DATA621_HW1_team

Matthew Baker, Misha Kollontai, Erinda Budo, Don Padmaperuma, Subhalaxmi Rout

9/20/2020

Overview

In this homework assignment, we will explore, analyze and model a data set containing approximately 2200 records. Each record represents a professional baseball team from the years 1871 to 2006 inclusive. Each record has the performance of the team for the given year, with all of the statistics adjusted to match the performance of a 162 game season. Our objective is to build a multiple linear regression model on the training data to predict the number of wins for the team.

Libraries

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

```
library(ggcorrplot)
library(pastecs)
```

```
##
## Attaching package: 'pastecs'
```

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

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

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

```
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
```

```
library(ggplot2)
library(mice)
```

```
##
## Attaching package: 'mice'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(VIM)
```

```
## Loading required package: colorspace
```

```
## Loading required package: grid
```

```
## VIM is ready to use.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
```

```
##
## Attaching package: 'VIM'
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(car)
```

```
## 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
```

```
library(jtools)
```

Data Import and Prep

The original data was loaded containing 2276 rows and 16 columns related to batting, base run, pitching, and fielding. A summary and boxplots of all variables suggest some columns have missing data, and some may contain outliers.

```
#Read the training and test data
MB_train <- read.csv("https://raw.githubusercontent.com/mkollontai/DATA621_GroupWork/master/HW1/moneybal
MB_test <- read.csv("https://raw.githubusercontent.com/mkollontai/DATA621_GroupWork/master/HW1/moneybal
```

```r
dim(MB_train)
```

```
## [1] 2276   17
```

```r
head(MB_train)
```

```
##   INDEX TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## 1     1          39           1445             194              39
## 2     2          70           1339             219              22
## 3     3          86           1377             232              35
## 4     4          70           1387             209              38
## 5     5          82           1297             186              27
## 6     6          75           1279             200              36
##   TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## 1              13             143             842              NA
## 2             190             685            1075              37
## 3             137             602             917              46
## 4              96             451             922              43
## 5             102             472             920              49
## 6              92             443             973             107
##   TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
## 1              NA               NA            9364               84
## 2              28               NA            1347              191
## 3              27               NA            1377              137
## 4              30               NA            1396               97
## 5              39               NA            1297              102
## 6              59               NA            1279               92
##   TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## 1              927             5456            1011               NA
## 2              689             1082             193              155
## 3              602              917             175              153
## 4              454              928             164              156
## 5              472              920             138              168
## 6              443              973             123              149
```

```r
#elimination index column in both data files
MB_train <- MB_train[-1]
MB_test <- MB_train[-1]
```

```r
#subset the batting  stats
 MB_train_bat<-MB_train[c("TEAM_BATTING_H"  , "TEAM_BATTING_2B" , "TEAM_BATTING_3B" , "TEAM_BATTING_HR"

#subset the baserunning  stats
 MB_train_base<-MB_train[c("TEAM_BASERUN_SB" , "TEAM_BASERUN_CS")]

#subset the fielding  stats
 MB_train_field<-MB_train[c("TEAM_FIELDING_E",  "TEAM_FIELDING_DP")]

#subset the pitching  stats
 MB_train_pitch<-MB_train[c("TEAM_PITCHING_H" ,"TEAM_PITCHING_HR" ,"TEAM_PITCHING_BB" ,"TEAM_PITCHING_S
```

Data Summaries

We can only use the variables given to us (or variables that we derive from the variables provided). Below codes shows the variables of interest in the data set:

```
summary(MB_train)
```

```
##   TARGET_WINS     TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
##  Min.   :  0.00   Min.   : 891   Min.   : 69.0   Min.   :  0.00
##  1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0   1st Qu.: 34.00
##  Median : 82.00   Median :1454   Median :238.0   Median : 47.00
##  Mean   : 80.79   Mean   :1469   Mean   :241.2   Mean   : 55.25
##  3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0   3rd Qu.: 72.00
##  Max.   :146.00   Max.   :2554   Max.   :458.0   Max.   :223.00
##
##  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB
##  Min.   :  0.00   Min.   :  0.0   Min.   :   0.0   Min.   :  0.0
##  1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 548.0   1st Qu.: 66.0
##  Median :102.00   Median :512.0   Median : 750.0   Median :101.0
##  Mean   : 99.61   Mean   :501.6   Mean   : 735.6   Mean   :124.8
##  3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 930.0   3rd Qu.:156.0
##  Max.   :264.00   Max.   :878.0   Max.   :1399.0   Max.   :697.0
##                                   NA's   :102      NA's   :131
##  TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
##  Min.   :  0.0   Min.   :29.00    Min.   : 1137   Min.   :  0.0
##  1st Qu.: 38.0   1st Qu.:50.50    1st Qu.: 1419   1st Qu.: 50.0
##  Median : 49.0   Median :58.00    Median : 1518   Median :107.0
##  Mean   : 52.8   Mean   :59.36    Mean   : 1779   Mean   :105.7
##  3rd Qu.: 62.0   3rd Qu.:67.00    3rd Qu.: 1682   3rd Qu.:150.0
##  Max.   :201.0   Max.   :95.00    Max.   :30132   Max.   :343.0
##  NA's   :772     NA's   :2085
##  TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E  TEAM_FIELDING_DP
##  Min.   :   0.0   Min.   :    0.0   Min.   :  65.0   Min.   : 52.0
##  1st Qu.: 476.0   1st Qu.:  615.0   1st Qu.: 127.0   1st Qu.:131.0
##  Median : 536.5   Median :  813.5   Median : 159.0   Median :149.0
##  Mean   : 553.0   Mean   :  817.7   Mean   : 246.5   Mean   :146.4
##  3rd Qu.: 611.0   3rd Qu.:  968.0   3rd Qu.: 249.2   3rd Qu.:164.0
##  Max.   :3645.0   Max.   :19278.0   Max.   :1898.0   Max.   :228.0
##                   NA's   :102                        NA's   :286
```

Few more descriptive statistics of MB_train data.The descriptive statistics below shows the the mean, mode, **standard deviation**, minimum and maximum of each variable in the dataset.

```
stat.desc(MB_train, basic = F)
```

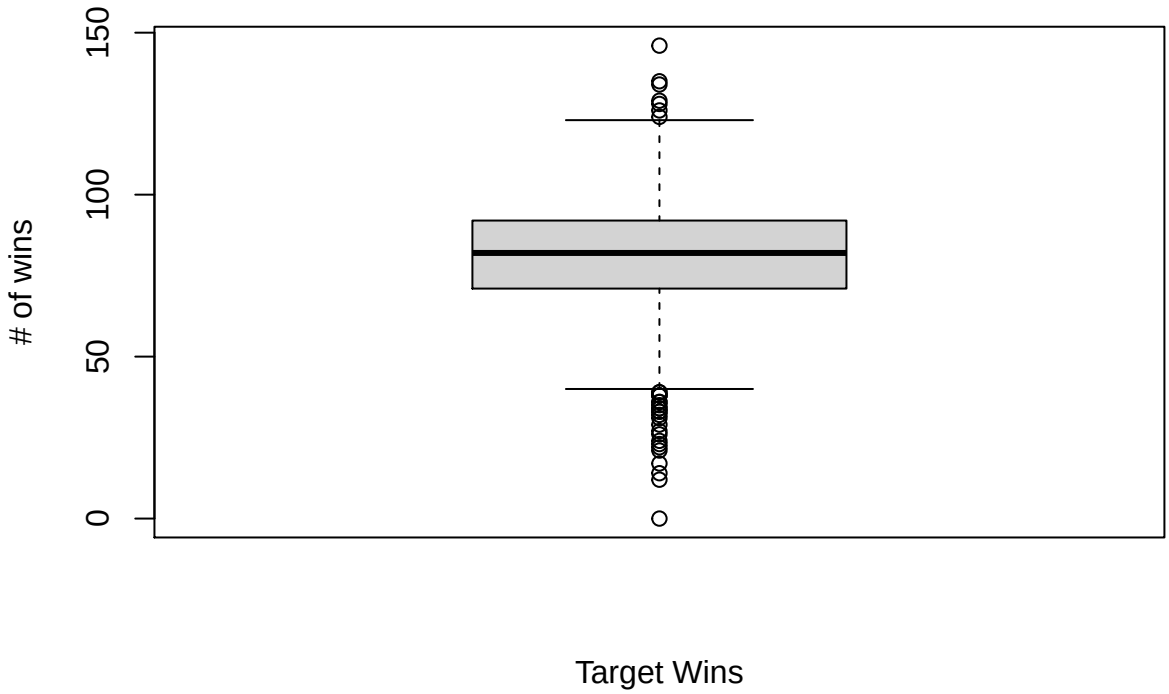
```
##              TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## median        82.0000000   1.454000e+03     238.0000000      47.0000000
## mean          80.7908612   1.469270e+03     241.2469244      55.2500000
## SE.mean        0.3301823   3.030789e+00       0.9810087       0.5856226
## CI.mean.0.95   0.6474899   5.943400e+00       1.9237652       1.1484102
## var          248.1303077   2.090661e+04    2190.3724081     780.5629670
## std.dev       15.7521525   1.445912e+02      46.8014146      27.9385570
## coef.var       0.1949744   9.841024e-02       0.1939980       0.5056752
##              TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## median           102.0000000    5.120000e+02    7.500000e+02     101.0000000
## mean              99.6120387    5.015589e+02    7.356053e+02     124.7617716
```

```
## SE.mean            1.2691285    2.571315e+00    5.330191e+00       1.8955584
## CI.mean.0.95       2.4887702    5.042367e+00    1.045281e+01       3.7173247
## var             3665.9237056    1.504814e+04    6.176538e+04    7707.2888364
## std.dev           60.5468720    1.226709e+02    2.485264e+02      87.7911660
## coef.var           0.6078269    2.445792e-01    3.378529e-01       0.7036704
##              TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
## median            49.0000000       58.0000000    1.518000e+03      107.0000000
## mean              52.8038564       59.3560209    1.779210e+03      105.6985940
## SE.mean            0.5919414        0.9382681    2.948896e+01        1.2848886
## CI.mean.0.95       1.1611188        1.8507602    5.782807e+01        2.5196759
## var              526.9934382      168.1462662    1.979207e+06     3757.5363673
## std.dev           22.9563376       12.9671225    1.406843e+03       61.2987469
## coef.var           0.4347474        0.2184635    7.907119e-01        0.5799391
##              TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## median           5.365000e+02     8.135000e+02    1.590000e+02      149.0000000
## mean             5.530079e+02     8.177305e+02    2.464807e+02      146.3879397
## SE.mean          3.487032e+00     1.186212e+01    4.774328e+00        0.5879114
## CI.mean.0.95     6.838095e+00     2.326228e+01    9.362492e+00        1.1529868
## var              2.767477e+04     3.059031e+05    5.187962e+04      687.8232833
## std.dev          1.663574e+02     5.530850e+02    2.277710e+02       26.2263853
## coef.var         3.008228e-01     6.763659e-01    9.240926e-01        0.1791567
```

