Baseball Statistics

Stryder R. Coleman, Padraic Darby, Chesteran Holligan, Shyan Walker

5/21/2021

## Abstract

In our project, we cleaned, transformed, modeled baseball statistics from the Baseball Databank, in particular the player data. We explored different variables to determine what is a predictor of position players’ and pitchers’ future performance, both in recognition (awards) and performance statistics (WAR and FIP); while also analyzing the relationship between regular season and postseason OPS. We discovered that the overall performance of a player can be quantified in several different ways.

## Introduction

The title of our database is called Baseball Databank. In our project, we will be analyzing our Baseball database and writing brief documentation about it. The data set contains Excel sheets consisting of all the baseball statistical data from the years 1871-2015, we will only be using the data from 1995 to 2015; and will be including 7 data frames. The first dataframe is batting which contains statistics related to batting. The second data frame, battingPost, contains batting statistics in the postseason. The third data frame, Fielding, contains variables related to fielding statistics. The fourth data frame, Pitching, contains pitching statistics for a given player on a given year. The fifth data frame, pitchingPost, contains statistics for pitching in the postseason. The sixth data frame, master, contains data about each player’s name, date of birth, and biographical information in the dataset. Finally the data frame, Salaries, contains a player’s salary in a given year. Every data frame is linked together with PlayerID and YearID.

For our dataset, we are going to answer a series of 7 questions that we believe our dataset can answer. First, we want to know if we can predict the number of runs a batter will score over a season? Second, we are going to investigate what makes a good position player; by looking into the predictor variables of WAR. We are going to investigate whether salary, birth location, age, and/or other variables are good predictor variables for determining a player’s war. The third question is about determining FIP. We want to know how to predict the FIP, therefore we are looking into whether salary, birth location, age, and/or other variables are good predictor variables for determining a pitcher’s FIP. Fourth, we want to determine the shortfalls of using traditional baseball statistics to rate a player’s performance. In order to figure out this question, we will compare an advanced statistic such as wOBA and compare it with a traditional statistical counterpart such as Batting Average. Fifth ,we want to know if there is a correlation between salaries and the performance of players? To determine the correlation we are going to compare player’s WAR to their salaries. Sixth, we are going to investigate how to predict the number of runs a pitcher will allow over a season. Seven, we want to know how the pressure of the postseason impacts a player’s performance. We will solve this by comparing a player’s OPS in the regular season to a player’s OPS in the postseason.

## Intial Goals

1. Initial Goals
2. Can we predict the number of runs a player will score over a season?
3. Can we predict a position player’s performance over a season?
4. Can we predict a pitcher’s performance over a season?
5. What are the flaws of traditional baseball statistics?
6. Is salary a good predictor of Awards won?
7. Can we predict the number of earned runs (ER) a pitcher will allow over a season?
8. Can we predict a position player’s batting postseason performance by using their regular season data?

## Library used

Every library we used

## Warning: package 'ggbeeswarm' was built under R version 4.0.5

## Loading required package: ggplot2

## Warning: package 'ggforce' was built under R version 4.0.5

## Warning: package 'tidyverse' was built under R version 4.0.5

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

## v tibble 3.1.0 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v purrr 0.3.4 v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.0.4

## Warning: package 'tidyr' was built under R version 4.0.4

## Warning: package 'dplyr' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

## Warning: package 'rpart' was built under R version 4.0.5

## Warning: package 'car' was built under R version 4.0.5

## Loading required package: carData

##   
## Attaching package: 'car'

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

## The following object is masked from 'package:purrr':  
##   
## some

## Warning: package 'e1071' was built under R version 4.0.5

##   
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':  
##   
## cross

## The following object is masked from 'package:ggplot2':  
##   
## alpha

## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

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

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

##   
## Attaching package: 'sjmisc'

## The following object is masked from 'package:purrr':  
##   
## is\_empty

## The following object is masked from 'package:tidyr':  
##   
## replace\_na

## The following object is masked from 'package:tibble':  
##   
## add\_case

## Warning: package 'lme4' was built under R version 4.0.5

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

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

## Registered S3 methods overwritten by 'lme4':  
## method from  
## cooks.distance.influence.merMod car   
## influence.merMod car   
## dfbeta.influence.merMod car   
## dfbetas.influence.merMod car

## Warning: package 'ModelMetrics' was built under R version 4.0.5

##   
## Attaching package: 'ModelMetrics'

## The following objects are masked from 'package:caret':  
##   
## confusionMatrix, precision, recall, sensitivity, specificity

## The following object is masked from 'package:base':  
##   
## kappa

## Importing the data frames

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## playerID = col\_character(),  
## awardID = col\_character(),  
## yearID = col\_double(),  
## lgID = col\_character(),  
## tie = col\_logical(),  
## notes = col\_character()  
## )

## Warning: 45 parsing failures.  
## row col expected actual file  
## 2711 tie 1/0/T/F/TRUE/FALSE Y 'AwardsPlayers.csv'  
## 2712 tie 1/0/T/F/TRUE/FALSE Y 'AwardsPlayers.csv'  
## 3052 tie 1/0/T/F/TRUE/FALSE Y 'AwardsPlayers.csv'  
## 3053 tie 1/0/T/F/TRUE/FALSE Y 'AwardsPlayers.csv'  
## 3141 tie 1/0/T/F/TRUE/FALSE Y 'AwardsPlayers.csv'  
## .... ... .................. ...... ...................  
## See problems(...) for more details.

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_double(),  
## playerID = col\_character(),  
## teamID = col\_character(),  
## lgID = col\_character(),  
## SF = col\_logical(),  
## GIDP = col\_logical()  
## )  
## i Use `spec()` for the full column specifications.

## Warning: 44702 parsing failures.  
## row col expected actual file  
## 25013 GIDP 1/0/T/F/TRUE/FALSE 2 'Batting.csv'  
## 25014 GIDP 1/0/T/F/TRUE/FALSE 10 'Batting.csv'  
## 25016 GIDP 1/0/T/F/TRUE/FALSE 4 'Batting.csv'  
## 25026 GIDP 1/0/T/F/TRUE/FALSE 8 'Batting.csv'  
## 25028 GIDP 1/0/T/F/TRUE/FALSE 3 'Batting.csv'  
## ..... .... .................. ...... .............  
## See problems(...) for more details.

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_double(),  
## round = col\_character(),  
## playerID = col\_character(),  
## teamID = col\_character(),  
## lgID = col\_character()  
## )  
## i Use `spec()` for the full column specifications.

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## playerID = col\_character(),  
## yearID = col\_double(),  
## stint = col\_double(),  
## teamID = col\_character(),  
## lgID = col\_logical(),  
## POS = col\_character(),  
## G = col\_double(),  
## GS = col\_logical(),  
## InnOuts = col\_logical(),  
## PO = col\_double(),  
## A = col\_double(),  
## E = col\_double(),  
## DP = col\_double(),  
## PB = col\_double(),  
## WP = col\_logical(),  
## SB = col\_logical(),  
## CS = col\_logical(),  
## ZR = col\_logical()  
## )

## Warning: 336722 parsing failures.  
## row col expected actual file  
## 1504 lgID 1/0/T/F/TRUE/FALSE NL 'Fielding.csv'  
## 1505 lgID 1/0/T/F/TRUE/FALSE NL 'Fielding.csv'  
## 1506 lgID 1/0/T/F/TRUE/FALSE NL 'Fielding.csv'  
## 1507 lgID 1/0/T/F/TRUE/FALSE NL 'Fielding.csv'  
## 1508 lgID 1/0/T/F/TRUE/FALSE NL 'Fielding.csv'  
## .... .... .................. ...... ..............  
## See problems(...) for more details.

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_character(),  
## birthYear = col\_double(),  
## birthMonth = col\_double(),  
## birthDay = col\_double(),  
## deathYear = col\_double(),  
## deathMonth = col\_double(),  
## deathDay = col\_double(),  
## weight = col\_double(),  
## height = col\_double(),  
## debut = col\_date(format = ""),  
## finalGame = col\_date(format = "")  
## )  
## i Use `spec()` for the full column specifications.

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_double(),  
## playerID = col\_character(),  
## teamID = col\_character(),  
## lgID = col\_character(),  
## IBB = col\_logical(),  
## SH = col\_logical(),  
## SF = col\_logical(),  
## GIDP = col\_logical()  
## )  
## i Use `spec()` for the full column specifications.

## Warning: 25692 parsing failures.  
## row col expected actual file  
## 14581 IBB 1/0/T/F/TRUE/FALSE 4 'Pitching.csv'  
## 14582 IBB 1/0/T/F/TRUE/FALSE 5 'Pitching.csv'  
## 14583 IBB 1/0/T/F/TRUE/FALSE 6 'Pitching.csv'  
## 14587 IBB 1/0/T/F/TRUE/FALSE 2 'Pitching.csv'  
## 14588 IBB 1/0/T/F/TRUE/FALSE 4 'Pitching.csv'  
## ..... ... .................. ...... ..............  
## See problems(...) for more details.

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_double(),  
## playerID = col\_character(),  
## round = col\_character(),  
## teamID = col\_character(),  
## lgID = col\_character()  
## )  
## i Use `spec()` for the full column specifications.

## Warning: 1 parsing failure.  
## row col expected actual file  
## 4875 BAOpp a double - 'PitchingPost.csv'

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## yearID = col\_double(),  
## teamID = col\_character(),  
## lgID = col\_character(),  
## playerID = col\_character(),  
## salary = col\_double()  
## )

## Filtering the data set

The years 1995-2015 were selected because we wanted to use data that represented modern baseball. Before 1995 the game was played differently with less reliance on relief pitching and homeruns. Another issue with the data prior 1995, was that baseball was segregated until 1947 when MLB started to desegregate slowly. Leaving talent split across segregated leagues and allowing worse players to get into the Majors. Third, the amount of players born internationally has grown significantly; with more international players than ever before. So we wanted to select an era of baseball that reflects that.

##Filtering dataframes to only include data between 1995-2015   
AwardsPlayers <- AwardsPlayers %>% filter(yearID >= 1995)  
Batting <- Batting %>% filter(yearID >= 1995)  
BattingPost <- BattingPost %>% filter(yearID >= 1995)  
Fielding <- Fielding %>% filter(yearID >= 1995)  
Pitching <- Pitching %>% filter(yearID >= 1995)  
PitchingPost <- PitchingPost %>% filter(yearID >= 1995)  
Master <- Master %>% filter(finalGame >= as.Date("1995-04-25"))  
Salaries <- Salaries %>% filter(yearID >= 1995)

## Transforming the Batting Statistics

We needed to tweak the batting data frame to meet our goals; since the players information was split over stints in a given year, therefore we needed to combine the observations into one single observation of a player’s batting statistics over a single season. Once that was done we needed to create several statistics SLG, XBH, SBP, 1B, OBA, wOBA, and OPS.

## Transforming Batting Statistics  
## removing undesirable variable  
Batting2 <- Batting %>% subset(select = -c(teamID, lgID, SF, GIDP, stint))  
## na is equivalent to 0  
Batting2[is.na(Batting2)] = 0  
## Combining all player batting stats into seasons  
Batting3 <- aggregate(. ~ playerID + yearID, data = Batting2, sum)  
## Creating on base average (OBA) higher indicates better players  
Batting3 <- Batting3 %>% mutate(OBA = ((H + BB + HBP)/(AB+ BB + HBP)))  
## Creating extra base hits (XBH)  
Batting3 <- Batting3 %>% mutate(XBH = (`2B` + `3B` + HR))  
## Creating Slugging (SLG) higher indicates better players  
Batting3 <- Batting3 %>% mutate(SLG = (((H-XBH)+(2\*`2B`)+(3\*`3B`)+(4\*HR))/(AB)))  
## Creating On Base Plus Slugging (OPS) higher indicates better players  
Batting3 <- Batting3 %>% mutate(OPS = (OBA + SLG))  
## Creating on base average Stolen Base Percentage (SBP) higher indicates better players  
Batting3 <- Batting3 %>% mutate(SBP = ((SB)/(SB + CS)))  
## Creating on singles stat  
Batting3 <- Batting3 %>% mutate(`1B` = (H-XBH))  
## Creating on singles stat  
Batting3 <- Batting3 %>% mutate(BO = (AB - H))   
## Creating BA  
Batting3 <- Batting3 %>% mutate(BA = (AB/H))  
## Creating wOBA  
Batting3 <- Batting3 %>% mutate(wOBA = (.69\*(BB) +.72\*(HBP) +.89\*(H - XBH) + 1.27\*(`2B`) + 1.62\*(`3B`) + HR\*2.10) / (AB + BB - IBB + SH +HBP))

## Transforming the Fielding (Position Players) Statistics

Similarly to the Batting dataframe, we combined the stint data into single seasonal observation. Once combined, we created a fielding statistic called FP.

##Transforming fielding statistics  
## removing undesirable variables   
SFielding2 <- Fielding %>% subset(select = -c(teamID, lgID,InnOuts, WP, SB, CS, ZR, GS, stint))  
## removing pitchers from fielding  
SFielding2 <- SFielding2 %>% filter(POS != "P")  
## removing position  
SFielding2 <- SFielding2 %>% subset(select = -c(POS))  
## na is equivalent to 0  
SFielding2[is.na(SFielding2)] = 0  
## Combining all player fielding stats into seasons  
SFielding3 <- aggregate(. ~ playerID + yearID, data = SFielding2, sum)  
## Combines the data frames batting and fielding together contain only position player observations   
## creating Fielding Percentage(FP)  
SFielding3 <- SFielding3 %>% mutate(FP = ((PO + A)/(PO + A + E)))

## Tranforming the Pitching Statistics

First, we removed all non numeric variables in the pitching dataframe. Then, we combined all the stats from players who played for multiple teams within a single league year. Then, we created a new variable (ERA) by using the following formula ERA = ((ER *9)/(IPouts/3)). Then, we removed the stints variable within the pitching data frame. Then, we removed all non numeric variables in the fielding data frame. Then, we filtered out albyl non pitchers in the fielding data frame and immediately removed the position variable since it was negladgle. Then, we created a completely new data frame (pitcher\_stats) by joining the pitching and the fielding data frames with the playerID and YearID variables as the forien key. Then, we created an exact copy of the pitcher\_stats dataframe (pitcher\_stats2) and removed the playerID variable from it. Then, we aggregated the whole pitcher\_stats3 data frame by yearId in order to calculate the yearly average stats. Then, we removed all the non important stats only leaving the yearID, ERA, BB, HBP, SO, IPouts. Then we renamed all the variables ERA to lgERA,, HR to lgHR, BB to lgBB, HBP to lgHBP, SO to lgSO and IPouts to lgIP. Then, we divided the lgIP by three since the IPouts statistic is just the IP multiplied by three. Then, we used the following formula FIPconstant = lgERA - (((13*lgHR)+(3*(lgBB+lgHBP))-(2*lgSO))/lgIP) to calculate the FIP constant league wide from the years 1995 to 2015 in the pitcher\_stats3 dataframe. Then, we created a new data frame (pitcher\_stats\_complete) by left joining the pitcher\_stats dataframe and the pitcher\_stats3 data frame. Then, we converted the Ipouts stat into IP that was in the original pitcher\_stats by dividing it by three and then renaming it. Then, we used the following formula FIP = ((13*HR)+(3*(BB+HBP))-(2\*SO))/IP + FIPconstant in order to calculate the FIP for all pitchers that played from the years 1995 to 2015. Finally, we filtered out all the pitchers who barely played (IP >= 20) in order to get rid of some outliers and filtered out all infinities and NAs.

## removing non numeric variables in pitching  
pitching2 <- Pitching %>% subset(select = -c(teamID, lgID, IBB, SH, SF, GIDP, ERA))  
## combining all the stats from the different stints of the pitchers  
pitching2 <- aggregate(. ~ playerID + yearID, data = pitching2, sum)  
##Creating a combined ERA  
pitching2 <- pitching2 %>% mutate(ERA = ((ER \*9)/(IPouts/3)))  
## removing the stint column  
pitching2 <- pitching2 %>% subset(select = -c(stint))  
## removing non numeric variables in fielding  
fielding2 <- Fielding %>% subset(select = -c(teamID, lgID, GS, InnOuts,PB, WP, SB, CS, ZR))  
## filtering out all non pitchers  
fielding2 <- fielding2 %>% filter(POS == "P")  
## removing the position column  
fielding2 <- fielding2 %>% subset(select = -c(POS))  
## combining all the stats from the different stints of the pitchers  
fielding2 <- aggregate(. ~ playerID + yearID, data = fielding2, sum)  
## removing the stint column  
fielding2 <- fielding2 %>% subset(select = -c(stint))  
## complete data frame containing all statistics recorded for pitchers  
pitcher\_stats <- left\_join(fielding2, pitching2, by = c("playerID", "yearID"))  
## removing the playerID column  
pitcher\_stats2 <- pitcher\_stats %>% subset(select = -c(playerID))  
## combining everything by year and averaging all the states league wide  
pitcher\_stats2 <- aggregate(. ~ yearID, data = pitcher\_stats2, mean)  
## removing the all non important columns  
pitcher\_stats3 <- pitcher\_stats2 %>% subset(select = c(yearID, ERA, HR, BB, HBP, SO, IPouts))  
## converting IPouts to IP  
pitcher\_stats3$IPouts <- pitcher\_stats3$IPouts/3  
##renaming all the columns  
pitcher\_stats3 <- pitcher\_stats3 %>% rename(lgERA = ERA, lgHR = HR, lgBB = BB, lgHBP = HBP, lgSO = SO, lgIP = IPouts)  
##calculating the FIP constant  
pitcher\_stats3 <- pitcher\_stats3 %>% mutate(FIPconstant = lgERA - (((13\*lgHR)+(3\*(lgBB+lgHBP))-(2\*lgSO))/lgIP))  
## create new dataframe with all the league avg states and the FIP constant per year  
pitcher\_stats\_complete <- left\_join(pitcher\_stats, pitcher\_stats3, by = c("yearID"))  
## removing G.y and renaming G.x  
pitcher\_stats\_complete <- pitcher\_stats\_complete %>% subset(select = -c(G.y))  
pitcher\_stats\_complete <- pitcher\_stats\_complete %>% rename(G = G.x)  
## converting and renaming IPouts to IP  
pitcher\_stats\_complete$IPouts <- pitcher\_stats\_complete$IPouts/3  
pitcher\_stats\_complete <- pitcher\_stats\_complete %>% rename(IP = IPouts)  
## calculating the FIP for every player   
pitcher\_stats\_complete <- pitcher\_stats\_complete %>% mutate(FIP = ((13\*HR)+(3\*(BB+HBP))-(2\*SO))/IP + FIPconstant)  
##filtering out pitchers with low playing time  
pitcher\_stats\_complete <- pitcher\_stats\_complete %>% filter(IP >= 20)  
##filtering out infinites   
pitcher\_stats\_complete <- pitcher\_stats\_complete %>% filter(is.finite(FIP))

## Creating WAR

To create war, we created a new data frame Player\_WAR by joining batting and fielding together by year and player. Within this new data frame we created a statistic called WAR, which is the sum of OPS/mean(OPS), SBP/mean(SBP), FP/mean(FP). WAR is great stat because it indicates a position player’s performance in all aspects of play (Fielding, Batting, and Base Running).

