Homework 01

CUNY MSDS DATA 621

Duubar Villalobos Jimenez

Homework #1

Overview

In this homework assignment, you 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.

Your objective is to build a multiple linear regression model on the training data to predict the number of wins for the team. You can only use the variables given to you (or variables that you derive from the variables provided).

Deliverables:

- A write-up submitted in PDF format. Your write-up should have four sections. Each one is described below. You may assume you are addressing me as a fellow data scientist, so do not need to shy away from technical details.
- Assigned predictions (the number of wins for the team) for the evaluation data set.
- Include your R statistical programming code in an Appendix.

1. DATA EXPLORATION

Data acquisition

First, we need to explore our given data set. For reproducibility purposes, I have put the original data sets in my git-hub account and then I will read them from there.

```
git_user <- 'https://raw.githubusercontent.com/dvillalobos/'
git_dir <- 'MSDS/master/621/Homeworks/assignment-01/data/'
baseball.train = read.csv(paste(git_user, git_dir, "moneyball-training-data.csv", sep = ""))
baseball.eval = read.csv(paste(git_user, git_dir, "moneyball-evaluation-data.csv", sep = ""))
```

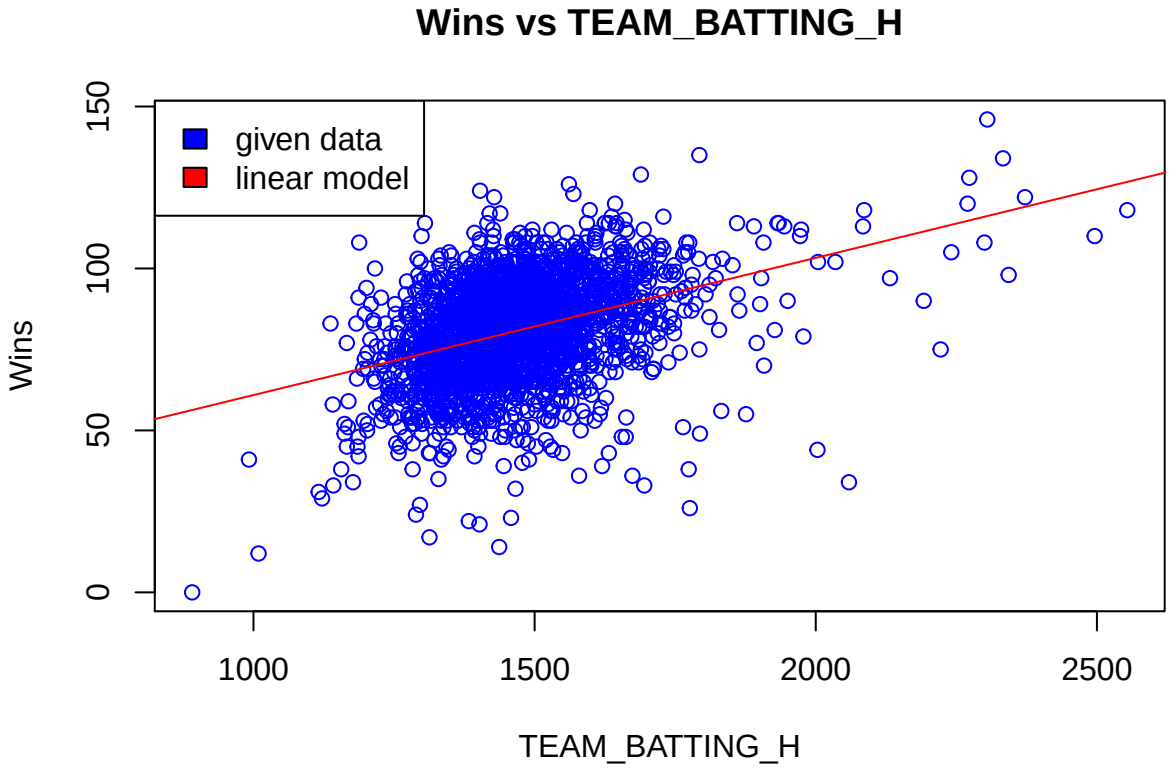
Simple Example

This example will help determine the ideas to follow in order to solve our problem; this is for explanatory purposes on how this problem will be approached.

```
lm.tr <- lm(baseball.train$TARGET_WINS ~ baseball.train$TEAM_BATTING_H)
summary(lm.tr)
```

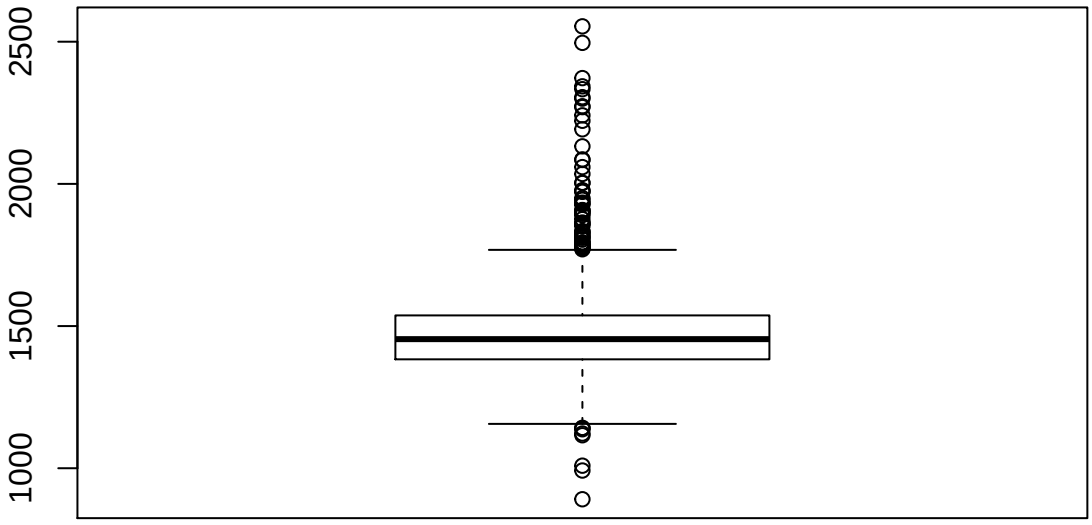
```
##
## Call:
## lm(formula = baseball.train$TARGET_WINS ~ baseball.train$TEAM_BATTING_H)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -71.768  -8.757   0.856   9.762  46.016
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  18.562326   3.107523   5.973 2.69e-09 ***
## baseball.train$TEAM_BATTING_H  0.042353   0.002105  20.122  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.52 on 2274 degrees of freedom
## Multiple R-squared:  0.1511, Adjusted R-squared:  0.1508
## F-statistic: 404.9 on 1 and 2274 DF,  p-value: < 2.2e-16
```

Since, this is just an example, I would not describe much to it at this point in time.



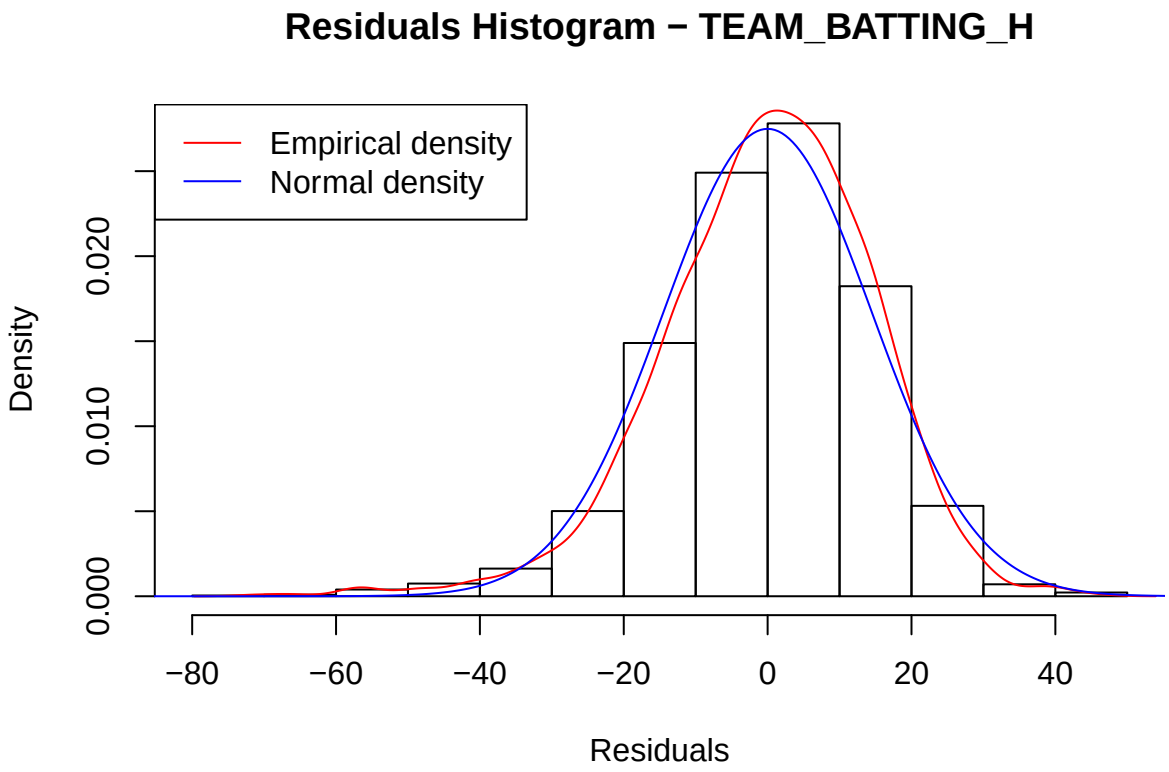
Let's see a more detailed box plot from our original given data.

```
boxplot(baseball.train$TEAM_BATTING_H,
      data=baseball.train,
      main=paste('Box Plot - TEAM_BATTING_H'),
      xlab='TEAM_BATTING_H')
```



