WAA Report

James Pinedo, II

3/8/2022

R Markdown

Overview

There was a common trope in the United States that is now beginning to fade. That trope is the idea of a father and son playing baseball together. Cut off from this underlying source of interest and passion, baseball too is now fading. Ironically, even while baseball is fading, the data analytics side of the sport is growing by leaps and bounds. It would not be an overstatement to posit that baseball is blazing a trail not just with sports data science but also from the point of view of how man is working to understand how each and every action can be measured and tabled for future analysis. Needless to say, the amount of data is daunting.

Although this is well and good, all that data does not help anyone like me one bit. I am a father with a young son, and I would like to play baseball with my son. I also appreciate, thanks to Harvard's generous set of courses on the subject, how data analysis is an incredibly practical tool for understanding patterns in every day life and helping to visualize what set of actions are conducive to successful outcomes. I would like to pair this understanding in a practical way to playing baseball with my son, but a little league field is not wired with radar guns or cameras, nor is a little tee-baller capable hitting the ball over the fence (a "home-run" - the best thing that a offensive player can accomplish) or striking someone out (the best thing that a defensive player can accomplish), and these two outcomes are the ones that advanced baseball statistics such as "WAR" most value. In fact, baseball on the play ground has a lot more in common with the teams that played in 1875 than the teams that are currently refusing to play right now (in case the reader is unaware, baseball games are being cancelled now as professional baseball players and team owners are fighting over wages).

The point of this project is to meld these two things: a desire to play baseball with my son and my new-found data science skills. Through this combination, I would like to help my son improve at baseball, perhaps even helping other young players as well. I will look at very simple counting stats, something that a dad in the stands or coach with access to a piece of paper and a pencil, can keep track of and then compile into a single statistic that weights each stat based upon the historical value that these simple stats have contributed to generating winning teams.

The Data Set I will look at every shred of available data, going back as far as 1871 and as recent as the past major league season, and I will look at as many different leagues as possible, including the famed "Negro Leagues" of old and leagues that are long defunct. This compilation of data will never truly be completed, as any team data is useful, and baseball has been played anywhere and everywhere. However, since I need to present my findings at some point, I opted to use all available team hitting (offensive) data that could be scraped from Stathead Baseball's website (stathead.com). This data was then compiled into a spreadsheet and is available for use on my project's github page, you can also access it on the Stathead site here - https://stathead.com/sharing/UqErJ

Key Steps Key steps that were performed on this data set were to first even out the data so that it could be compared on a average statistic per game basis. For example, modern Major League baseball plays a 162 game season. Most teams in this data set did not, so it was important to look at the averages of each stat on a per game basis. I first used linear regression to use these averaged-out statistics as predictors, and then used K Nearest Neighbors, and finally Random Forest.

Of these, the linear regression model was the most illuminating, for it showed how stats that are valued highly in the major leagues today - for example the walk (or “base-on-balls”, which is when the defensive team allows a hitter on the offensive team to reach first base without that hitter team actually hitting the ball), are not effective predictors of winning teams from a historical perspective. I will speak more on that in the methods/analysis section.