## Contains both fielding and batting statistics for each player by season  
Player\_WAR <- full\_join(Batting3, SFielding3, by = c("playerID", "yearID"))   
## Removing undesirable variables  
##Player\_WAR <- Player\_WAR %>% subset(select = c(playerID, yearID, OPS, SBP, FP, G.x))  
## Calculating the means  
OPSmean <- mean(Player\_WAR$OPS, na.rm ="TRUE")  
SBPmean <- mean(Player\_WAR$SBP, na.rm = "TRUE")  
FPmean <- mean(Player\_WAR$FP, na.rm = "TRUE")  
## Creating a statistic that determines value of player based on fielding, batting, and base running  
Player\_WAR <- Player\_WAR %>% mutate(WAR = ((OPS/OPSmean)+(SBP/SBPmean)+(FP/FPmean)))

## Batting Post

Again like the batting data frame we needed to combine rows. However, in the postseason this was because a player’s information was separated by series level. Once combined into observations by player and year, we created the OPS for the postseason.

##Comparing Regular VS Postseason OPS  
## removing undesirable variable  
BattingPost2 <- BattingPost %>% subset(select = -c(teamID, lgID, SF, GIDP, round))  
## na is equivalent to 0  
BattingPost2[is.na(BattingPost2)] = 0  
## Combining all player batting stats into seasons  
BattingPost3 <- aggregate(. ~ playerID + yearID, data = BattingPost2, sum)  
## Creating on base average (OBA)  
BattingPost3 <- BattingPost3 %>% mutate(OBA = ((H + BB + HBP)/(AB+ BB + HBP)))  
## Creating extra base hits (XBH)  
BattingPost3 <- BattingPost3 %>% mutate(XBH = (`2B` + `3B` + HR))  
## Creating Slugging (SLG)  
BattingPost3 <- BattingPost3 %>% mutate(SLG = (((H-XBH)+(2\*`2B`)+(3\*`3B`)+(4\*HR))/(AB)))  
## Creating On Base Plus Slugging (OPS)  
BattingPost3 <- BattingPost3 %>% mutate(OPS = (OBA + SLG))  
## Creating on singles stat  
BattingPost3 <- BattingPost3 %>% mutate(`1B` = (H-XBH))  
## Creating on singles stat  
BattingPost3 <- BattingPost3 %>% mutate(BO = (AB - H))

## Functions

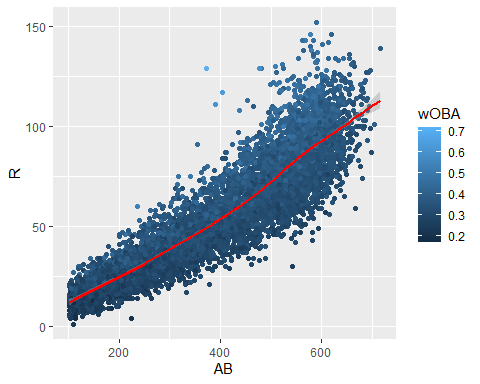
## Question 1 Predicting Runs (R)

In this question, we wanted to see what variables are good predictors for the number of runs a player may score over a season and if we can predict them.

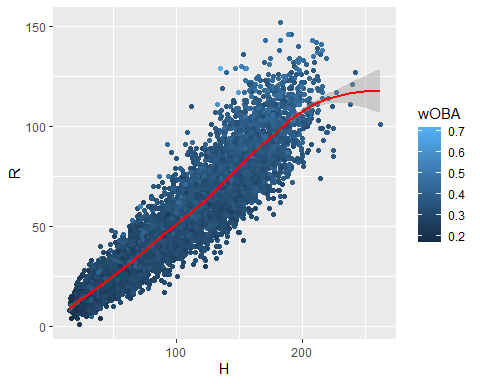
##WAR vs. Salary  
WAR\_Salary <- left\_join(Player\_WAR, Salaries,by = c("playerID", "yearID"))  
WAR\_Salary <- WAR\_Salary %>% rename(G = G.x)  
WAR\_Salary2 <- na.omit(WAR\_Salary)  
WAR\_Salary2 <- WAR\_Salary2 %>% subset(select = -c(playerID, yearID))  
WAR\_Salary2 <- WAR\_Salary2 %>% filter(AB > 100)  
WAR\_Salary2 <- WAR\_Salary2 %>% filter(G > 20)

## Visualization

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

 As you can see, there is a positive correlation with runs increasing as either at bats or hits increases. Indicating that runs are likely linked to the number of at bats and how often a player gets on base. ## partitioning the data

## training data

## MLR

modelMLRR <- lm(R ~ `1B` + `2B` + `3B` + HR + BB + HBP + SB + CS ,data = R.train)  
vif(modelMLRR)

## `1B` `2B` `3B` HR BB HBP SB CS   
## 3.203403 3.449698 1.508730 2.341152 2.320964 1.221213 2.603885 2.433571

summary(modelMLRR)

##   
## Call:  
## lm(formula = R ~ `1B` + `2B` + `3B` + HR + BB + HBP + SB + CS,   
## data = R.train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -26.7880 -4.4046 -0.1907 4.2071 31.3980   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.909143 0.228761 -8.346 <2e-16 \*\*\*  
## `1B` 0.291833 0.005071 57.545 <2e-16 \*\*\*  
## `2B` 0.421530 0.015625 26.978 <2e-16 \*\*\*  
## `3B` 1.145033 0.047998 23.856 <2e-16 \*\*\*  
## HR 0.891583 0.013621 65.456 <2e-16 \*\*\*  
## BB 0.246505 0.005862 42.048 <2e-16 \*\*\*  
## HBP 0.333131 0.026596 12.526 <2e-16 \*\*\*  
## SB 0.380402 0.015142 25.123 <2e-16 \*\*\*  
## CS 0.078158 0.045694 1.710 0.0872 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 7.036 on 5630 degrees of freedom  
## Multiple R-squared: 0.9411, Adjusted R-squared: 0.941   
## F-statistic: 1.124e+04 on 8 and 5630 DF, p-value: < 2.2e-16

predictedR <- predict(modelMLRR, R.train)  
RMSE(predictedR, R.train$R)

## [1] 7.030765

R2(predictedR, R.train$R)

## [1] 0.9410622

## SVR

modelSVRR <- svm(R ~ `1B` + `2B` + `3B` + HR + BB + HBP + SB + CS , data = R.train)  
predictedR2 <- predict(modelSVRR, R.train)  
RMSE(predictedR2, R.train$R)

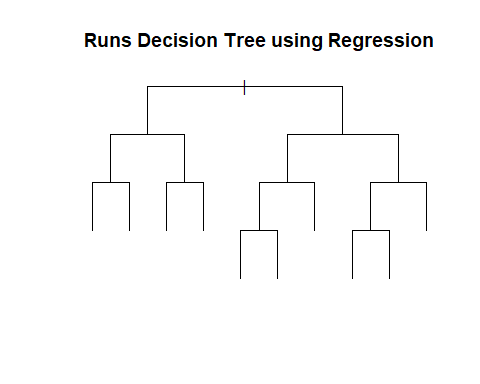
## [1] 6.270867

R2(predictedR2, R.train$R)

## [1] 0.9531665

## DT

fitR <- rpart( R ~ `1B` + `2B` + `3B` + HR + BB + HBP + SB + CS, data = R.train, method = "anova")  
plot(fitR, uniform = TRUE,  
 main = "Runs Decision Tree using Regression")



summary(fitR)

## Call:  
## rpart(formula = R ~ `1B` + `2B` + `3B` + HR + BB + HBP + SB +   
## CS, data = R.train, method = "anova")  
## n= 5639   
##   
## CP nsplit rel error xerror xstd  
## 1 0.55523623 0 1.0000000 1.0003206 0.015249331  
## 2 0.08963254 1 0.4447638 0.4491849 0.009632938  
## 3 0.06196292 2 0.3551312 0.3593137 0.008322376  
## 4 0.03379117 3 0.2931683 0.2960216 0.006612779  
## 5 0.02054200 4 0.2593771 0.2644553 0.006014543  
## 6 0.01848988 5 0.2388351 0.2440393 0.005976794  
## 7 0.01841610 6 0.2203453 0.2305853 0.005701301  
## 8 0.01332584 7 0.2019292 0.2104547 0.005347158  
## 9 0.01158237 8 0.1886033 0.1998347 0.005078591  
## 10 0.01000000 9 0.1770210 0.1892112 0.004403815  
##   
## Variable importance  
## 1B 2B BB HR CS SB   
## 29 19 19 16 8 8   
##   
## Node number 1: 5639 observations, complexity param=0.5552362  
## mean=55.07342, MSE=838.7089   
## left son=2 (2627 obs) right son=3 (3012 obs)  
## Primary splits:  
## 1B < 65.5 to the left, improve=0.5552362, (0 missing)  
## 2B < 20.5 to the left, improve=0.5425680, (0 missing)  
## BB < 39.5 to the left, improve=0.4706399, (0 missing)  
## HR < 13.5 to the left, improve=0.3944487, (0 missing)  
## SB < 6.5 to the left, improve=0.1868621, (0 missing)  
## Surrogate splits:  
## 2B < 19.5 to the left, agree=0.844, adj=0.665, (0 split)  
## BB < 28.5 to the left, agree=0.777, adj=0.522, (0 split)  
## HR < 9.5 to the left, agree=0.715, adj=0.389, (0 split)  
## CS < 2.5 to the left, agree=0.673, adj=0.298, (0 split)  
## SB < 3.5 to the left, agree=0.668, adj=0.287, (0 split)  
##   
## Node number 2: 2627 observations, complexity param=0.06196292  
## mean=31.9665, MSE=258.812   
## left son=4 (2020 obs) right son=5 (607 obs)  
## Primary splits:  
## HR < 10.5 to the left, improve=0.4310233, (0 missing)  
## 2B < 15.5 to the left, improve=0.4026995, (0 missing)  
## 1B < 39.5 to the left, improve=0.3911620, (0 missing)  
## BB < 31.5 to the left, improve=0.3781705, (0 missing)  
## HBP < 3.5 to the left, improve=0.1376118, (0 missing)  
## Surrogate splits:  
## BB < 40.5 to the left, agree=0.837, adj=0.293, (0 split)  
## 2B < 18.5 to the left, agree=0.820, adj=0.219, (0 split)  
## HBP < 6.5 to the left, agree=0.783, adj=0.059, (0 split)  
## 1B < 60.5 to the left, agree=0.775, adj=0.026, (0 split)  
##   
## Node number 3: 3012 observations, complexity param=0.08963254  
## mean=75.22676, MSE=472.6435   
## left son=6 (1747 obs) right son=7 (1265 obs)  
## Primary splits:  
## BB < 51.5 to the left, improve=0.29777650, (0 missing)  
## HR < 18.5 to the left, improve=0.27187290, (0 missing)  
## 2B < 27.5 to the left, improve=0.25651580, (0 missing)  
## 1B < 89.5 to the left, improve=0.19989790, (0 missing)  
## SB < 14.5 to the left, improve=0.08722196, (0 missing)  
## Surrogate splits:  
## HR < 20.5 to the left, agree=0.700, adj=0.285, (0 split)  
## 2B < 33.5 to the left, agree=0.655, adj=0.179, (0 split)  
## SB < 35.5 to the left, agree=0.586, adj=0.013, (0 split)  
## CS < 9.5 to the left, agree=0.584, adj=0.009, (0 split)  
## HBP < 13.5 to the left, agree=0.583, adj=0.008, (0 split)  
##   
## Node number 4: 2020 observations, complexity param=0.020542  
## mean=26.17673, MSE=118.6188   
## left son=8 (1190 obs) right son=9 (830 obs)  
## Primary splits:  
## 1B < 39.5 to the left, improve=0.4054631, (0 missing)  
## 2B < 10.5 to the left, improve=0.3313871, (0 missing)  
## BB < 16.5 to the left, improve=0.2919918, (0 missing)  
## SB < 4.5 to the left, improve=0.1787154, (0 missing)  
## HR < 5.5 to the left, improve=0.1639227, (0 missing)  
## Surrogate splits:  
## 2B < 11.5 to the left, agree=0.745, adj=0.380, (0 split)  
## BB < 21.5 to the left, agree=0.696, adj=0.260, (0 split)  
## SB < 7.5 to the left, agree=0.657, adj=0.166, (0 split)  
## CS < 2.5 to the left, agree=0.657, adj=0.165, (0 split)  
## 3B < 2.5 to the left, agree=0.649, adj=0.146, (0 split)  
##   
## Node number 5: 607 observations, complexity param=0.01158237  
## mean=51.23394, MSE=242.5647   
## left son=10 (555 obs) right son=11 (52 obs)  
## Primary splits:  
## HR < 27.5 to the left, improve=0.3720441, (0 missing)  
## BB < 68.5 to the left, improve=0.3496750, (0 missing)  
## 2B < 18.5 to the left, improve=0.2449622, (0 missing)  
## 1B < 48.5 to the left, improve=0.1824067, (0 missing)  
## SB < 3.5 to the left, improve=0.1107503, (0 missing)  
## Surrogate splits:  
## BB < 79.5 to the left, agree=0.947, adj=0.385, (0 split)  
##   
## Node number 6: 1747 observations, complexity param=0.03379117  
## mean=65.13165, MSE=319.503   
## left son=12 (1273 obs) right son=13 (474 obs)  
## Primary splits:  
## 1B < 104.5 to the left, improve=0.2863180, (0 missing)  
## 2B < 27.5 to the left, improve=0.2578114, (0 missing)  
## HR < 16.5 to the left, improve=0.1959969, (0 missing)  
## BB < 34.5 to the left, improve=0.1250563, (0 missing)  
## SB < 11.5 to the left, improve=0.1138351, (0 missing)  
## Surrogate splits:  
## SB < 26.5 to the left, agree=0.755, adj=0.097, (0 split)  
## 2B < 36.5 to the left, agree=0.751, adj=0.082, (0 split)  
## CS < 12.5 to the left, agree=0.745, adj=0.059, (0 split)  
## 3B < 8.5 to the left, agree=0.742, adj=0.051, (0 split)  
## HBP < 19.5 to the left, agree=0.732, adj=0.013, (0 split)  
##   
## Node number 7: 1265 observations, complexity param=0.0184161  
## mean=89.16838, MSE=349.0238   
## left son=14 (904 obs) right son=15 (361 obs)  
## Primary splits:  
## HR < 28.5 to the left, improve=0.1972720, (0 missing)  
## BB < 71.5 to the left, improve=0.1562635, (0 missing)  
## 2B < 30.5 to the left, improve=0.1301045, (0 missing)  
## 1B < 84.5 to the left, improve=0.1160720, (0 missing)  
## SB < 13.5 to the left, improve=0.0844483, (0 missing)  
## Surrogate splits:  
## BB < 91.5 to the left, agree=0.750, adj=0.125, (0 split)  
## 1B < 66.5 to the right, agree=0.715, adj=0.003, (0 split)  
##   
## Node number 8: 1190 observations  
## mean=20.38487, MSE=57.89893   
##   
## Node number 9: 830 observations  
## mean=34.48072, MSE=88.62312   
##   
## Node number 10: 555 observations  
## mean=48.32613, MSE=137.3657   
##   
## Node number 11: 52 observations  
## mean=82.26923, MSE=311.9275   
##   
## Node number 12: 1273 observations, complexity param=0.01848988  
## mean=59.29537, MSE=218.2615   
## left son=24 (913 obs) right son=25 (360 obs)  
## Primary splits:  
## HR < 16.5 to the left, improve=0.31473260, (0 missing)  
## 2B < 26.5 to the left, improve=0.22919980, (0 missing)  
## BB < 34.5 to the left, improve=0.15660330, (0 missing)  
## 1B < 78.5 to the left, improve=0.15173270, (0 missing)  
## SB < 4.5 to the left, improve=0.07066474, (0 missing)  
## Surrogate splits:  
## 2B < 33.5 to the left, agree=0.756, adj=0.136, (0 split)  
## HBP < 16.5 to the left, agree=0.718, adj=0.003, (0 split)  
##   
## Node number 13: 474 observations  
## mean=80.80591, MSE=254.2408   
##   
## Node number 14: 904 observations, complexity param=0.01332584  
## mean=83.92478, MSE=297.0519   
## left son=28 (525 obs) right son=29 (379 obs)  
## Primary splits:  
## 1B < 104.5 to the left, improve=0.23469680, (0 missing)  
## SB < 16.5 to the left, improve=0.15225300, (0 missing)  
## 2B < 33.5 to the left, improve=0.14238130, (0 missing)  
## 3B < 4.5 to the left, improve=0.10071950, (0 missing)  
## BB < 71.5 to the left, improve=0.09202184, (0 missing)  
## Surrogate splits:  
## SB < 24.5 to the left, agree=0.656, adj=0.179, (0 split)  
## 2B < 36.5 to the left, agree=0.643, adj=0.148, (0 split)  
## CS < 5.5 to the left, agree=0.632, adj=0.121, (0 split)  
## 3B < 6.5 to the left, agree=0.605, adj=0.058, (0 split)  
## HR < 6.5 to the right, agree=0.603, adj=0.053, (0 split)  
##   
## Node number 15: 361 observations  
## mean=102.2992, MSE=237.8994   
##   
## Node number 24: 913 observations  
## mean=54.09091, MSE=143.6292   
##   
## Node number 25: 360 observations  
## mean=72.49444, MSE=164.6277   
##   
## Node number 28: 525 observations  
## mean=76.83048, MSE=228.1751   
##   
## Node number 29: 379 observations  
## mean=93.75198, MSE=226.1707

predictedR3 <- predict(fitR, R.train, method = "anova")  
RMSE(predictedR3, R.train$R)

## [1] 12.18479

R2(predictedR3, R.train$R)

## [1] 0.822979

## testing data

## MLR

predictedR4 <- predict(modelMLRR, R.test)  
RMSE(R.test$R,predictedR4)

## [1] 7.207785

R2(R.test$R,predictedR4)

## [1] 0.9364848

## SVR

predictedR5 <- predict(modelSVRR, R.test)  
RMSE(predictedR5, R.test$R)

## [1] 7.304765

R2(predictedR5, R.test$R)

## [1] 0.9349128

## DT

predictedR6 <- predict(fitR, R.test, method = "anova")  
RMSE(predictedR6, R.test$R)

## [1] 12.74174

R2(predictedR6, R.test$R)

## [1] 0.8019361

##Conclusion

The best model for explaining the data was the SVR. With the test data the SVR model had an R-Squared of 0.9522258 and a RMSE of 6.311195. Indicating the model can explain about 95.3 % of the batting statistics. However, for predicting the multiple linear regression had the highest R-Squared of 0.9420112 and the lowest RMSE of 7.004749 indicating that this is the best model for predicting. The variables were chosen because they had low collinearity and great p-values. The variable with the greatest influence on the model was triples (3B), with a slope of 1.098654 and with home runs (HR) falling shortly behind.

