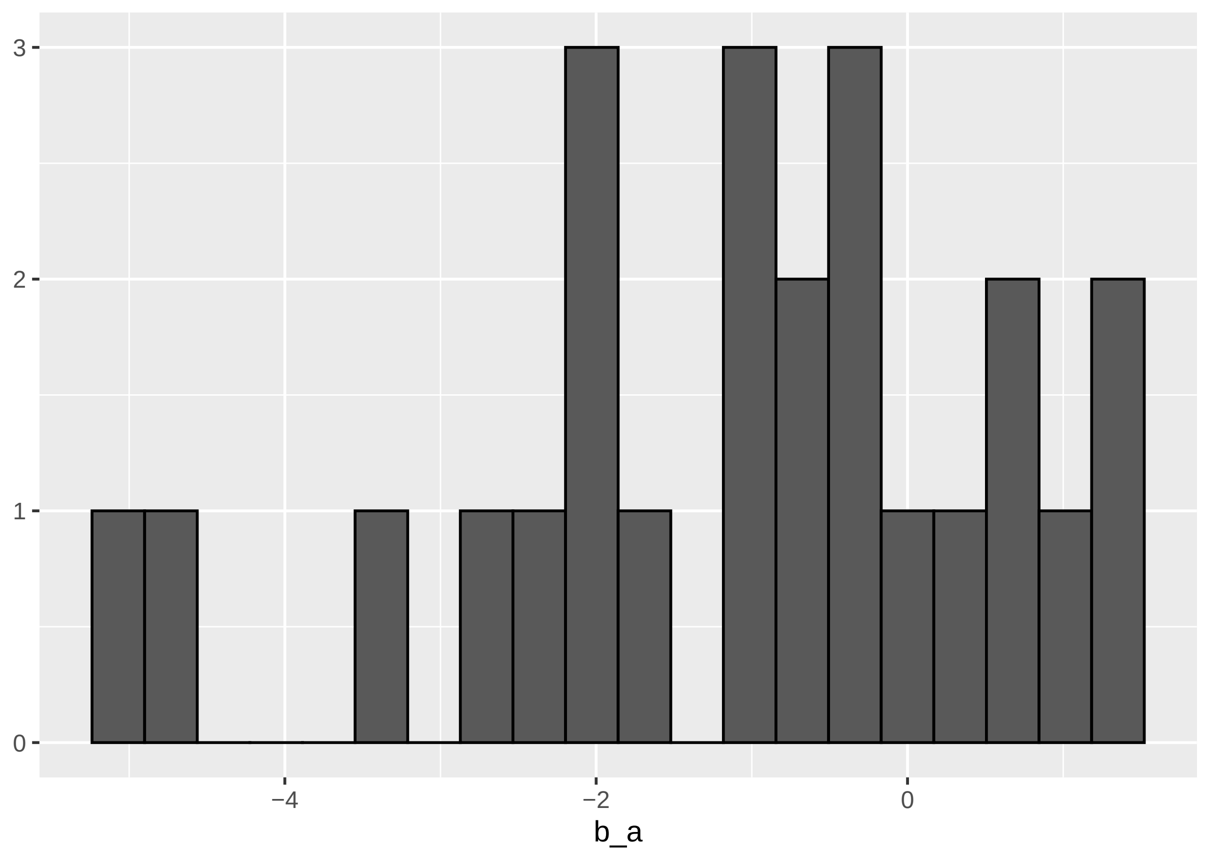
```
## # A tibble: 5 x 2
##   method                        RMSE
##   <chr>                        <dbl>
## 1 Using mean only         7.90216151
## 2 Player History Model    5.97132342
## 3 Add Home Model          5.97118446
## 4 Add Position Model      6.03393443
## 5 Position without Player 7.86274606
```

```
#Clearly using a players direct history is more useful than the position alone
```

That is a much worse RMSE value, so we know player is a better predictor and we can remove position from the model

Next we can check age

```
age_ave <- train_set %>%
  group_by(Age) %>%
  summarize(b_a = mean(PTS - mu))
age_ave %>% qplot(b_a, geom ="histogram", bins = 20, data = ., color = I("black"))
```



```
predicted_age_ave <- test_set %>%
  left_join(player_ave, by='Player') %>%
  left_join(home_ave, by='Home') %>%
  left_join(age_ave, by='Age') %>%
  mutate(pred = mu + b_p + b_h + b_a)
rmse_age <- RMSE(test_set_full$PTS,predicted_age_ave$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Add Age Model (Position removed)",
                                     RMSE = rmse_age ))

rmse_results
```