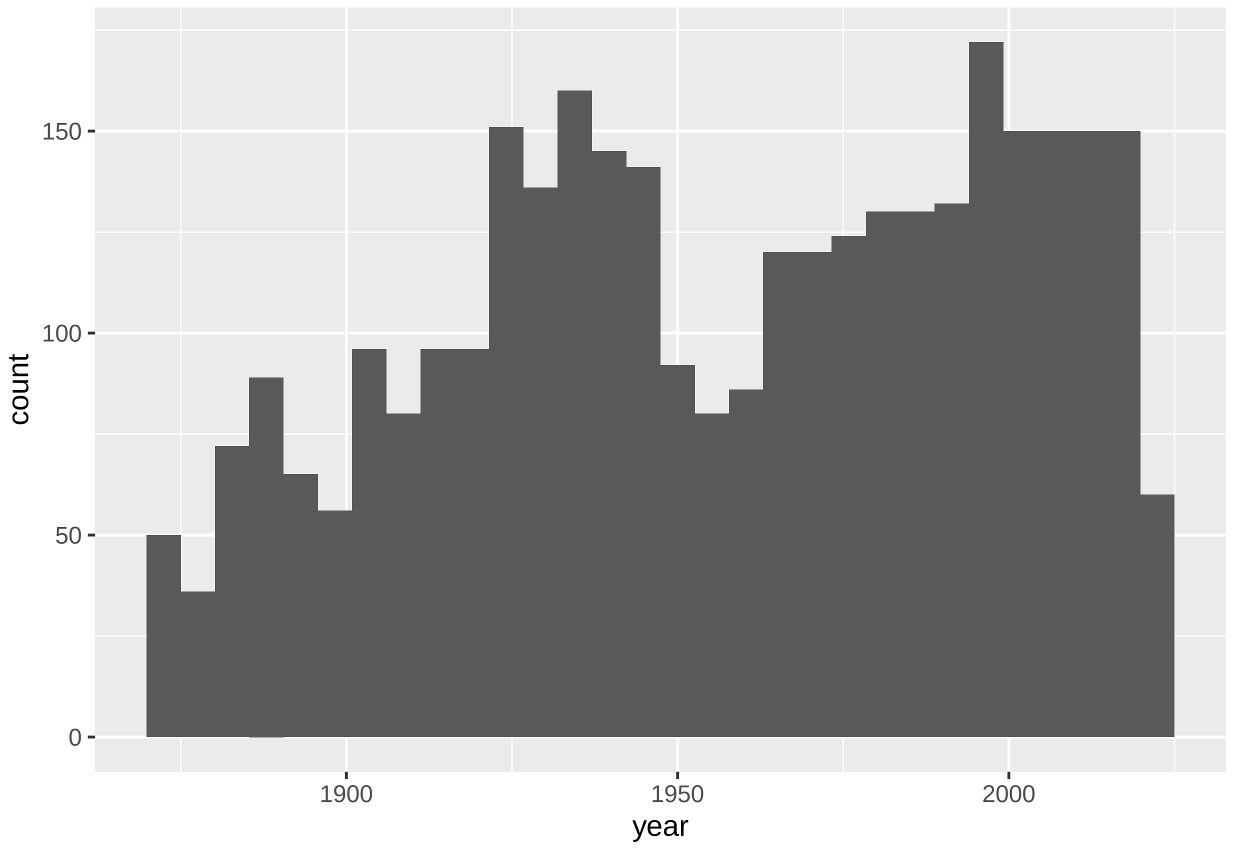
In brief, this study was able to use simple offensive stats to predict how successful (or not) a team would be, by predicting the “Wins Above Average” (WAA). Average wins were determined by taking the winning percentages of all the teams in the data set and averaging it out to equal a 162 game schedule. Using only basic hitting statistics, I was able to predict the correct WAA within an average error of 13.3 games given a 162 game season. This is an improvement of the standard error of 16 games should we guess the average at all times. I will speak more on how this could be improved in the conclusion.

Methods/Analysis

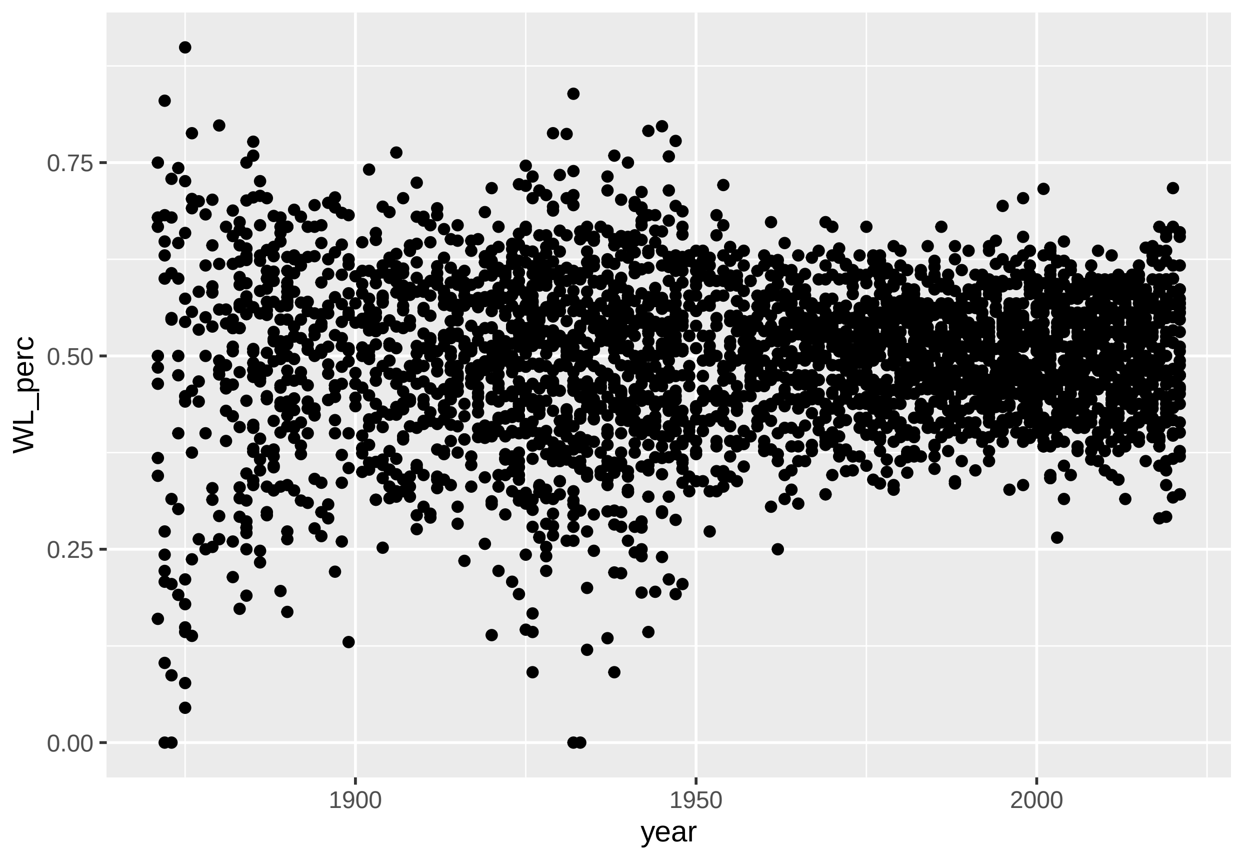
Before starting out on the methods used to start predicting, I will comment on the methods used for obtaining the data and then cleaning it for use in analysis. I relied on the data found on the Stathead Baseball site for this project. Unfortunately, the data there is split up into many pages, so I downloaded the data from each page and combined them into a spreadsheet. This spreadsheet was then uploaded to Github and then imported into my R project using the RCurl package.