Data Exploration Visualizations

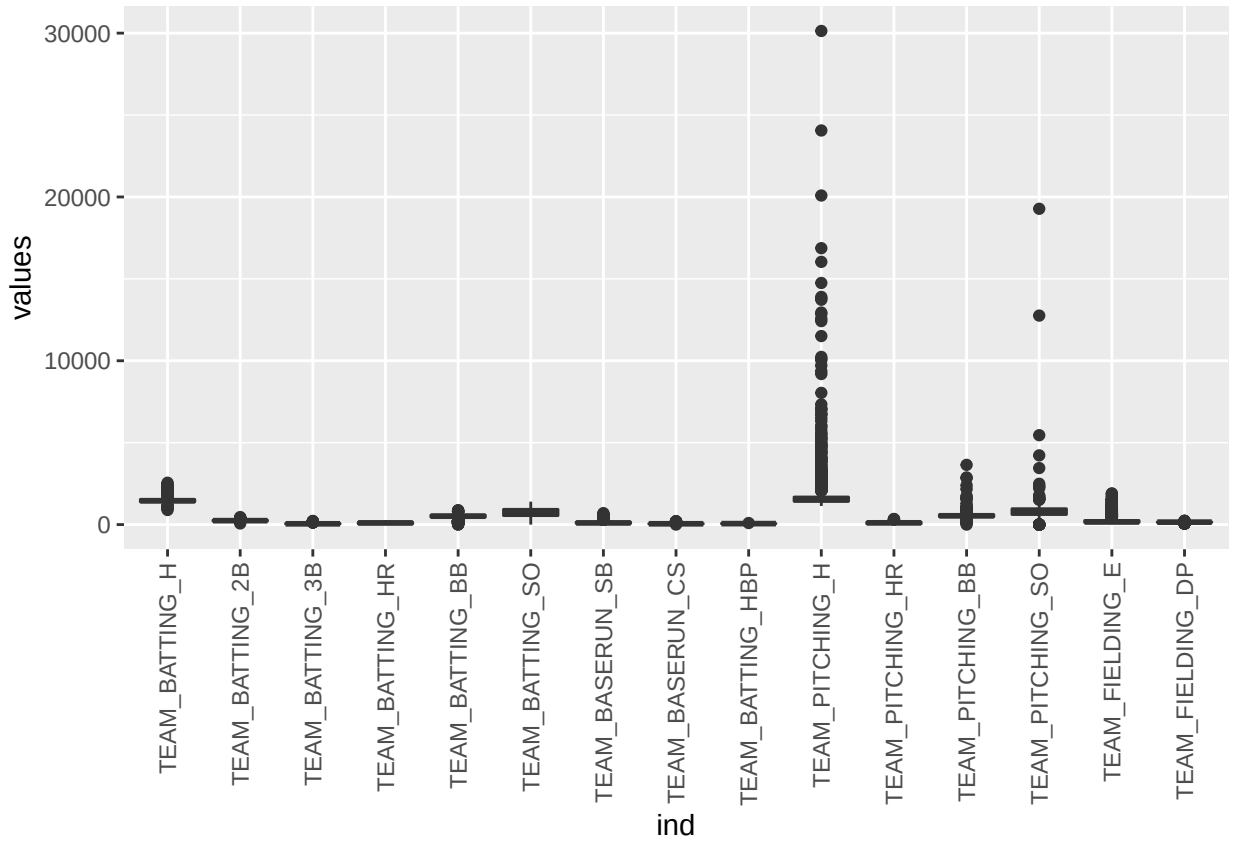
Let's take a closer look at the number of wins or **TARGET_WINS** variable.

```
boxplot(MB_train$TARGET_WINS, xlab = "Target Wins", ylab = "# of wins")
```

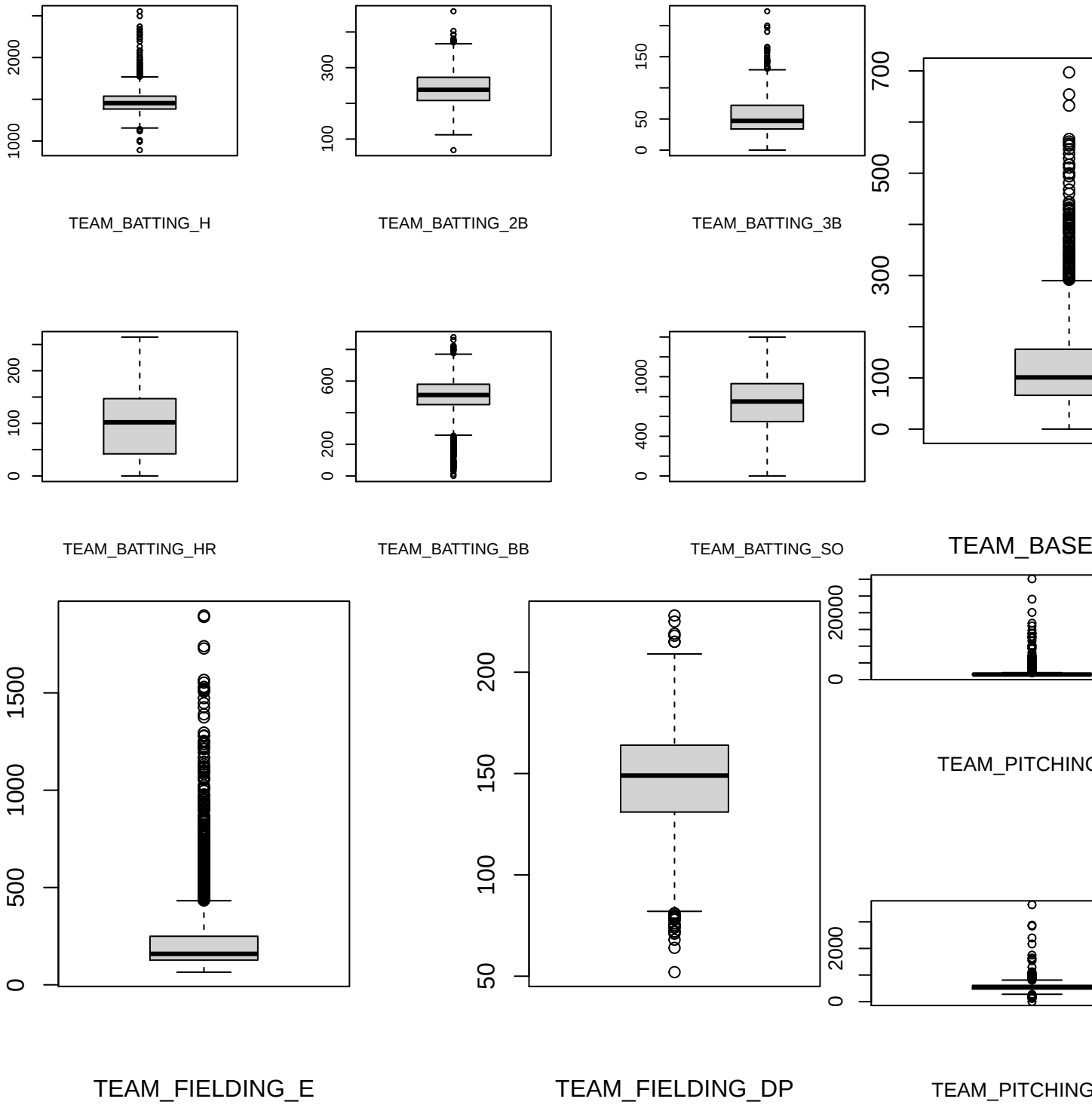


Let's look at all of the metrics in order to evaluate the presence of outliers and quality of the data overall.

```
## Warning: Removed 3478 rows containing non-finite values (stat_boxplot).
```