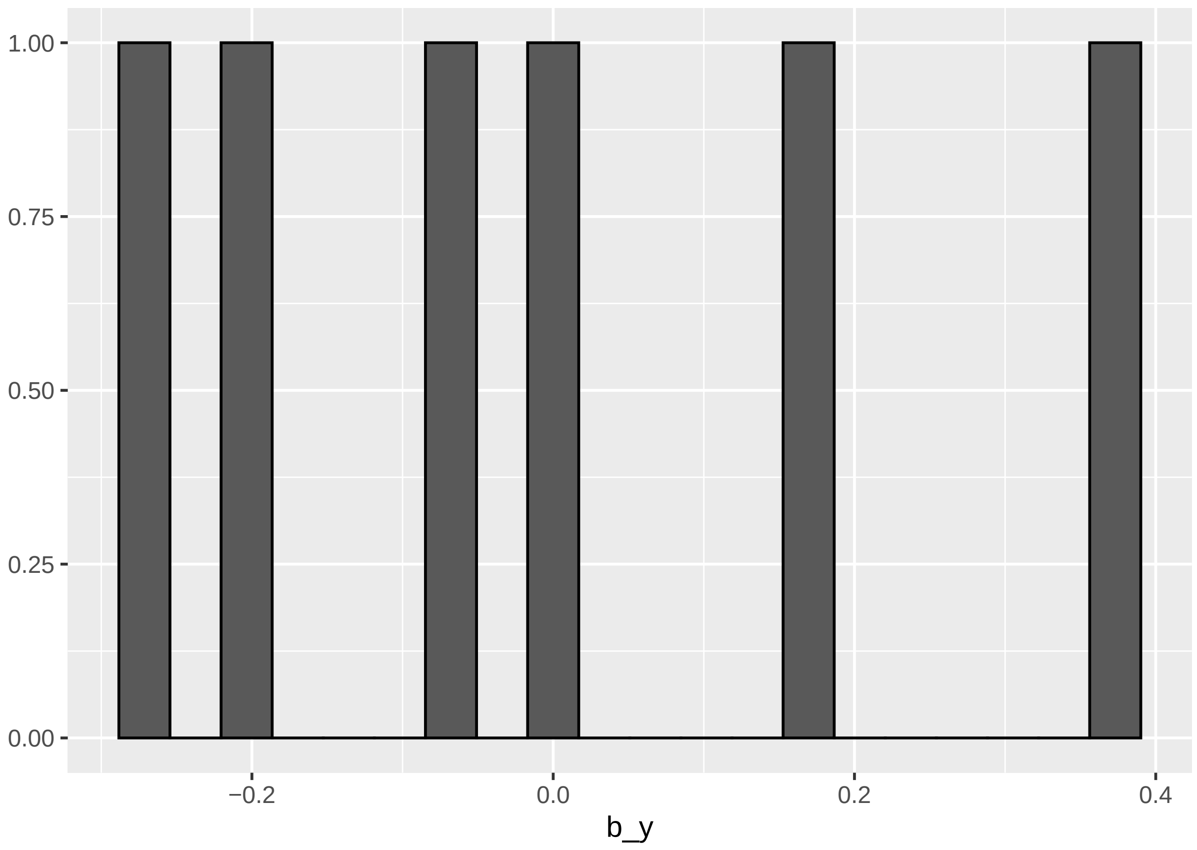
```
## # A tibble: 6 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Using mean only                 7.90216151
## 2 Player History Model            5.97132342
## 3 Add Home Model                  5.97118446
## 4 Add Position Model              6.03393443
## 5 Position without Player         7.86274606
## 6 Add Age Model (Position removed) 5.99375036
```

```
#We see the same result with Age as with position. Clearly the "Player" factors are covered by using th
#Instead we will focus on the "game" factors.
```

Age seems to be similar to position, in that it is already included by virtue of having Player in the model. We will change our focus to "Game" factors, since player factors are likely automatically included in Player

We know that as years pass, there is a higher average points - so we can include year in the model as well.

```r
year_ave <- train_set %>%
  group_by(Year) %>%
  summarize(b_y = mean(PTS - mu))
year_ave %>% qplot(b_y, geom ="histogram", bins = 20, data = ., color = I("black"))
```