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

#Loading in the data - all team hitting data available

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

waa_hitting <-
  read.csv(text=
             getURL("https://raw.githubusercontent.com/jamespinedoii/BaseballWAA/main/WAA%20Hitting%20Da

waa_hitting <- waa_hitting %>% mutate(year = Year) %>% select(-Year)

#Exploring the data a bit:
```

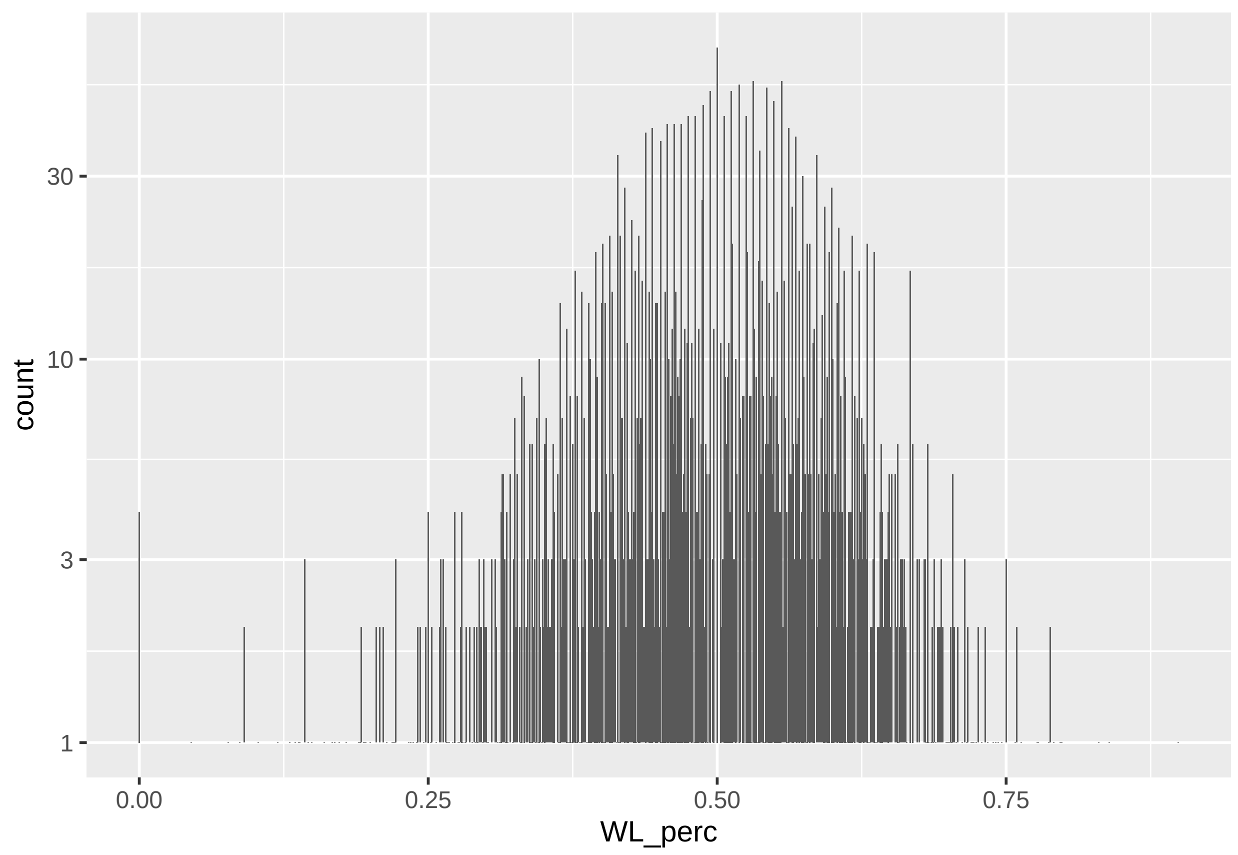
```
#Number of teams per year
waa_hitting %>%
  ggplot(aes(year))+
  geom_histogram()
```



```
#Data is sparser before 1900, which is expected.