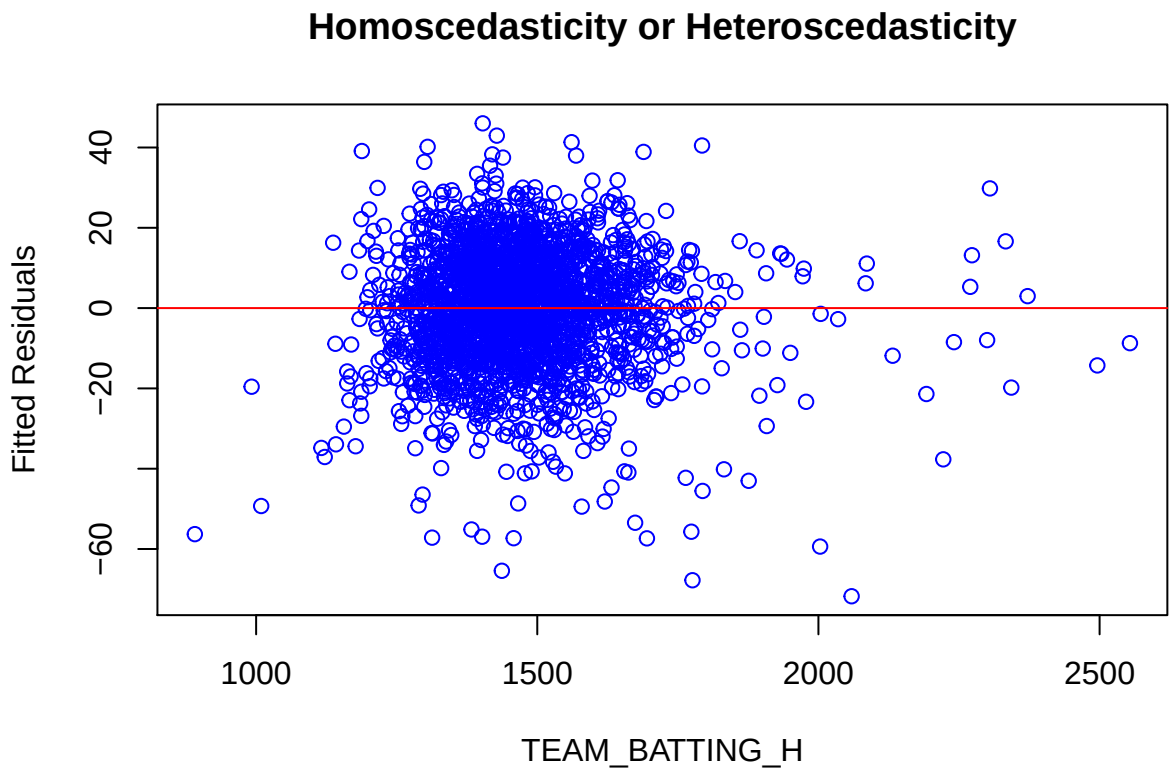
Let's see how the residuals histogram plot from our linear model data look like.

```
hist(lm.tr$residuals, freq = FALSE,
   main = paste('Residuals Histogram - TEAM_BATTING_H'),
   xlab = 'Residuals',
   ylab = 'Density')
lines(density(lm.tr$residuals), col="red")
lines(seq(-400, 500, by=.5),
    dnorm(seq(-400, 500, by=.5),
          mean(lm.tr$residuals),
          sd(lm.tr$residuals)),
    col="blue")
lnames <- c('Empirical density', 'Normal density')
legend('topleft', lnames, col = c('red','blue'), lty = 1)
```



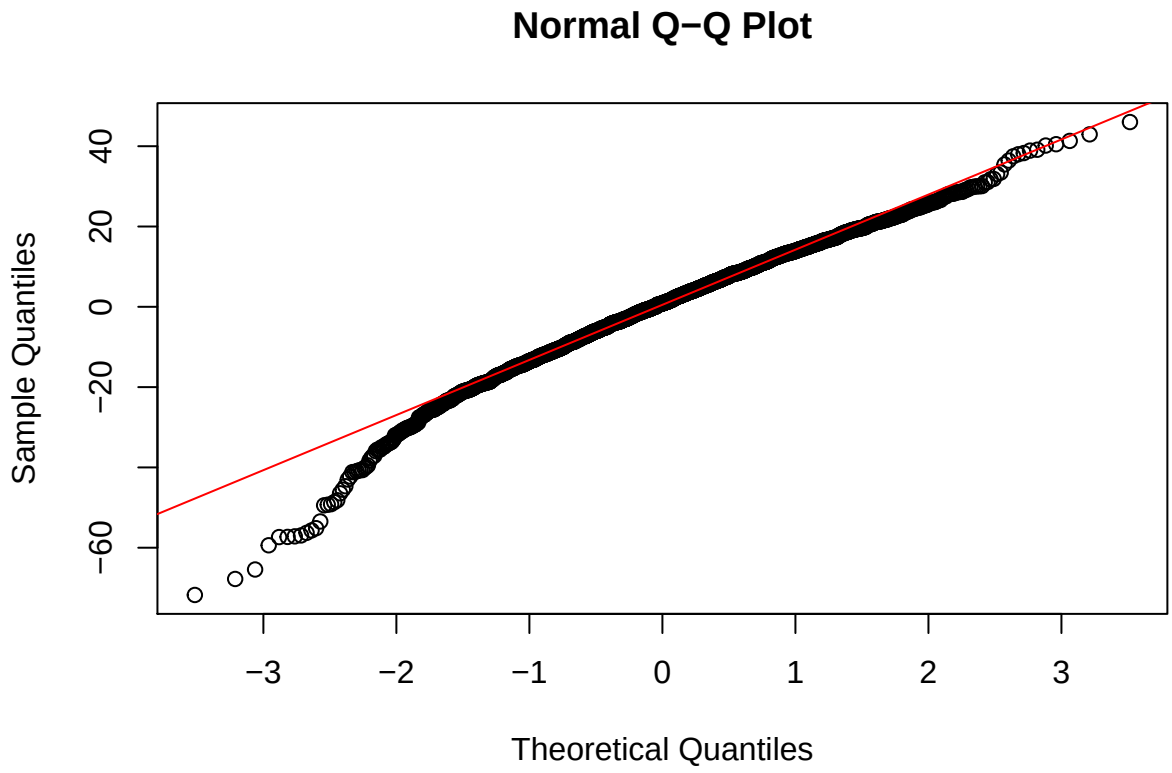
Let's plot our residuals data in order to visualize Homoscedasticity or Heteroscedasticity.

```
plot(lm.tr$residuals ~ baseball.train$TEAM_BATTING_H,
      type="p",
      col="blue",
      main='Homoscedasticity or Heteroscedasticity',
      xlab='TEAM_BATTING_H',
      ylab="Fitted Residuals")
abline(h=0, col="red")
```



Let's plot and visualize a Q-Q Plot to determine normality.

```
qqnorm(resid(lm.tr))
qqline(resid(lm.tr), col='red')
```



As you can see, from our previous example, we can deduce a few things, in particular, that in this case a

single predictor variable might not be enough to predict the **TARGET_WINS**.

General exploration

The below process will help obtain insights from the data.

Dimensions

Let's see the dimensions of our data set.

```
dim(baseball.train)
```

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

As we can notice, the training data set has a total of 17 different variables. The total number or records available are 2276.

Structure

The below structure is currently present in the data, for simplicity purposes, I have loaded and treated this data set as a data frame in which all the variables are integers.

```
str(baseball.train)
```

```
## 'data.frame':    2276 obs. of  17 variables:
##  $ INDEX           : int  1 2 3 4 5 6 7 8 11 12 ...
##  $ TARGET_WINS     : int  39 70 86 70 82 75 80 85 86 76 ...
##  $ TEAM_BATTING_H  : int  1445 1339 1377 1387 1297 1279 1244 1273 1391 1271 ...
##  $ TEAM_BATTING_2B : int  194 219 232 209 186 200 179 171 197 213 ...
##  $ TEAM_BATTING_3B : int  39 22 35 38 27 36 54 37 40 18 ...
##  $ TEAM_BATTING_HR : int  13 190 137 96 102 92 122 115 114 96 ...
##  $ TEAM_BATTING_BB : int  143 685 602 451 472 443 525 456 447 441 ...
##  $ TEAM_BATTING_SO : int  842 1075 917 922 920 973 1062 1027 922 827 ...
##  $ TEAM_BASERUN_SB : int  NA 37 46 43 49 107 80 40 69 72 ...
##  $ TEAM_BASERUN_CS : int  NA 28 27 30 39 59 54 36 27 34 ...
##  $ TEAM_BATTING_HBP: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ TEAM_PITCHING_H : int  9364 1347 1377 1396 1297 1279 1244 1281 1391 1271 ...
##  $ TEAM_PITCHING_HR: int  84 191 137 97 102 92 122 116 114 96 ...
##  $ TEAM_PITCHING_BB: int  927 689 602 454 472 443 525 459 447 441 ...
##  $ TEAM_PITCHING_SO: int  5456 1082 917 928 920 973 1062 1033 922 827 ...
##  $ TEAM_FIELDING_E : int  1011 193 175 164 138 123 136 112 127 131 ...
##  $ TEAM_FIELDING_DP: int  NA 155 153 156 168 149 186 136 169 159 ...
```