##Question 2 Position Position Player performance

## WAR vs Salary

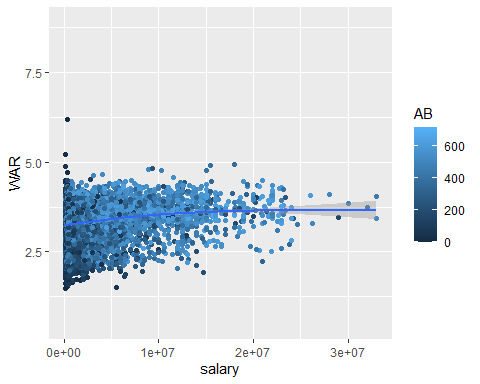
In this question, we wanted to determine whether how much a player is paid can predict his future performance. ## visualization

ggplot(data = WAR\_Salary, mapping = aes(x = salary, y = WAR, color = AB)) + geom\_point() + geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 18497 rows containing non-finite values (stat\_smooth).

## Warning: Removed 18497 rows containing missing values (geom\_point).

 The Player War converges towards a best fit line which means that there is less variability in the performance of a position player the higher the salary. ## modeling # #MLR

model1b <- lm(WAR ~ SB + CS + SO + IBB + HBP + SH + PB + FP + H, data = train.data)  
summary(model1b)

##   
## Call:  
## lm(formula = WAR ~ SB + CS + SO + IBB + HBP + SH + PB + FP +   
## H, data = train.data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.44467 -0.16150 0.04934 0.23777 1.29944   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.217e+00 3.433e-01 6.457 1.16e-10 \*\*\*  
## SB 3.549e-02 8.006e-04 44.333 < 2e-16 \*\*\*  
## CS -1.006e-01 2.535e-03 -39.700 < 2e-16 \*\*\*  
## SO -7.882e-05 1.944e-04 -0.406 0.68512   
## IBB 1.095e-02 1.403e-03 7.808 6.88e-15 \*\*\*  
## HBP 7.608e-03 1.505e-03 5.055 4.44e-07 \*\*\*  
## SH -1.171e-02 1.901e-03 -6.161 7.73e-10 \*\*\*  
## PB -2.016e-02 2.591e-03 -7.784 8.31e-15 \*\*\*  
## FP 9.158e-01 3.509e-01 2.610 0.00909 \*\*   
## H 2.454e-03 1.637e-04 14.985 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3932 on 5628 degrees of freedom  
## Multiple R-squared: 0.3818, Adjusted R-squared: 0.3808   
## F-statistic: 386.2 on 9 and 5628 DF, p-value: < 2.2e-16

##train Data  
predictedR <- predict(model1b, train.data)  
RMSE(predictedR, train.data$WAR)

## [1] 0.3928754

R2(predictedR, train.data$WAR)

## [1] 0.3817942

##test Data  
predictedR <- predict(model1b, test.data)  
RMSE(predictedR, test.data$WAR)

## [1] 0.3942034

R2(predictedR, test.data$WAR)

## [1] 0.3941182

##SVR

model\_1\_SVR <- svm(WAR ~ SB + CS + SO + IBB + HBP + SH + PB + FP + H, data = train.data)  
  
##Train Data  
predicted.classes <- predict(model\_1\_SVR, train.data)  
RMSE(predicted.classes, train.data$WAR)

## [1] 0.2668796

R2(predicted.classes, train.data$WAR)

## [1] 0.7350932

##Test Data  
predicted.classes <- predict(model\_1\_SVR, test.data)  
  
RMSE(predicted.classes, test.data$WAR)

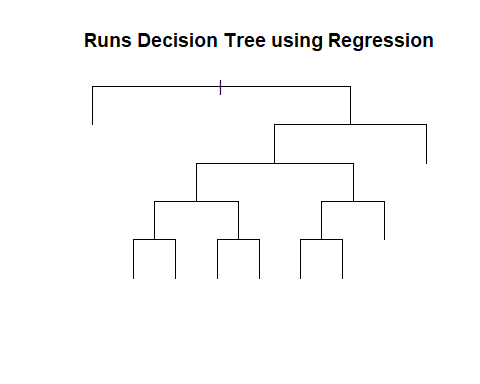
## [1] 0.2928498

R2(predicted.classes, test.data$WAR)

## [1] 0.6867841

##DT

DT\_1\_WAR <- rpart(WAR ~ SB + CS + SO + IBB + HBP + SH + PB + FP + H, data = train.data, method = "anova")  
  
plot(DT\_1\_WAR, uniform = TRUE,  
 main = "Runs Decision Tree using Regression")



summary(DT\_1\_WAR)

## Call:  
## rpart(formula = WAR ~ SB + CS + SO + IBB + HBP + SH + PB + FP +   
## H, data = train.data, method = "anova")  
## n= 5638   
##   
## CP nsplit rel error xerror xstd  
## 1 0.46979762 0 1.0000000 1.0002739 0.019792095  
## 2 0.12846290 1 0.5302024 0.5304832 0.010014578  
## 3 0.10615589 2 0.4017395 0.4020191 0.008096163  
## 4 0.02176682 3 0.2955836 0.2959805 0.005935761  
## 5 0.02076831 4 0.2738168 0.2805710 0.005687721  
## 6 0.01281181 5 0.2530485 0.2549118 0.005170963  
## 7 0.01216830 6 0.2402366 0.2461873 0.005056002  
## 8 0.01038848 7 0.2280683 0.2324354 0.004722182  
## 9 0.01000000 8 0.2176799 0.2247499 0.004633467  
##   
## Variable importance  
## SB CS H SO PB   
## 71 22 4 1 1   
##   
## Node number 1: 5638 observations, complexity param=0.4697976  
## mean=3.349464, MSE=0.2496759   
## left son=2 (559 obs) right son=3 (5079 obs)  
## Primary splits:  
## SB < 0.5 to the left, improve=0.46979760, (0 missing)  
## CS < 0.5 to the right, improve=0.19827100, (0 missing)  
## H < 126.5 to the left, improve=0.08525808, (0 missing)  
## SO < 63.5 to the left, improve=0.04098043, (0 missing)  
## IBB < 1.5 to the left, improve=0.03464606, (0 missing)  
## Surrogate splits:  
## SO < 8.5 to the left, agree=0.901, adj=0.004, (0 split)  
##   
## Node number 2: 559 observations  
## mean=2.317114, MSE=0.04461739   
##   
## Node number 3: 5079 observations, complexity param=0.1284629  
## mean=3.463085, MSE=0.1420379   
## left son=6 (4316 obs) right son=7 (763 obs)  
## Primary splits:  
## CS < 0.5 to the right, improve=0.25066690, (0 missing)  
## H < 140.5 to the left, improve=0.08026945, (0 missing)  
## SB < 4.5 to the left, improve=0.06348762, (0 missing)  
## IBB < 7.5 to the left, improve=0.05039475, (0 missing)  
## SH < 0.5 to the right, improve=0.04947284, (0 missing)  
## Surrogate splits:  
## H < 18.5 to the right, agree=0.85, adj=0.003, (0 split)  
##   
## Node number 6: 4316 observations, complexity param=0.1061559  
## mean=3.383749, MSE=0.1163584   
## left son=12 (1425 obs) right son=13 (2891 obs)  
## Primary splits:  
## SB < 3.5 to the left, improve=0.29755460, (0 missing)  
## H < 141.5 to the left, improve=0.16612480, (0 missing)  
## SO < 63.5 to the left, improve=0.07642422, (0 missing)  
## IBB < 7.5 to the left, improve=0.06497702, (0 missing)  
## HBP < 4.5 to the left, improve=0.04735717, (0 missing)  
## Surrogate splits:  
## CS < 1.5 to the left, agree=0.728, adj=0.177, (0 split)  
## H < 51.5 to the left, agree=0.701, adj=0.093, (0 split)  
## PB < 0.5 to the right, agree=0.694, adj=0.074, (0 split)  
## SO < 33.5 to the left, agree=0.676, adj=0.018, (0 split)  
## FP < 0.9294946 to the left, agree=0.672, adj=0.008, (0 split)  
##   
## Node number 7: 763 observations  
## mean=3.911861, MSE=0.05029334   
##   
## Node number 12: 1425 observations, complexity param=0.02176682  
## mean=3.118717, MSE=0.09388415   
## left son=24 (819 obs) right son=25 (606 obs)  
## Primary splits:  
## CS < 1.5 to the right, improve=0.22902850, (0 missing)  
## SB < 1.5 to the left, improve=0.18127190, (0 missing)  
## H < 141.5 to the left, improve=0.09753069, (0 missing)  
## IBB < 6.5 to the left, improve=0.07396909, (0 missing)  
## SH < 0.5 to the right, improve=0.06489453, (0 missing)  
## Surrogate splits:  
## H < 45.5 to the right, agree=0.606, adj=0.074, (0 split)  
## SO < 30.5 to the right, agree=0.598, adj=0.054, (0 split)  
## FP < 0.9966132 to the left, agree=0.592, adj=0.041, (0 split)  
## SB < 1.5 to the right, agree=0.581, adj=0.015, (0 split)  
## HBP < 0.5 to the right, agree=0.580, adj=0.013, (0 split)  
##   
## Node number 13: 2891 observations, complexity param=0.02076831  
## mean=3.514386, MSE=0.07574719   
## left son=26 (1706 obs) right son=27 (1185 obs)  
## Primary splits:  
## H < 140.5 to the left, improve=0.13350220, (0 missing)  
## SH < 0.5 to the right, improve=0.10720230, (0 missing)  
## IBB < 5.5 to the left, improve=0.10000730, (0 missing)  
## SB < 12.5 to the left, improve=0.09011670, (0 missing)  
## CS < 1.5 to the right, improve=0.07615778, (0 missing)  
## Surrogate splits:  
## SO < 79.5 to the left, agree=0.668, adj=0.190, (0 split)  
## SB < 19.5 to the left, agree=0.667, adj=0.187, (0 split)  
## IBB < 3.5 to the left, agree=0.664, adj=0.181, (0 split)  
## CS < 6.5 to the left, agree=0.644, adj=0.132, (0 split)  
## HBP < 6.5 to the left, agree=0.633, adj=0.105, (0 split)  
##   
## Node number 24: 819 observations, complexity param=0.01281181  
## mean=2.992582, MSE=0.07412005   
## left son=48 (272 obs) right son=49 (547 obs)  
## Primary splits:  
## SB < 1.5 to the left, improve=0.29709320, (0 missing)  
## H < 142.5 to the left, improve=0.14007430, (0 missing)  
## IBB < 6.5 to the left, improve=0.11851170, (0 missing)  
## CS < 2.5 to the right, improve=0.09503738, (0 missing)  
## SH < 0.5 to the right, improve=0.07814210, (0 missing)  
## Surrogate splits:  
## SO < 21.5 to the left, agree=0.672, adj=0.011, (0 split)  
## H < 24 to the left, agree=0.672, adj=0.011, (0 split)  
## CS < 8.5 to the right, agree=0.669, adj=0.004, (0 split)  
## SH < 13 to the right, agree=0.669, adj=0.004, (0 split)  
## FP < 0.9251168 to the left, agree=0.669, adj=0.004, (0 split)  
##   
## Node number 25: 606 observations, complexity param=0.01038848  
## mean=3.289186, MSE=0.07003305   
## left son=50 (281 obs) right son=51 (325 obs)  
## Primary splits:  
## SB < 1.5 to the left, improve=0.34457050, (0 missing)  
## H < 93.5 to the left, improve=0.21484740, (0 missing)  
## SO < 55.5 to the left, improve=0.10384680, (0 missing)  
## IBB < 2.5 to the left, improve=0.10167570, (0 missing)  
## HBP < 4.5 to the left, improve=0.08409666, (0 missing)  
## Surrogate splits:  
## SO < 35.5 to the left, agree=0.564, adj=0.060, (0 split)  
## H < 32.5 to the left, agree=0.561, adj=0.053, (0 split)  
## PB < 1.5 to the right, agree=0.554, adj=0.039, (0 split)  
## FP < 0.9943483 to the right, agree=0.553, adj=0.036, (0 split)  
## IBB < 15.5 to the right, agree=0.541, adj=0.011, (0 split)  
##   
## Node number 26: 1706 observations, complexity param=0.0121683  
## mean=3.430576, MSE=0.06538546   
## left son=52 (1126 obs) right son=53 (580 obs)  
## Primary splits:  
## CS < 2.5 to the right, improve=0.15355760, (0 missing)  
## SH < 0.5 to the right, improve=0.07579684, (0 missing)  
## SB < 12.5 to the left, improve=0.04997570, (0 missing)  
## H < 92.5 to the left, improve=0.03150348, (0 missing)  
## IBB < 0.5 to the left, improve=0.02665777, (0 missing)  
## Surrogate splits:  
## H < 53.5 to the right, agree=0.693, adj=0.098, (0 split)  
## SB < 5.5 to the right, agree=0.693, adj=0.097, (0 split)  
## SO < 22.5 to the right, agree=0.671, adj=0.033, (0 split)  
## PB < 0.5 to the left, agree=0.661, adj=0.003, (0 split)  
##   
## Node number 27: 1185 observations  
## mean=3.635044, MSE=0.06599371   
##   
## Node number 48: 272 observations  
## mean=2.782144, MSE=0.04293016   
##   
## Node number 49: 547 observations  
## mean=3.097224, MSE=0.056659   
##   
## Node number 50: 281 observations  
## mean=3.122124, MSE=0.04895661   
##   
## Node number 51: 325 observations  
## mean=3.433631, MSE=0.04326042   
##   
## Node number 52: 1126 observations  
## mean=3.35866, MSE=0.05784717   
##   
## Node number 53: 580 observations  
## mean=3.570191, MSE=0.05048739

##train  
predictedR3 <- predict(DT\_1\_WAR, train.data, method = "anova")  
RMSE(predictedR3, train.data$WAR)

## [1] 0.2331296

R2(predictedR3, train.data$WAR)

## [1] 0.7823201

##test  
predictedR3 <- predict(DT\_1\_WAR, test.data, method = "anova")  
RMSE(predictedR3, test.data$WAR)

## [1] 0.2377595

R2(predictedR3, test.data$WAR)

## [1] 0.7789366

The Best Model for both explaining and predicting the data is the Decision Tree. While predicting, the model has a value of 0.778 for the R^2 and an RMSE value of 0.247. While explaining, the model has a value of 0.782 for the R^2 and an RMSE value of 0.233.

## WAR vs AGE

In this question, we wanted to determine whether the age of a position player can determine his future performance.

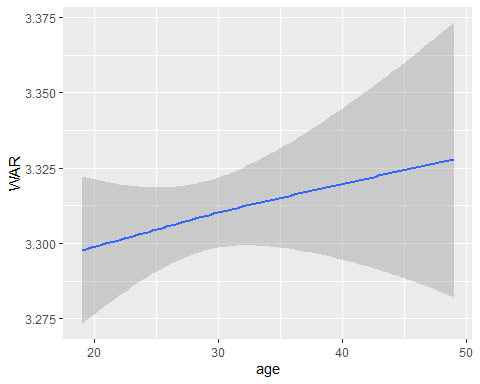
## removing all unnecessary columns in master  
master1 <- Master %>% subset(select = c("playerID", "birthYear", "birthCountry", "birthState", "birthCity", "nameFirst", "nameLast", "nameGiven"))  
  
## adding the contents of master to the pitching stats and calculating the age for every player for every year  
pitcher\_stats\_complete\_AGE <- left\_join(pitcher\_stats\_complete, master1, by = c("playerID"))  
pitcher\_stats\_complete\_AGE <- pitcher\_stats\_complete\_AGE %>% mutate(age = yearID-birthYear)  
  
## adding the contents of master to the player war and calculating the age for every player for every year  
Player\_WAR\_AGE <- left\_join(Player\_WAR, master1, by = c("playerID"))  
Player\_WAR\_AGE <- Player\_WAR\_AGE %>% mutate(age = yearID-birthYear)

##visualization

ggplot(data = Player\_WAR\_AGE) + geom\_smooth(mapping = aes(x = age, y = WAR))

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 16736 rows containing non-finite values (stat\_smooth).



##MLR

Player\_WAR\_AGE <- na.omit(Player\_WAR\_AGE)  
  
set.seed(123)  
training.samples <- Player\_WAR\_AGE$WAR %>% createDataPartition(p = 0.8, list = FALSE)  
train.data <- Player\_WAR\_AGE[training.samples, ]  
test.data <- Player\_WAR\_AGE[-training.samples, ]

MLR\_5\_WAR <- lm(WAR ~ H + SB + CS + SH + SLG + PB + age , data = train.data)  
summary(MLR\_5\_WAR)

##   
## Call:  
## lm(formula = WAR ~ H + SB + CS + SH + SLG + PB + age, data = train.data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.01732 -0.15876 0.07648 0.24769 1.18553   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.3510341 0.0417814 56.270 < 2e-16 \*\*\*  
## H 0.0005305 0.0001253 4.234 2.33e-05 \*\*\*  
## SB 0.0429872 0.0008412 51.103 < 2e-16 \*\*\*  
## CS -0.1195367 0.0026267 -45.509 < 2e-16 \*\*\*  
## SH 0.0067338 0.0019651 3.427 0.000614 \*\*\*  
## SLG 2.4490757 0.0606271 40.396 < 2e-16 \*\*\*  
## PB -0.0193730 0.0026680 -7.261 4.25e-13 \*\*\*  
## age -0.0010034 0.0012220 -0.821 0.411601   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4142 on 7004 degrees of freedom  
## Multiple R-squared: 0.4536, Adjusted R-squared: 0.4531   
## F-statistic: 830.8 on 7 and 7004 DF, p-value: < 2.2e-16

##train Data  
predictedR <- predict(model1b, train.data)  
RMSE(predictedR, train.data$WAR)

## [1] 0.4577968

R2(predictedR, train.data$WAR)

## [1] 0.3353106

##test Data  
predictedR <- predict(model1b, test.data)  
  
RMSE(predictedR, test.data$WAR)

## [1] 0.4564401

R2(predictedR, test.data$WAR)

## [1] 0.3298506

##SVR

SVR\_5\_WAR <- svm(WAR ~ H + SB + CS + SH + SLG + PB + age , data = train.data)  
  
##Train Data  
predicted.classes <- predict(SVR\_5\_WAR, train.data)  
RMSE(predicted.classes, train.data$WAR)

## [1] 0.2497527

R2(predicted.classes, train.data$WAR)

## [1] 0.8232196

##Test Data  
predicted.classes <- predict(SVR\_5\_WAR, test.data)  
  
RMSE(predicted.classes, test.data$WAR)

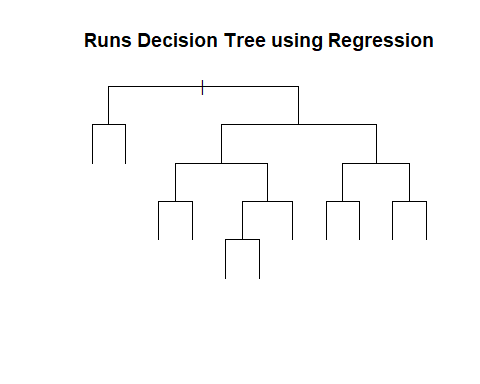
## [1] 0.2568473

R2(predicted.classes, test.data$WAR)