#Year vs WL_perc
waa_hitting%>%
  ggplot(aes(year, WL_perc))+geom_point()
```



```r
#More parity as baseball becomes more modern

#WL_perc Data
waa_hitting %>%
  ggplot(aes(WL_perc)) +
  geom_bar() + scale_y_log10()
```



```
#Most teams are around 0.5 WL percentage
```

Now that the data was in the project, I found the average win-loss percentage ("WL_perc", the number of wins divided by number of games, an near universally tracked average that was already in the data set) for all teams, which was 0.4964048 (teams on average won about half their games, which makes perfect sense - every time one team wins, another has to lose).

```
ave_w_perc <- mean(waa_hitting$WL_perc)
ave_w_perc
```

```
## [1] 0.4964048
```

```
#shortening name
hitting <- waa_hitting
```

Then a column was added to show the wins above average percentage (WAAP) for each team. For example, if a team played 10 games and won seven of them, their WL_perc would be .7, thus their WAAP would be 0.2035952.

```
hitting <- hitting %>% mutate(WAAP = WL_perc - ave_w_perc)
```

To explore the range of data that we have here, I checked what the min and max WAAP was.

```
hitting <- hitting %>% mutate(WAAP = WL_perc - ave_w_perc)
which.max(hitting$WAAP)
```

```
## [1] 49
```

```
kable(hitting[49,] %>% select(year, Tm, W, L, WL_perc, G, X2B, HR, RBI, BB, BA, OBP, SLG,
                              WAAP))
```

	year	Tm	W	L	WL_perc	G	X2B	HR	RBI	BB	BA	OBP	SLG	WAAP
49	1875	BOS	71	8	0.899	82	167	15	565	33	0.321	0.327	0.41	0.4025952

```
which.min(hitting$WAAP)
```

```
## [1] 10
```

```
kable(hitting[10,] %>% select(year, Tm, W, L, WL_perc, G, X2B, HR, RBI, BB, BA, OBP, SLG,
                              WAAP))
```

	year	Tm	W	L	WL_perc	G	X2B	HR	RBI	BB	BA	OBP	SLG	WAAP
10	1872	NAT	0	11	0	11	6	0	50	1	0.22	0.221	0.237	-0.4964048

So average wins percentage is almost .5 and the best team was .4 above that. The worst team didn't win a game. That's quite a spread! Not surprisingly, the ends of the spectrum are found in shorter seasons from the 19th century, showing larger sample sizes serve to bring teams closer to a mean.

This WAAPe is a little difficult to visualize, so I opted to add another column to the data, one that took WAAP and multiplied it by 162, this showing what each team's WAA would be should they have played 162 games or "WAA_162". This is the column that we will be trying to predict moving forward, with using simple counting stats as predictors.

```
hitting <- hitting %>% mutate(WAA_162 = WAAP*162)

#for modern baseball fans, showing the most recent year
kable(hitting %>% filter(year == 2021) %>% select(year, Tm, WAA_162, W, L, WL_perc))
```

year	Tm	WAA_162	W	L	WL_perc
2021	CLE	-0.3895819	80	82	0.494
2021	TEX	-20.4775819	60	102	0.370
2021	CHC	-9.4615819	71	91	0.438
2021	DET	-3.4675819	77	85	0.475
2021	SEA	9.6544181	90	72	0.556
2021	LAA	-3.4675819	77	85	0.475
2021	MIA	-13.3495819	67	95	0.414
2021	BAL	-28.4155819	52	110	0.321
2021	KCR	-6.3835819	74	88	0.457
2021	TOR	10.6264181	91	71	0.562

year	Tm	WAA_162	W	L	WL_perc
2021	COL	-5.8975819	74	87	0.460
2021	ATL	8.1964181	88	73	0.547
2021	TBR	19.5364181	100	62	0.617
2021	MIN	-7.3555819	73	89	0.451
2021	OAK	5.6044181	86	76	0.531
2021	STL	9.6544181	90	72	0.556
2021	BOS	11.5984181	92	70	0.568
2021	NYM	-3.4675819	77	85	0.475
2021	NYY	11.5984181	92	70	0.568
2021	CIN	2.5264181	83	79	0.512
2021	MIL	14.5144181	95	67	0.586
2021	PHI	1.5544181	82	80	0.506
2021	SDP	-1.3615819	79	83	0.488
2021	SFG	26.5024181	107	55	0.660
2021	CHW	12.5704181	93	69	0.574
2021	PIT	-19.3435819	61	101	0.377
2021	ARI	-28.4155819	52	110	0.321
2021	HOU	14.5144181	95	67	0.586
2021	LAD	25.5304181	106	56	0.654
2021	WSN	-15.4555819	65	97	0.401

```
#Still the same best and worst teams

which.max(hitting$WAA_162)
```

```
## [1] 49
```

```
kable(hitting[49,] %>% select(year, Tm, W, L, WL_perc, G, X2B, HR, RBI, BB, BA, OBP, SLG,
                                WAAP))
```

	year	Tm	W	L	WL_perc	G	X2B	HR	RBI	BB	BA	OBP	SLG	WAAP
49	1875	BOS	71	8	0.899	82	167	15	565	33	0.321	0.327	0.41	0.4025952