Summary

The below is a summary extracted from our given training data.

```
train.summary <- data.frame(unclass(summary(baseball.train[2:17])),
                            check.names = FALSE,
                            row.names = NULL,
                            stringsAsFactors = FALSE)
train.summary
```

```
##        TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B  TEAM_BATTING_3B
## 1 Min.   :  0.00   Min.   : 891   Min.   : 69.0   Min.   :  0.00
## 2 1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0   1st Qu.: 34.00
## 3 Median : 82.00   Median :1454   Median :238.0   Median : 47.00
## 4 Mean   : 80.79   Mean   :1469   Mean   :241.2   Mean   : 55.25
## 5 3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0   3rd Qu.: 72.00
## 6 Max.   :146.00   Max.   :2554   Max.   :458.0   Max.   :223.00
## 7             <NA>           <NA>           <NA>             <NA>
##    TEAM_BATTING_HR TEAM_BATTING_BB  TEAM_BATTING_SO TEAM_BASERUN_SB
## 1 Min.   :  0.00   Min.   :  0.0   Min.   :   0.0   Min.   :  0.0
## 2 1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 548.0   1st Qu.: 66.0
## 3 Median :102.00   Median :512.0   Median : 750.0   Median :101.0
## 4 Mean   : 99.61   Mean   :501.6   Mean   : 735.6   Mean   :124.8
## 5 3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 930.0   3rd Qu.:156.0
## 6 Max.   :264.00   Max.   :878.0   Max.   :1399.0   Max.   :697.0
## 7             <NA>            <NA>    NA's   :102     NA's   :131
##   TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
## 1 Min.   :  0.0    Min.   :29.00   Min.   : 1137    Min.   :  0.0
## 2 1st Qu.: 38.0    1st Qu.:50.50   1st Qu.: 1419    1st Qu.: 50.0
## 3 Median : 49.0    Median :58.00   Median : 1518    Median :107.0
## 4 Mean   : 52.8    Mean   :59.36   Mean   : 1779    Mean   :105.7
## 5 3rd Qu.: 62.0    3rd Qu.:67.00   3rd Qu.: 1682    3rd Qu.:150.0
## 6 Max.   :201.0    Max.   :95.00   Max.   :30132    Max.   :343.0
## 7   NA's   :772     NA's   :2085              <NA>             <NA>
##   TEAM_PITCHING_BB  TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
## 1 Min.   :   0.0   Min.   :    0.0   Min.   :  65.0    Min.   : 52.0
## 2 1st Qu.: 476.0   1st Qu.:  615.0   1st Qu.: 127.0    1st Qu.:131.0
## 3 Median : 536.5   Median :  813.5   Median : 159.0    Median :149.0
## 4 Mean   : 553.0   Mean   :  817.7   Mean   : 246.5    Mean   :146.4
## 5 3rd Qu.: 611.0   3rd Qu.:  968.0   3rd Qu.: 249.2    3rd Qu.:164.0
## 6 Max.   :3645.0   Max.   :19278.0   Max.   :1898.0    Max.   :228.0
## 7             <NA>     NA's   :102               <NA>    NA's   :286
```

From the summary statistics, we can identify the need to work with some **NA's** in diverse columns. There's no clear answer as to why these values are not available or included in the data set, each variable needs to be analyzed individually and find the best approach.

```
train.summary$TEAM_BATTING_SO[7]
```

```
## [1] "NA's   :102  "
```

```
train.summary$TEAM_BASERUN_SB[7]
```

```
## [1] "NA's   :131  "
```

```
train.summary$TEAM_BASERUN_CS[7]
```

```
## [1] "NA's   :772  "
```

```
train.summary$TEAM_BATTING_HBP[7]
```

```
## [1] "NA's   :2085  "
```

```
train.summary$TEAM_PITCHING_SO[7]
```

```
## [1] "NA's   :102  "
```

```
train.summary$TEAM_FIELDING_DP[7]
```

```
## [1] "NA's   :286  "
```

Also, there's a need to find out a little bit more information related to **ZERO** values, these are reported in the minimum value for some of the columns. Need to verify if it is feasible to have such values, could these values considered as entry errors? we need to analyze them in more detail with hopes of avoid possible misconceptions for the model.

```
train.summary$` TARGET_WINS`[1]
```

```
## [1] "Min.   :  0.00  "
```

```
train.summary$TEAM_BATTING_3B[1]
```

```
## [1] "Min.   :  0.00  "
```

```
train.summary$TEAM_BATTING_HR[1]
```

```
## [1] "Min.   :  0.00  "
```

```
train.summary$TEAM_BATTING_BB[1]
```

```
## [1] "Min.   :  0.0  "
```

```
train.summary$TEAM_BATTING_SO[1]
```

```
## [1] "Min.   :   0.0  "
```

```
train.summary$TEAM_BASERUN_SB[1]
```

```
## [1] "Min.   :  0.0  "
```

```
train.summary$TEAM_BASERUN_CS[1]
```

```
## [1] "Min.   :  0.0  "
```

```
train.summary$TEAM_PITCHING_HR[1]
```

```
## [1] "Min.   :  0.0  "
```

```
train.summary$TEAM_PITCHING_BB[1]
```

```
## [1] "Min.   :   0.0  "
```

```
train.summary$TEAM_PITCHING_SO[1]
```

```
## [1] "Min.   :    0.0  "
```

```
train.summary$TEAM_PITCHING_HR[1]
```

```
## [1] "Min.   :  0.0  "
```

```
train.summary$TEAM_PITCHING_BB[1]
```

```
## [1] "Min.   :   0.0  "
```

```
train.summary$TEAM_PITCHING_SO[1]
```

```
## [1] "Min.   :    0.0  "
```

Visualizations

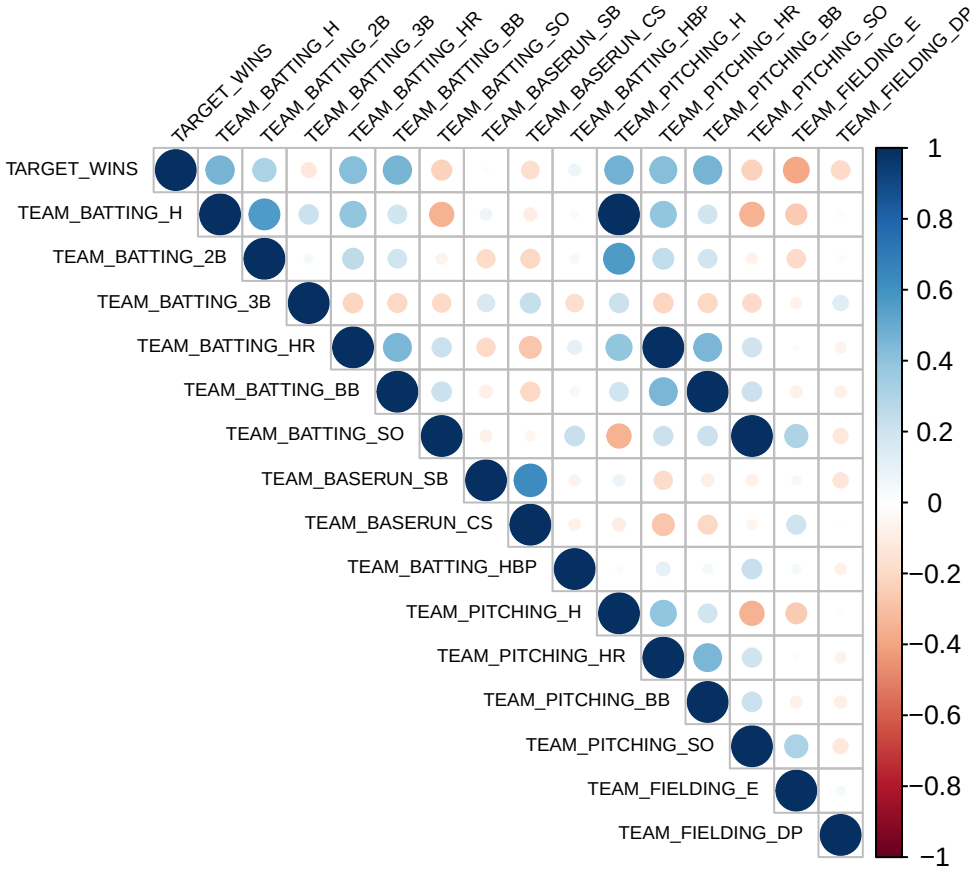
Let's compare the various relationships, in particular we will focus on the first row since that row represent the **y_axis = TARGET_WINS** vs the other variable on the **x_axis**. The idea is to imagine some sort of linearity in between **TARGET_WINS** vs the other related variable on a one by one cases.

Correlation matrix

Let's do some visualizations for the correlation matrix.

```
my_matrix <- baseball.train[c(2:17)]
cor_res <- cor(my_matrix, use = "na.or.complete")
```

```
corrplot(cor_res,
         type = "upper",
         order = "original",
         tl.col = "black",
         tl.srt = 45,
         tl.cex = 0.55)
```



Let's read the correlations respective values:

```
##                  TARGET_WINS
## TARGET_WINS       1.00000000
## TEAM_BATTING_H    0.46994665
## TEAM_BATTING_2B   0.31298400
## TEAM_BATTING_3B  -0.12434586
## TEAM_BATTING_HR   0.42241683
```

```
## TEAM_BATTING_BB   0.46868793
## TEAM_BATTING_SO  -0.22889273
## TEAM_BASERUN_SB   0.01483639
## TEAM_BASERUN_CS  -0.17875598
## TEAM_BATTING_HBP  0.07350424
## TEAM_PITCHING_H   0.47123431
## TEAM_PITCHING_HR  0.42246683
## TEAM_PITCHING_BB  0.46839882
## TEAM_PITCHING_SO -0.22936481
## TEAM_FIELDING_E  -0.38668800
## TEAM_FIELDING_DP -0.19586601
```

In particular, is important to note some very strong positive correlations represented in the graph above among themselves.

some of them are:

- **TEAM_BATTING_H** is strongly correlated in a positive way with **TEAM_PITCHING_H**.
- **TEAM_BATTING_HR** is strongly correlated in a positive way with **TEAM_PITCHING_HR**.
- **TEAM_BATTING_BB** is strongly correlated in a positive way with **TEAM_PITCHING_BB**.
- **TEAM_BATTING_SO** is strongly correlated in a positive way with **TEAM_PITCHING_SO**.