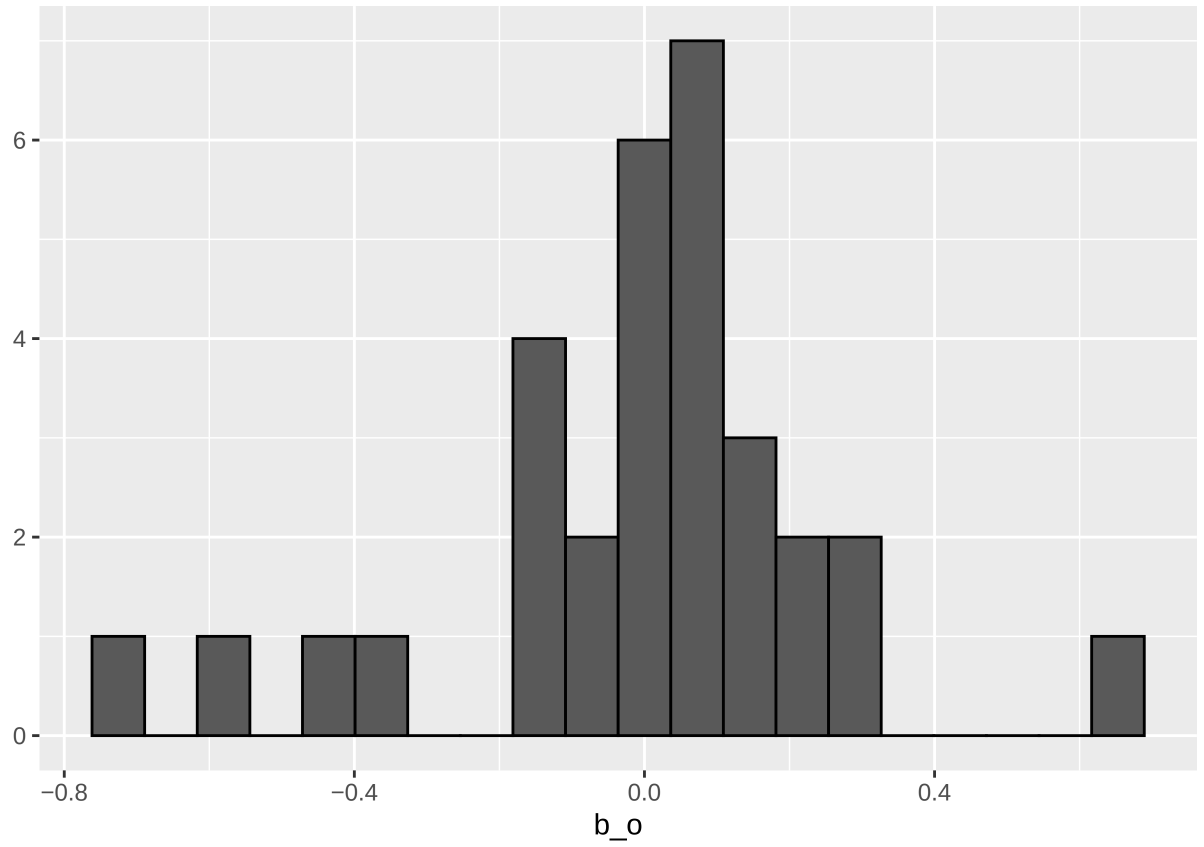
```r
predicted_year_ave <- test_set %>%
  left_join(player_ave, by='Player') %>%
  left_join(home_ave, by='Home') %>%
  left_join(year_ave, by='Year') %>%
  mutate(pred = mu + b_p + b_h + b_y)
rmse_year <- RMSE(test_set_full$PTS,predicted_year_ave$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Add Year Model (Age Removed)",
                                     RMSE = rmse_year ))
rmse_results
```

```
## # A tibble: 7 x 2
##   method                                 RMSE
##   <chr>                                 <dbl>
## 1 Using mean only                  7.90216151
## 2 Player History Model             5.97132342
## 3 Add Home Model                   5.97118446
## 4 Add Position Model               6.03393443
## 5 Position without Player          7.86274606
## 6 Add Age Model (Position removed) 5.99375036
## 7 Add Year Model (Age Removed)     5.96757729
```