```
which.min(hitting$WAA_162)
```

```
## [1] 10
```

```
kable(hitting[10,] %>% select(year, Tm, W, L, WL_perc, G, X2B, HR, RBI, BB, BA, OBP, SLG,
                                WAAP))
```

	year	Tm	W	L	WL_perc	G	X2B	HR	RBI	BB	BA	OBP	SLG	WAAP
10	1872	NAT	0	11	0	11	6	0	50	1	0.22	0.221	0.237	-0.4964048

```
top_teams <- hitting[order(hitting$WAA_162),]
#worst teams
kable(head(top_teams) %>% select(year, Tm, WAA_162))
```

	year	Tm	WAA_162
10	1872	NAT	-80.41758
25	1873	MAR	-80.41758
1048	1932	NWB	-80.41758
1076	1933	CG	-80.41758
48	1875	BRA	-73.12758
38	1875	WES	-67.94358

```r
#best teams
kable(top_n(top_teams, 6, WAA_162) %>% select(year, Tm, WAA_162))
```

year	Tm	WAA_162
1943	HG	47.72442
1945	CBE	48.69642
1880	CHC	48.85842
1872	BOS	54.04242
1932	DW	55.50042
1875	BOS	65.22042

A quick note on the predictors (you may skip this paragraph if you are already familiar with baseball), when playing baseball the way that a team wins is by scoring more "runs" than the other team. Each team has nine tries to score runs, and they switch off between being the offensive and defensive side throughout these nine tries (each set of tries by both teams is called an "inning"). While one team is on the offensive, they are the "hitting" team, as this is the time that they are attempting to hit the baseball from the starting position called "home plate". The defensive team is called the "fielding" team - this is when their player, called the "pitcher", is throwing the baseball - trying not to let the hitting team hit it but also needing to keep the ball within a narrowly defined area (called the "strike zone") over home plate. A baseball field is shaped in a diamond with four safe areas (or "bases") that a player on the hitting team my touch safely, and if the fielding team catches them off of this base they are called "out" and must leave the field. A run is scored every time the hitting team manages to safely run around the baseball diamond without being called out three times in one inning. The best chance that the hitting team has at running around these bases (starting from the home base [called "home plate"], to first base, second base, third base and then back to home) is by hitting the baseball where the fielders are not. This is called a "hit". Each hitters attempt to do this is called a "plate appearance" (PA). Should the hitter hit the ball where the fielders are not and make it all the way to second base, that is called a double. Should they make it to third base that is called a triple. Should they manage to run around all the bases in single hit that would be called a home run. There are other ways that a hitter may reach base, mostly due to the fielding team making a mistake, but for the purposes of this study, I will only explain only one of those - the hitter may reach base without hitting the ball when the opposing team's pitcher fails four times to throw the baseball within the strike zone. Should this happen, the batter is awarded with first base and this is counted as a "walk" or "base-on-balls". A hitter is credited for a "run-batted-in" if they hit the ball and allow one of their team mates that were already stationed on one of the bases to run home. All of these actions are counted and have been counted reliably since the 1870's and are of great use to this project.

Of course, I needed to average out these stats to reflect that most teams in the data set did not play a 162 game season. To that end, I divided the hits (H), doubles (X2B), triples (X3B), home runs (HR), runs batted in (RBI), and walks (BB) by the number of games that each team played. Easily tracked by any parent sitting the stands watching their child play, these averages will be used as predictors.

```
hitting <- hitting %>%
  mutate(h_g = H/G, x2b_g = X2B/G, x3b_g = X3B/G, hr_g = HR/G, rbi_g = RBI/G,
         bb_g = BB/G)
```

So will as the following averages: the average amount each hitter reached base per plate appearance (on-base percentage or OBP); the batting average (BA) or average hit per "at-bat" (at bats are plate appearances minus sacrifices [an action that was not tracked reliably in the early days of baseball, so we did not use it here] or walks); and the average amount of bases that the player manages to run around per attempt (called the slugging percentage or SLG). Again, these averages are easy to determine with simple division, so these are the nine predictors that we will use.

A note on why I didn't choose other numbers in the data set for predictors - the point of this project is to help us understand better through data analysis the value of each action that any baseball player (whether playing in the major leagues on on a local little league) is able to accomplish. This rules out the use of several predictors that would definitely make our algorithm more accurate, but would defeat the purpose of the project. The easiest example of this is the year. At the end of the analysis, I did use the year as a predictor, and was able to be more accurate, but this would be of no use to a parent in the stands, so I do not count it in the larger project. That guiding principal - hoping to be of some practical use, also rules out the use of games, plate appearances, and league - none of these would help us be able to determine the value of an action that my young son playing baseball is taking on his baseball team, for example.

Now that I have predictors that are evened out for all the teams in the data set - I can go to work!

In order to avoid over training a linear regression model, I split up the data into a test set and training set. The test set was made from a random sample of 10% of the overall data while the rest was in the training set. I did try out other divisions of the data (50/50, for example), but I opted to present this algorithm trained with a 90/10 as I found this split to be the most challenging to create an algorithm that successfully predicted WAA.