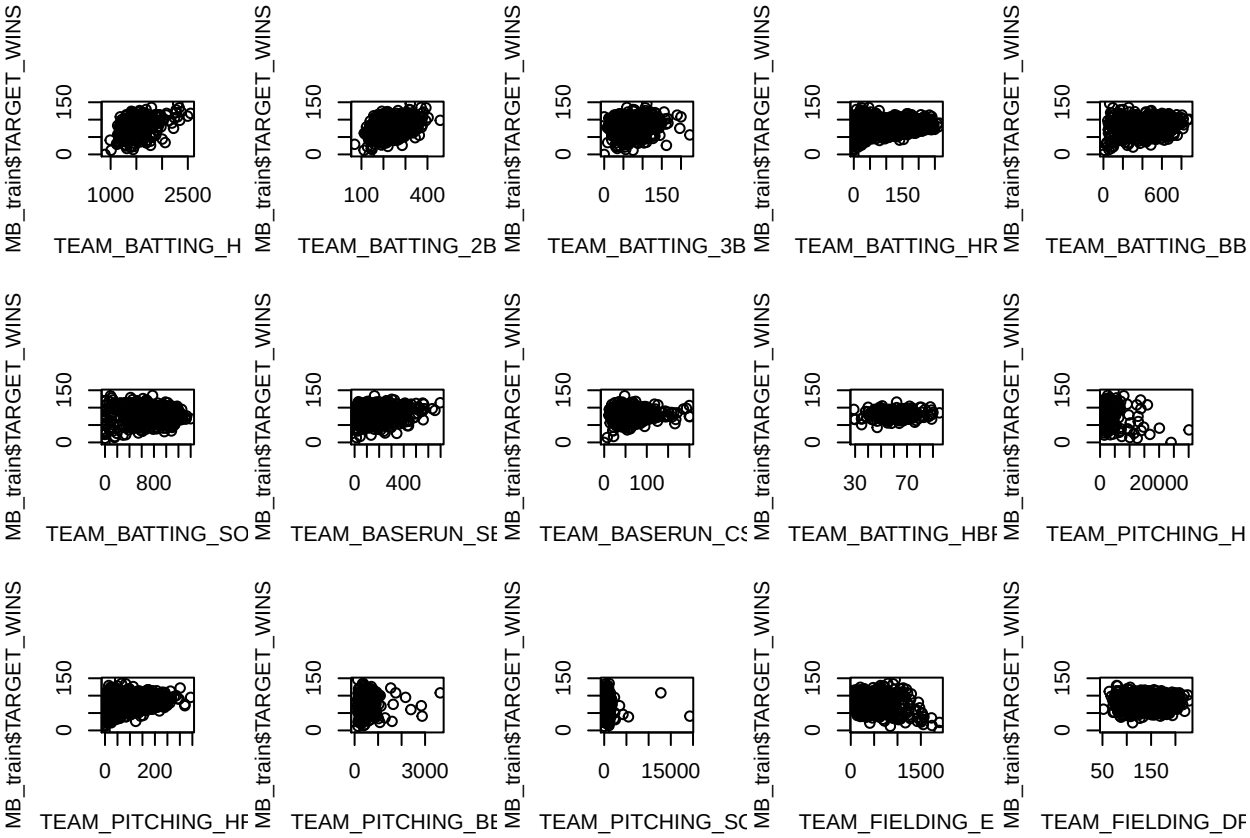

```
par(mfrow=c(2,3))
x <- c(1:6)
for (val in x) {
  boxplot(MB_train_bat[,val], xlab=names(MB_train_bat[val]))$out
}
```



Relationship with target variable TARGET_WIN

Using the plots below we can get a quick overview of relationships between our input variables and the target variable. Unfortunately nearly all of our variables in their current state are fairly widely distributed and do not show an obvious trend in the relationship with the wins. It appears that we need to account for multiple variables in order to create an accurate model.

```
#### Target Wins vs all variables
par(mfrow=c(3,5))
x <- c(2:16)
for (val in x) {
plot(MB_train[,val],MB_train$TARGET_WINS, xlab=names(MB_train[val]))
}
```



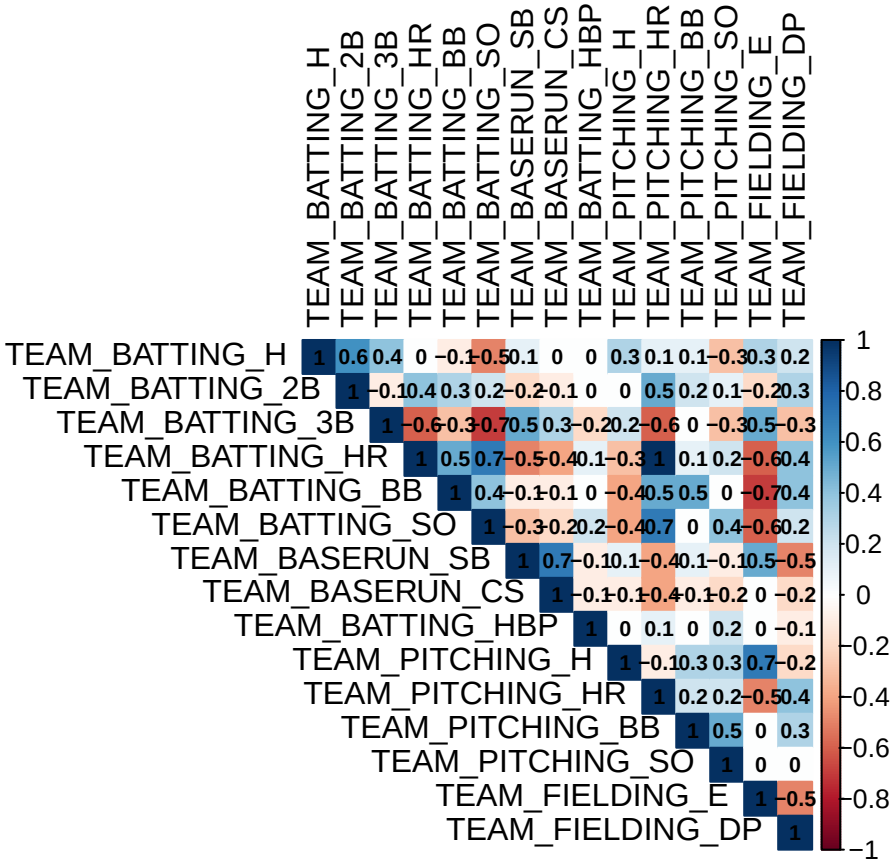
Finding Correlation among predictor variables

```
par(mfrow=c(1,1))
cor(MB_train, use = "complete.obs")[,1]
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##       1.00000000       0.46994665       0.31298400      -0.12434586
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##       0.42241683       0.46868793      -0.22889273       0.01483639
##  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR
##      -0.17875598       0.07350424       0.47123431       0.42246683
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##       0.46839882      -0.22936481      -0.38668800      -0.19586601
```

By looking at correlation between our predictor variables we can get an idea of those that are tied closely together - this may help us determine when including multiples may be redundant and removing one or two of the closely related variables may simplify our model without a strong negative impact on our model accuracy.

```
corr<- round(cor(MB_train[-1], use="pairwise.complete.obs", method = "pearson"),1)
corrplot(corr, method = "color",
         type = "upper", order = "original", number.cex = .7,
         addCoef.col = "black", # Add coefficient of correlation
         tl.col = "black", tl.srt = 90, # Text label color and rotation
                  # hide correlation coefficient on the principal diagonal
         diag = TRUE)
```



Cleaning Data based on what we know about the game

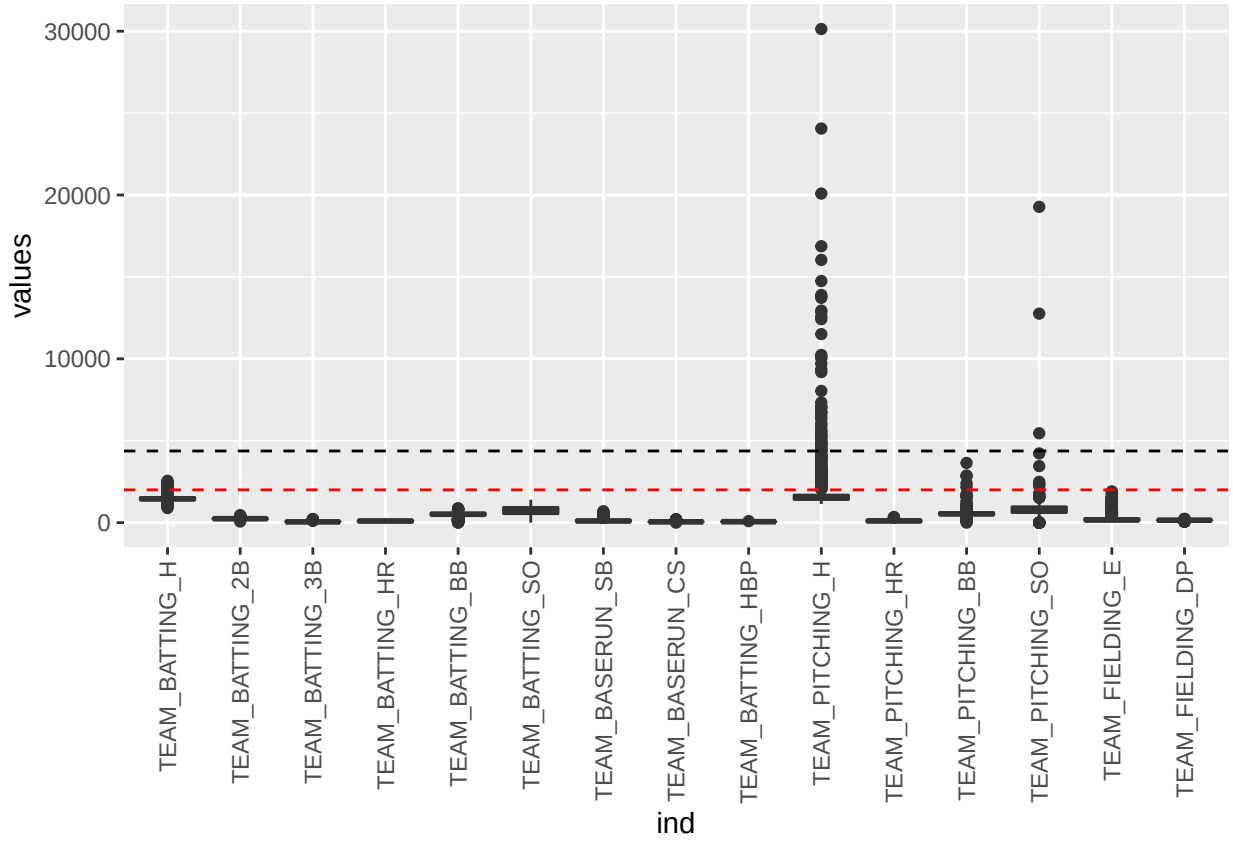
Let's take a look at the number of strikeouts a team pitching staff has achieved in a given season. First, let's calculate the maximum number strikeouts a team can achieve per season assuming 162 games. This would require 3 strikeouts every 9 innings.