In particular, I will extract the correlation values for the strong correlations identified above.

```
cor_res.df['TEAM_BATTING_H','TEAM_PITCHING_H']
```

```
## [1] 0.9991927
```

```
cor_res.df['TEAM_BATTING_HR','TEAM_PITCHING_HR']
```

```
## [1] 0.9999326
```

```
cor_res.df['TEAM_BATTING_BB','TEAM_PITCHING_BB']
```

```
## [1] 0.9998814
```

```
cor_res.df['TEAM_BATTING_SO','TEAM_PITCHING_SO']
```

```
## [1] 0.9997684
```

If we think about the process, those correlations make sense since this is a dual process in which one action tracks a response. **PITCHING** is correlated to **BATTING**.

From the above results, we could take one variable in function of the other one.

Something interesting to note is that in effect, the above identified correlations have very similar correlations as well with **TARGET_WINS**.

TEAM_BATTING_H 0.46994665 &
TEAM_PITCHING_H 0.47123431

TEAM_BATTING_HR 0.42241683 &
TEAM_PITCHING_HR 0.42246683

TEAM_BATTING_BB 0.46868793 &
TEAM_PITCHING_BB 0.46839882

TEAM_BATTING_SO -0.22889273 &
TEAM_PITCHING_SO -0.22936481

So, basically our initial table will be reduced as follows:

Original Table: Predictor elimination due to "repeated" value.

```
TARGET_WINS         <- To predict
TEAM_BATTING_H      <- Correlated to TEAM_PITCHING_H <- Discard
TEAM_BATTING_2B
TEAM_BATTING_3B
TEAM_BATTING_HR     <- Correlated to TEAM_PITCHING_HR <- Discard
TEAM_BATTING_BB     <- Correlated to TEAM_PITCHING_BB <- Keep
TEAM_BATTING_SO     <- Correlated to TEAM_PITCHING_SO <- Keep
TEAM_BASERUN_SB
TEAM_BASERUN_CS
TEAM_BATTING_HBP
TEAM_PITCHING_H     <- Correlated to TEAM_BATTING_H <- Keep
TEAM_PITCHING_HR    <- Correlated to TEAM_BATTING_HR <- Keep
TEAM_PITCHING_BB    <- Correlated to TEAM_BATTING_BB <- Discard
TEAM_PITCHING_SO    <- Correlated to TEAM_BATTING_SO <- Discard
TEAM_FIELDING_E
TEAM_FIELDING_DP
```

Since these correlations are related among themselves and are also related to **TARGET_WINS**, I will exclude the variables that have the smallest correlation related to **TARGET_WINS**.

New Reduced Table: Predictor eliminated due to "repeated" value.

```
TARGET_WINS         <- To predict
TEAM_BATTING_2B
TEAM_BATTING_3B
TEAM_BATTING_BB
TEAM_BATTING_SO
TEAM_BASERUN_SB
TEAM_BASERUN_CS
TEAM_BATTING_HBP
TEAM_PITCHING_H
TEAM_PITCHING_HR
TEAM_FIELDING_E
TEAM_FIELDING_DP
```

```r
exclude <- c('INDEX', 'TEAM_BATTING_H', 'TEAM_BATTING_HR', 'TEAM_PITCHING_BB', 'TEAM_PITCHING_SO')
newvars <- names( baseball.train) %in% exclude
reduced.train <- baseball.train[!newvars]
```

Primary insigths

Based on the above data exploration, we could note the following:

- **Missing values**

It is confirmed the presence of missing values and these need to be address in a case by case.

- **Zero values**

It is confirmed the presence of Zero values as minimum entries in the data. Need to verify and accept or reject the feasibility of such values for some of the variables.

- **Correlations**

There seems to be very strong correlations to the target variable.

In regards to correlations related to other variables, it is confirmed that such correlation exist and those 'duplicate' correlated variables among themselves were removed.

2. DATA PREPARATION

The following steps and/or assumptions will be considered. The idea is to make our given training data set more homogeneous and workable. The final goal is to be able to predict the **TARGET_WINS** with our data.

Missing values NA's

Previously, we identified the need to analyze in more detail these variables. The below is a list of variables that include NA's.

- **TEAM_BATTING_SO**
- TEAM_BASERUN_SB
- TEAM_BASERUN_CS
- TEAM_BATTING_HBP
- **TEAM_PITCHING_SO** <- Discarded
- TEAM_FIELDING_DP

From the above NA list, we previously identifies that **TEAM_BATTING_SO** and **TEAM_PITCHING_SO** were correlated among themselves and I have already eliminated the one with the lowest correlation of the two compared to **TARGET_WINS**.

Hence, our list of variables presenting **missing** values has been reduced to:

- TEAM_BATTING_SO
- TEAM_BASERUN_SB
- TEAM_BASERUN_CS
- TEAM_BATTING_HBP
- TEAM_FIELDING_DP

Proportionalities

Let's see the proportion of missing values in order to determine the best approach for these variables.

```
TEAM_BATTING_SO.p <- round(sum(is.na(reduced.train$TEAM_BATTING_SO))/dim(reduced.train)[1]*100,2)
TEAM_BASERUN_SB.p <- round(sum(is.na(reduced.train$TEAM_BASERUN_SB))/dim(reduced.train)[1]*100,2)
TEAM_BASERUN_CS.p <- round(sum(is.na(reduced.train$TEAM_BASERUN_CS))/dim(reduced.train)[1]*100,2)
TEAM_BATTING_HBP.p <- round(sum(is.na(reduced.train$TEAM_BATTING_HBP))/dim(reduced.train)[1]*100,2)
TEAM_FIELDING_DP.p <- round(sum(is.na(reduced.train$TEAM_FIELDING_DP))/dim(reduced.train)[1]*100,2)
```

The below table display the respective missing value percentages for each variable.

- TEAM_BATTING_SO = 4.48 % missing data.
- TEAM_BASERUN_SB = 5.76 % missing data.
- TEAM_BASERUN_CS = 33.92 % missing data.
- **TEAM_BATTING_HBP = 91.61 % missing data.**
- TEAM_FIELDING_DP = 12.57 % missing data.

From the above results and by analyzing the given data, we could "discard" the **TEAM_BATTING_HBP** due to the high percentage of missing data; particularly, replacing it by "ZERO" should not be advisable since the minimum value recorded is 29 and replacing it with a median value won't be advisable neither due

to the high percentage of missing values. From my perspective, is best not to consider that variable due to low impact and high inaccuracy.

```
exclude <- c('TEAM_BATTING_HBP')
newvars <- names( reduced.train) %in% exclude
reduced.train <- reduced.train[!newvars]
```

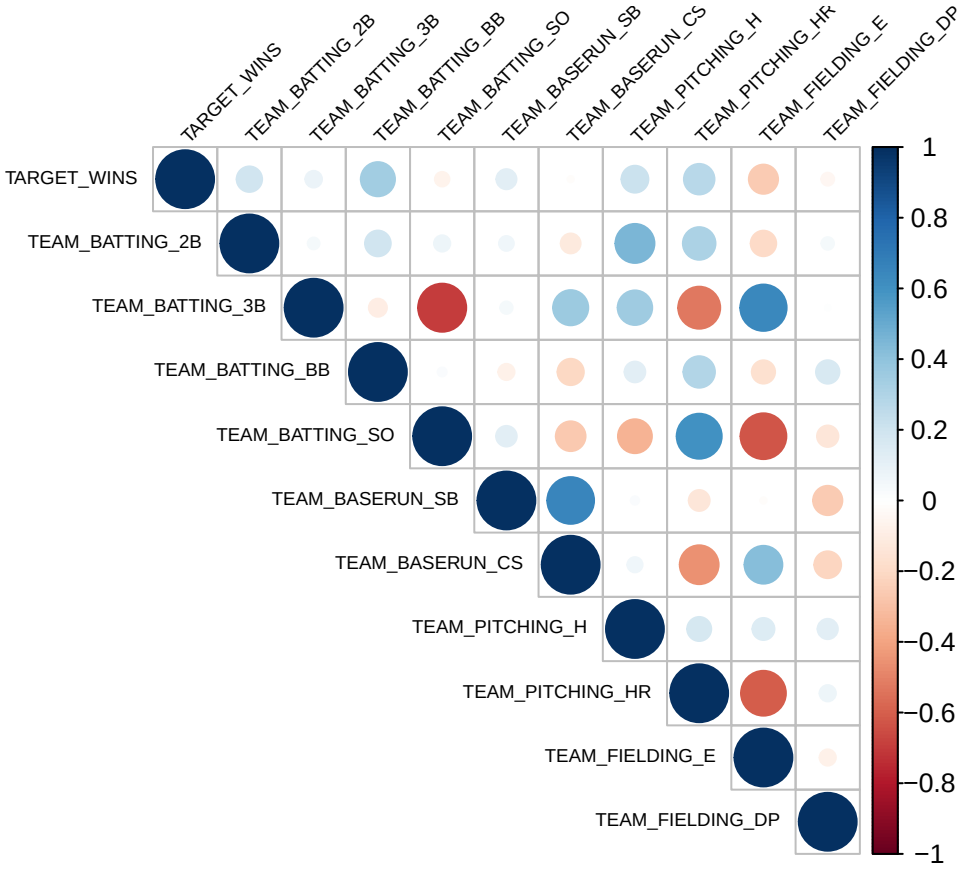
From here onward, I will try to obtain the best predictors in order to predict our **TARGET_WINS**.