```
set.seed(1, sample.kind="Rounding")
test_index_hitting <- createDataPartition(y = hitting$WAA_162,
                                          times = 1, p = 0.1, list = FALSE)
train_hitting <- hitting[-test_index_hitting,]
test_hitting <- hitting[test_index_hitting,]
```

First, I set a baseline. I found that if I guessed the average WAA_162 (which on our train set is -0.001531599) for every team in the test set, I would be on average about 16 games off (I use the root mean squared error or "RMSE" formula for this). An RMSE of 16 is what we will need to get better.

```
ave_y <- mean(train_hitting$WAA_162)
ave_y
```

```
## [1] -0.001531599
```

```
naive_rmse <- sqrt(mean((ave_y - test_hitting$WAA_162)^2))
naive_rmse
```

```
## [1] 16.7083
```

```
table <- tibble(Method = "Guess the Ave", RMSE = naive_rmse)
kable(table)
```

Method	RMSE
Guess the Ave	16.7083

Results (and Methods continued)

Now, I started with all nine predictors to train a linear model on the train set, and then used the predict function with the fitted model to make predictions on the test set. I then checked the RMSE of the those predictions on the actual WAA_162 of the test set and found that this gave me a RMSE of 13.40354.

```
fit <- lm(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + x2b_g + x3b_g +
            rbi_g + bb_g, data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm_all <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm_all
```

```
## [1] 13.40354
```

```
#compare rmses
table <- bind_rows(table,
                   tibble(Method = "LM_All Feats", RMSE = rmse_lm_all))
kable(table)
```

Method	RMSE
Guess the Ave	16.70830
LM_All Feats	13.40354

Thinking that I perhaps had some noisy predictors, I then started cycling them out one by one to see if I could improve my RMSE. I did this several times with different slices of the data, for the slice that I presented in the accompanying code, I found that only by removing the RBI/G (rbi_g) predictor did the overall RMSE improve to 13.29213. Removing any of the other predictors did not make for a better score. There were a few notes of interest at this point, however. By far the predictor that was the least helpful was walks, which only improved our predicted WAA_162's RMSE by 0.00492. In fact, in other slices of the data this predictor was not helpful at all. Also of noted interest is that in triples/game only improved our RMSE in this slice of the data by 0.02256. In other slices of the data it was not helpful either.

```
#Taking away BB ratio below
fit <- lm(WAA_162 ~ hr_g + BA + OBP + SLG + h_g +
            x2b_g + x3b_g + rbi_g, data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.40846
```

```r
#better to look at walks - taking away x3b ratio below

fit <- lm(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + bb_g +
            x2b_g + rbi_g, data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.4261
```

```r
#will keep x3b in the equation. Try taking out rbi ratio:

fit <- lm(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + bb_g + x3b_g +
            x2b_g, data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.29213
```

```r
#RBI is a bit noisy in this data set. Will try to take out x2b

fit <- lm(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + bb_g + x3b_g,
          data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.38123
```

```r
#it is still better to keep it in. Removing hits/game

fit <- lm(WAA_162 ~ hr_g + BA + OBP + SLG + x2b_g + bb_g + x3b_g,
          data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.36721
```

```r
#better to include H/G ratio. Taking away SLG here

fit <- lm(WAA_162 ~ hr_g + BA + OBP + x2b_g + bb_g + x3b_g + h_g,
          data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.37954
```

```r
#better to look at SLG. Taking away OBP

fit <- lm(WAA_162 ~ hr_g + BA + SLG + x2b_g + bb_g + x3b_g + h_g,
          data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.3203
```

```r
#Better to keep OBP. Taking away BA

fit <- lm(WAA_162 ~ hr_g + OBP + SLG + x2b_g + bb_g + x3b_g + h_g,
          data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.37988
```

```r
#Better to look at BA. Taking away HR

fit <- lm(WAA_162 ~ BA + OBP + SLG + x2b_g + bb_g + x3b_g + h_g,
          data = train_hitting)

y_hat <- predict(fit, test_hitting)

rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.35767
```

```r
#Better to look at HR/G. Here is the best that LM can do:

fit <- lm(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + bb_g + x3b_g +
            x2b_g, data = train_hitting)

y_hat <- predict(fit, test_hitting)
```

```
rmse_lm <- sqrt(mean((y_hat - test_hitting$WAA_162)^2))
rmse_lm
```

```
## [1] 13.29213
```

```
#compare rmses
table <- bind_rows(table,
                   tibble(Method = "LM_Best Feats", RMSE = rmse_lm))
kable(table)
```

Method	RMSE
Guess the Ave	16.70830
LM_All Feats	13.40354
LM_Best Feats	13.29213

Given than our linear model only resulted in an improvement to an RMSE of 13.29213, meaning that we are about 13 games off on average, I went beyond linear regression in hopes of finding some better results.

First, I tried K-Nearest Neighbors. Using our top 8 predictors for this data slice, I fitted a model using the knn3 function on the train set. I then used this fitted model to make predictions on the test set and found that these predictions resulted in a RMSE of 16.70832.