That has a slight effect as well, so we will include it in the model

Finally we will check the opponent

```
#opponent included
opp_ave <- train_set %>%
  group_by(Opp) %>%
  summarize(b_o = mean(PTS - mu))
opp_ave %>% qplot(b_o, geom ="histogram", bins = 20, data = ., color = I("black"))
```



```
predicted_opp_ave <- test_set %>%
  left_join(player_ave, by='Player') %>%
  left_join(home_ave, by='Home') %>%
  left_join(year_ave, by='Year') %>%
  left_join(opp_ave, by='Opp') %>%
  mutate(pred = mu + b_p + b_h + b_y + b_o)
rmse_opp <- RMSE(test_set_full$PTS,predicted_opp_ave$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Add Opponent",
```

```r
                                                RMSE = rmse_opp ))
rmse_results
```

```
## # A tibble: 8 x 2
##   method                              RMSE
##   <chr>                              <dbl>
## 1 Using mean only                 7.90216151
## 2 Player History Model            5.97132342
## 3 Add Home Model                  5.97118446
## 4 Add Position Model              6.03393443
## 5 Position without Player         7.86274606
## 6 Add Age Model (Position removed) 5.99375036
## 7 Add Year Model (Age Removed)    5.96757729
## 8 Add Opponent                    5.96671254
```

Again there is a slight effect

```r
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	7.902162
Player History Model	5.971323
Add Home Model	5.971184
Add Position Model	6.033934
Position without Player	7.862746
Add Age Model (Position removed)	5.993750
Add Year Model (Age Removed)	5.967577
Add Opponent	5.966712

Now that we have learned from our liner regression, we can attempt to use the useful variables to create a decision tree that will help predict whether a player will reach a score of 22 points or more First we will create a dataset with just the useful factors to make it easier to manage. Then we will set the test and training sets again

```r
#create a dataset that uses only the colums we found to be good predicters in the linear regression mod
Kdat <- dat %>% select(Player, Pos, Year, Opp, Home, Gr22)

#Dependant variable is a factor
Kdat$Gr22 <- factor(Kdat$Gr22)

#Make year a factor instead of numerical
Kdat$Year <- factor(Kdat$Year)

#maintain consistency with test and training sets
set.seed(1)


test_index <- createDataPartition(y = dat$PTS, times = 1, p = 0.5, list = FALSE)
```

```r
train_set <- Kdat[-test_index,]
temp <- Kdat[test_index,]

# Make sure all players in test set are also in train set
test_set <- temp %>%
  semi_join(train_set, by = "Player")

# Add rows removed from test_set set back into train_set set

removed <- anti_join(temp, test_set)
```

```
## Joining, by = c("Player", "Pos", "Year", "Opp", "Home", "Gr22")
```

```r
train_set <- rbind(train_set, removed)