Eliminating low correlated values

From our correlations table, we found strong correlations among our data; let's run a new correlation data but with the reduced amount of data.

```
my_matrix1 <- reduced.train
cor_res1 <- cor(my_matrix1, use = "na.or.complete")
```

```
corrplot(cor_res1,
         type = "upper",
         order = "original",
         tl.col = "black",
         tl.srt = 45,
         tl.cex = 0.55)
```



Let's read the correlations respective values:

```
##                    TARGET_WINS
## TARGET_WINS        1.00000000
```

```
## TEAM_BATTING_2B   0.19377063
## TEAM_BATTING_3B   0.08273171
## TEAM_BATTING_BB   0.34829184
## TEAM_BATTING_SO  -0.06289963
## TEAM_BASERUN_SB   0.12035129
## TEAM_BASERUN_CS  -0.01119995
## TEAM_PITCHING_H   0.21610375
## TEAM_PITCHING_HR  0.27872264
## TEAM_FIELDING_E  -0.25450741
## TEAM_FIELDING_DP -0.04949709
```

From the correlation visuals and the above table, we could eliminate a few more predictor variables, and the reason is due to it's low correlated value.

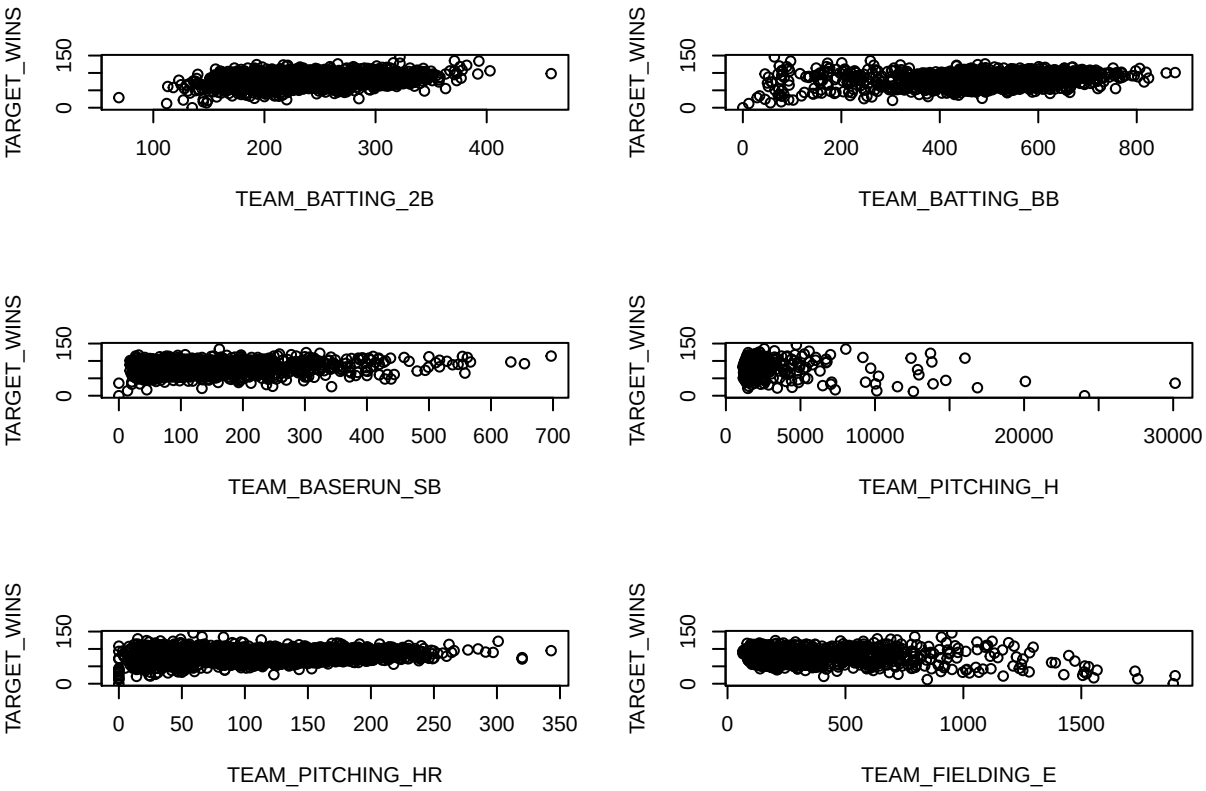
For example, we could eliminate the values whose absolute value is below 0.10, leaving the ones that are considered some how important to our model.

```
cor_res1.df <- cor_res1.df[which(abs(cor_res1.df$TARGET_WINS) > 0.10),]
cor_res1.df[1]
```

```
##                  TARGET_WINS
## TARGET_WINS        1.0000000
## TEAM_BATTING_2B    0.1937706
## TEAM_BATTING_BB    0.3482918
## TEAM_BASERUN_SB    0.1203513
## TEAM_PITCHING_H    0.2161037
## TEAM_PITCHING_HR   0.2787226
## TEAM_FIELDING_E   -0.2545074
```

```
exclude <- c('TEAM_BATTING_3B', 'TEAM_BATTING_SO', 'TEAM_BASERUN_CS', 'TEAM_FIELDING_DP')
newvars <- names( reduced.train) %in% exclude
reduced.train <- reduced.train[!newvars]
```

From here moving forward, we could look at the data composition with a little bit more detail.



From the above graphs, it seems that we have a need to reduce some excessive large values from our predictor variables in order o make our reduced table of predictors more workable.

A possible transformation is to use the square root on both sides of the model, this is advised since all the variables have the same units and also all of them are counts.

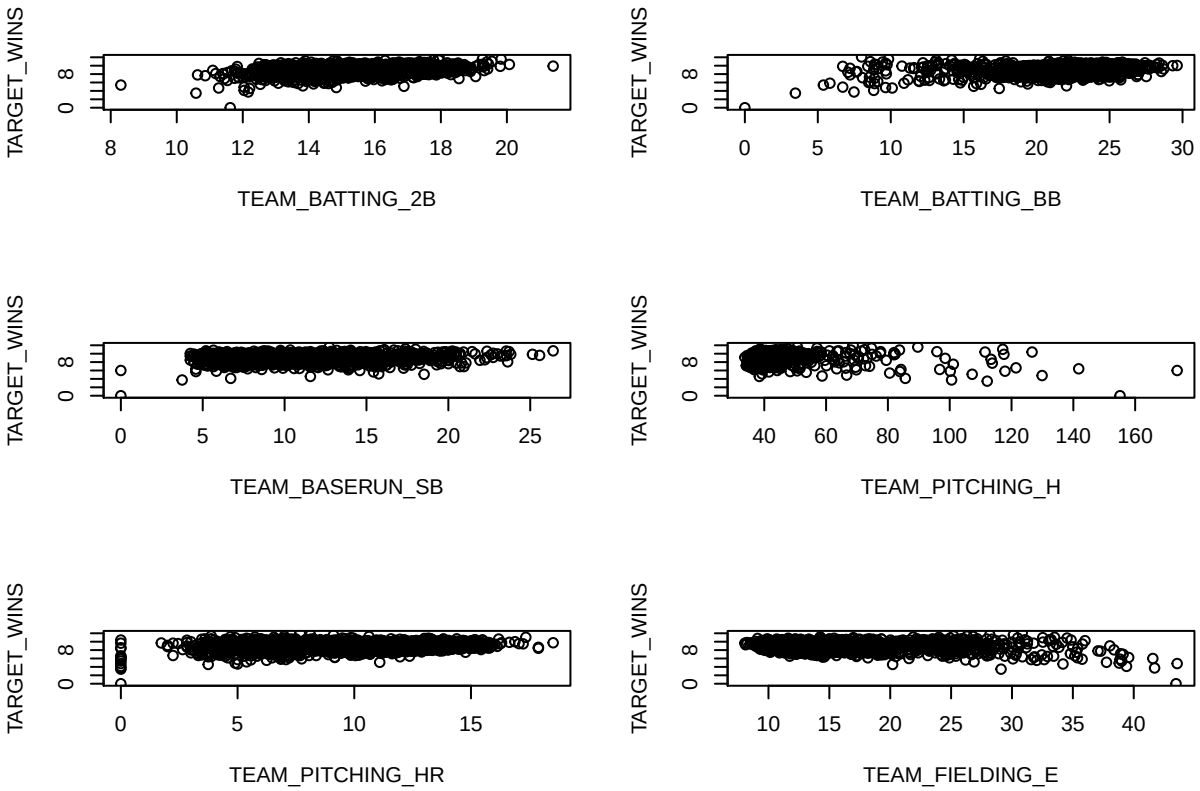
That is if we have a model:

$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_5 + e$

We could transform to:

$y^{'} = b_0^{'} + b_1^{'}x_1^{'} + b_2^{'}x_2^{'} + b_3^{'}x_3^{'} + b_4^{'}x_4^{'} + b_5^{'}x_5^{'} + e^{'}$

where $y^{'} = \sqrt{y}$ and each $x_i^{'} = \sqrt{x_i}$ with $b_i^{'}$ and $e^{'}$ are new constants obtained due to the transformation process.



From the above graphs, we can notice how our given data follows what seems to be linear, the only exception is the **TEAM_PITCHING_H**. We noticed how it has some very large values with low **TARGET_WINS**. Also, we noticed that the correlation of **TEAM_PITCHING_H** with **TARGET_WINS** is positive but our theoretical effect is that it should produce a negative impact on wins, thus contradicting our hypothesis; based on that and with the low distribution with large outliers, I will remove this variable as well.

```
exclude <- c('TEAM_PITCHING_H')
newvars <- names( reduced.train) %in% exclude
reduced.train <- reduced.train[!newvars]
```

3. BUILD MODELS

a) Build linear model with raw data.

In this section I will build a linear model utilizing the raw data, with no transformation and no changes whatsoever in order to have it as starting point and compare it's results with a more refined model down the road.