## [1] 0.8045462

##DT

DT\_5\_WAR <- rpart(WAR ~ H + SB + CS + SH + SLG + PB + age, data = train.data, method = "anova")  
  
plot(DT\_5\_WAR, uniform = TRUE,  
 main = "Runs Decision Tree using Regression")



summary(DT\_5\_WAR)

## Call:  
## rpart(formula = WAR ~ H + SB + CS + SH + SLG + PB + age, data = train.data,   
## method = "anova")  
## n= 7012   
##   
## CP nsplit rel error xerror xstd  
## 1 0.50405440 0 1.0000000 1.0002775 0.023483810  
## 2 0.10785848 1 0.4959456 0.4961170 0.016563408  
## 3 0.08304040 2 0.3880871 0.3883957 0.014082742  
## 4 0.03800830 3 0.3050467 0.3073798 0.013559855  
## 5 0.03623698 4 0.2670384 0.2742769 0.013357748  
## 6 0.01627525 5 0.2308015 0.2346571 0.011959105  
## 7 0.01574037 6 0.2145262 0.2186522 0.011832138  
## 8 0.01212961 7 0.1987858 0.2014237 0.011666898  
## 9 0.01029746 8 0.1866562 0.1915604 0.011440391  
## 10 0.01011420 9 0.1763588 0.1827176 0.009537887  
## 11 0.01000000 10 0.1662446 0.1797842 0.009516720  
##   
## Variable importance  
## SB SLG CS H SH   
## 64 14 12 7 2   
##   
## Node number 1: 7012 observations, complexity param=0.5040544  
## mean=3.310151, MSE=0.3135916   
## left son=2 (863 obs) right son=3 (6149 obs)  
## Primary splits:  
## SB < 0.5 to the left, improve=0.50405440, (0 missing)  
## CS < 0.5 to the right, improve=0.20611690, (0 missing)  
## SLG < 0.4231153 to the left, improve=0.10799560, (0 missing)  
## H < 126.5 to the left, improve=0.07093453, (0 missing)  
## PB < 0.5 to the right, improve=0.01971345, (0 missing)  
##   
## Node number 2: 863 observations, complexity param=0.01574037  
## mean=2.248901, MSE=0.0786421   
## left son=4 (321 obs) right son=5 (542 obs)  
## Primary splits:  
## SLG < 0.3591784 to the left, improve=0.509982700, (0 missing)  
## H < 52.5 to the left, improve=0.164361700, (0 missing)  
## age < 25.5 to the left, improve=0.020661670, (0 missing)  
## CS < 1.5 to the left, improve=0.009267948, (0 missing)  
## SH < 1.5 to the right, improve=0.006922585, (0 missing)  
## Surrogate splits:  
## H < 22.5 to the left, agree=0.722, adj=0.252, (0 split)  
## age < 22.5 to the left, agree=0.634, adj=0.016, (0 split)  
## SH < 5.5 to the right, agree=0.632, adj=0.009, (0 split)  
##   
## Node number 3: 6149 observations, complexity param=0.1078585  
## mean=3.459095, MSE=0.1663146   
## left son=6 (4776 obs) right son=7 (1373 obs)  
## Primary splits:  
## CS < 0.5 to the right, improve=0.23191330, (0 missing)  
## SLG < 0.470359 to the left, improve=0.17089410, (0 missing)  
## H < 140.5 to the left, improve=0.05279105, (0 missing)  
## SH < 0.5 to the right, improve=0.04367049, (0 missing)  
## SB < 5.5 to the left, improve=0.03650914, (0 missing)  
## Surrogate splits:  
## H < 16.5 to the right, agree=0.810, adj=0.151, (0 split)  
## SLG < 0.2254212 to the right, agree=0.787, adj=0.047, (0 split)  
## SB < 1.5 to the right, agree=0.783, adj=0.028, (0 split)  
##   
## Node number 4: 321 observations  
## mean=1.988674, MSE=0.05299022   
##   
## Node number 5: 542 observations  
## mean=2.40302, MSE=0.02997545   
##   
## Node number 6: 4776 observations, complexity param=0.0830404  
## mean=3.353794, MSE=0.125877   
## left son=12 (1288 obs) right son=13 (3488 obs)  
## Primary splits:  
## SB < 2.5 to the left, improve=0.30372810, (0 missing)  
## SLG < 0.4425426 to the left, improve=0.21830790, (0 missing)  
## H < 115.5 to the left, improve=0.16831530, (0 missing)  
## SH < 0.5 to the right, improve=0.02172099, (0 missing)  
## PB < 1.5 to the right, improve=0.01345964, (0 missing)  
## Surrogate splits:  
## H < 23.5 to the left, agree=0.749, adj=0.069, (0 split)  
## CS < 1.5 to the left, agree=0.743, adj=0.047, (0 split)  
## PB < 1.5 to the right, agree=0.735, adj=0.019, (0 split)  
## SLG < 0.2086111 to the left, agree=0.733, adj=0.011, (0 split)  
##   
## Node number 7: 1373 observations, complexity param=0.03623698  
## mean=3.825385, MSE=0.1342386   
## left son=14 (663 obs) right son=15 (710 obs)  
## Primary splits:  
## SLG < 0.3724166 to the left, improve=0.43232490, (0 missing)  
## H < 50.5 to the left, improve=0.12943110, (0 missing)  
## SB < 1.5 to the left, improve=0.02292804, (0 missing)  
## SH < 0.5 to the right, improve=0.01656236, (0 missing)  
## age < 27.5 to the left, improve=0.01407688, (0 missing)  
## Surrogate splits:  
## H < 35.5 to the left, agree=0.735, adj=0.451, (0 split)  
## SB < 1.5 to the left, agree=0.580, adj=0.131, (0 split)  
## SH < 0.5 to the right, agree=0.580, adj=0.130, (0 split)  
## age < 27.5 to the left, agree=0.550, adj=0.068, (0 split)  
##   
## Node number 12: 1288 observations, complexity param=0.01627525  
## mean=3.032024, MSE=0.09574528   
## left son=24 (597 obs) right son=25 (691 obs)  
## Primary splits:  
## SLG < 0.387676 to the left, improve=0.29020220, (0 missing)  
## CS < 1.5 to the right, improve=0.21040330, (0 missing)  
## SB < 1.5 to the left, improve=0.15028840, (0 missing)  
## H < 102.5 to the left, improve=0.07515719, (0 missing)  
## SH < 0.5 to the right, improve=0.05862163, (0 missing)  
## Surrogate splits:  
## H < 63.5 to the left, agree=0.717, adj=0.390, (0 split)  
## SH < 0.5 to the right, agree=0.623, adj=0.188, (0 split)  
## PB < 0.5 to the right, agree=0.546, adj=0.020, (0 split)  
## age < 43 to the right, agree=0.538, adj=0.003, (0 split)  
##   
## Node number 13: 3488 observations, complexity param=0.0380083  
## mean=3.472613, MSE=0.08465334   
## left son=26 (2259 obs) right son=27 (1229 obs)  
## Primary splits:  
## SLG < 0.4425995 to the left, improve=0.28305060, (0 missing)  
## H < 134.5 to the left, improve=0.13764890, (0 missing)  
## SB < 7.5 to the left, improve=0.11472050, (0 missing)  
## SH < 0.5 to the right, improve=0.07297869, (0 missing)  
## CS < 1.5 to the right, improve=0.05302439, (0 missing)  
## Surrogate splits:  
## SH < 0.5 to the right, agree=0.728, adj=0.227, (0 split)  
## H < 143.5 to the left, agree=0.719, adj=0.202, (0 split)  
##   
## Node number 14: 663 observations, complexity param=0.0101142  
## mean=3.576088, MSE=0.05909715   
## left son=28 (146 obs) right son=29 (517 obs)  
## Primary splits:  
## SLG < 0.2376917 to the left, improve=0.56761990, (0 missing)  
## H < 4.5 to the left, improve=0.35094980, (0 missing)  
## SH < 2.5 to the left, improve=0.02840930, (0 missing)  
## PB < 2.5 to the left, improve=0.02233332, (0 missing)  
## age < 26.5 to the left, improve=0.02090981, (0 missing)  
## Surrogate splits:  
## H < 4.5 to the left, agree=0.857, adj=0.349, (0 split)  
##   
## Node number 15: 710 observations, complexity param=0.01212961  
## mean=4.058179, MSE=0.09217829   
## left son=30 (683 obs) right son=31 (27 obs)  
## Primary splits:  
## SLG < 0.646851 to the left, improve=0.407536300, (0 missing)  
## H < 2.5 to the right, improve=0.165055600, (0 missing)  
## SH < 0.5 to the right, improve=0.031834800, (0 missing)  
## PB < 0.5 to the right, improve=0.006185946, (0 missing)  
## age < 22.5 to the right, improve=0.003488734, (0 missing)  
## Surrogate splits:  
## H < 2.5 to the right, agree=0.966, adj=0.111, (0 split)  
##   
## Node number 24: 597 observations  
## mean=2.852691, MSE=0.06207292   
##   
## Node number 25: 691 observations  
## mean=3.186962, MSE=0.07304585   
##   
## Node number 26: 2259 observations, complexity param=0.01029746  
## mean=3.358438, MSE=0.05852299   
## left son=52 (1207 obs) right son=53 (1052 obs)  
## Primary splits:  
## SB < 8.5 to the left, improve=0.171274800, (0 missing)  
## SLG < 0.3394217 to the left, improve=0.161902200, (0 missing)  
## CS < 1.5 to the right, improve=0.066768300, (0 missing)  
## H < 107.5 to the left, improve=0.058351650, (0 missing)  
## age < 25.5 to the left, improve=0.005283756, (0 missing)  
## Surrogate splits:  
## CS < 4.5 to the left, agree=0.747, adj=0.456, (0 split)  
## H < 101.5 to the left, agree=0.665, adj=0.281, (0 split)  
## SH < 3.5 to the left, agree=0.616, adj=0.176, (0 split)  
## SLG < 0.372499 to the left, agree=0.539, adj=0.010, (0 split)  
## age < 22.5 to the right, agree=0.535, adj=0.001, (0 split)  
##   
## Node number 27: 1229 observations  
## mean=3.682476, MSE=0.06467926   
##   
## Node number 28: 146 observations  
## mean=3.231436, MSE=0.06513958   
##   
## Node number 29: 517 observations  
## mean=3.673418, MSE=0.01437309   
##   
## Node number 30: 683 observations  
## mean=4.019643, MSE=0.02930333   
##   
## Node number 31: 27 observations  
## mean=5.033002, MSE=0.6948351   
##   
## Node number 52: 1207 observations  
## mean=3.26497, MSE=0.06018338   
##   
## Node number 53: 1052 observations  
## mean=3.465678, MSE=0.03509408

##train  
predictedR3 <- predict(DT\_5\_WAR, train.data, method = "anova")  
RMSE(predictedR3, train.data$WAR)

## [1] 0.2283263

R2(predictedR3, train.data$WAR)

## [1] 0.8337554

##test  
predictedR3 <- predict(DT\_1\_WAR, test.data, method = "anova")  
RMSE(predictedR3, test.data$WAR)

## [1] 0.2796959

R2(predictedR3, test.data$WAR)

## [1] 0.7514515

## Conclusion

The model that is the best at predicting the data is the SVR with an R^2 value of 0.804 and an RMSE value of 0.256. The model that is the best at explaining the data is the decision tree with an R^2 value of 0.8333 and an RMSE of 0.228

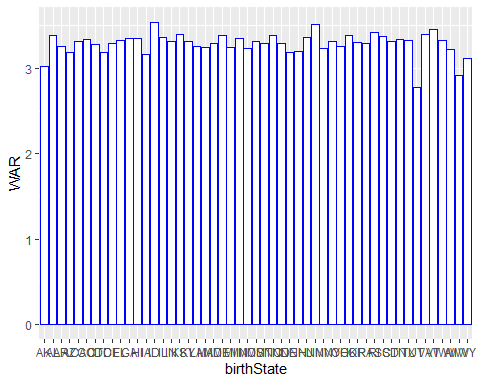
## WAR vs birthState

In this question, we wanted to determine whether the origin of a position player is a good predictor their future performance in the MLB

stateavgFIP <- pitcher\_stats\_complete\_AGE %>% subset(select = c(birthCountry, birthState, FIP))  
stateavgWAR <- Player\_WAR\_AGE %>% subset(select = c(birthCountry, birthState, WAR))  
  
stateavgFIP <- stateavgFIP %>% filter(birthCountry == "USA")  
stateavgWAR <- stateavgWAR %>% filter(birthCountry == "USA")  
  
stateavgFIP <- stateavgFIP %>% subset(select = -c(birthCountry))  
stateavgWAR <- stateavgWAR %>% subset(select = -c(birthCountry))  
  
stateavgFIP <- stateavgFIP %>% filter(is.finite(FIP))  
stateavgWAR <- stateavgWAR %>% filter(is.finite(WAR))  
  
stateavgFIP <- aggregate(. ~ birthState, data = stateavgFIP, mean)  
stateavgWAR <- aggregate(. ~ birthState, data = stateavgWAR, mean)  
  
stateFIP <- pitcher\_stats\_complete\_AGE  
stateWAR <- Player\_WAR\_AGE  
  
stateFIP <- stateFIP %>% filter(birthCountry == "USA")  
stateWAR <- stateWAR %>% filter(birthCountry == "USA")  
  
stateFIP$birthState <- as.factor(stateFIP$birthState)  
stateWAR$birthState <- as.factor(stateWAR$birthState)  
  
stateFIP <- na.omit(stateFIP)  
stateWAR <- na.omit(stateWAR)

## Visualization

ggplot(data=stateavgWAR, aes(x=birthState, y=WAR)) + geom\_bar(stat="identity", color="blue", fill="white")



There is not a noticeable difference in the WAR of position players regardless of which state that they were born in, outside of players who were born in utah.

## Modeling

## MLR

set.seed(123)  
training.samples <- stateWAR$WAR %>% createDataPartition(p = 0.8, list = FALSE)  
train.data <- stateWAR[training.samples, ]  
test.data <- stateWAR[-training.samples, ]

MVR\_6\_WAR <- lm(WAR ~ AB + R + SB + CS + SH + OBA + SLG + birthState, data = train.data)  
summary(MVR\_6\_WAR)

##   
## Call:  
## lm(formula = WAR ~ AB + R + SB + CS + SH + OBA + SLG + birthState,   
## data = train.data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.25979 -0.14797 0.07798 0.23429 1.05912   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.006e+00 1.852e-01 10.835 < 2e-16 \*\*\*  
## AB -4.252e-04 9.125e-05 -4.659 3.25e-06 \*\*\*  
## R 3.397e-03 6.161e-04 5.514 3.66e-08 \*\*\*  
## SB 4.269e-02 1.006e-03 42.427 < 2e-16 \*\*\*  
## CS -1.220e-01 3.023e-03 -40.353 < 2e-16 \*\*\*  
## SH 6.489e-03 2.357e-03 2.753 0.00592 \*\*   
## OBA 2.036e+00 1.612e-01 12.628 < 2e-16 \*\*\*  
## SLG 1.520e+00 9.518e-02 15.969 < 2e-16 \*\*\*  
## birthStateAL 1.212e-01 1.871e-01 0.648 0.51698   
## birthStateAR 8.798e-02 1.967e-01 0.447 0.65476   
## birthStateAZ 2.439e-02 1.878e-01 0.130 0.89666   
## birthStateCA 4.936e-02 1.826e-01 0.270 0.78687   
## birthStateCO 1.311e-01 2.073e-01 0.633 0.52699   
## birthStateCT 1.164e-02 1.887e-01 0.062 0.95084   
## birthStateDC 1.376e-01 2.103e-01 0.654 0.51293   
## birthStateDE -5.521e-02 1.974e-01 -0.280 0.77972   
## birthStateFL 5.591e-02 1.832e-01 0.305 0.76017   
## birthStateGA 8.288e-02 1.840e-01 0.450 0.65249   
## birthStateHI 3.703e-02 2.019e-01 0.183 0.85449   
## birthStateIA 2.460e-02 1.941e-01 0.127 0.89913   
## birthStateID 2.104e-01 2.744e-01 0.767 0.44328   
## birthStateIL 6.561e-02 1.843e-01 0.356 0.72180   
## birthStateIN 4.914e-02 1.861e-01 0.264 0.79174   
## birthStateKS 4.772e-02 1.919e-01 0.249 0.80366   
## birthStateKY 1.113e-01 1.888e-01 0.589 0.55558   
## birthStateLA 1.100e-01 1.868e-01 0.589 0.55600   
## birthStateMA 6.288e-03 1.894e-01 0.033 0.97352   
## birthStateMD 4.864e-02 1.891e-01 0.257 0.79702   
## birthStateME 2.016e-01 2.732e-01 0.738 0.46056   
## birthStateMI 4.909e-03 1.882e-01 0.026 0.97919   
## birthStateMN 7.502e-02 1.956e-01 0.384 0.70136   
## birthStateMO -2.258e-02 1.890e-01 -0.119 0.90494   
## birthStateMS 3.985e-02 1.861e-01 0.214 0.83045   
## birthStateMT -1.037e-01 3.406e-01 -0.304 0.76081   
## birthStateNC 7.535e-02 1.856e-01 0.406 0.68472   
## birthStateND -3.289e-02 2.088e-01 -0.158 0.87484   
## birthStateNE -4.333e-03 1.996e-01 -0.022 0.98268   
## birthStateNH -1.943e-02 2.466e-01 -0.079 0.93719   
## birthStateNJ 2.864e-02 1.861e-01 0.154 0.87769   
## birthStateNM 1.838e-01 2.123e-01 0.866 0.38658   
## birthStateNV 7.524e-02 1.973e-01 0.381 0.70291   
## birthStateNY 4.816e-02 1.839e-01 0.262 0.79339   
## birthStateOH 1.914e-02 1.845e-01 0.104 0.91738   
## birthStateOK 1.167e-01 1.888e-01 0.618 0.53641   
## birthStateOR -3.201e-02 1.876e-01 -0.171 0.86449   
## birthStatePA 6.451e-02 1.855e-01 0.348 0.72801   
## birthStateRI 5.176e-02 2.123e-01 0.244 0.80739   
## birthStateSC 5.026e-02 1.882e-01 0.267 0.78949   
## birthStateSD -1.313e-02 2.038e-01 -0.064 0.94863   
## birthStateTN 3.225e-02 1.913e-01 0.169 0.86612   
## birthStateTX 5.137e-02 1.836e-01 0.280 0.77964   
## birthStateUT -1.547e-01 2.465e-01 -0.627 0.53039   
## birthStateVA 8.784e-02 1.871e-01 0.470 0.63870   
## birthStateVT -3.683e-02 2.324e-01 -0.158 0.87408   
## birthStateWA 1.013e-01 1.864e-01 0.543 0.58683   
## birthStateWI -3.731e-03 1.930e-01 -0.019 0.98458   
## birthStateWV -2.364e-01 2.576e-01 -0.918 0.35883   
## birthStateWY -5.722e-02 2.073e-01 -0.276 0.78255   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.407 on 5341 degrees of freedom  
## Multiple R-squared: 0.4624, Adjusted R-squared: 0.4566   
## F-statistic: 80.58 on 57 and 5341 DF, p-value: < 2.2e-16