rm( test_index, temp, removed)
```

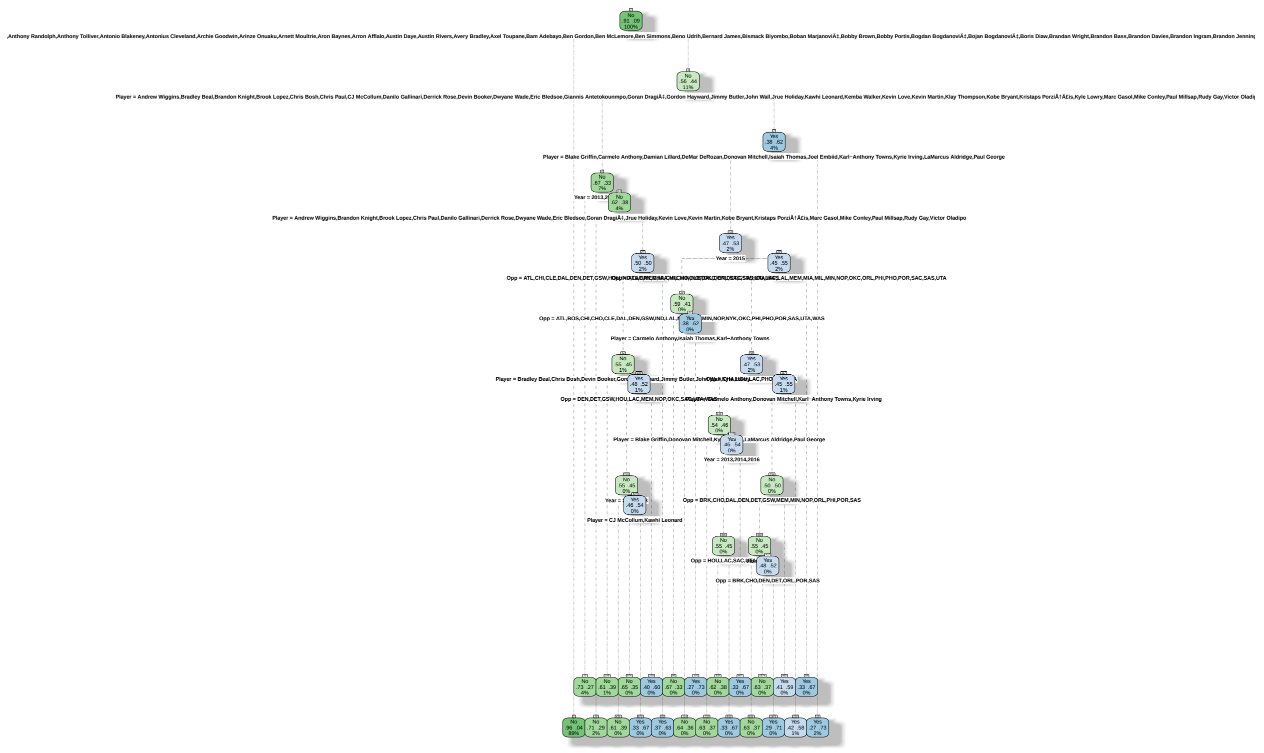
Now we can create a decision tree using Player, Home/Away, Opponent and Year as variables for splitting nodes

```r
#create a tree using Player, year, opponent and home variables (the ones linear regression showed us a

mytree <- rpart(
  Gr22 ~.,
  data = train_set[1:6],
  method = "class",
control =rpart.control(minsplit =1,minbucket=1, cp=0.001)
)