The record number of strikeouts by a pitching staff in a season is 1,687 by the Houston Astros in 2018 (this datapoint is not included here since the data only covers up to 2006). Since some of the data is extrapolated to assume a 162 game season, it's possible some earlier seasons may equate to more, so let's use 2000 as a max possible value.

```
Max_SO <- 162*9*3 #This assumes a season in which every out was a strikeout (obviously never happened)

ggplot(stack(MB_train[,-1]), aes(x = ind, y = values)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +   geom_hline(yintercept=Max_SO
  geom_hline(yintercept=2000, linetype = 'dashed', color='red')
```

```
## Warning: Removed 3478 rows containing non-finite values (stat_boxplot).
```

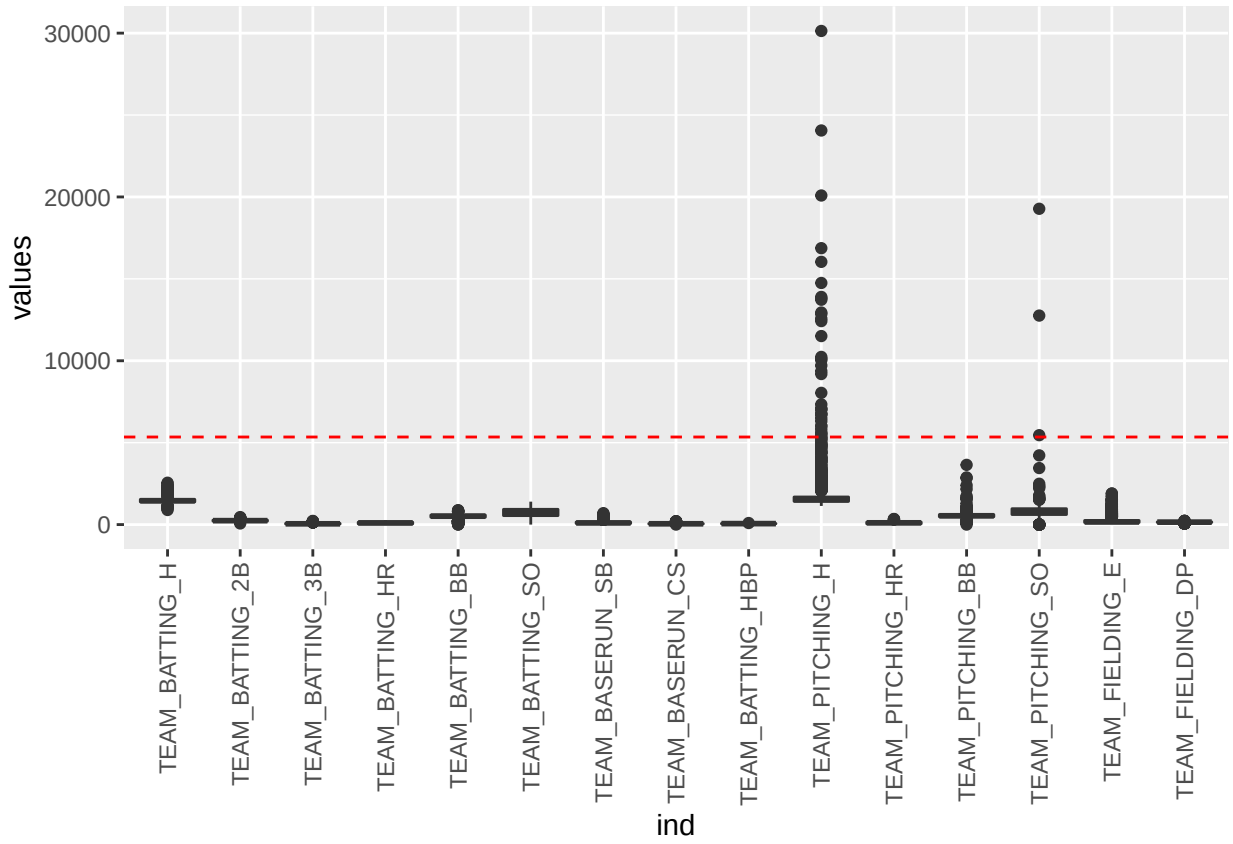


According to MLB.com the most hits allowed in a game was 33. Assuming a 162 game seaosn, the highest number of hits allowed in a season possible would be $162 * 33 = 5346$ hits. Anything above this is impossible.

```
most_hits <- 162*33

ggplot(stack(MB_train[,-1]), aes(x = ind, y = values)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  geom_hline(yintercept=most_hits, linetype = 'dashed', color='red')
```

```
## Warning: Removed 3478 rows containing non-finite values (stat_boxplot).
```



Based on the graph above, it looks like we are potentially leaving a large amount of erroneous data in the set, but we have no factual basis to remove it on.

We can clearly see that there is a fair amount of outliers in our data. To begin, let's simply remove these rows of data since we can't have faith in the rest of the data associated with the measurement. We can also remove the rows associated with other clearly erroneous data:

- No team has ever hit 0 home runs in a season according to baseball-almanac1, so remove rows that claim such a season.
- No team has ever had 0 hitter strikeouts in a season according to baseball-almanac2, so remove rows that claim such a season.
- According to this article "The fewest home runs given up by a pitching staff in one season was the four relinquished by the marvelous mound corps of the 1902 Pittsburgh Pirates over a 140-game schedule.
- Trusting the research of user BowlOfRed the highest number of errors by a team in a season is Washington in 1886 committing 867 errors in 122 games. If we extrapolate that number to 162 games we can cut out the false data points here as well.

```
most_hits <- 162*33
max_errors <- 867/122*162
Max_SO <- 162*9*3 #This assumes a season in which every out was a strikeout (obviously never happened)
MB_train_clean <- MB_train[!(MB_train$TEAM_PITCHING_SO > 2000),]
MB_train_clean <- MB_train_clean[!(MB_train_clean$TEAM_BATTING_HR == 0),]
MB_train_clean <- MB_train_clean[!(MB_train_clean$TEAM_BATTING_SO == 0),]
MB_train_clean <- MB_train_clean[!(MB_train_clean$TEAM_PITCHING_HR < 4),]
MB_train_clean <- MB_train_clean[!(MB_train_clean$TEAM_PITCHING_H > most_hits),]
MB_train_clean <- MB_train_clean[!(MB_train_clean$TEAM_FIELDING_E > max_errors),]
```