```
knn_fit <- knn3(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + bb_g + x3b_g +
                  x2b_g, data = train_hitting)

y_hat_knn <- predict(knn_fit, test_hitting, type = "prob")

rmse_knn <- sqrt(mean((y_hat_knn - test_hitting$WAA_162)^2))
rmse_knn
```

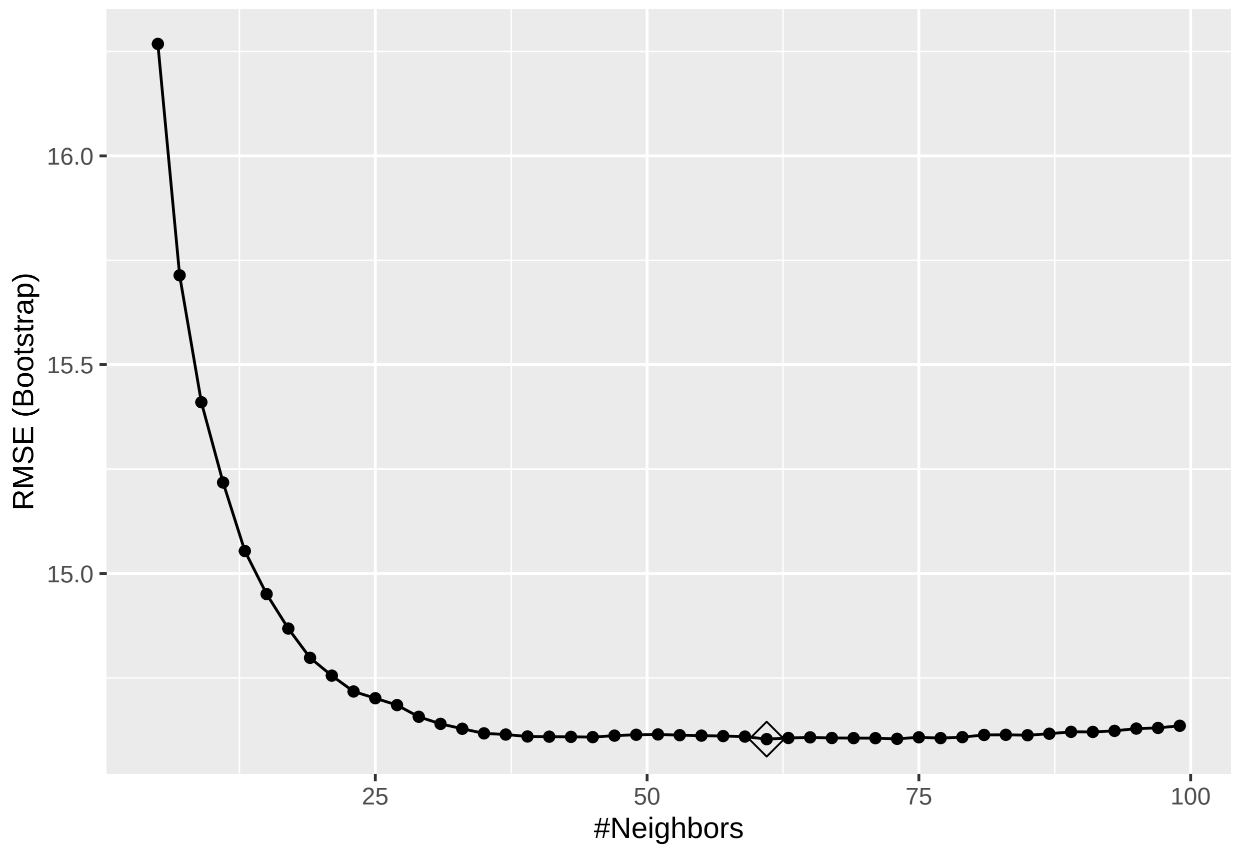
```
## [1] 16.70832
```

```
#compare rmses
table <- bind_rows(table,
                   tibble(Method = "KNN", RMSE = rmse_knn))
kable(table)
```

Method	RMSE
Guess the Ave	16.70830
LM_All Feats	13.40354
LM_Best Feats	13.29213
KNN	16.70832

Not what I was hoping for, but perhaps I could choose a better tuning parameter for this function? Understanding that this tuning parameter will randomly sample segments of the data many times, I opted to train it on the entire data set. I used the train function to with the tuning parameter set to k = seq(5, 99, 2). I then plotted the results to find that the K value with the lowest RMSE was K=61. This tune resulted in a RMSE of 14.60331 - not an improvement on our linear model.