##train Data  
predictedR <- predict(MVR\_6\_WAR, train.data)  
  
RMSE(predictedR, train.data$WAR)

## [1] 0.4048419

R2(predictedR, train.data$WAR)

## [1] 0.4623657

##test Data  
predictedR <- predict(MVR\_6\_WAR, test.data)  
  
RMSE(predictedR, test.data$WAR)

## [1] 0.4129065

R2(predictedR, test.data$WAR)

## [1] 0.4796459

##SVR

SVR\_6\_WAR <- svm(WAR ~ AB + R + SB + CS + SH + OBA + SLG + birthState, data = train.data)  
  
##Train Data  
predicted.classes <- predict(SVR\_6\_WAR, train.data)  
RMSE(predicted.classes, train.data$WAR)

## [1] 0.3300996

R2(predicted.classes, train.data$WAR)

## [1] 0.683664

##Test Data  
predicted.classes <- predict(SVR\_6\_WAR, test.data)  
  
RMSE(predicted.classes, test.data$WAR)

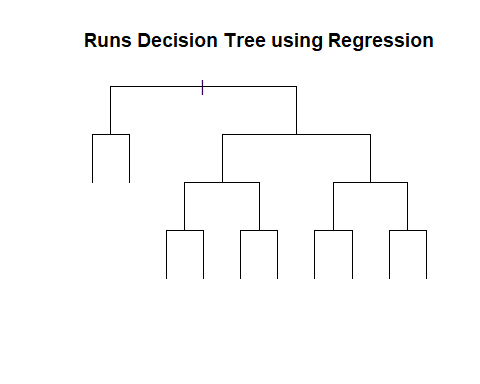
## [1] 0.3615384

R2(predicted.classes, test.data$WAR)

## [1] 0.642366

##DT

DT\_6\_WAR <- rpart(WAR ~ AB + R + SB + CS + SH + OBA + SLG + birthState, data = train.data, method = "anova")  
  
plot(DT\_6\_WAR, uniform = TRUE,  
 main = "Runs Decision Tree using Regression")



summary(DT\_6\_WAR)

## Call:  
## rpart(formula = WAR ~ AB + R + SB + CS + SH + OBA + SLG + birthState,   
## data = train.data, method = "anova")  
## n= 5399   
##   
## CP nsplit rel error xerror xstd  
## 1 0.51116747 0 1.0000000 1.0002836 0.021481621  
## 2 0.11407578 1 0.4888325 0.4890948 0.011771308  
## 3 0.08414570 2 0.3747567 0.3751198 0.009221386  
## 4 0.03597719 3 0.2906110 0.2921979 0.008186696  
## 5 0.03557327 4 0.2546339 0.2582639 0.007615153  
## 6 0.01605662 5 0.2190606 0.2226063 0.006409756  
## 7 0.01554638 6 0.2030040 0.2081144 0.006125460  
## 8 0.01131040 7 0.1874576 0.1924143 0.005891679  
## 9 0.01057853 8 0.1761472 0.1846860 0.005588674  
## 10 0.01000000 9 0.1655686 0.1763493 0.005325972  
##   
## Variable importance  
## SB SLG CS R OBA AB SH   
## 57 12 11 6 6 5 1   
##   
## Node number 1: 5399 observations, complexity param=0.5111675  
## mean=3.316253, MSE=0.3048484   
## left son=2 (665 obs) right son=3 (4734 obs)  
## Primary splits:  
## SB < 0.5 to the left, improve=0.51116750, (0 missing)  
## CS < 0.5 to the right, improve=0.21680480, (0 missing)  
## OBA < 0.329879 to the left, improve=0.10946440, (0 missing)  
## SLG < 0.4148616 to the left, improve=0.10637810, (0 missing)  
## R < 59.5 to the left, improve=0.08080649, (0 missing)  
##   
## Node number 2: 665 observations, complexity param=0.01554638  
## mean=2.263014, MSE=0.07105369   
## left son=4 (280 obs) right son=5 (385 obs)  
## Primary splits:  
## SLG < 0.3718681 to the left, improve=0.54152450, (0 missing)  
## OBA < 0.3085145 to the left, improve=0.45929010, (0 missing)  
## R < 11.5 to the left, improve=0.18100180, (0 missing)  
## AB < 229.5 to the left, improve=0.12356110, (0 missing)  
## birthState splits as LLLLRLLLLLRLL-RLRRRRLRRRRR-LRLRLLRLRRLRRRRRRLRRRLRR, improve=0.03746789, (0 missing)  
## Surrogate splits:  
## OBA < 0.2969997 to the left, agree=0.785, adj=0.489, (0 split)  
## R < 11.5 to the left, agree=0.704, adj=0.296, (0 split)  
## AB < 158 to the left, agree=0.669, adj=0.214, (0 split)  
## birthState splits as LRRLRLRLRLRRL-RLRRLRRRRRRR-RRRRRLRRRLRRRRRRRLRRRRRR, agree=0.611, adj=0.075, (0 split)  
## SH < 4.5 to the right, agree=0.588, adj=0.021, (0 split)  
##   
## Node number 3: 4734 observations, complexity param=0.1140758  
## mean=3.464205, MSE=0.159972   
## left son=6 (3656 obs) right son=7 (1078 obs)  
## Primary splits:  
## CS < 0.5 to the right, improve=0.24792400, (0 missing)  
## SLG < 0.4699624 to the left, improve=0.16827090, (0 missing)  
## OBA < 0.35063 to the left, improve=0.14505090, (0 missing)  
## R < 71.5 to the left, improve=0.06508121, (0 missing)  
## SB < 4.5 to the left, improve=0.03704488, (0 missing)  
## Surrogate splits:  
## R < 11.5 to the right, agree=0.805, adj=0.143, (0 split)  
## AB < 56.5 to the right, agree=0.802, adj=0.131, (0 split)  
## OBA < 0.2401044 to the right, agree=0.788, adj=0.068, (0 split)  
## SLG < 0.2254983 to the right, agree=0.783, adj=0.046, (0 split)  
## SB < 1.5 to the right, agree=0.782, adj=0.041, (0 split)  
##   
## Node number 4: 280 observations  
## mean=2.033, MSE=0.03535499   
##   
## Node number 5: 385 observations  
## mean=2.430297, MSE=0.03055556   
##   
## Node number 6: 3656 observations, complexity param=0.0841457  
## mean=3.356064, MSE=0.1213936   
## left son=12 (1018 obs) right son=13 (2638 obs)  
## Primary splits:  
## SB < 2.5 to the left, improve=0.3120522, (0 missing)  
## SLG < 0.4421894 to the left, improve=0.2287672, (0 missing)  
## R < 58.5 to the left, improve=0.2215877, (0 missing)  
## OBA < 0.3419208 to the left, improve=0.2138262, (0 missing)  
## AB < 357.5 to the left, improve=0.1472982, (0 missing)  
## Surrogate splits:  
## R < 17.5 to the left, agree=0.756, adj=0.124, (0 split)  
## AB < 117.5 to the left, agree=0.748, adj=0.093, (0 split)  
## CS < 1.5 to the left, agree=0.741, adj=0.071, (0 split)  
## OBA < 0.2324507 to the left, agree=0.729, adj=0.027, (0 split)  
## SLG < 0.2095721 to the left, agree=0.725, adj=0.012, (0 split)  
##   
## Node number 7: 1078 observations, complexity param=0.03557327  
## mean=3.830959, MSE=0.1166398   
## left son=14 (521 obs) right son=15 (557 obs)  
## Primary splits:  
## SLG < 0.3745639 to the left, improve=0.46564580, (0 missing)  
## OBA < 0.3215534 to the left, improve=0.41306100, (0 missing)  
## R < 20.5 to the left, improve=0.15678070, (0 missing)  
## AB < 326.5 to the left, improve=0.11758110, (0 missing)  
## SB < 1.5 to the left, improve=0.02702261, (0 missing)  
## Surrogate splits:  
## OBA < 0.3201027 to the left, agree=0.772, adj=0.528, (0 split)  
## R < 20.5 to the left, agree=0.743, adj=0.468, (0 split)  
## AB < 161 to the left, agree=0.694, adj=0.367, (0 split)  
## SB < 1.5 to the left, agree=0.591, adj=0.154, (0 split)  
## SH < 0.5 to the right, agree=0.579, adj=0.129, (0 split)  
##   
## Node number 12: 1018 observations, complexity param=0.01605662  
## mean=3.042754, MSE=0.09227101   
## left son=24 (461 obs) right son=25 (557 obs)  
## Primary splits:  
## SLG < 0.3874437 to the left, improve=0.2813445, (0 missing)  
## OBA < 0.337143 to the left, improve=0.2435648, (0 missing)  
## CS < 1.5 to the right, improve=0.1960676, (0 missing)  
## SB < 1.5 to the left, improve=0.1452568, (0 missing)  
## R < 57.5 to the left, improve=0.1170141, (0 missing)  
## Surrogate splits:  
## OBA < 0.3148356 to the left, agree=0.760, adj=0.471, (0 split)  
## R < 26.5 to the left, agree=0.752, adj=0.453, (0 split)  
## AB < 276.5 to the left, agree=0.704, adj=0.347, (0 split)  
## SH < 0.5 to the right, agree=0.616, adj=0.152, (0 split)  
## birthState splits as RLLRRRLLRRRLLLLLRRLLL-RRRRLRRRLRRLRRRRLRLRRRRLLRR-R, agree=0.610, adj=0.139, (0 split)  
##   
## Node number 13: 2638 observations, complexity param=0.03597719  
## mean=3.47697, MSE=0.08013254   
## left son=26 (1670 obs) right son=27 (968 obs)  
## Primary splits:  
## SLG < 0.4425955 to the left, improve=0.28011790, (0 missing)  
## OBA < 0.3484286 to the left, improve=0.23810950, (0 missing)  
## R < 66.5 to the left, improve=0.18347240, (0 missing)  
## SB < 7.5 to the left, improve=0.11755010, (0 missing)  
## AB < 481.5 to the left, improve=0.08836526, (0 missing)  
## Surrogate splits:  
## OBA < 0.354922 to the left, agree=0.754, adj=0.329, (0 split)  
## R < 80.5 to the left, agree=0.742, adj=0.298, (0 split)  
## SH < 0.5 to the right, agree=0.701, adj=0.186, (0 split)  
## AB < 518.5 to the left, agree=0.684, adj=0.138, (0 split)  
## birthState splits as -LRLLLLLLLLLLLLLLRLRL-LLLL-LLLLLLRLLRLLLLLRLLLLLLLL, agree=0.641, adj=0.021, (0 split)  
##   
## Node number 14: 521 observations, complexity param=0.01057853  
## mean=3.58999, MSE=0.05936973   
## left son=28 (110 obs) right son=29 (411 obs)  
## Primary splits:  
## SLG < 0.2372294 to the left, improve=0.56288530, (0 missing)  
## OBA < 0.2348346 to the left, improve=0.49740940, (0 missing)  
## R < 4.5 to the left, improve=0.20912530, (0 missing)  
## AB < 60.5 to the left, improve=0.18752770, (0 missing)  
## birthState splits as LRLLRLRRRLLRR-LLLRLRRRRRLL-LRL-LRRLRRRR-LRLLRRRRR-L, improve=0.05476073, (0 missing)  
## Surrogate splits:  
## OBA < 0.2039273 to the left, agree=0.875, adj=0.409, (0 split)  
## R < 1.5 to the left, agree=0.821, adj=0.155, (0 split)  
## AB < 12.5 to the left, agree=0.816, adj=0.127, (0 split)  
## birthState splits as RRRRRLRRRRRRR-RRRRRRRRRRRL-RRL-RRRRRRRR-RRRRRRRRR-R, agree=0.795, adj=0.027, (0 split)  
##   
## Node number 15: 557 observations, complexity param=0.0113104  
## mean=4.056353, MSE=0.06509306   
## left son=30 (484 obs) right son=31 (73 obs)  
## Primary splits:  
## SLG < 0.543855 to the left, improve=0.51343480, (0 missing)  
## OBA < 0.3818374 to the left, improve=0.40885620, (0 missing)  
## AB < 22.5 to the right, improve=0.26647270, (0 missing)  
## R < 5.5 to the right, improve=0.12154860, (0 missing)  
## birthState splits as -LRLRLLLLLRLLRLLLLRLRLLRLL-R-L-LRLLLLRLLLLLL-L-LL-L, improve=0.05855287, (0 missing)  
## Surrogate splits:  
## OBA < 0.4075714 to the left, agree=0.908, adj=0.301, (0 split)  
## AB < 22.5 to the right, agree=0.885, adj=0.123, (0 split)  
## R < 107.5 to the left, agree=0.882, adj=0.096, (0 split)  
## birthState splits as -LLLLLLLLLLLLRLLLLLLLLLLLL-L-L-LLLLLLLLLLLLL-L-LL-L, agree=0.871, adj=0.014, (0 split)  
##   
## Node number 24: 461 observations  
## mean=2.865649, MSE=0.05812419   
##   
## Node number 25: 557 observations  
## mean=3.189334, MSE=0.07308692   
##   
## Node number 26: 1670 observations  
## mean=3.362905, MSE=0.05678698   
##   
## Node number 27: 968 observations  
## mean=3.673757, MSE=0.05923694   
##   
## Node number 28: 110 observations  
## mean=3.236631, MSE=0.05933381   
##   
## Node number 29: 411 observations  
## mean=3.684563, MSE=0.01701691   
##   
## Node number 30: 484 observations  
## mean=3.985354, MSE=0.01655196   
##   
## Node number 31: 73 observations  
## mean=4.527082, MSE=0.13192

##train  
predictedR3 <- predict(DT\_6\_WAR, train.data, method = "anova")  
RMSE(predictedR3, train.data$WAR)

## [1] 0.2246627

R2(predictedR3, train.data$WAR)

## [1] 0.8344314

##test  
predictedR3 <- predict(DT\_6\_WAR, test.data, method = "anova")  
RMSE(predictedR3, test.data$WAR)

## [1] 0.2530343

R2(predictedR3, test.data$WAR)

## [1] 0.8049066

## Conclusion

The best model for both explaining and predicting the data is the Decision Tree. While predicting the data, the Decision Tree has an R^2 value of 0.804 and an RMSE value of 0.25. While explaining the data, the Decision Tree has an R^2 value of 0.834 and an RMSE value of 0.22.

## Question 3

## FIP vs Salary

In this question, we wanted to determine whether how much a pitcher is paid can predict their future performance

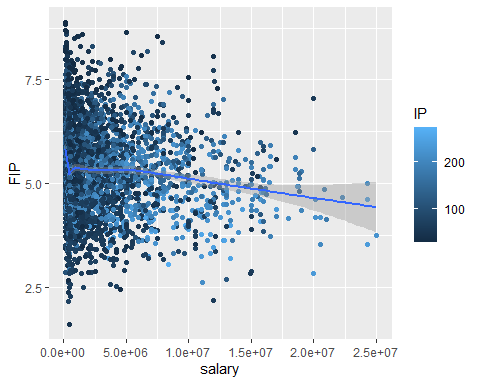
## visualization

FIP\_Salary <- left\_join(pitcher\_stats\_complete, Salaries,by = c("playerID", "yearID"))  
##Adding age  
##filtering out small sample sizes (games)  
FIP\_Salary <- FIP\_Salary %>% filter(IP > 30)  
##filtering out outliers(WAR)  
FIP\_Salary <- FIP\_Salary %>% filter(FIP < 9)  
##Visualization  
ggplot(data = FIP\_Salary, mapping = aes(x = salary, y = FIP, color = IP)) + geom\_point() +geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 1001 rows containing non-finite values (stat\_smooth).

## Warning: Removed 1001 rows containing missing values (geom\_point).

 The FIP converges towards a best fit line which means that there is less variability in the performance of a pitcher the higher the salary.

##modeling

FIP\_Salary2 <- FIP\_Salary %>% subset(select = -c(playerID, yearID, teamID, lgID))  
  
set.seed(123)  
training.samples <- FIP\_Salary2$FIP %>% createDataPartition(p = 0.8, list = FALSE)  
train.data <- FIP\_Salary2[training.samples, ]  
test.data <- FIP\_Salary2[-training.samples, ]

##MLR

model1c <- lm(FIP ~ PO + E + DP + W + L + CG + HR + BB + BAOpp + WP + HBP + ERA, data = train.data)   
summary(model1c)

##   
## Call:  
## lm(formula = FIP ~ PO + E + DP + W + L + CG + HR + BB + BAOpp +   
## WP + HBP + ERA, data = train.data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.90597 -0.39616 -0.00822 0.37155 2.40484   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.3527375 0.0401035 83.602 < 2e-16 \*\*\*  
## PO -0.0095032 0.0025778 -3.687 0.00023 \*\*\*  
## E -0.0205076 0.0093641 -2.190 0.02858 \*   
## DP 0.0246935 0.0086485 2.855 0.00432 \*\*   
## W -0.0677634 0.0037407 -18.115 < 2e-16 \*\*\*  
## L -0.0895451 0.0041515 -21.569 < 2e-16 \*\*\*  
## CG -0.0448733 0.0102787 -4.366 1.3e-05 \*\*\*  
## HR 0.0670345 0.0021047 31.850 < 2e-16 \*\*\*  
## BB 0.0121056 0.0008296 14.593 < 2e-16 \*\*\*  
## BAOpp -0.1768345 0.0751628 -2.353 0.01868 \*   
## WP -0.0359825 0.0037140 -9.688 < 2e-16 \*\*\*  
## HBP 0.0107973 0.0037208 2.902 0.00373 \*\*   
## ERA 0.4507469 0.0083237 54.152 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.5917 on 4194 degrees of freedom  
## Multiple R-squared: 0.6585, Adjusted R-squared: 0.6576   
## F-statistic: 674 on 12 and 4194 DF, p-value: < 2.2e-16

##train Data  
predictedR <- predict(model1c, train.data)  
RMSE(predictedR, train.data$FIP)

## [1] 0.5907724

R2(predictedR, train.data$FIP)

## [1] 0.6585348

##test Data  
predictedR <- predict(model1c, test.data)  
RMSE(predictedR, test.data$FIP)

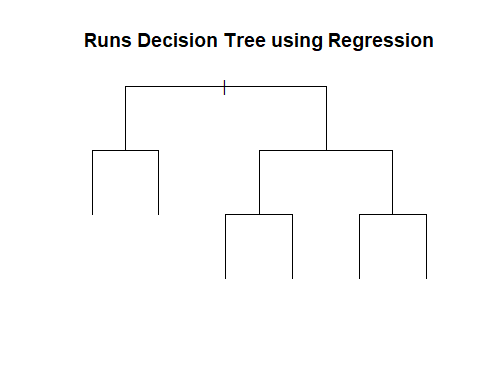
## [1] 0.5975134

R2(predictedR, test.data$FIP)