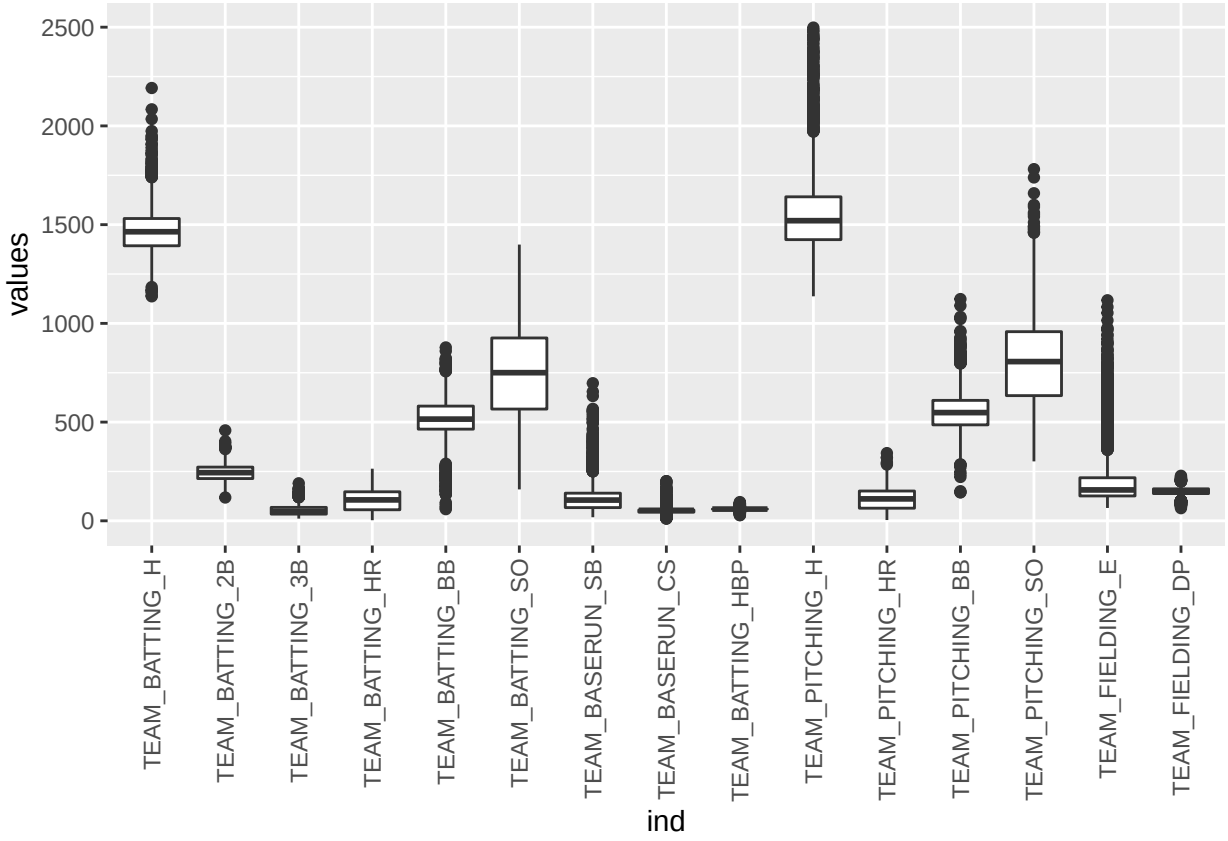
Impute Missing Data We can see that a large amount of datapoints are missing. Let's replace the NA's with the mean value for each column.

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

Visualizations of cleaned data

Another look at the variables. Being able to examine Skewness and outliers of the data will help us to chose the model. This is important as some models will require transformation of data.

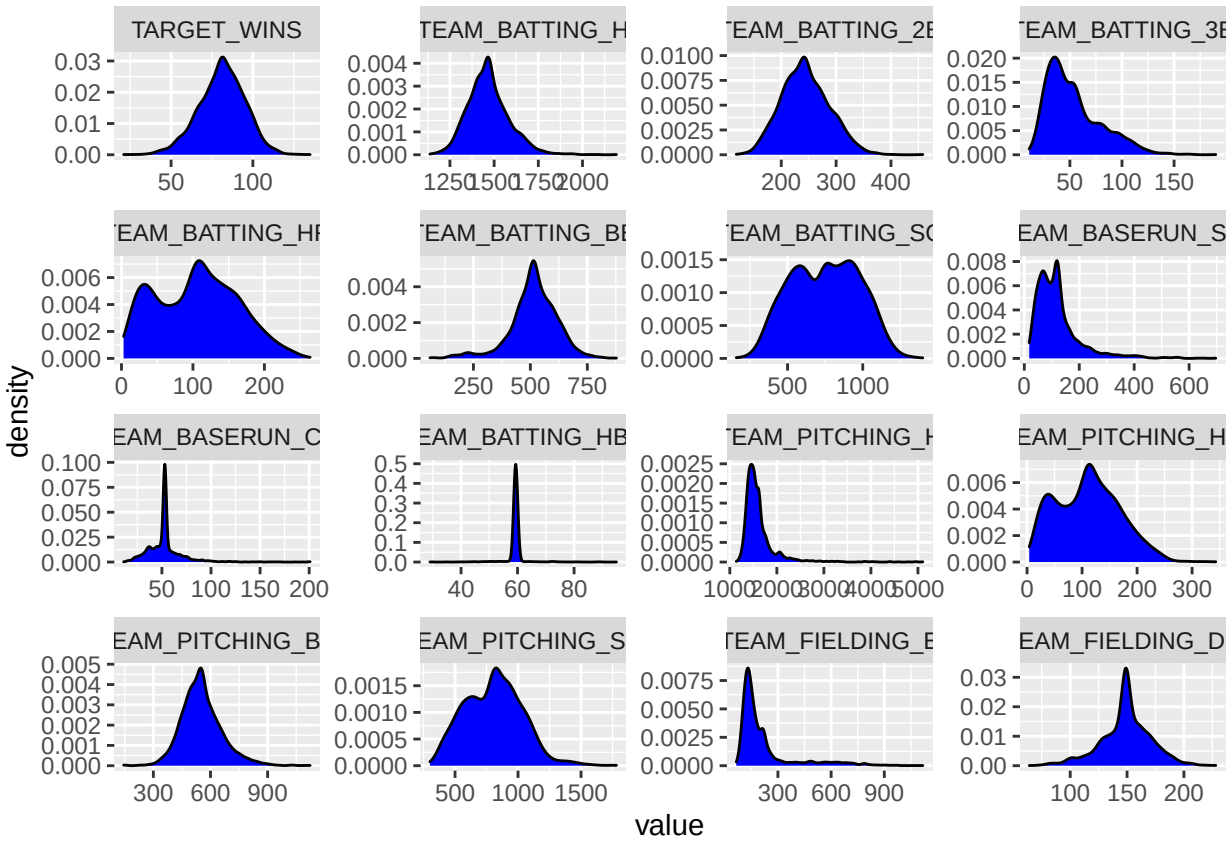
```
## Warning: Removed 60 rows containing non-finite values (stat_boxplot).
```




```
par(mfrow = c(3,5))
datasub = melt(MB_train_clean)
```

```
## Using  as id variables
```

```
ggplot(datasub, aes(x=value)) +
  geom_density(fill = 'blue') + facet_wrap(~variable, scales = 'free')
```



Prediction models

To begin let's take a look at the model using all of the original data. This model includes all outliers that we looked over in the section above.

```
model0 <- lm(TARGET_WINS ~., data = MB_train)
summary(model0)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = MB_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.8708  -5.6564  -0.0599   5.2545  22.9274
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      60.28826   19.67842   3.064  0.00253 **
## TEAM_BATTING_H    1.91348    2.76139   0.693  0.48927
## TEAM_BATTING_2B   0.02639    0.03029   0.871  0.38484
## TEAM_BATTING_3B  -0.10118    0.07751  -1.305  0.19348
## TEAM_BATTING_HR  -4.84371   10.50851  -0.461  0.64542
## TEAM_BATTING_BB  -4.45969    3.63624  -1.226  0.22167
## TEAM_BATTING_SO   0.34196    2.59876   0.132  0.89546
## TEAM_BASERUN_SB   0.03304    0.02867   1.152  0.25071
## TEAM_BASERUN_CS  -0.01104    0.07143  -0.155  0.87730
```

```
## TEAM_BATTING_HBP  0.08247    0.04960   1.663  0.09815 .
## TEAM_PITCHING_H  -1.89096    2.76095  -0.685  0.49432
## TEAM_PITCHING_HR  4.93043   10.50664   0.469  0.63946
## TEAM_PITCHING_BB  4.51089    3.63372   1.241  0.21612
## TEAM_PITCHING_SO -0.37364    2.59705  -0.144  0.88577
## TEAM_FIELDING_E  -0.17204    0.04140  -4.155 5.08e-05 ***
## TEAM_FIELDING_DP -0.10819    0.03654  -2.961  0.00349 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.467 on 175 degrees of freedom
##   (2085 observations deleted due to missingness)
## Multiple R-squared:  0.5501, Adjusted R-squared:  0.5116
## F-statistic: 14.27 on 15 and 175 DF,  p-value: < 2.2e-16
```

We can clearly see that only a few (3) of our variables seem to have low enough p-values to be deemed statistically significant predictors. There are also 2085 observation deleted due to missing data - this is a very large chunk of wasted datapoints!

Next let's create a model that removes the outliers and patently false values identified in the previous section as well as the missing datapoints.