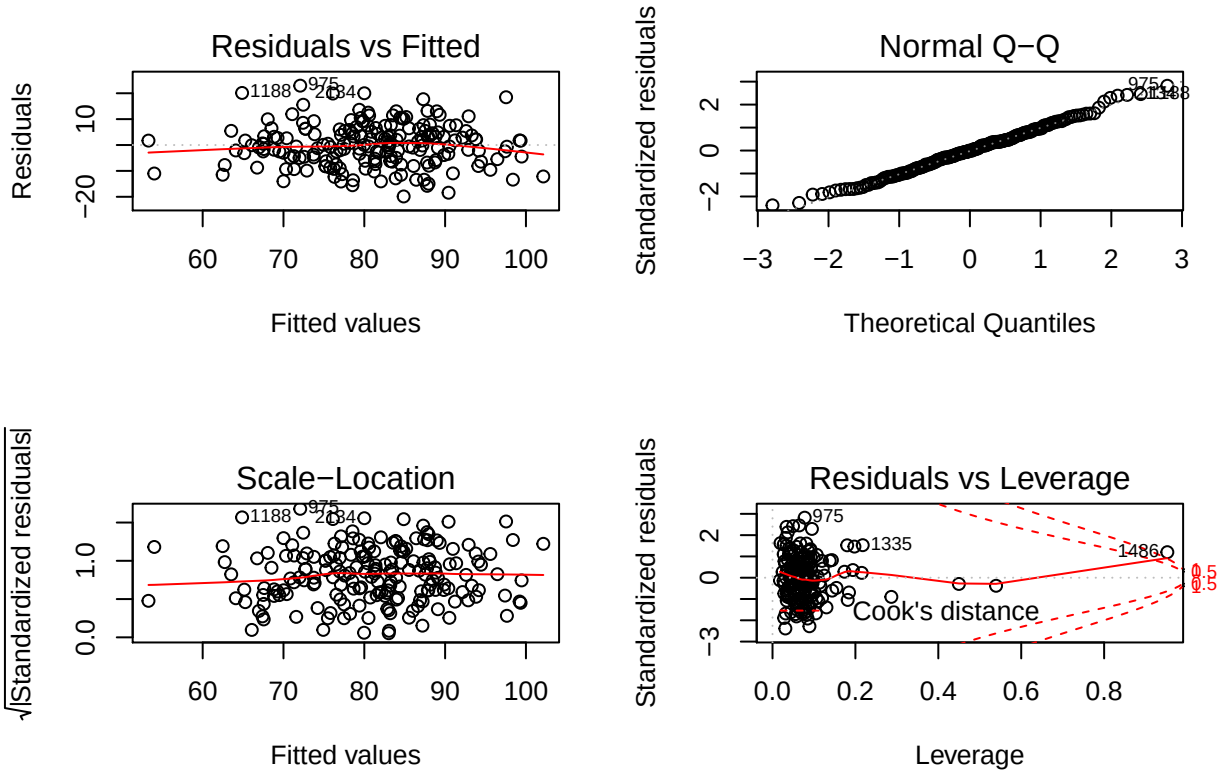
```
raw.lm <- lm(TARGET_WINS ~ . - INDEX,
             data = baseball.train)

summary(raw.lm)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - INDEX, data = baseball.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
```

From the above, we notice how only 2 predictors are statistically significant in the calculation prediction of the target variable; also, we can notice how the R^2 value is 0.55; also, some of the standard errors are very high; something interesting to note is the numbers of degrees of freedom equals to 175; which means that this model could be improved.

```
par(mfrow=c(2,2))
plot(raw.lm)
```



Even though it seems that the Normal Q-Q plot follows the data, we noticed the tails on both ends, this caused due to outliers most likely. The Residuals vs Fitted values plot, shows what seems to be a good homoscedastic pattern, that means with no variance nor a visible pattern; but yet again this is debatable due to the fact of the existence of high p-values in our given model.

b) Build linear model with excluded data.

I will create a model as follows:

```
TARGET_WINS          <- To predict
TEAM_BATTING_2B
TEAM_BATTING_BB
TEAM_BASERUN_SB
TEAM_PITCHING_HR
TEAM_FIELDING_E
```

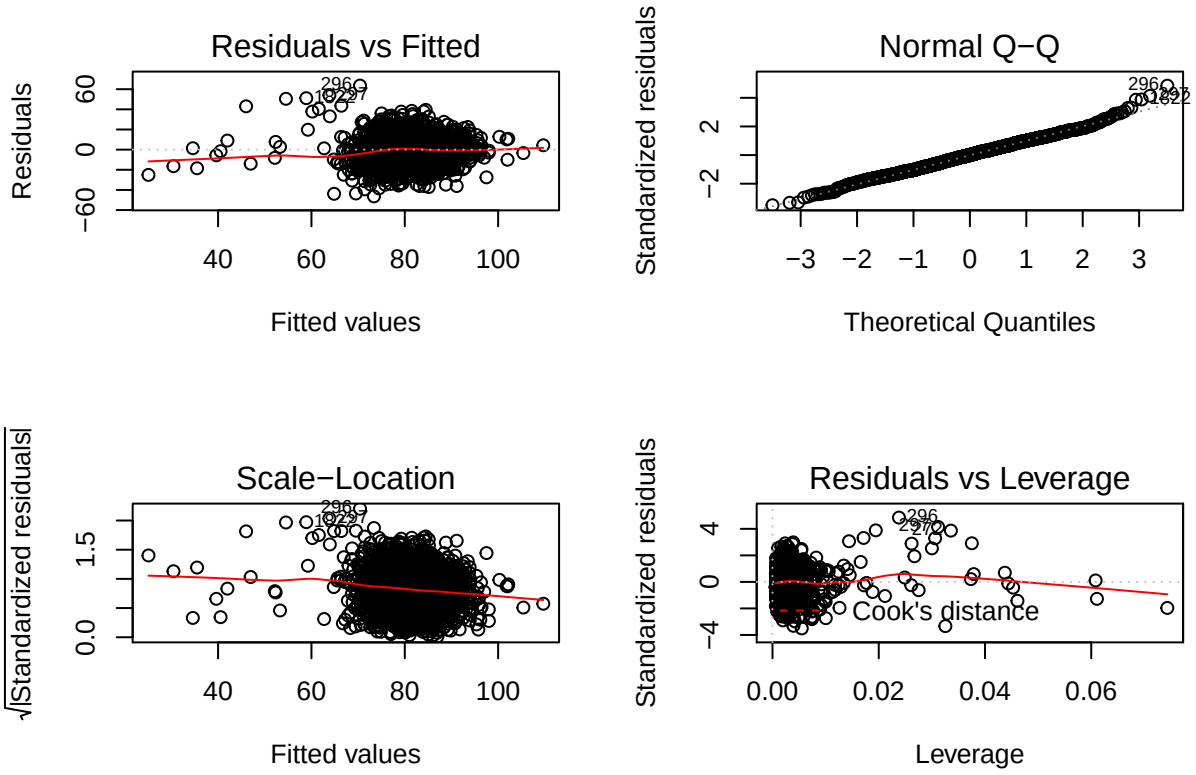
```
model1.lm <- lm(TARGET_WINS ~ .,
                data = reduced.train)

summary(model1.lm)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = reduced.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.397  -9.304   0.116   8.882  63.593
##
## Coefficients:
```

```
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      46.987630   2.333243  20.138  < 2e-16 ***
## TEAM_BATTING_2B   0.063641   0.007101   8.962  < 2e-16 ***
## TEAM_BATTING_BB   0.024162   0.003424   7.057 2.30e-12 ***
## TEAM_BASERUN_SB   0.054539   0.004006  13.613  < 2e-16 ***
## TEAM_PITCHING_HR  0.022468   0.006199   3.624 0.000296 ***
## TEAM_FIELDING_E  -0.016136   0.002221  -7.265 5.21e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.28 on 2139 degrees of freedom
##   (131 observations deleted due to missingness)
## Multiple R-squared:  0.1933, Adjusted R-squared:  0.1915
## F-statistic: 102.5 on 5 and 2139 DF,  p-value: < 2.2e-16
```

Now, if we compare this model to our original model, we see a tremendous difference in terms of p values; now our **model1** shows that all predictors are statistically significant; our degrees of freedom increased dramatically; our standard errors are much better; however, we noticed how our R^2 has lowered compared to our previous results; in the mean time I would not worry too much about it, since I will try to refine this model from now on.

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



Something nice to look is that with this model, it seems that we actually seem to approach a very good linear regression given by the Q-Q plot with a very nice looking randomly plot of Residuals vs fitted values. The current downside with this model is that there seems to be some leverage and outliers still present and that **TEAM_PITCHING_HR** contradicts the theoretical effect since currently seems to project positive values instead of negative.

Let's remove that predictor and see what happens:

c) Build linear model with excluded data.