## [1] 0.6555688

##DT

DT\_1\_FIP <- rpart(FIP ~ PO + E + DP + W + L + CG + HR + BB + BAOpp + WP + HBP + ERA, data = train.data, method = "anova")  
  
plot(DT\_1\_FIP, uniform = TRUE,  
 main = "Runs Decision Tree using Regression")



summary(DT\_1\_FIP)

## Call:  
## rpart(formula = FIP ~ PO + E + DP + W + L + CG + HR + BB + BAOpp +   
## WP + HBP + ERA, data = train.data, method = "anova")  
## n= 4207   
##   
## CP nsplit rel error xerror xstd  
## 1 0.35262219 0 1.0000000 1.0001367 0.02261521  
## 2 0.07040909 1 0.6473778 0.6562659 0.01488947  
## 3 0.06306076 2 0.5769687 0.5895975 0.01374755  
## 4 0.01974605 3 0.5139080 0.5291982 0.01203378  
## 5 0.01276962 4 0.4941619 0.5042508 0.01152797  
## 6 0.01000000 5 0.4813923 0.4946918 0.01125976  
##   
## Variable importance  
## ERA BAOpp HR L W BB   
## 58 20 10 5 4 4   
##   
## Node number 1: 4207 observations, complexity param=0.3526222  
## mean=5.417703, MSE=1.022101   
## left son=2 (2043 obs) right son=3 (2164 obs)  
## Primary splits:  
## ERA < 4.170961 to the left, improve=0.35262220, (0 missing)  
## HR < 5.5 to the left, improve=0.14705380, (0 missing)  
## BAOpp < 0.2635 to the left, improve=0.12119050, (0 missing)  
## W < 2.5 to the right, improve=0.02358530, (0 missing)  
## BB < 21.5 to the left, improve=0.02265184, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.2595 to the left, agree=0.734, adj=0.453, (0 split)  
## HR < 7.5 to the left, agree=0.625, adj=0.229, (0 split)  
## L < 4.5 to the left, agree=0.577, adj=0.129, (0 split)  
## W < 10.5 to the right, agree=0.563, adj=0.101, (0 split)  
## BB < 29.5 to the left, agree=0.556, adj=0.087, (0 split)  
##   
## Node number 2: 2043 observations, complexity param=0.07040909  
## mean=4.799834, MSE=0.6228545   
## left son=4 (645 obs) right son=5 (1398 obs)  
## Primary splits:  
## ERA < 3.024167 to the left, improve=0.23792510, (0 missing)  
## HR < 5.5 to the left, improve=0.13097490, (0 missing)  
## BAOpp < 0.2345 to the left, improve=0.05126493, (0 missing)  
## BB < 21.5 to the left, improve=0.03517228, (0 missing)  
## HBP < 1.5 to the left, improve=0.01947285, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.2245 to the left, agree=0.740, adj=0.177, (0 split)  
## HR < 3.5 to the left, agree=0.704, adj=0.064, (0 split)  
## BB < 6.5 to the left, agree=0.688, adj=0.012, (0 split)  
## W < 20.5 to the right, agree=0.687, adj=0.009, (0 split)  
## PO < 31.5 to the right, agree=0.686, adj=0.006, (0 split)  
##   
## Node number 3: 2164 observations, complexity param=0.06306076  
## mean=6.001024, MSE=0.6983458   
## left son=6 (1697 obs) right son=7 (467 obs)  
## Primary splits:  
## ERA < 5.933593 to the left, improve=0.17943120, (0 missing)  
## HR < 5.5 to the left, improve=0.07124372, (0 missing)  
## W < 8.5 to the right, improve=0.03153193, (0 missing)  
## BAOpp < 0.3035 to the left, improve=0.02202515, (0 missing)  
## PO < 8.5 to the right, improve=0.01735863, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.6535 to the left, agree=0.788, adj=0.017, (0 split)  
## BB < 5.5 to the right, agree=0.785, adj=0.002, (0 split)  
##   
## Node number 4: 645 observations  
## mean=4.23309, MSE=0.5431053   
##   
## Node number 5: 1398 observations  
## mean=5.061315, MSE=0.4430837   
##   
## Node number 6: 1697 observations, complexity param=0.01974605  
## mean=5.815328, MSE=0.53005   
## left son=12 (695 obs) right son=13 (1002 obs)  
## Primary splits:  
## ERA < 4.708883 to the left, improve=0.094394790, (0 missing)  
## HR < 5.5 to the left, improve=0.079414670, (0 missing)  
## BAOpp < 0.1815 to the right, improve=0.016831850, (0 missing)  
## W < 8.5 to the right, improve=0.012067870, (0 missing)  
## WP < 5.5 to the right, improve=0.007668703, (0 missing)  
## Surrogate splits:  
## W < 10.5 to the right, agree=0.609, adj=0.046, (0 split)  
## BAOpp < 0.2675 to the left, agree=0.602, adj=0.029, (0 split)  
## PO < 15.5 to the right, agree=0.598, adj=0.017, (0 split)  
## DP < 5.5 to the right, agree=0.598, adj=0.017, (0 split)  
## HR < 3.5 to the left, agree=0.594, adj=0.009, (0 split)  
##   
## Node number 7: 467 observations, complexity param=0.01276962  
## mean=6.67581, MSE=0.7292619   
## left son=14 (279 obs) right son=15 (188 obs)  
## Primary splits:  
## ERA < 6.806542 to the left, improve=0.16122940, (0 missing)  
## HR < 7.5 to the left, improve=0.11363610, (0 missing)  
## BB < 21.5 to the left, improve=0.03964839, (0 missing)  
## BAOpp < 0.282 to the right, improve=0.03044145, (0 missing)  
## W < 1.5 to the right, improve=0.01354335, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.3325 to the left, agree=0.647, adj=0.122, (0 split)  
## W < 1.5 to the right, agree=0.612, adj=0.037, (0 split)  
## PO < 0.5 to the right, agree=0.600, adj=0.005, (0 split)  
## WP < 12 to the left, agree=0.600, adj=0.005, (0 split)  
## HBP < 0.5 to the right, agree=0.600, adj=0.005, (0 split)  
##   
## Node number 12: 695 observations  
## mean=5.546748, MSE=0.4338636   
##   
## Node number 13: 1002 observations  
## mean=6.001618, MSE=0.512028   
##   
## Node number 14: 279 observations  
## mean=6.394335, MSE=0.577586   
##   
## Node number 15: 188 observations  
## mean=7.093532, MSE=0.6622856

##Train Data  
predictedR3 <- predict(DT\_1\_FIP, train.data, method = "anova")  
RMSE(predictedR3, train.data$FIP)

## [1] 0.7014497

R2(predictedR3, train.data$FIP)

## [1] 0.5186077

##Test Data  
predictedR <- predict(DT\_1\_FIP, data = train.data, method = "anova")  
RMSE(predictedR, train.data$FIP)

## [1] 0.7014497

R2(predictedR, train.data$FIP)

## [1] 0.5186077

The best model for both explaining and predicting the data is the Multiple Linear regression. For explaining the data, the Multiple Linear Regression has an R^2 value of 0.658 and an RMSE value of 0.590. For predicting the data, the Multiple Linear Regression has an R^2 value of 0.655 and an RMSE value of 0.597

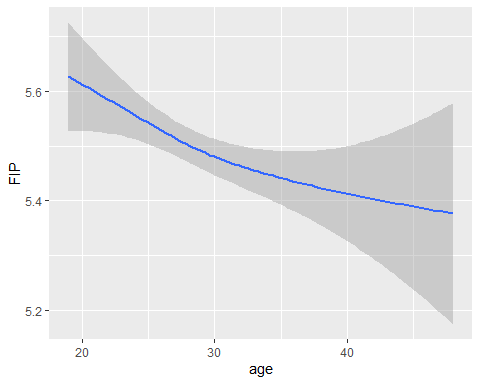
## FIP vs Age

In this question, we wanted to determine whether the origin of a pitcher is a good predictor of their future performance in the MLB.

## visualization

ggplot(data = pitcher\_stats\_complete\_AGE) + geom\_smooth(mapping = aes(x = age, y = FIP))

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



We have determined that there is little variation in the average performance of a player regardless of which state that they were born in besides Delaware, Washington DC, Alaska, and New Mexico.

##Modeling ##MLR

set.seed(123)  
training.samples <- pitcher\_stats\_complete\_AGE$FIP %>% createDataPartition(p = 0.8, list = FALSE)  
train.data1 <- pitcher\_stats\_complete\_AGE[training.samples, ]  
test.data1 <- pitcher\_stats\_complete\_AGE[-training.samples, ]

MVR\_5\_FIP <- lm(FIP ~ PO + A + E + DP + W + L + CG + HR + BB + BAOpp + WP + HBP + BK + ERA, data = train.data)  
  
##Train Data  
predicted.classes <- predict(MVR\_5\_FIP, train.data)  
  
RMSE(predicted.classes, train.data$FIP)

## [1] 0.5900913

R2(predicted.classes, train.data$FIP)

## [1] 0.6593217

##Test Data  
predicted.classes <- predict(MVR\_5\_FIP, test.data)  
  
RMSE(predicted.classes, test.data$FIP)

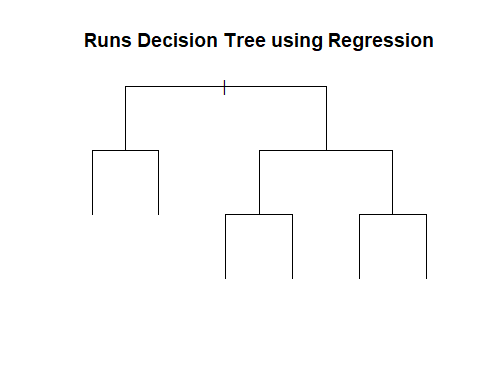
## [1] 0.5975063

R2(predicted.classes, test.data$FIP)

## [1] 0.6555865

##DT

DT\_5\_FIP <- rpart(FIP ~ PO + A + E + DP + W + L + CG + HR + BB + BAOpp + WP + HBP + BK + ERA, data = train.data, method = "anova")  
  
plot(DT\_5\_FIP, uniform = TRUE,  
 main = "Runs Decision Tree using Regression")



summary(DT\_5\_FIP)

## Call:  
## rpart(formula = FIP ~ PO + A + E + DP + W + L + CG + HR + BB +   
## BAOpp + WP + HBP + BK + ERA, data = train.data, method = "anova")  
## n= 4207   
##   
## CP nsplit rel error xerror xstd  
## 1 0.35262219 0 1.0000000 1.0006872 0.02262627  
## 2 0.07040909 1 0.6473778 0.6580623 0.01499323  
## 3 0.06306076 2 0.5769687 0.5837706 0.01374276  
## 4 0.01974605 3 0.5139080 0.5343724 0.01223663  
## 5 0.01276962 4 0.4941619 0.5104774 0.01168822  
## 6 0.01000000 5 0.4813923 0.4985295 0.01144253  
##   
## Variable importance  
## ERA BAOpp HR L W BB   
## 58 20 10 5 4 4   
##   
## Node number 1: 4207 observations, complexity param=0.3526222  
## mean=5.417703, MSE=1.022101   
## left son=2 (2043 obs) right son=3 (2164 obs)  
## Primary splits:  
## ERA < 4.170961 to the left, improve=0.35262220, (0 missing)  
## HR < 5.5 to the left, improve=0.14705380, (0 missing)  
## BAOpp < 0.2635 to the left, improve=0.12119050, (0 missing)  
## W < 2.5 to the right, improve=0.02358530, (0 missing)  
## BB < 21.5 to the left, improve=0.02265184, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.2595 to the left, agree=0.734, adj=0.453, (0 split)  
## HR < 7.5 to the left, agree=0.625, adj=0.229, (0 split)  
## L < 4.5 to the left, agree=0.577, adj=0.129, (0 split)  
## W < 10.5 to the right, agree=0.563, adj=0.101, (0 split)  
## BB < 29.5 to the left, agree=0.556, adj=0.087, (0 split)  
##   
## Node number 2: 2043 observations, complexity param=0.07040909  
## mean=4.799834, MSE=0.6228545   
## left son=4 (645 obs) right son=5 (1398 obs)  
## Primary splits:  
## ERA < 3.024167 to the left, improve=0.23792510, (0 missing)  
## HR < 5.5 to the left, improve=0.13097490, (0 missing)  
## BAOpp < 0.2345 to the left, improve=0.05126493, (0 missing)  
## BB < 21.5 to the left, improve=0.03517228, (0 missing)  
## HBP < 1.5 to the left, improve=0.01947285, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.2245 to the left, agree=0.740, adj=0.177, (0 split)  
## HR < 3.5 to the left, agree=0.704, adj=0.064, (0 split)  
## BB < 6.5 to the left, agree=0.688, adj=0.012, (0 split)  
## W < 20.5 to the right, agree=0.687, adj=0.009, (0 split)  
## PO < 31.5 to the right, agree=0.686, adj=0.006, (0 split)  
##   
## Node number 3: 2164 observations, complexity param=0.06306076  
## mean=6.001024, MSE=0.6983458   
## left son=6 (1697 obs) right son=7 (467 obs)  
## Primary splits:  
## ERA < 5.933593 to the left, improve=0.17943120, (0 missing)  
## HR < 5.5 to the left, improve=0.07124372, (0 missing)  
## W < 8.5 to the right, improve=0.03153193, (0 missing)  
## BAOpp < 0.3035 to the left, improve=0.02202515, (0 missing)  
## A < 12.5 to the right, improve=0.02142409, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.6535 to the left, agree=0.788, adj=0.017, (0 split)  
## BB < 5.5 to the right, agree=0.785, adj=0.002, (0 split)  
##   
## Node number 4: 645 observations  
## mean=4.23309, MSE=0.5431053   
##   
## Node number 5: 1398 observations  
## mean=5.061315, MSE=0.4430837   
##   
## Node number 6: 1697 observations, complexity param=0.01974605  
## mean=5.815328, MSE=0.53005   
## left son=12 (695 obs) right son=13 (1002 obs)  
## Primary splits:  
## ERA < 4.708883 to the left, improve=0.094394790, (0 missing)  
## HR < 5.5 to the left, improve=0.079414670, (0 missing)  
## BAOpp < 0.1815 to the right, improve=0.016831850, (0 missing)  
## W < 8.5 to the right, improve=0.012067870, (0 missing)  
## WP < 5.5 to the right, improve=0.007668703, (0 missing)  
## Surrogate splits:  
## W < 10.5 to the right, agree=0.609, adj=0.046, (0 split)  
## BAOpp < 0.2675 to the left, agree=0.602, adj=0.029, (0 split)  
## PO < 15.5 to the right, agree=0.598, adj=0.017, (0 split)  
## A < 37.5 to the right, agree=0.598, adj=0.017, (0 split)  
## DP < 5.5 to the right, agree=0.598, adj=0.017, (0 split)  
##   
## Node number 7: 467 observations, complexity param=0.01276962  
## mean=6.67581, MSE=0.7292619   
## left son=14 (279 obs) right son=15 (188 obs)  
## Primary splits:  
## ERA < 6.806542 to the left, improve=0.16122940, (0 missing)  
## HR < 7.5 to the left, improve=0.11363610, (0 missing)  
## BB < 21.5 to the left, improve=0.03964839, (0 missing)  
## BAOpp < 0.282 to the right, improve=0.03044145, (0 missing)  
## A < 2.5 to the right, improve=0.02196229, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.3325 to the left, agree=0.647, adj=0.122, (0 split)  
## W < 1.5 to the right, agree=0.612, adj=0.037, (0 split)  
## A < 2.5 to the right, agree=0.602, adj=0.011, (0 split)  
## BK < 2.5 to the left, agree=0.602, adj=0.011, (0 split)  
## PO < 0.5 to the right, agree=0.600, adj=0.005, (0 split)  
##   
## Node number 12: 695 observations  
## mean=5.546748, MSE=0.4338636   
##   
## Node number 13: 1002 observations  
## mean=6.001618, MSE=0.512028   
##   
## Node number 14: 279 observations  
## mean=6.394335, MSE=0.577586   
##   
## Node number 15: 188 observations  
## mean=7.093532, MSE=0.6622856

##train  
predictedR3 <- predict(DT\_5\_FIP, train.data, method = "anova")  
RMSE(predictedR3, train.data$FIP)

## [1] 0.7014497

R2(predictedR3, train.data$FIP)

## [1] 0.5186077

##test  
predictedR3 <- predict(DT\_5\_FIP, test.data, method = "anova")  
RMSE(predictedR3, test.data$FIP)

## [1] 0.7224929

R2(predictedR3, test.data$FIP)

## [1] 0.4960003

## Conclusion

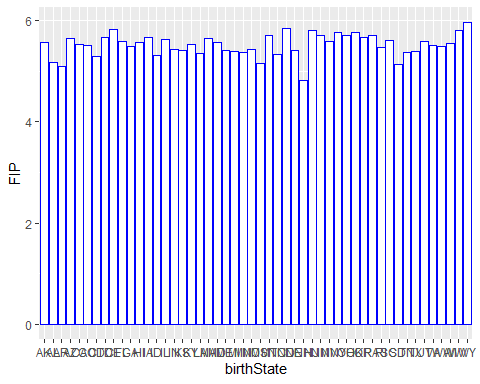
The best model for both explaining and predicting the data is the SVR. For explaining the data, the SVR has an R^2 value of 0.879 and an RMSE value of 0.387. For predicting the data, the SVR has an R^2 value of 0.866 and an RMSE value of 0.423.

## FIP vs birthState

For this question, we wanted to determine whether the age of pitchers is a good predictor of their future performance.

## Visualization

ggplot(data=stateavgFIP, aes(x=birthState, y=FIP)) + geom\_bar(stat="identity", color="blue", fill="white")



There is a negative correlation between the FIP and age which due to the nature of the FIP statistics in which a lower value indicates a better player means that pitchers are at their worst when first making it into the MLB and gradually get better as they age.