Model 1

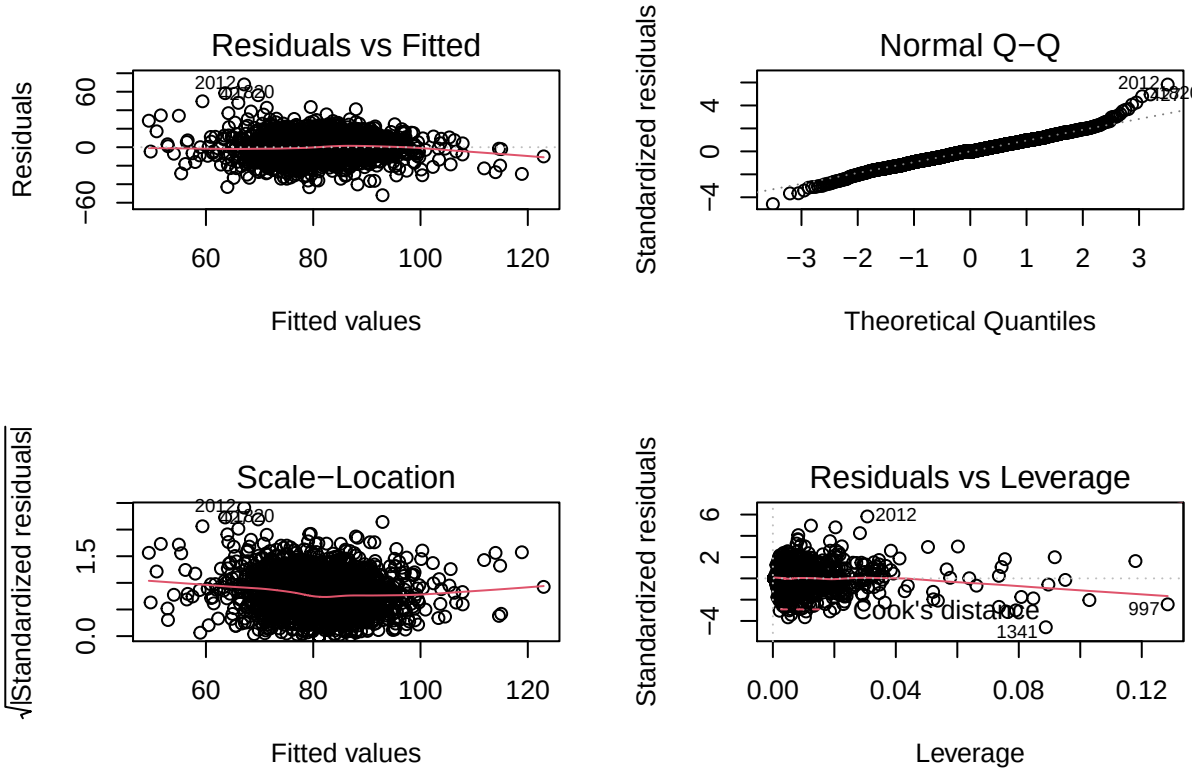
```
model1 <- lm(TARGET_WINS ~., data = MB_train_clean)
summary(model1)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = MB_train_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.923  -7.504   0.000   7.492  67.865
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      28.239071   6.952878   4.061 5.05e-05 ***
## TEAM_BATTING_H    0.014808   0.005444   2.720 0.006576 **
## TEAM_BATTING_2B  -0.019688   0.009059  -2.173 0.029859 *
## TEAM_BATTING_3B   0.139605   0.017975   7.767 1.23e-14 ***
## TEAM_BATTING_HR   0.197748   0.055627   3.555 0.000386 ***
## TEAM_BATTING_BB   0.107079   0.016291   6.573 6.14e-11 ***
## TEAM_BATTING_SO  -0.053991   0.009403  -5.742 1.06e-08 ***
## TEAM_BASERUN_SB   0.044307   0.004676   9.476  < 2e-16 ***
## TEAM_BASERUN_CS  -0.014355   0.015313  -0.937 0.348637
## TEAM_BATTING_HBP  0.081460   0.066341   1.228 0.219617
## TEAM_PITCHING_H   0.015360   0.002346   6.547 7.28e-11 ***
## TEAM_PITCHING_HR -0.104425   0.051629  -2.023 0.043234 *
## TEAM_PITCHING_BB -0.078529   0.014513  -5.411 6.95e-08 ***
## TEAM_PITCHING_SO  0.041993   0.008354   5.027 5.39e-07 ***
## TEAM_FIELDING_E  -0.039787   0.003638 -10.938  < 2e-16 ***
## TEAM_FIELDING_DP -0.077240   0.013038  -5.924 3.64e-09 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.84 on 2202 degrees of freedom
## Multiple R-squared:  0.3006, Adjusted R-squared:  0.2958
## F-statistic: 63.08 on 15 and 2202 DF,  p-value: < 2.2e-16
```

We can immediately see how much more statistically significant each of our cleaned predictors are - nearly every single p-value is noticeably lower than it was in the initial model. This suggests that the data in this model is more likely to have a relationship with our target variable.

```
par(mfrow=c(2,2))
plot(model1)
```



Diagnostics plots

Model 2

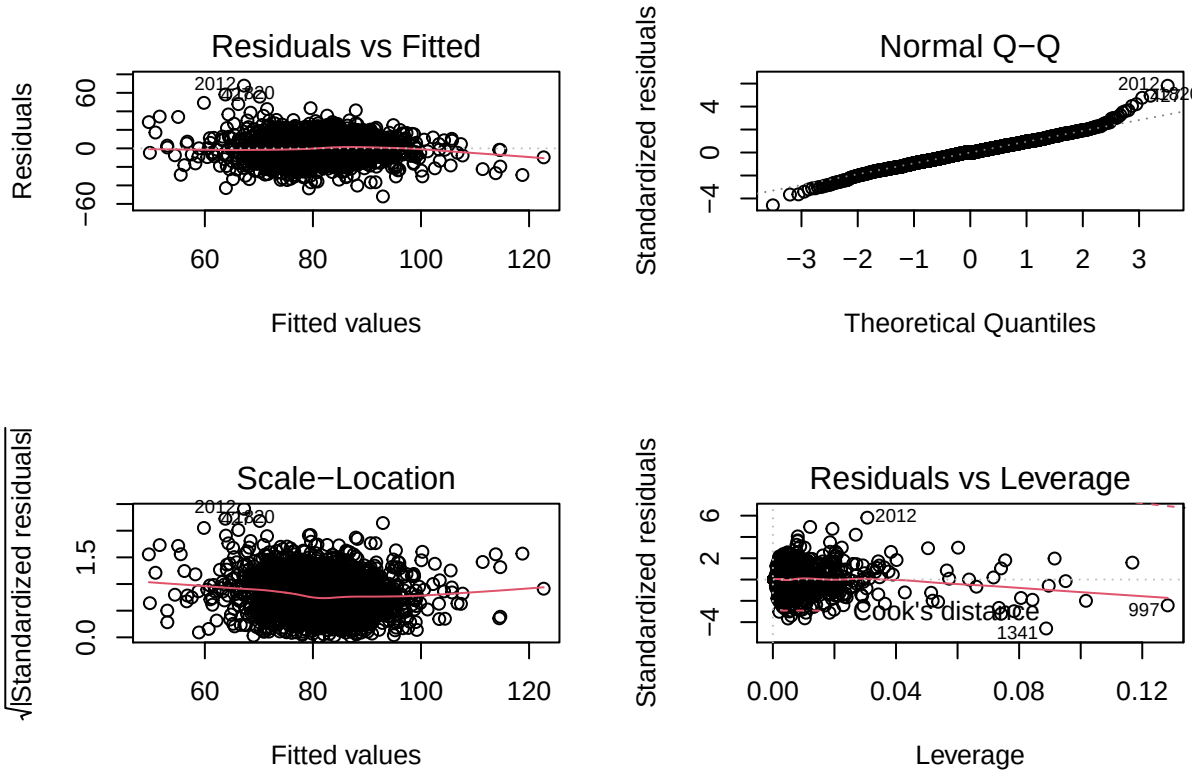
Variables will be removed one by one until the most optimal output is achieved. We did this in order to remove those features that do not have a significant effect on the dependent variable or prediction of output and simplify our model. To start we wikll remove the variables with the highest p-values.