```r
set.seed(1)
train_knn <- train(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + bb_g + x3b_g + x2b_g,
                   method = "knn", data = hitting,
                   tuneGrid = data.frame(k = seq(5, 99, 2)))

plot_knn <- ggplot(train_knn, highlight = TRUE)
plot_knn
```



```r
which.min(train_knn$results$RMSE)
```

```
## [1] 29
```

```r
kable(train_knn$results[29,])
```

	k	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
29	61	14.60331	0.2179161	11.1259	0.2780908	0.0235136	0.1826634

```r
#compare rmses
table <- bind_rows(table,
                   tibble(Method = "KNN_Tuned", RMSE = 14.60331))
kable(table)
```

Method	RMSE
Guess the Ave	16.70830
LM_All Feats	13.40354
LM_Best Feats	13.29213
KNN	16.70832
KNN_Tuned	14.60331

So I turned to the Random Forest package. Again understanding that it would take multiple random samples of the data, I used the entire data set for it. Using the top 8 features, I made predictions on the data set and found that this resulted in an RMSE of 13.98472 - better than KNN but not than the linear model.

```
if(!require(randomForest)) install.packages('randomForest',
                                           repos = "http://cran.us.r-project.org")
library(randomForest)
set.seed(1)
fit_rf <- randomForest(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + bb_g + x3b_g +
                         x2b_g, data = hitting)

rmse_rf_somefeats <- sqrt(mean((fit_rf$predicted - hitting$WAA_162)^2))
rmse_rf_somefeats
```

```
## [1] 13.98472
```

```
#compare rmses
table <- bind_rows(table,
                   tibble(Method = "RF", RMSE = rmse_rf_somefeats))
kable(table)
```

Method	RMSE
Guess the Ave	16.70830
LM_All Feats	13.40354
LM_Best Feats	13.29213
KNN	16.70832
KNN_Tuned	14.60331
RF	13.98472

Thinking that the nature of Random Forest might help with noisier predictors, I tried a random forest fit with all nine predictors. This resulted in a better RMSE of 13.66735.

```
set.seed(1)
fit_rf_all <- randomForest(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + x2b_g +
                             x3b_g + rbi_g + bb_g, data = hitting)

rmse_rf_all <- sqrt(mean((fit_rf_all$predicted - hitting$WAA_162)^2))
rmse_rf_all
```

```
## [1] 13.66735
```

```r
#compare rmses
table <- bind_rows(table,
                   tibble(Method = "RF_All Feats", RMSE = rmse_rf_all))
kable(table)
```

Method	RMSE
Guess the Ave	16.70830
LM_All Feats	13.40354
LM_Best Feats	13.29213
KNN	16.70832
KNN_Tuned	14.60331
RF	13.98472
RF_All Feats	13.66735

Finally, to see the outcome of adding the year and number of games to the algorithm, I added those two features to the random forest fit and found that the predictions that results improved. The resulting RMSE was 12.84503.

```r
set.seed(1)
fit_rf_x <- randomForest(WAA_162 ~ hr_g + BA + OBP + SLG + h_g + x2b_g +
                             x3b_g + rbi_g + bb_g + year + G, data = hitting)

rmse_rf_x <- sqrt(mean((fit_rf_x$predicted - hitting$WAA_162)^2))
rmse_rf_x
```

```
## [1] 12.84503
```

This makes sense. Random Forest was tracking that there is a correlation in accurately predicting WAA_162 based upon the year and the number of games played. The algorithm was now tracking that the game of baseball was changing with the years, but given that a young boy playing baseball with his local team has statistically more in common with a team of 1875 than a major league team of 2021 (he will hit an average home run total closer to 1875 than now, for example) this improvement only shows that I am using the data analytics tools at my disposal properly, but not much more.

Results Summary

For easy review, here is a summary of the different RMSE scores:

```r
#compare rmses
table <- bind_rows(table,
                   tibble(Method = "RF_x", RMSE = rmse_rf_x))
kable(table)
```

Method	RMSE
Guess the Ave	16.70830
LM_All Feats	13.40354
LM_Best Feats	13.29213
KNN	16.70832

Method	RMSE
KNN_Tuned	14.60331
RF	13.98472
RF_All Feats	13.66735
RF_x	12.84503

Conclusion

In summary, the predictions resulting from an algorithm built on a linear model of eight predictors (Home Runs per game, batting average, on-base percentage, slugging percentage, hits per game, walks per game, triples per game, and doubles per game) gave me the best results. However, there is much more work that can be done. Due to the constraints in my data set, defensive data (pitching and fielding) was not used. This algorithm could easily by plugged into other data sets (the Lahman data set for example) that have pitching and fielding data. Armed with this additional data, I believe it would be possible to get the RMSE below 8 games, if not, more. In addition, this algorithm is simple enough for others to use to train on their local league's stat sheet, meaning that it is easy to check to see if it applies for your individual situation.

Once you have a predicted WAA_162 that can accurately reflect the reality of your situation, I would suggest that we take the amount that each stat was able to accurately lower the resulting RMSE and weight that stat accordingly. Then that weighted score could be compiled into a single individual player stat that is able to simply and reliably show how much that individual player has contributed to wins above average - that is, the success of the team as a whole. For example, if a young player had 5 walks, we would multiply 5*0.00492 to their over all WAA score. This will allow us to compile and weight statistics in way that has more direct bearing than other modern advanced statistics.

This may not seem like an earth shattering study or find, but I believe that it can be a practical help to me and my son as we play baseball together. I hope that you, the reader, may find it of some use to you as well.