##models ##MLR

set.seed(123)  
training.samples <- stateFIP$FIP %>% createDataPartition(p = 0.8, list = FALSE)  
train.data <- stateFIP[training.samples, ]  
test.data <- stateFIP[-training.samples, ]

## SVR

SVR\_6\_FIP <- svm(FIP ~ G + PO + A + E + DP + W + L + CG + SHO + SV + HR + BB + SO + BAOpp + WP + HBP + GF + ERA + birthState, data = train.data)  
  
##Train Data  
predicted.classes <- predict(SVR\_6\_FIP, train.data)  
RMSE(predicted.classes, train.data$FIP)

## [1] 0.3876918

R2(predicted.classes, train.data$FIP)

## [1] 0.8794718

##Test Data  
predicted.classes <- predict(SVR\_6\_FIP, test.data)  
  
RMSE(predicted.classes, test.data$FIP)

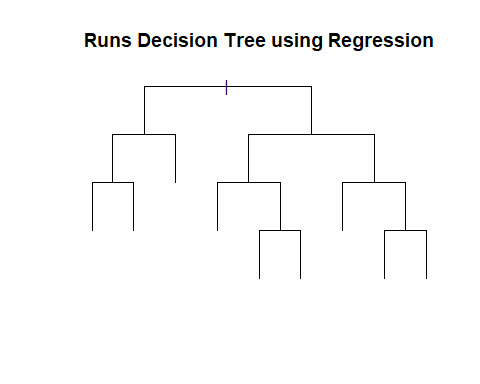
## [1] 0.4231111

R2(predicted.classes, test.data$FIP)

## [1] 0.8665801

## DT

DT\_6\_FIP <- rpart(FIP ~ G + PO + A + E + DP + W + L + CG + SHO + SV + HR + BB + SO + BAOpp + WP + HBP + GF + ERA + birthState, data = train.data, method = "anova")  
  
plot(DT\_6\_FIP, uniform = TRUE,  
 main = "Runs Decision Tree using Regression")



summary(DT\_6\_FIP)

## Call:  
## rpart(formula = FIP ~ G + PO + A + E + DP + W + L + CG + SHO +   
## SV + HR + BB + SO + BAOpp + WP + HBP + GF + ERA + birthState,   
## data = train.data, method = "anova")  
## n= 3585   
##   
## CP nsplit rel error xerror xstd  
## 1 0.33986556 0 1.0000000 1.0004291 0.02747026  
## 2 0.08083038 1 0.6601344 0.6672086 0.01852510  
## 3 0.06304195 2 0.5793041 0.5978074 0.01745691  
## 4 0.01944257 3 0.5162621 0.5258767 0.01481009  
## 5 0.01218607 5 0.4773770 0.4876143 0.01269581  
## 6 0.01179991 6 0.4651909 0.4728495 0.01232783  
## 7 0.01000000 8 0.4415911 0.4578938 0.01182348  
##   
## Variable importance  
## ERA BAOpp SO G HR W BB A L SV GF   
## 51 15 10 10 5 4 2 1 1 1 1   
##   
## Node number 1: 3585 observations, complexity param=0.3398656  
## mean=5.515043, MSE=1.196697   
## left son=2 (2099 obs) right son=3 (1486 obs)  
## Primary splits:  
## ERA < 4.583737 to the left, improve=0.33986560, (0 missing)  
## G < 27.5 to the right, improve=0.11008630, (0 missing)  
## BAOpp < 0.2595 to the left, improve=0.10545340, (0 missing)  
## SO < 39.5 to the right, improve=0.10014830, (0 missing)  
## HR < 4.5 to the left, improve=0.08725566, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.2795 to the left, agree=0.735, adj=0.361, (0 split)  
## G < 27.5 to the right, agree=0.679, adj=0.225, (0 split)  
## SO < 35.5 to the right, agree=0.637, adj=0.124, (0 split)  
## W < 1.5 to the right, agree=0.618, adj=0.079, (0 split)  
## HR < 25.5 to the left, agree=0.594, adj=0.021, (0 split)  
##   
## Node number 2: 2099 observations, complexity param=0.08083038  
## mean=4.978445, MSE=0.7008803   
## left son=4 (772 obs) right son=5 (1327 obs)  
## Primary splits:  
## ERA < 3.332624 to the left, improve=0.23571740, (0 missing)  
## HR < 5.5 to the left, improve=0.10478320, (0 missing)  
## BAOpp < 0.2405 to the left, improve=0.05714446, (0 missing)  
## SV < 33.5 to the right, improve=0.05543970, (0 missing)  
## GF < 45.5 to the right, improve=0.04834079, (0 missing)  
## Surrogate splits:  
## BAOpp < 0.2255 to the left, agree=0.711, adj=0.215, (0 split)  
## HR < 3.5 to the left, agree=0.675, adj=0.115, (0 split)  
## SV < 1.5 to the right, agree=0.660, adj=0.075, (0 split)  
## GF < 40.5 to the right, agree=0.656, adj=0.063, (0 split)  
## G < 60.5 to the right, agree=0.647, adj=0.041, (0 split)  
##   
## Node number 3: 1486 observations, complexity param=0.06304195  
## mean=6.272996, MSE=0.915837   
## left son=6 (1044 obs) right son=7 (442 obs)  
## Primary splits:  
## ERA < 6.121856 to the left, improve=0.19873120, (0 missing)  
## SO < 21.5 to the right, improve=0.09660637, (0 missing)  
## G < 24.5 to the right, improve=0.06634532, (0 missing)  
## HR < 3.5 to the left, improve=0.06503480, (0 missing)  
## W < 2.5 to the right, improve=0.04338431, (0 missing)  
## Surrogate splits:  
## SO < 18.5 to the right, agree=0.732, adj=0.100, (0 split)  
## G < 12.5 to the right, agree=0.725, adj=0.075, (0 split)  
## W < 0.5 to the right, agree=0.717, adj=0.050, (0 split)  
## BAOpp < 0.3215 to the left, agree=0.717, adj=0.048, (0 split)  
## BB < 7.5 to the right, agree=0.707, adj=0.014, (0 split)  
##   
## Node number 4: 772 observations, complexity param=0.01218607  
## mean=4.445546, MSE=0.6044628   
## left son=8 (363 obs) right son=9 (409 obs)  
## Primary splits:  
## SO < 64.5 to the right, improve=0.11203400, (0 missing)  
## ERA < 2.352742 to the left, improve=0.09826156, (0 missing)  
## SV < 21.5 to the right, improve=0.07056488, (0 missing)  
## GF < 41.5 to the right, improve=0.06351264, (0 missing)  
## birthState splits as LLLRLLLLRRL-LLRLLLLLRLLLLRRLLLLRLLRRRLRRRLLLLLLRL-, improve=0.05462448, (0 missing)  
## Surrogate splits:  
## BB < 24.5 to the right, agree=0.786, adj=0.545, (0 split)  
## W < 6.5 to the right, agree=0.767, adj=0.504, (0 split)  
## HR < 5.5 to the right, agree=0.740, adj=0.446, (0 split)  
## A < 12.5 to the right, agree=0.732, adj=0.430, (0 split)  
## L < 3.5 to the right, agree=0.731, adj=0.427, (0 split)  
##   
## Node number 5: 1327 observations  
## mean=5.288466, MSE=0.4956499   
##   
## Node number 6: 1044 observations, complexity param=0.01179991  
## mean=5.995406, MSE=0.5719095   
## left son=12 (105 obs) right son=13 (939 obs)  
## Primary splits:  
## HR < 3.5 to the left, improve=0.07261214, (0 missing)  
## ERA < 5.36297 to the left, improve=0.05045604, (0 missing)  
## SO < 20.5 to the right, improve=0.04159446, (0 missing)  
## birthState splits as LLLLRLLRLRRLLLRRRLRRRRRLLL-LRRLLRRRRRRRLLRRRLRLRL-, improve=0.03233708, (0 missing)  
## G < 53.5 to the right, improve=0.02911562, (0 missing)  
## Surrogate splits:  
## BB < 7.5 to the left, agree=0.904, adj=0.048, (0 split)  
## G < 4.5 to the left, agree=0.902, adj=0.029, (0 split)  
## birthState splits as RRRRRRRRRRRRRRRRRRRRRRRRRR-RRRRRRRRRRRRRRRRRLRRRR-, agree=0.901, adj=0.019, (0 split)  
##   
## Node number 7: 442 observations, complexity param=0.01944257  
## mean=6.92866, MSE=1.11629   
## left son=14 (104 obs) right son=15 (338 obs)  
## Primary splits:  
## HR < 4.5 to the left, improve=0.16527410, (0 missing)  
## ERA < 7.367532 to the left, improve=0.09998837, (0 missing)  
## SO < 17.5 to the right, improve=0.06582494, (0 missing)  
## G < 8.5 to the right, improve=0.04817233, (0 missing)  
## birthState splits as LRLLLRR-RRLRRRLRRRRLRLLLRL-L-LRRLLRLRLLRLLLLLRRLL-, improve=0.04652759, (0 missing)  
## Surrogate splits:  
## SO < 11.5 to the left, agree=0.774, adj=0.038, (0 split)  
## A < 1.5 to the left, agree=0.769, adj=0.019, (0 split)  
## BB < 4.5 to the left, agree=0.767, adj=0.010, (0 split)  
##   
## Node number 8: 363 observations  
## mean=4.169318, MSE=0.4502488   
##   
## Node number 9: 409 observations  
## mean=4.690707, MSE=0.613508   
##   
## Node number 12: 105 observations  
## mean=5.386001, MSE=0.5823869   
##   
## Node number 13: 939 observations, complexity param=0.01179991  
## mean=6.063551, MSE=0.5245666   
## left son=26 (825 obs) right son=27 (114 obs)  
## Primary splits:  
## SO < 26.5 to the right, improve=0.11753140, (0 missing)  
## A < 3.5 to the right, improve=0.06348392, (0 missing)  
## ERA < 5.36297 to the left, improve=0.05726918, (0 missing)  
## G < 20.5 to the right, improve=0.04027221, (0 missing)  
## W < 1.5 to the right, improve=0.03808870, (0 missing)  
## Surrogate splits:  
## BB < 13.5 to the right, agree=0.913, adj=0.281, (0 split)  
## G < 9.5 to the right, agree=0.911, adj=0.263, (0 split)  
## A < 1.5 to the right, agree=0.887, adj=0.070, (0 split)  
## W < 0.5 to the right, agree=0.881, adj=0.018, (0 split)  
## birthState splits as LLLLLLLRLLLLLLLLLLLLLLLLLL-LLLLLRLLLLLLLLLLLLLLLL-, agree=0.881, adj=0.018, (0 split)  
##   
## Node number 14: 104 observations  
## mean=6.154318, MSE=0.8776785   
##   
## Node number 15: 338 observations, complexity param=0.01944257  
## mean=7.166919, MSE=0.9484482   
## left son=30 (236 obs) right son=31 (102 obs)  
## Primary splits:  
## SO < 22.5 to the right, improve=0.26601270, (0 missing)  
## ERA < 7.08927 to the left, improve=0.18726940, (0 missing)  
## A < 6.5 to the right, improve=0.11281010, (0 missing)  
## G < 7.5 to the right, improve=0.09028590, (0 missing)  
## birthState splits as LRLLRRR-RRLRRRRRRRRRRLLLRR-R-LLRLLRLLLLRRLRLLRRLL-, improve=0.06591158, (0 missing)  
## Surrogate splits:  
## BB < 16.5 to the right, agree=0.811, adj=0.373, (0 split)  
## G < 8.5 to the right, agree=0.784, adj=0.284, (0 split)  
## L < 2.5 to the right, agree=0.784, adj=0.284, (0 split)  
## A < 3.5 to the right, agree=0.775, adj=0.255, (0 split)  
## ERA < 7.806693 to the left, agree=0.754, adj=0.186, (0 split)  
##   
## Node number 26: 825 observations  
## mean=5.971251, MSE=0.4412677   
##   
## Node number 27: 114 observations  
## mean=6.731513, MSE=0.6195616   
##   
## Node number 30: 236 observations  
## mean=6.8367, MSE=0.6233214   
##   
## Node number 31: 102 observations  
## mean=7.930955, MSE=0.8646518

##train  
predictedR3 <- predict(DT\_6\_FIP, train.data, method = "anova")  
RMSE(predictedR3, train.data$FIP)

## [1] 0.7269462

R2(predictedR3, train.data$FIP)

## [1] 0.5584089

##test  
predictedR3 <- predict(DT\_6\_FIP, test.data, method = "anova")  
RMSE(predictedR3, test.data$FIP)

## [1] 0.7446652

R2(predictedR3, test.data$FIP)

## [1] 0.5595935

## Conclusion

The best model for both explaining and predicting the data is the Multiple Linear Regression. For explaining the data, the Multiple Linear Regression has an R^2 value of 0.659 and an RMSE value of 0.590. For predicting the data, the Multiple Linear Regression has an R^2 value of 0.655 and an RMSE value of 0.597

## Question 4 What are the flaws of traditional baseball statistics?

Note: (Shyan didn’t show or give us his updated code nor did he write this section, which he worked on alone)

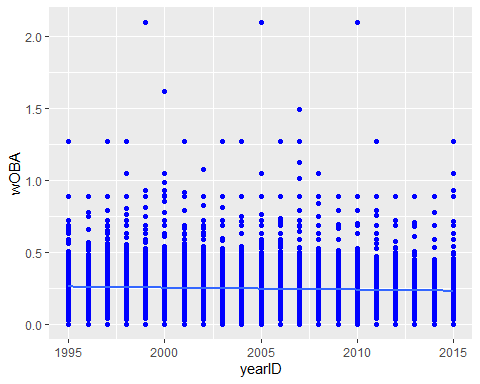
## Visualization

ggplot(data = Batting3, mapping = aes(x = yearID, y = wOBA)) + geom\_point(color = "blue") + geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 6273 rows containing non-finite values (stat\_smooth).

## Warning: Removed 6273 rows containing missing values (geom\_point).

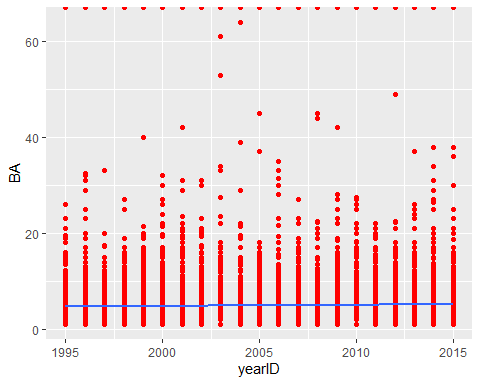


ggplot(data = Batting3, mapping = aes(x = yearID, y = BA)) + geom\_point(color = "red") + geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 9998 rows containing non-finite values (stat\_smooth).

## Warning: Removed 6434 rows containing missing values (geom\_point).

 There is much of a noticeable difference in both variation of wOBA and BA. ## Modeling

#Linear regression  
  
Batting3[is.na(Batting3)] = 0  
Batting3$BA[is.infinite(Batting$BA)] = 0

## Warning: Unknown or uninitialised column: `BA`.

Batting3$wOBA[is.infinite(Batting$wOBA)] = 0

## Warning: Unknown or uninitialised column: `wOBA`.

##model <-lm(wOBA ~ BA, data = Batting3)  
##summary(model)  
  
##view(Batting3)  
#Multiple Linear Regression  
##modelMult <- lm(wOBA ~ BA + G + AB + R + H + XBH + OBP + `2B` + `3B` + RBI + SB + CS + BB + SO + IBB+ HBP + SH, data = Batting3)  
##vif(modelMult)  
##summary(modelMult)

(Shyan did not give me the working code for this section)

## Question 5 - IS SALARY A GOOD PREDICTOR OF AWARDS WON?

## Transformation

I removed the tie and notes variables from the dataframes, then I removed the teamID column since I didn’t need it

# Remove tie and notes variables from AwardsPlayers dataframe  
Awards2 <- AwardsPlayers[ -c(5:6)]  
kable(Awards2[1:15, ], caption = "Awards2 df after removal and filter")

Awards2 df after removal and filter

|  |  |  |  |
| --- | --- | --- | --- |
| playerID | awardID | yearID | lgID |
| hershor01 | ALCS MVP | 1995 | AL |
| coninje01 | All-Star Game MVP | 1995 | ML |
| glavito02 | Babe Ruth Award | 1995 | NL |
| gwynnto01 | Branch Rickey Award | 1995 | ML |
| johnsra05 | Cy Young Award | 1995 | AL |
| maddugr01 | Cy Young Award | 1995 | NL |
| snowjt01 | Gold Glove | 1995 | AL |
| alomaro01 | Gold Glove | 1995 | AL |
| boggswa01 | Gold Glove | 1995 | AL |
| rodriiv01 | Gold Glove | 1995 | AL |
| griffke02 | Gold Glove | 1995 | AL |
| loftoke01 | Gold Glove | 1995 | AL |
| whitede03 | Gold Glove | 1995 | AL |
| langsma01 | Gold Glove | 1995 | AL |
| vizquom01 | Gold Glove | 1995 | AL |

Salaries2 <- Salaries[ -c(2)]  
kable(Salaries2[1:15, ], caption = "Salaries2 df after removal and filter")

Salaries2 df after removal and filter

|  |  |  |  |
| --- | --- | --- | --- |
| yearID | lgID | playerID | salary |
| 1995 | NL | averyst01 | 4000000 |
| 1995 | NL | bedrost01 | 750000 |
| 1995 | NL | bellira01 | 550000 |
| 1995 | NL | blausje01 | 3420000 |
| 1995 | NL | borbope02 | 111500 |
| 1995 | NL | clontbr01 | 109000 |
| 1995 | NL | glavito02 | 4750000 |
| 1995 | NL | grissma02 | 4950000 |
| 1995 | NL | jonesch06 | 114000 |
| 1995 | NL | justida01 | 5200000 |
| 1995 | NL | kellymi02 | 111500 |
| 1995 | NL | kleskry01 | 190000 |
| 1995 | NL | lemkema01 | 1250000 |
| 1995 | NL | lopezja01 | 150000 |
| 1995 | NL | maddugr01 | 6500000 |

From here, I joined the two dataframes using a left join by their key and omitted any na values.

AwardedSalaries <- left\_join(Salaries2, Awards2, by = c("playerID","yearID"))  
AwardedSalaries <- na.omit(AwardedSalaries)  
kable(AwardedSalaries[1:15, ], caption = "Dataframe created from Left Join")

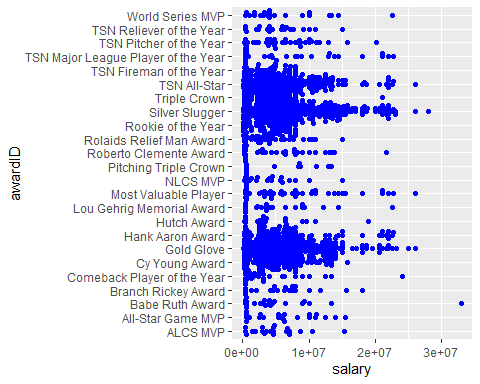
Dataframe created from Left Join

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| yearID | lgID.x | playerID | salary | awardID | lgID.y |
| 1995 | NL | glavito02 | 4750000 | Babe Ruth Award | NL |
| 1995 | NL | glavito02 | 4750000 | Silver Slugger | NL |
| 1995 | NL | glavito02 | 4750000 | World Series MVP | ML |
| 1995 | NL | grissma02 | 4950000 | Gold Glove | NL |
| 1995 | NL | maddugr01 | 6500000 | Cy Young Award | NL |
| 1995 | NL | maddugr01 | 6500000 | Gold Glove | NL |
| 1995 | NL | maddugr01 | 6500000 | TSN All-Star | NL |
| 1995 | NL | maddugr01 | 6500000 | TSN Pitcher of the Year | NL |
| 1995 | AL | mussimi01 | 2925000 | TSN All-Star | AL |
| 1995 | AL | ripkeca01 | 6700000 | TSN All-Star | AL |
| 1995 | AL | valenjo02 | 637500 | Silver Slugger | AL |
| 1995 | AL | vaughmo01 | 2775000 | Most Valuable Player | AL |
| 1995 | AL | vaughmo01 | 2775000 | Silver Slugger | AL |
| 1995 | AL | vaughmo01 | 2775000 | TSN All-Star | AL |
| 1995 | AL | edmonji01 | 167500 | TSN All-Star | AL |

## Visualization

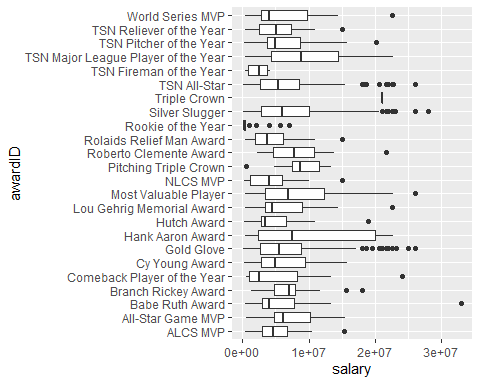
To find any correlation among this data, I created a scatterplot visualization of the AwardedSalaries dataframe.

ggplot(AwardedSalaries,aes(awardID, salary)) +   
 geom\_beeswarm(color = "blue") + coord\_flip()



For more exploration on this question, I created a boxplot as well.

ggplot(AwardedSalaries,aes(awardID, salary)) + geom\_boxplot() + coord\_flip()



## Testing for Correlation b/w Salaries and Awards

Furthermore, I conducted a test to compare the variance of Salaries for each Award to the overall variance of the AwardedSalaries dataframe. I broke the main dataframe into smaller ones and computed the variance for each, then composed them in a list.