Summary and vif

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - TEAM_BASERUN_CS - TEAM_BATTING_HBP,
##     data = MB_train_clean)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -51.950  -7.526   0.000   7.489  67.737
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      31.970344   5.694430   5.614 2.22e-08 ***
## TEAM_BATTING_H    0.014872   0.005444   2.732 0.006348 **
## TEAM_BATTING_2B  -0.019862   0.009058  -2.193 0.028432 *
## TEAM_BATTING_3B   0.139146   0.017974   7.742 1.48e-14 ***
## TEAM_BATTING_HR   0.203085   0.055296   3.673 0.000246 ***
## TEAM_BATTING_BB   0.104805   0.016137   6.495 1.03e-10 ***
## TEAM_BATTING_SO  -0.052917   0.009361  -5.653 1.78e-08 ***
## TEAM_BASERUN_SB   0.042941   0.004485   9.574  < 2e-16 ***
## TEAM_PITCHING_H   0.015293   0.002345   6.520 8.68e-11 ***
## TEAM_PITCHING_HR -0.107818   0.051486  -2.094 0.036365 *
## TEAM_PITCHING_BB -0.076129   0.014327  -5.314 1.18e-07 ***
## TEAM_PITCHING_SO  0.041055   0.008313   4.939 8.46e-07 ***
## TEAM_FIELDING_E  -0.038959   0.003557 -10.952  < 2e-16 ***
## TEAM_FIELDING_DP -0.077138   0.013026  -5.922 3.68e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.84 on 2204 degrees of freedom
## Multiple R-squared:  0.2998, Adjusted R-squared:  0.2957
## F-statistic: 72.59 on 13 and 2204 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(model2)
```



Diagnostics plots

Model 3

This model will focus only on the variables that are maximally statistically significant - in order to see if only those variables allow for a better model.

Summary and vif

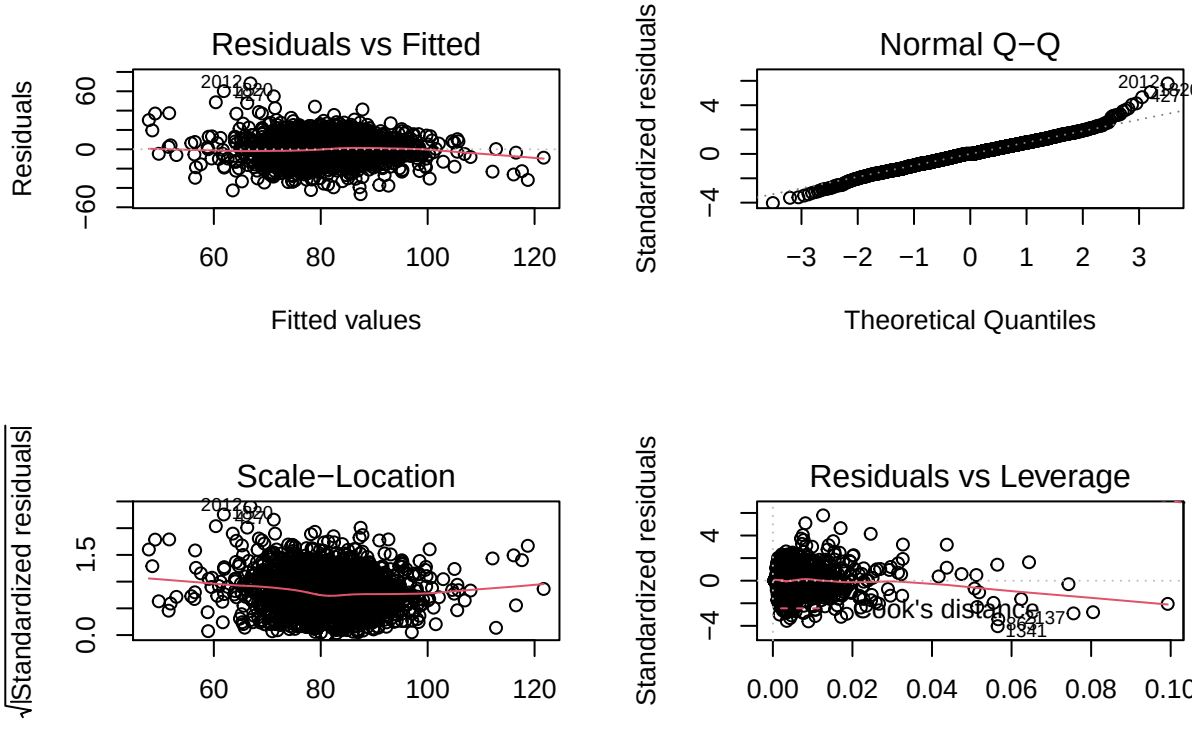
```
##
## Call:
## lm(formula = TARGET_WINS ~ . - TEAM_BASERUN_CS - TEAM_BATTING_HBP -
##     TEAM_BATTING_H - TEAM_BATTING_2B - TEAM_PITCHING_HR, data = MB_train_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.454  -7.537   0.000   7.476  68.149
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      41.097981   4.186019   9.818  < 2e-16 ***
## TEAM_BATTING_3B   0.151931   0.016258   9.345  < 2e-16 ***
## TEAM_BATTING_HR   0.095681   0.008798  10.876  < 2e-16 ***
## TEAM_BATTING_BB   0.122744   0.013833   8.873  < 2e-16 ***
## TEAM_BATTING_SO  -0.042450   0.008542  -4.970 7.22e-07 ***
## TEAM_BASERUN_SB   0.044993   0.004371  10.294  < 2e-16 ***
## TEAM_PITCHING_H   0.019730   0.001533  12.873  < 2e-16 ***
## TEAM_PITCHING_BB -0.093053   0.012048  -7.724 1.70e-14 ***
## TEAM_PITCHING_SO  0.029066   0.007313   3.975 7.27e-05 ***
## TEAM_FIELDING_E  -0.037287   0.003158 -11.805  < 2e-16 ***
```

```
## TEAM_FIELDING_DP -0.072405   0.012752  -5.678 1.54e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.86 on 2207 degrees of freedom
## Multiple R-squared:  0.2963, Adjusted R-squared:  0.2932
## F-statistic: 92.95 on 10 and 2207 DF,  p-value: < 2.2e-16
```

```
vif(model3)
```

```
##  TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO
##         2.895882         3.945930        33.232173        57.805669
##  TEAM_BASERUN_SB  TEAM_PITCHING_H TEAM_PITCHING_BB TEAM_PITCHING_SO
##         2.052647         4.955136        25.999917        43.298983
##  TEAM_FIELDING_E TEAM_FIELDING_DP
##         4.233002         1.247533
```

```
par(mfrow=c(2,2))
plot(model3)
```



Diagnostics plots

Looking at the plots above we can see a fairly linear Q-Q plot outside of the extreme values. The standardized residuals also appear to be fairly randomly distributed, another good sign.

Oddly enough, the strikeout metrics (along with the Fielding DP) have the highest p-values, so we can further simplify our model by removing them and see how that impacts our model.

```
model4 <- lm(TARGET_WINS ~ . - TEAM_BASERUN_CS - TEAM_BATTING_HBP - TEAM_BATTING_H    - TEAM_BATTING_2B
summary(model4)
```

```
##
```

```
## Call:
## lm(formula = TARGET_WINS ~ . - TEAM_BASERUN_CS - TEAM_BATTING_HBP -
##     TEAM_BATTING_H - TEAM_BATTING_2B - TEAM_PITCHING_HR - TEAM_BATTING_SO -
##     TEAM_PITCHING_SO - TEAM_FIELDING_DP, data = MB_train_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.676  -7.720   0.000   7.681  65.082
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      17.910403   2.768168   6.470  1.2e-10 ***
## TEAM_BATTING_3B   0.169952   0.014891  11.413  < 2e-16 ***
## TEAM_BATTING_HR   0.060278   0.006989   8.624  < 2e-16 ***
## TEAM_BATTING_BB   0.088487   0.008928   9.911  < 2e-16 ***
## TEAM_BASERUN_SB   0.038427   0.003988   9.635  < 2e-16 ***
## TEAM_PITCHING_H   0.024650   0.001375  17.923  < 2e-16 ***
## TEAM_PITCHING_BB -0.062429   0.007514  -8.308  < 2e-16 ***
## TEAM_FIELDING_E  -0.035933   0.003059 -11.746  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.02 on 2210 degrees of freedom
## Multiple R-squared:  0.2765, Adjusted R-squared:  0.2742
## F-statistic: 120.7 on 7 and 2210 DF,  p-value: < 2.2e-16
```

Prediction

Let's attempt a prediction using the latest model. To do so we must take a subset of our test dataset that includes only the columns we choose to use in the prediction.

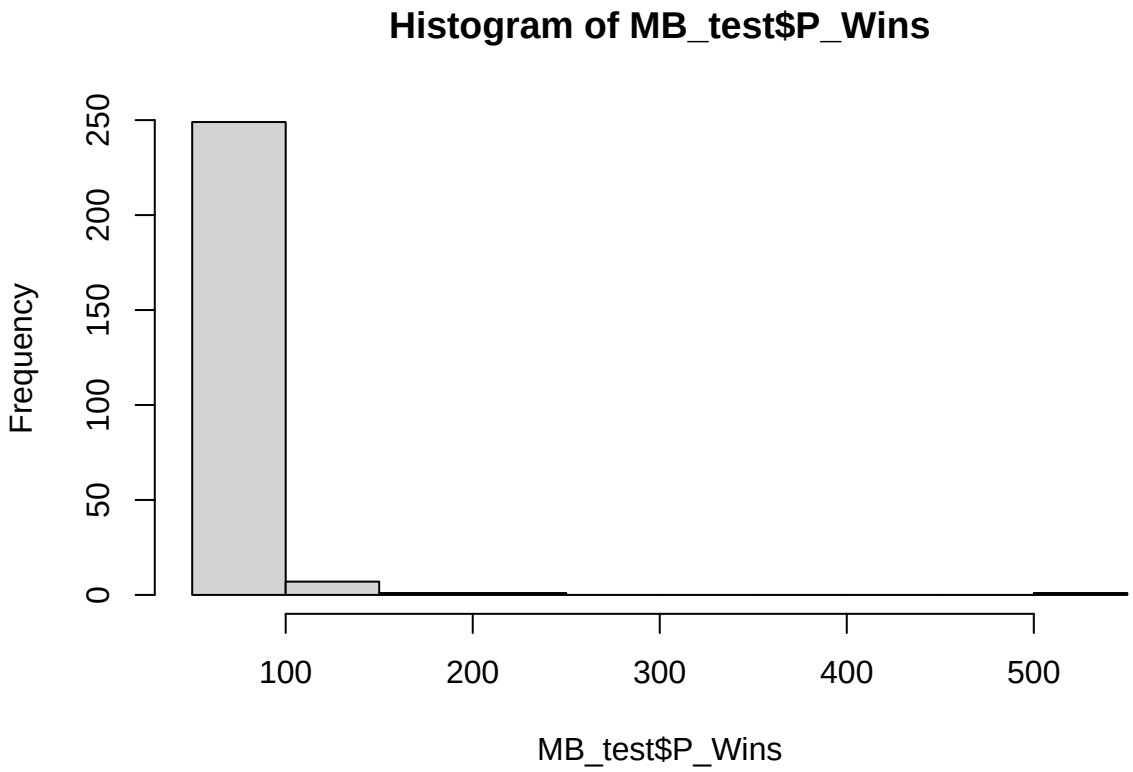
```
MB_test<- read.csv("https://raw.githubusercontent.com/mkollontai/DATA621_GroupWork/master/HW1/moneyball
```

Let's identify all of the rows containing obvious outliers in the data:

```
Out <- function(x){
  if (x[2] > most_hits * 1.25 | x[11] > most_hits*1.25) {
    return (1)
  } else if (x[7] > Max_SO | x[14] > Max_SO) {
    return (1)
  } else if (x[15] > max_errors){
    return (1)
  } else {
    return (0)
  }
}
```

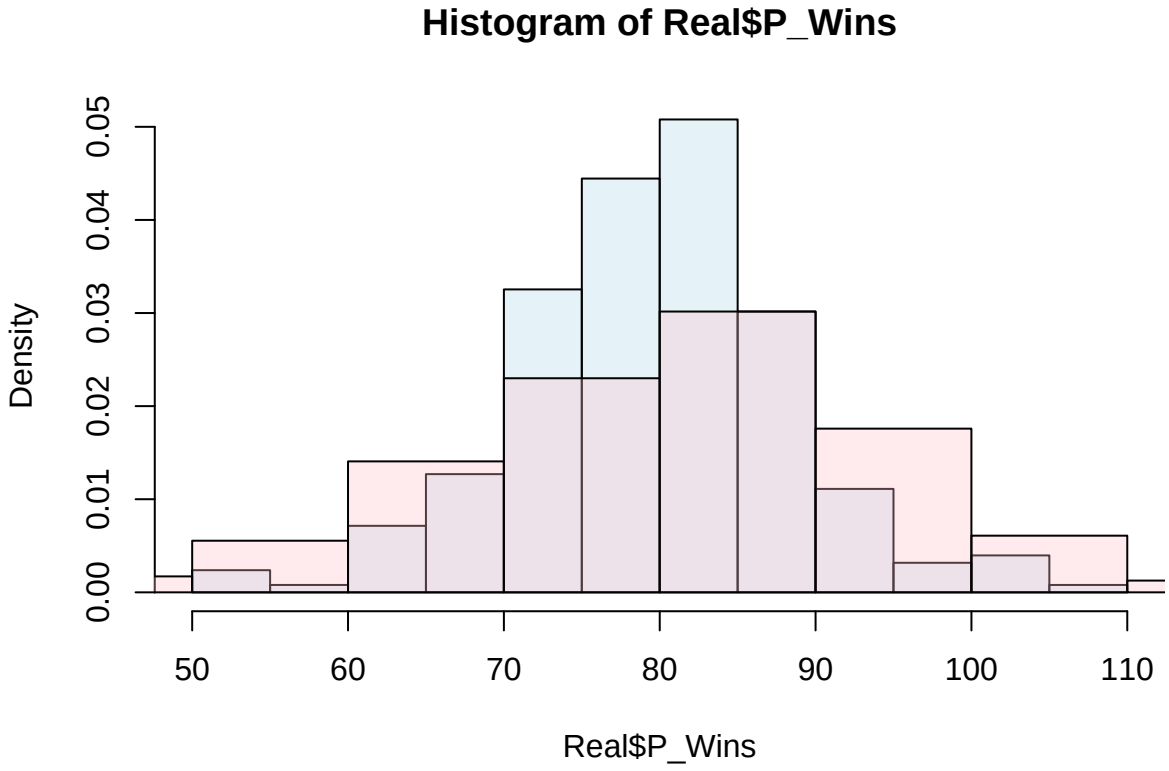
```
#First we replace NAs with the median values
for(i in 1:ncol(MB_test)){
  MB_test[is.na(MB_test[,i]), i] <- median(MB_test[,i], na.rm = TRUE)
}

MB_test$Outliers <- apply(MB_test, 1, FUN = Out)
MB_test$P_Wins <- round(predict(model4,  newdata = MB_test),0)
hist(MB_test$P_Wins)
```



Based on this histogram we can see that there are some obvious outliers in our predictions. Let us look only at the rows not tagged as having outliers in the variables used for prediction.

```
plot(histA, col = c1, freq = FALSE)
plot(histB, col = c2, freq = FALSE, add = TRUE, xlim = c(0,150))
```



This histogram overlayed on our original data suggest our predictions at least follow the values of the training dataset. The predicted data is in blue and the training data is in pink. The distribution seems close to normal, centered around 80/90 wins. The training data showed a peak between 80-85. There is nothing from this distribution that immediately jumps out as erroneous. The removal of the "Outlier" rows seems to have helped with that.

We can't test out the accuracy of our model as we don't have the true 'Win' values for our test dataset, but below see a list of the predicted wins for all of the rows not containing outliers in the data:

```
Real[c('INDEX','P_Wins')]
```

```
##     INDEX P_Wins
## 1       9     67
## 2      10     69
## 3      14     73
## 4      47     84
## 5      60    100
## 6      63     72
## 7      74     76
## 8      83     68
## 9      98     73
## 10    120     71
## 11    123     71
## 12    135     83
## 13    138     84
## 14    140     78
## 15    151     76
## 16    153     76
## 17    171     72
## 18    184     81
## 19    193     64
## 20    213     89
## 21    217     83
## 22    226     83
## 23    230     83
## 24    241     72
## 25    291     79
## 26    294     83
## 28    348     71
## 29    350     81
## 30    357     70
## 31    367     90
## 32    368     86
## 33    372     86
## 34    382     87
## 35    388     82
## 36    396     82
## 37    398     77
## 38    403     90
## 39    407     83
## 40    410     85
## 41    412     80
## 42    414     86
## 44    440    104
## 45    476     88
```

```
## 46    479     91
## 47    481     94
## 48    501     74
## 49    503     69
## 50    506     75
## 51    519     76
## 52    522     80
## 53    550     79
## 54    554     74
## 55    566     75
## 56    578     76
## 57    596     90
## 58    599     73
## 59    605     63
## 60    607     77
## 61    614     85
## 62    644     83
## 63    692     84
## 64    699     85
## 65    700     83
## 66    716     97
## 67    721     73
## 68    722     78
## 69    729     78
## 70    731     90
## 71    746     90
## 72    763     74
## 73    774     83
## 74    776     86
## 75    788     76
## 76    789     83
## 77    792     85
## 78    811     80
## 79    835     69
## 80    837     74
## 81    861     83
## 82    862     88
## 83    863     95
## 84    871     79
## 85    879     85
## 86    887     76
## 87    892     78
## 88    904     81
## 89    909     84
## 90    925     91
## 91    940     76
## 93    976     70
## 94    981     77
## 95    983     78
## 96    984     79
## 97    989     90
## 98    995     96
## 99   1000     89
## 100  1001     93
```

```
## 101  1007     83
## 102  1016     72
## 103  1027     81
## 104  1033     81
## 105  1070     81
## 106  1081     81
## 107  1084     51
## 108  1098     81
## 109  1150     86
## 110  1160     58
## 111  1169     86
## 112  1172     86
## 113  1174     90
## 114  1176     87
## 115  1178     80
## 116  1184     81
## 117  1193     90
## 118  1196     83
## 119  1199     75
## 120  1207     73
## 121  1218     90
## 122  1223     65
## 123  1226     67
## 124  1227     62
## 125  1229     72
## 126  1241     81
## 127  1244     85
## 128  1246     74
## 129  1248     87
## 130  1249     91
## 131  1253     83
## 132  1261     78
## 133  1305     74
## 134  1314     84
## 135  1323     86
## 136  1328     68
## 137  1353     79
## 138  1363     74
## 139  1371     85
## 140  1372     80
## 141  1389     63
## 142  1393     70
## 143  1421     92
## 144  1431     77
## 145  1437     75
## 146  1442     75
## 147  1450     80
## 148  1463     81
## 149  1464     82
## 150  1470     81
## 151  1471     84
## 152  1484     80
## 154  1507     68
## 155  1514     76
```

```
## 156  1526     73
## 157  1549     89
## 158  1552     63
## 159  1556     90
## 160  1564     69
## 161  1585    104
## 162  1586    105
## 163  1590     91
## 164  1591    104
## 165  1592     97
## 166  1603     92
## 167  1612     86
## 168  1634     82
## 169  1645     74
## 170  1647     80
## 171  1673     91
## 172  1674     87
## 173  1687     80
## 174  1688     88
## 175  1700     81
## 176  1708     78
## 177  1713     79
## 178  1717     75
## 179  1721     75
## 180  1730     81
## 181  1737     87
## 182  1748     87
## 183  1749     84
## 184  1763     85
## 186  1778     92
## 187  1780     83
## 188  1782     52
## 189  1784     55
## 190  1794    104
## 191  1803     65
## 192  1804     78
## 193  1819     72
## 194  1832     74
## 195  1833     75
## 196  1844     65
## 197  1847     73
## 198  1854     84
## 199  1855     81
## 200  1857     86
## 201  1864     79
## 202  1865     80
## 203  1869     78
## 204  1880     85
## 205  1881     78
## 206  1882     83
## 207  1894     80
## 208  1896     78
## 209  1916     79
## 210  1918     71
```

```
## 211  1921    106
## 212  1926     90
## 213  1938     87
## 214  1979     68
## 215  1982     75
## 216  1987     84
## 217  1997     84
## 218  2004     91
## 219  2011     75
## 220  2015     80
## 221  2022     80
## 222  2025     76
## 223  2027     84
## 224  2031     78
## 226  2066     78
## 227  2073     77
## 228  2087     84
## 229  2092     83
## 230  2125     67
## 231  2148     74
## 232  2162     92
## 233  2191     84
## 234  2203     84
## 235  2218     79
## 236  2221     74
## 237  2225     79
## 238  2232     80
## 239  2267     91
## 240  2291     75
## 241  2299     88
## 242  2317     86
## 243  2318     80
## 244  2353     82
## 245  2403     65
## 246  2411     81
## 247  2415     75
## 248  2424     84
## 249  2441     72
## 250  2464     87
## 251  2465     84
## 252  2472     68
## 253  2481     90
## 255  2500     69
## 256  2501     78
## 257  2520     80
## 258  2521     83
## 259  2525     78
```

Potential Future Work

One approach we looked into pursuing was to split up the data into 'Batting', 'Baserun', 'Pitching' and 'Fielding' data and investigating whether or not some trends hold true for one category, but not others. Perhaps apply uniform coefficients across the categories.