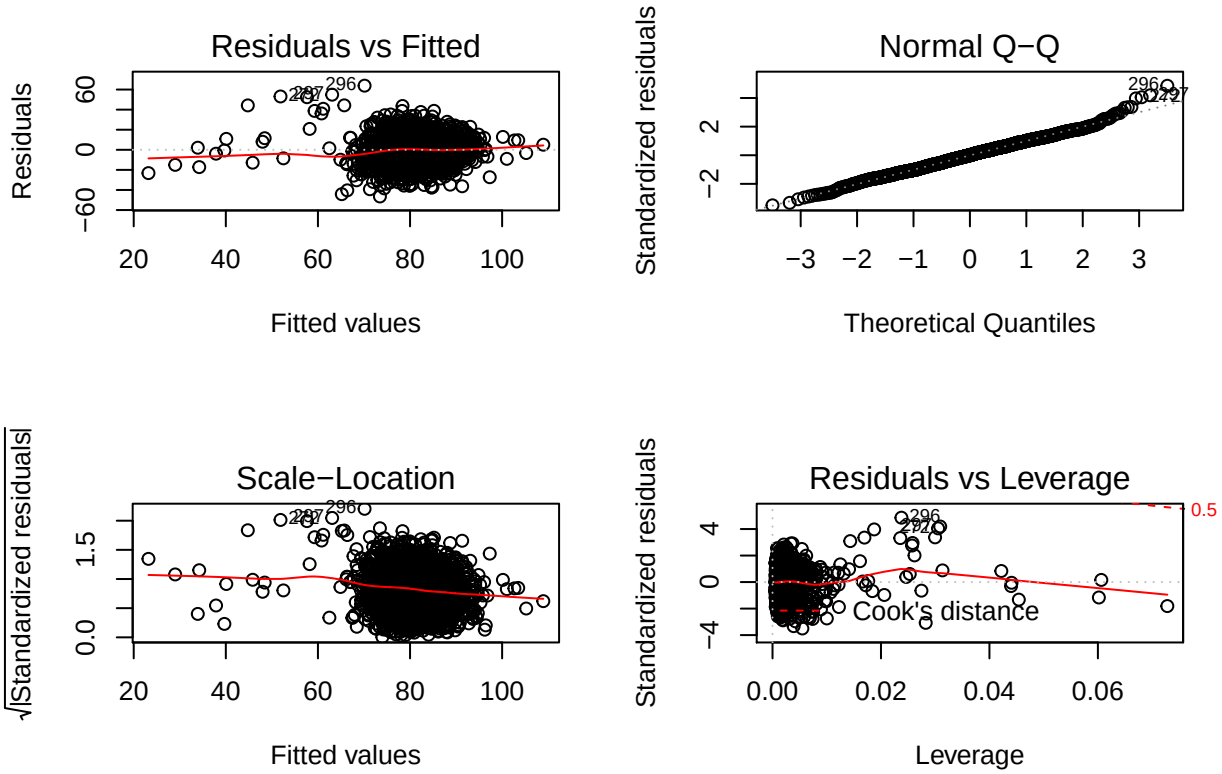
```
exclude <- c('TEAM_PITCHING_HR')
newvars <- names( reduced.train) %in% exclude
reduced.train <- reduced.train[!newvars]
```

```
model2.lm <- lm(TARGET_WINS ~ .,
             data = reduced.train)

summary(model2.lm)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = reduced.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.465  -9.119   0.086   8.913  63.868
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     46.376336   2.333729  19.872  < 2e-16 ***
## TEAM_BATTING_2B  0.073190   0.006613  11.068  < 2e-16 ***
## TEAM_BATTING_BB  0.027077   0.003338   8.113 8.25e-16 ***
## TEAM_BASERUN_SB  0.050777   0.003881  13.085  < 2e-16 ***
## TEAM_FIELDING_E -0.017487   0.002196  -7.964 2.68e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.31 on 2140 degrees of freedom
##   (131 observations deleted due to missingness)
## Multiple R-squared:  0.1884, Adjusted R-squared:  0.1869
## F-statistic: 124.2 on 4 and 2140 DF,  p-value: < 2.2e-16
```

For this new model in which **TEAM_PITCHING_HR** got excluded, we can notice not may significant changes.

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



From this model, we can still notice that there are some problems with the residuals, it seems that some variability is still present.

c) Build linear model more refined from above.

In order to refine the above model, I will create a model as follows:

Let's transform the values in our data frame by calculating the square root on both sides of the linear relationship; this is due to all predictor variables having the same units and are basically counts.

```
reduced.train$TARGET_WINS <- sqrt(baseball.train$TARGET_WINS)
reduced.train$TEAM_BATTING_2B <- sqrt(baseball.train$TEAM_BATTING_2B)
reduced.train$TEAM_BATTING_BB <- sqrt(baseball.train$TEAM_BATTING_BB)
reduced.train$TEAM_BASERUN_SB <- sqrt(baseball.train$TEAM_BASERUN_SB)
reduced.train$TEAM_FIELDING_E <- sqrt(baseball.train$TEAM_FIELDING_E)
```

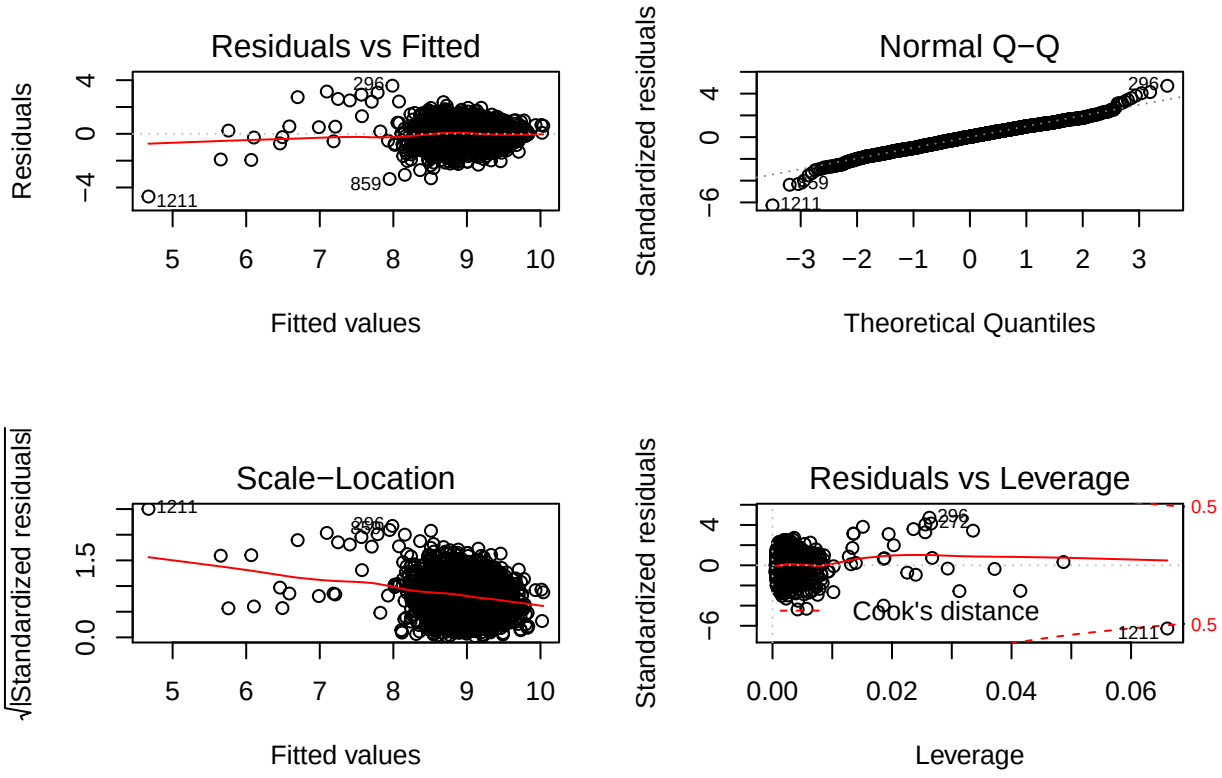
```
model3.lm <- lm(TARGET_WINS ~ . ,
             data = reduced.train)

summary(model3.lm)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = reduced.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6716 -0.5055  0.0248  0.5182  3.5884
##
## Coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.166610   0.278611  18.544   <2e-16 ***
## TEAM_BATTING_2B  0.124039   0.011946  10.383   <2e-16 ***
## TEAM_BATTING_BB  0.073113   0.008314   8.794   <2e-16 ***
## TEAM_BASERUN_SB  0.077234   0.005733  13.472   <2e-16 ***
## TEAM_FIELDING_E -0.044537   0.005298  -8.406   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7709 on 2140 degrees of freedom
##   (131 observations deleted due to missingness)
## Multiple R-squared:  0.2126, Adjusted R-squared:  0.2112
## F-statistic: 144.5 on 4 and 2140 DF,  p-value: < 2.2e-16
```

Now, as you can see, the performance has increased tremendously, the residuals have decreased dramatically, the median is near zero and all the p-values indicate that the current predictor variables are statistically significant, the standard errors are low; but once again, we noticed how the residual standard error has decreased and also, we noticed that the median is more considerable near zero with what seems to be a good fit. The current down side is that there seems to be some variability still present in the data.