#Explore the tree
mytree
```

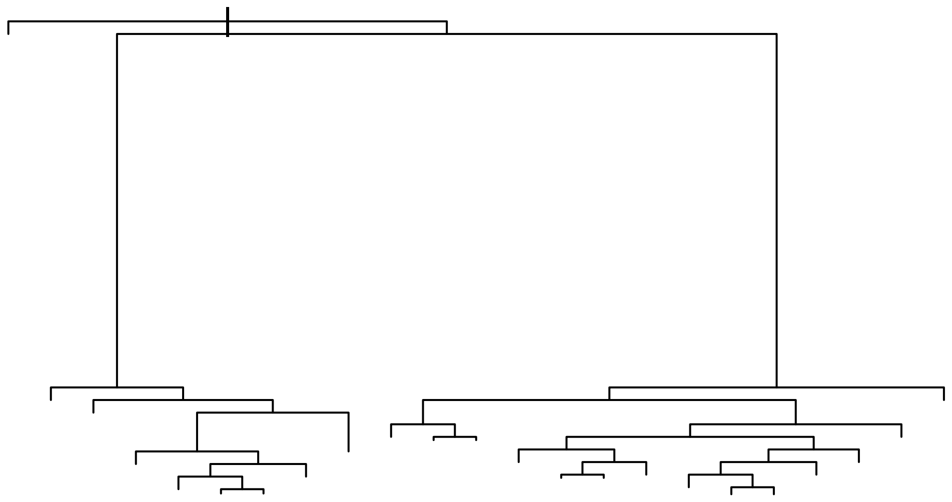
```
## n= 64762
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##    1) root 64762 5669 No (0.91246410 0.08753590)
##      2) Player=Ã-mer AÅŸÄ±k,A.J. Hammons,A.J. Price,Ã lex Abrines,Aaron Brooks,Aaron Gordon,Aaron Gra
##      3) Player=Andrew Wiggins,Anthony Davis,Blake Griffin,Bradley Beal,Brandon Knight,Brook Lopez,Ca
##        6) Player=Andrew Wiggins,Bradley Beal,Brandon Knight,Brook Lopez,Chris Bosh,Chris Paul,CJ McC
##         12) Year=2013,2014,2015 2271  615 No (0.72919419 0.27080581) *
##         13) Year=2016,2017,2018 2315  888 No (0.61641469 0.38358531)
##           26) Player=Andrew Wiggins,Brandon Knight,Brook Lopez,Chris Paul,Danilo Gallinari,Derrick R
##           27) Player=Bradley Beal,Chris Bosh,CJ McCollum,Devin Booker,Giannis Antetokounmpo,Gordon H
##             54) Opp=ATL,CHI,CLE,DAL,DEN,DET,GSW,HOU,IND,LAC,MEM,MIA,MIL,MIN,NOP,OKC,ORL,SAC,SAS,UTA,
##              108) Player=Bradley Beal,Chris Bosh,Devin Booker,Gordon Hayward,Jimmy Butler,John Wall,
##              109) Player=CJ McCollum,Giannis Antetokounmpo,Kawhi Leonard,Kemba Walker,Klay Thompson
##                218) Opp=DEN,DET,GSW,HOU,LAC,MEM,NOP,OKC,SAC,UTA,WAS 186   83 No (0.55376344 0.446236
##                  436) Year=2016,2018 118   46 No (0.61016949 0.38983051) *
##                  437) Year=2017 68   31 Yes (0.45588235 0.54411765)
##                    874) Player=CJ McCollum,Kawhi Leonard 26    9 No (0.65384615 0.34615385) *
##                      875) Player=Giannis Antetokounmpo,Kemba Walker,Klay Thompson 42   14 Yes (0.33333
##                  219) Opp=ATL,CHI,CLE,DAL,IND,MIA,MIL,MIN,ORL,SAS 144   57 Yes (0.39583333 0.60416667)
##               55) Opp=BOS,BRK,CHO,LAL,NYK,PHI,PHO,POR,TOR 309  115 Yes (0.37216828 0.62783172) *
##         7) Player=Anthony Davis,Blake Griffin,Carmelo Anthony,Damian Lillard,DeMar DeRozan,DeMarcus C
##           14) Player=Blake Griffin,Carmelo Anthony,Damian Lillard,DeMar DeRozan,Donovan Mitchell,Isaia
##             28) Year=2015 257  106 No (0.58754864 0.41245136)
##               56) Opp=ATL,BOS,CHI,CHO,CLE,DAL,DEN,GSW,IND,LAL,MEM,MIL,MIN,NOP,NYK,OKC,PHI,PHO,POR,SAS,
##               57) Opp=BRK,DET,HOU,LAC,MIA,ORL,SAC,TOR 73   28 Yes (0.38356164 0.61643836)
##                114) Player=Carmelo Anthony,Isaiah Thomas,Karl-Anthony Towns 22    8 No (0.63636364 0.3
##                115) Player=Blake Griffin,Damian Lillard,DeMar DeRozan,Kyrie Irving,LaMarcus Aldridge,P
##             29) Year=2013,2014,2016,2017,2018 1296  578 Yes (0.44598765 0.55401235)
##               58) Opp=ATL,BRK,CHA,CHI,CHO,CLE,DAL,DEN,DET,GSW,HOU,LAC,LAL,MEM,MIA,MIL,MIN,NOP,OKC,ORL,
##                116) Opp=CHA,HOU,LAC,PHO,SAC,UTA 219  100 No (0.54337900 0.45662100)
##                  232) Player=Blake Griffin,Donovan Mitchell,Kyrie Irving,LaMarcus Aldridge,Paul George
##                  233) Player=Carmelo Anthony,Damian Lillard,DeMar DeRozan,Isaiah Thomas,Joel Embiid,Ka
##                    466) Year=2013,2014,2016 71   32 No (0.54929577 0.45070423)
##                      932) Opp=HOU,LAC,SAC,UTA 53   20 No (0.62264151 0.37735849) *
##                      933) Opp=CHA,PHO 18    6 Yes (0.33333333 0.66666667) *
##                    467) Year=2017,2018 43   14 Yes (0.32558140 0.67441860) *
##                117) Opp=ATL,BRK,CHI,CHO,CLE,DAL,DEN,DET,GSW,LAL,MEM,MIA,MIL,MIN,NOP,OKC,ORL,PHI,POR,SA
##                  234) Player=Carmelo Anthony,Donovan Mitchell,Karl-Anthony Towns,Kyrie Irving 287  143
##                    468) Opp=BRK,CHO,DAL,DEN,DET,GSW,MEM,MIN,NOP,ORL,PHI,POR,SAS 182   81 No (0.5549450
##                      936) Home=A 92   34 No (0.63043478 0.36956522) *
##                      937) Home=H 90   43 Yes (0.47777778 0.52222222)
##                       1874) Opp=BRK,CHO,DEN,DET,ORL,POR,SAS 49   18 No (0.63265306 0.36734694) *
##                       1875) Opp=DAL,GSW,MEM,MIN,NOP,PHI 41   12 Yes (0.29268293 0.70731707) *
##                    469) Opp=ATL,CHI,CLE,LAL,MIA,MIL,OKC 105   43 Yes (0.40952381 0.59047619) *
##                  235) Player=Blake Griffin,Damian Lillard,DeMar DeRozan,Isaiah Thomas,Joel Embiid,LaMa
##               59) Opp=BOS,IND,NYK,TOR,WAS 220   73 Yes (0.33181818 0.66818182) *
##           15) Player=Anthony Davis,DeMarcus Cousins,James Harden,Kevin Durant,LeBron James,Russell Wes
```