# Filter every award into its own variable  
ALCSMVP <- AwardedSalaries %>% filter(awardID == "ALCS MVP")  
AllStarMVP <- AwardedSalaries %>% filter(awardID == "All-Star Game MVP")  
BabeRuthAward <- AwardedSalaries %>% filter(awardID == "Babe Ruth Award")  
ComebackPlayer <- AwardedSalaries %>% filter(awardID == "Comeback Player of the Year")  
CyYoungAward <- AwardedSalaries %>% filter(awardID == "Cy Young Award")  
GoldGloveAward <- AwardedSalaries %>% filter(awardID == "Gold Glove")  
HankAaronAward <- AwardedSalaries %>% filter(awardID == "Hank Aaron Award")  
MVPAward <- AwardedSalaries %>% filter(awardID == "Most Valuable Player")  
NLCSMVP <- AwardedSalaries %>% filter(awardID == "NLCS MVP")  
PTC <- AwardedSalaries %>% filter(awardID == "Pitching Triple Crown")  
RobertoAward <- AwardedSalaries %>% filter(awardID == "Roberto Clemente Award")  
RolaidsRelief <- AwardedSalaries %>% filter(awardID == "Rolaids Relief Man Award")  
RookieOfYear <- AwardedSalaries %>% filter(awardID == "Rookie of the Year")  
SilverSlugger <- AwardedSalaries %>% filter(awardID == "Silver Slugger")  
TripleCrown <- AwardedSalaries %>% filter(awardID == "Triple Crown")  
TSN\_AllStar <- AwardedSalaries %>% filter(awardID == "TSN All-Star")  
TSN\_Fireman <- AwardedSalaries %>% filter(awardID == "TSN Fireman of the Year")  
TSN\_Pitcher <- AwardedSalaries %>% filter(awardID == "TSN Pitcher of the Year")  
TSN\_Reliever <- AwardedSalaries %>% filter(awardID == "TSN Reliever of the Year")  
  
# Determine variance in salaries of each award  
ALCSMVPv = ALCSMVP$salary  
ALCSMVP\_Variance = var(ALCSMVPv)  
  
ALLStarMVPv = AllStarMVP$salary  
AllStarMVP\_Variance = var(ALLStarMVPv)  
  
BabeRuthAwardv = BabeRuthAward$salary  
BabeRuthAward\_Variance = var(BabeRuthAwardv)  
  
ComebackPlayerv = ComebackPlayer$salary  
ComebackPlayer\_Variance = var(ComebackPlayerv)  
  
CyYoungAwardv = CyYoungAward$salary  
CyYoungAward\_Variance = var(CyYoungAwardv)  
  
GoldGloveAwardv = GoldGloveAward$salary  
GoldGloveAward\_Variance = var(GoldGloveAwardv)  
  
HankAaronAwardv = HankAaronAward$salary  
HankAaronAward\_Variance = var(HankAaronAwardv)  
  
MVPAwardv = MVPAward$salary  
MVPAward\_Variance = var(MVPAwardv)  
  
NLCSMVPv = NLCSMVP$salary  
NLCSMVP\_Variance = var(NLCSMVPv)  
  
PTCv = PTC$salary  
PTC\_Variance = var(PTCv)  
  
RobertoAwardv = RobertoAward$salary  
RobertoAward\_Variance = var(RobertoAwardv)  
  
RolaidsReliefv = RolaidsRelief$salary  
RolaidsRelief\_Variance = var(RolaidsReliefv)  
  
RookieOfYearv = RookieOfYear$salary  
RookieOfYear\_Variance = var(RookieOfYearv)  
  
SilverSluggerv = SilverSlugger$salary  
SilverSlugger\_Variance = var(SilverSluggerv)  
  
TripleCrownv = TripleCrown$salary  
TripleCrown\_Variance = var(TripleCrownv)  
  
TSN\_AllStarv = TSN\_AllStar$salary  
TSN\_AllStar\_Variance = var(TSN\_AllStarv)  
  
TSN\_Firemanv = TSN\_Fireman$salary  
TSN\_Fireman\_Variance = var(TSN\_Firemanv)  
  
TSN\_Pitcherv = TSN\_Pitcher$salary  
TSN\_Pitcher\_Variance = var(TSN\_Pitcherv)  
  
TSN\_Relieverv = TSN\_Reliever$salary  
TSN\_Reliever\_Variance = var(TSN\_Relieverv)  
  
# Create a table compiling salary variances by Award  
Awards <- c("ALCSMVP","AllStarMVP","BabeRuthAward","ComebackPlayer",  
 "CyYoungAward","GoldGloveAward","HankAaronAward","MVPAward",  
 "NLCSMVP","PTC","RobertoAward","RolaidsRelief","RookieOfTheYear",  
 "Silver Slugger","Triple Crown","TSN All-Star","TSN Fireman",  
 "TSN Pitcher","TSN Reliever")  
  
SalaryVariance <- c(ALCSMVP\_Variance, AllStarMVP\_Variance, BabeRuthAward\_Variance,   
 ComebackPlayer\_Variance, CyYoungAward\_Variance,   
 GoldGloveAward\_Variance, HankAaronAward\_Variance, MVPAward\_Variance,   
 NLCSMVP\_Variance, PTC\_Variance, RobertoAward\_Variance, RolaidsRelief\_Variance,   
 RookieOfYear\_Variance, SilverSlugger\_Variance, TripleCrown\_Variance,   
 TSN\_AllStar\_Variance, TSN\_Fireman\_Variance, TSN\_Pitcher\_Variance,   
 TSN\_Reliever\_Variance)  
  
AwSalVar <- data.frame(Awards, SalaryVariance, stringsAsFactors = FALSE)  
  
kable(AwSalVar, caption = "Salary variance list by award")

Salary variance list by award

|  |  |
| --- | --- |
| Awards | SalaryVariance |
| ALCSMVP | 1.310385e+13 |
| AllStarMVP | 1.812633e+13 |
| BabeRuthAward | 5.421356e+13 |
| ComebackPlayer | 3.661157e+13 |
| CyYoungAward | 1.907542e+13 |
| GoldGloveAward | 2.613429e+13 |
| HankAaronAward | 7.151830e+13 |
| MVPAward | 4.778330e+13 |
| NLCSMVP | 1.463927e+13 |
| PTC | 1.819067e+13 |
| RobertoAward | 2.245915e+13 |
| RolaidsRelief | 1.217345e+13 |
| RookieOfTheYear | 2.812859e+12 |
| Silver Slugger | 3.387276e+13 |
| Triple Crown | NA |
| TSN All-Star | 2.518628e+13 |
| TSN Fireman | 2.326122e+12 |
| TSN Pitcher | 2.003707e+13 |
| TSN Reliever | 1.466733e+13 |

Finally, I computed the overall variance of the AwardedSalaries dataframe and compared it to the individual variances of each award to reach the conclusion that Salary is not a good predictor of awards won.

# Compute the variance of AwardedSalaries  
AwardedSalariesv = AwardedSalaries$salary  
AwardedSalaries\_Variance = var(AwardedSalariesv)  
  
kable(AwardedSalaries\_Variance, caption = "Variance of AwardedSalaries dataframe")

Variance of AwardedSalaries dataframe

|  |
| --- |
| x |
| 2.999782e+13 |

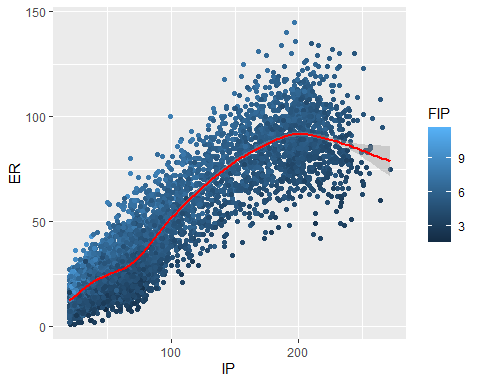
## Question 6 Predicting Earned Runs (ER)

For Question 6, we wanted to figure out how to predict the number of earned runs (ER) a pitcher may allow throughout a season.

## Visualization

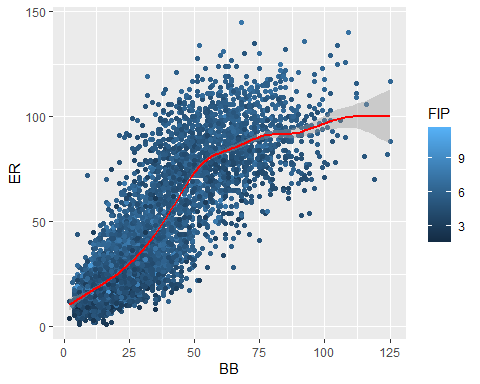
ggplot(data = pitcher\_stats\_complete, mapping = aes(x = IP, y = ER, color = FIP)) + geom\_point() + geom\_smooth(color = "red")

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



ggplot(data = pitcher\_stats\_complete, mapping = aes(x = BB, y = ER, color = FIP)) + geom\_point() + geom\_smooth(color = "red")

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



The first thing we noticed was that the number of runners on base and the number of innings a pitcher throws has a positive correlation with earned runners. In both graphs there is a strong positive correlation indicating a relationship.

## partitioning the data

ER.ts <- pitcher\_stats\_complete$ER %>% createDataPartition(p = .8, list = FALSE)  
ER.train <- pitcher\_stats\_complete[ER.ts,]  
ER.test <- pitcher\_stats\_complete[-ER.ts,]

## Training

## MLR

modelMLRER <- lm(ER ~ PO + A + CG + HR + BB + BAOpp + WP + HBP + SO,data = ER.train)  
vif(modelMLRER)

## PO A CG HR BB BAOpp WP HBP   
## 2.249864 2.893556 1.740708 2.721431 3.967850 1.039823 1.626087 1.707632   
## SO   
## 3.724320

summary(modelMLRER)

##   
## Call:  
## lm(formula = ER ~ PO + A + CG + HR + BB + BAOpp + WP + HBP +   
## SO, data = ER.train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -39.544 -5.131 -0.540 4.680 46.651   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.425436 0.380622 -6.372 2.04e-10 \*\*\*  
## PO 0.381865 0.036208 10.546 < 2e-16 \*\*\*  
## A 0.459005 0.021922 20.938 < 2e-16 \*\*\*  
## CG 0.650700 0.152305 4.272 1.97e-05 \*\*\*  
## HR 2.030204 0.025630 79.211 < 2e-16 \*\*\*  
## BB 0.415498 0.011814 35.169 < 2e-16 \*\*\*  
## BAOpp 11.903316 1.008835 11.799 < 2e-16 \*\*\*  
## WP 0.238404 0.054176 4.401 1.10e-05 \*\*\*  
## HBP 0.591204 0.052761 11.205 < 2e-16 \*\*\*  
## SO -0.048012 0.004879 -9.840 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 8.591 on 4723 degrees of freedom  
## Multiple R-squared: 0.9158, Adjusted R-squared: 0.9156   
## F-statistic: 5706 on 9 and 4723 DF, p-value: < 2.2e-16

predictER <- predict(modelMLRER, ER.train)  
RMSE(predictER, ER.train$ER)

## [1] 8.58236

R2(predictER, ER.train$ER)

## [1] 0.9157753

## SVR

modelSVRER <- svm(ER ~ PO + A + DP + L + CG + HR + BB + BAOpp + WP + HBP,data = ER.train)  
predictedER <- predict(modelSVRER, ER.train)  
RMSE(predictedER, ER.train$ER)

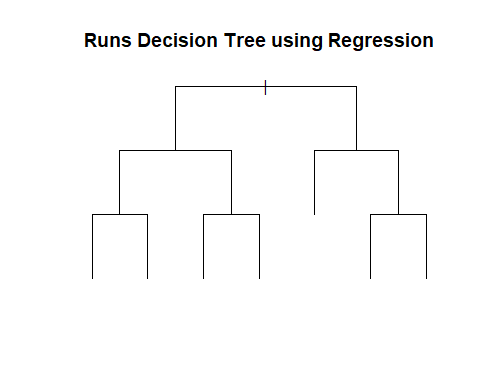
## [1] 6.148831

R2(predictedER, ER.train$ER)

## [1] 0.9569345

## DT

fitER <- rpart( ER ~ PO + A + DP + L + CG + HR + BB + BAOpp + WP + HBP, data = ER.train, method = "anova")  
plot(fitER, uniform = TRUE,  
 main = "Runs Decision Tree using Regression")



dev.off()

## null device   
## 1

print(fitER)

## n= 4733   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 4733 4139145.00 43.41242   
## 2) HR< 13.5 3341 683349.60 27.35768   
## 4) HR< 7.5 2285 182951.70 21.41751   
## 8) BB< 20.5 1256 59476.54 17.33678 \*  
## 9) BB>=20.5 1029 77030.64 26.39845 \*  
## 5) HR>=7.5 1056 245305.90 40.21117   
## 10) BB< 48.5 921 125130.50 36.57872 \*  
## 11) BB>=48.5 135 25116.99 64.99259 \*  
## 3) HR>=13.5 1392 527731.00 81.94612   
## 6) L< 8.5 521 143441.80 67.47217 \*  
## 7) L>=8.5 871 209854.30 90.60390   
## 14) HR< 21.5 415 75936.85 83.37108 \*  
## 15) HR>=21.5 456 92449.16 97.18640 \*

predictedER2 <- predict(fitER, ER.train, method = "anova")  
RMSE(predictedER2, ER.train$ER)

## [1] 11.24589

R2(predictedER2, ER.train$ER)

## [1] 0.855385

## Testing

## MLR

predictER4 <- predict(modelMLRER, ER.test)  
RMSE(predictER4, ER.test$ER)

## [1] 8.902452

R2(predictER4, ER.test$ER)

## [1] 0.9061605

## SVR

predictedER5 <- predict(modelSVRER,ER.test)  
RMSE(predictedER5, ER.test$ER)

## [1] 7.616933

R2(predictedER5, ER.test$ER)

## [1] 0.9314181

## DT

predictedER6 <- predict(fitER, ER.test, method = "anova")  
RMSE(predictedER6, ER.test$ER)

## [1] 12.08987

R2(predictedER6, ER.test$ER)

## [1] 0.8270685

For predicting earned runs the variables PO, A, DP , L , CG , HR , BB , BAOpp, WP, and HBP; with BAOpp having the highest slope of 11.619722. Three models were made for both training and test data; with SVR having the best explanatory and predictive models compared to the multiple linear regression and the decision tree. For explanatory, SVR has the highest R-squared of 0.956943 and lowest RMSE of 6.133889; and for predicting an R-Squared of 0.9288055 and RMSE of 7.815366 making it the best model for predicting earned runs.

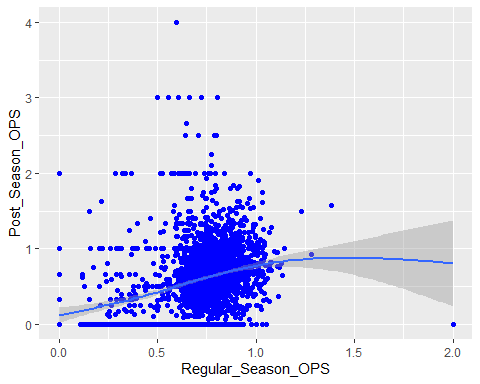
## Question 7 Regular Season OPS Vs Post Season OPS

For question seven, we wanted to look into predicting postseason OPS by using regular season statistics.

## Visualization

ggplot(data = RvPOPS, mapping = aes(y = Post\_Season\_OPS, x = Regular\_Season\_OPS)) + geom\_point(color = "blue") +geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



Clearly there is a slight positive correlation between Regular and Postseason OPS, indicating that a relationship is there. However, noticeably there is a lot of variation.

## correlation test

cor(RvPOPS$Regular\_Season\_OPS, RvPOPS$Post\_Season\_OPS)

## [1] 0.2923308

The correlation is small indicating a weak positive correlation of 0.1977545.

## MLR

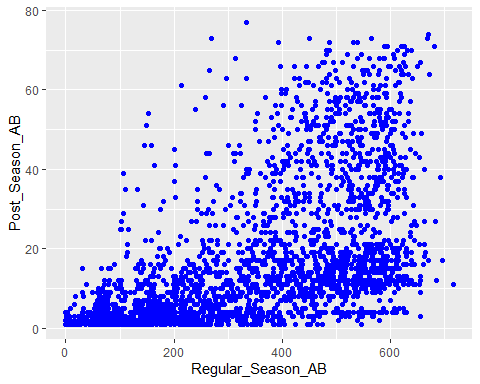
model\_PVRP <- lm(Post\_Season\_OPS ~ Regular\_Season\_OPS, data = RvPOPS)  
summary(model\_PVRP)

##   
## Call:  
## lm(formula = Post\_Season\_OPS ~ Regular\_Season\_OPS, data = RvPOPS)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.4681 -0.2828 -0.0277 0.2149 3.4908   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.10046 0.03347 3.001 0.00272 \*\*   
## Regular\_Season\_OPS 0.68382 0.04456 15.345 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4329 on 2520 degrees of freedom  
## Multiple R-squared: 0.08546, Adjusted R-squared: 0.08509   
## F-statistic: 235.5 on 1 and 2520 DF, p-value: < 2.2e-16

Since the multiple linear regression was at 0.03911, regular batting statistics are not a good predictor of postseason OPS.

## Investigating the low R-Squared Values

ggplot(data = RvPOPS, mapping=aes(x=Regular\_Season\_AB, y=Post\_Season\_AB))+geom\_point(color = "blue")



## Conclusion

When it comes to what makes a good baseball player there are a lot of factors. Is he a position player? Is he a pitcher? How does this player play? The amount of time that a player is on the field is the most important factor, with age, birth location, and salary not being a good predictor of talent. In reality, a player’s performance is directly linked to the amount of playing time and their skill; with better players getting much more out of their time. Players with less games played having less accurate reflections of their skills. This is a problem when predicting postseason performance, with at maximum a player playing around 15-20 in a given year. Which makes it very difficult to accurately predict future performances.