```r
#This one becomes a bit of a mess, because there are so many players names to include in the tree. We c
#however it would be a shame to lose our best available predictor and the overall accuracy would decrea
fancyRpartPlot(mytree, caption = NULL)
```

```
## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e
## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81
## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning: labs do not fit even at cex 0.15, there may be some overplotting

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font width unknown for character 0x81
## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x81

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x81

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x81

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x8d

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x8d

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x8d

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x81

## Warning in strwidth(s[i], units[i], cex[i], font[i], vfont = NULL): font width
## unknown for character 0x1e

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font width unknown for character 0x81

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x81

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x81

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x81

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x8d

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x8d

## Warning in text.default(xy$x, xy$y, sep.labs$in.box, cex = cex, font = font, :
## font metrics unknown for character 0x8d
```



```r
plot(mytree)
```



As we can see in the plotting, the visual of the tree is a bit messy, since there are player names inlcuded and we have 868 players which split into different groups but retain all of the names for future predictablility, so displaying the different splits makes a bit of a mess

We need to install a couple more packages

Results

```
#generate predictions for a test set
pred <- predict(mytree, test_set, type = "class")

#change Gr22 to a factor for comparison
test_set_full$Gr22 <- factor(test_set_full$Gr22)

#generate confusion matrix
confusionMatrix(pred, test_set_full$Gr22)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction    No   Yes
##        No  57960  3998
##        Yes  1109  1698
##
##                 Accuracy : 0.9211
##                  95% CI : (0.919, 0.9232)
##     No Information Rate : 0.9121
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3624
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9812
##             Specificity : 0.2981
##          Pos Pred Value : 0.9355
##          Neg Pred Value : 0.6049
##              Prevalence : 0.9121
##          Detection Rate : 0.8949
##    Detection Prevalence : 0.9567
##       Balanced Accuracy : 0.6397
##
##        'Positive' Class : No
##
```

```r
#check accuracy
accuracy <- postResample(pred, test_set_full$Gr22)
accuracy
```

```
##  Accuracy     Kappa
## 0.9211457 0.3623627
```

Generating a confusion matrix shows us that the model predicts whether a player will score 22 or more points with 92% accuracy. This sounds pretty good, but a deeper look shows that we have a lot of players that score 22 or greater that we fail to predict. This is not a terrible result, because we don't need to predict all of the players correctly, but we do want to be sure that we are not predicting too many players to be greater than 22 and getting it wrong. After all, we only need to choose 9 players for a team.

A more accurate model might take into account each of the different statistics that are tracked in daily fantasy sports, in order to get a more well rounded estimate. We might also do a formula where we divide projected points by the daily fantasy sports cost in order to get a points/dollar estimate of the players projected performance. That data is available from betting sites, but it is typically in the format of one nights games per excel sheet, so it would require more extensive scraping and stitching together of data to have a useful set for machine learning.

Conclusion

With a data set of points per game from previous games played, we can make a prediciton of the points a player will get within approximately 6 points using linear regression. Using a decision tree we can attempt to determine which players will score above a threshold with an accuracy of approximately 92%. With additional factors for daily fantasy sports we might be able to get a more accurate prediction. If we could include injury data and team lineups, we could attempt to predict when an injury will lead to increased playing time for a player. That can often predict whether they score well above their daily fantasy sports score.