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



By looking at the plots, we noticed how the fitted values vs the residuals are presenting some sort of variability, also we can notice how the tails on the Q-Q plot tend to deviate on both extremes and the leverage graph is showing the presence of what seem to be outliers.

d) Build linear model with identified leverage and outliers

In order to find values that will be considered as leverage; I will create a function that find leverage and outlier points.

```
IsOutlier <- function(target, variable){

  l.mean <- mean(variable, na.rm = TRUE)
  l.n <- length(variable)

  x_minus_xhat_sqrd <- (variable - l.mean)^2
  l.sum <- sum(x_minus_xhat_sqrd)

  # Obtaining leverage formula for each value
  leverage_manual <- round(1/l.n + x_minus_xhat_sqrd/l.sum,3)

  # Identifiying leverage points
  leverage <- data.frame(leverage = leverage_manual)
  leverage$is_leverage <- 0
  leverage$is_leverage[which(leverage$leverage > 4/l.n)] <- 1

  # Obtaining regular unchanged linear model in order to obtain original residuals
  leverage.lm <- lm(target ~ variable)
  #summary(leverage.lm)

  # Extracting residuals
  leverage$residuals <- round(leverage.lm$residuals,3)

  # Obtaining standardize residuals

  leverage.sd <- (1/(l.n - 2) * sum(leverage$residuals^2))^(1/2)
  leverage$r <- round(leverage.lm$residuals / (leverage.sd *(1 - leverage$leverage)^(1/2)),3)

  # Identifying outliers
  leverage$outlier <- 0
  leverage$outlier[which(abs(leverage$r) > 2)] <- 1

  return(leverage$outlier)
}
```

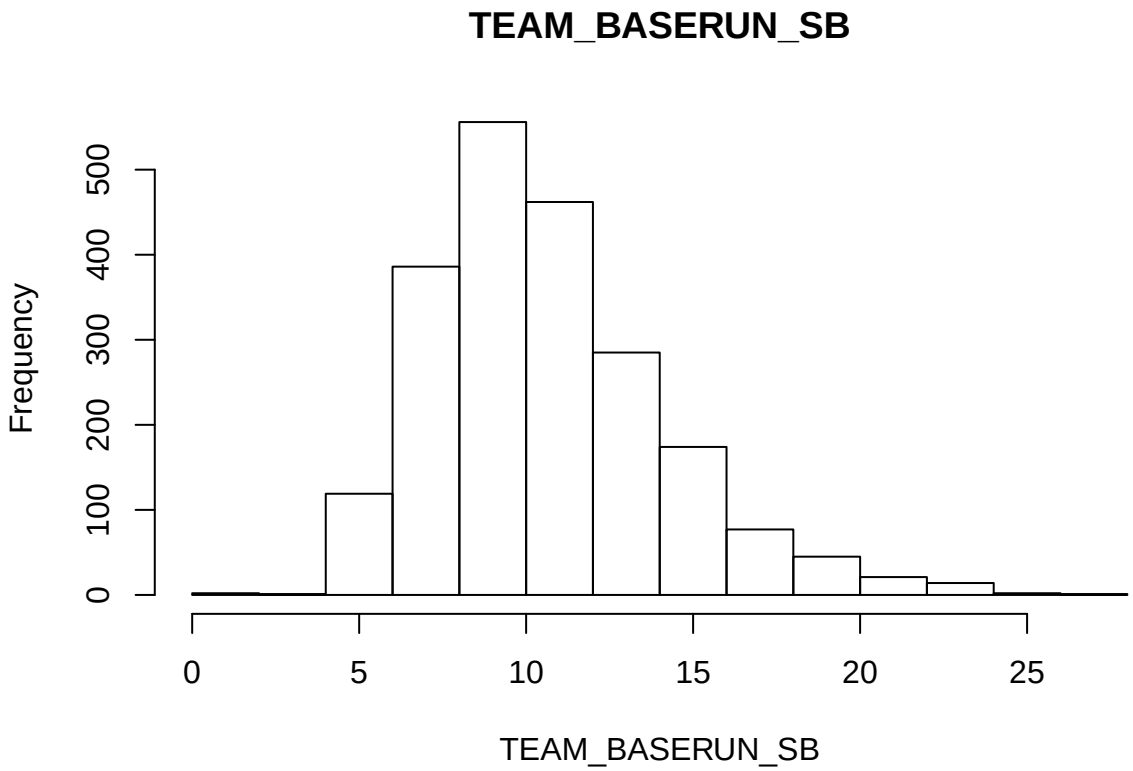
Let's identify outliers from the given model and let's see if we could use those inputs into our model.

```
reduced.train$TEAM_BATTING_2B_Outlier <- IsOutlier(reduced.train$TARGET_WINS,
                                                   reduced.train$TEAM_BATTING_2B)

reduced.train$TEAM_BATTING_BB_Outlier <- IsOutlier(reduced.train$TARGET_WINS,
                                                   reduced.train$TEAM_BATTING_BB)

reduced.train$TEAM_FIELDING_E_Outlier <- IsOutlier(reduced.train$TARGET_WINS,
                                                   reduced.train$TEAM_FIELDING_E)
```

Something to note from **TEAM_BASERUN_SB** is that it currently shows a small number of NAs. Previously, I made note that if we need to work with NA's we need to find a way to solve this "problem". Also, it was identified that the percentage of missing values for **TEAM_BASERUN_SB** = 5.76 %; hence, due to the low percentage of missing values compared to our given data, I consider that it's safe to replace

those missing values with randomly generated values. Let's see what's the current distribution.

```
hist(reduced.train$TEAM_BASERUN_SB,
     main = 'TEAM_BASERUN_SB',
     xlab = 'TEAM_BASERUN_SB')
```



Since the current distribution seems to follow a somehow normal curve, I will replace the missing NAs with randomly generated values from 0 to 26.4007576.

```
set.seed(123)
rand_values <- sample(0:max(reduced.train$TEAM_BASERUN_SB, na.rm = TRUE),
                      size=sum(is.na(reduced.train$TEAM_BASERUN_SB)),
                      replace = TRUE)

reduced.train$TEAM_BASERUN_SB[is.na(reduced.train$TEAM_BASERUN_SB)] <- rand_values

reduced.train$TEAM_BASERUN_SB_Outlier <- IsOutlier(reduced.train$TARGET_WINS,
                                                   reduced.train$TEAM_BASERUN_SB)
```

Let's see the current results by incorporating values related to outliers; that is creating "dummy" predictors with a value of 0 (is not outlier) or 1 (is outlier).

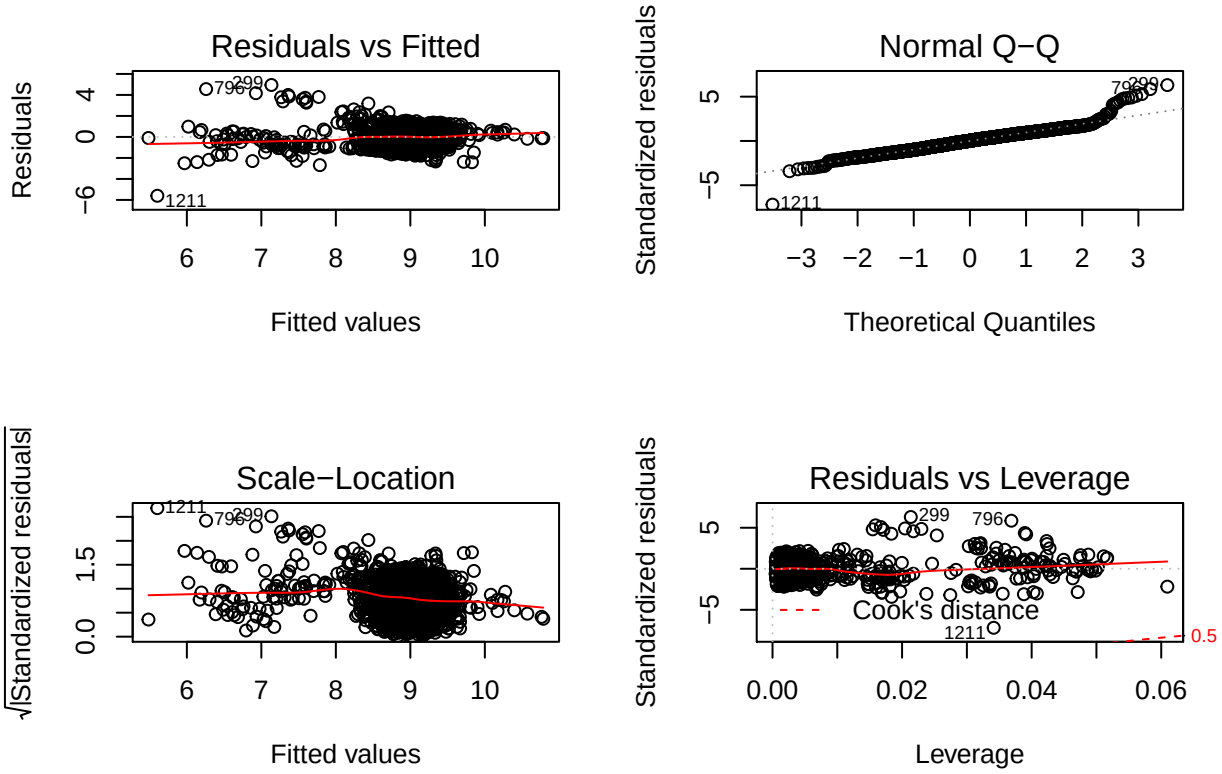
```
model4.lm <- lm(TARGET_WINS ~ . ,
             data = reduced.train)

summary(model4.lm)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = reduced.train)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -5.6040 -0.5205  0.0199  0.5106  4.9476
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                5.356364   0.272779  19.636  < 2e-16 ***
## TEAM_BATTING_2B            0.151149   0.011606  13.023  < 2e-16 ***
## TEAM_BATTING_BB            0.039496   0.007539   5.239 1.77e-07 ***
## TEAM_BASERUN_SB            0.041035   0.004955   8.281  < 2e-16 ***
## TEAM_FIELDING_E           -0.001046   0.005058  -0.207 0.836156
## TEAM_BATTING_2B_Outlier -0.512773   0.136832  -3.747 0.000183 ***
## TEAM_BATTING_BB_Outlier  0.217240   0.139736   1.555 0.120171
## TEAM_FIELDING_E_Outlier  0.786684   0.154864   5.080 4.09e-07 ***
## TEAM_BASERUN_SB_Outlier -1.954221   0.149023 -13.114  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7929 on 2267 degrees of freedom
## Multiple R-squared:  0.2773, Adjusted R-squared:  0.2748
## F-statistic: 108.7 on 8 and 2267 DF,  p-value: < 2.2e-16
```

From the above model, we can notice how by introducing 'new variables' related to outlier info, our model has changed dramatically once again. Let's remove the non statistically significant predictors from our model. Something interesting is that **TEAM_FIELDING_E_Outlier** shows a positive relationship but this contradicts the theoretical hypothesis as well.

```
par(mfrow=c(2,2))
plot(model4.lm)
```



e) Linear model with removed TEAM_FIELDING_E, TEAM_FIELDING_E_Outlier and TEAM_BATTING_BB_Outlier

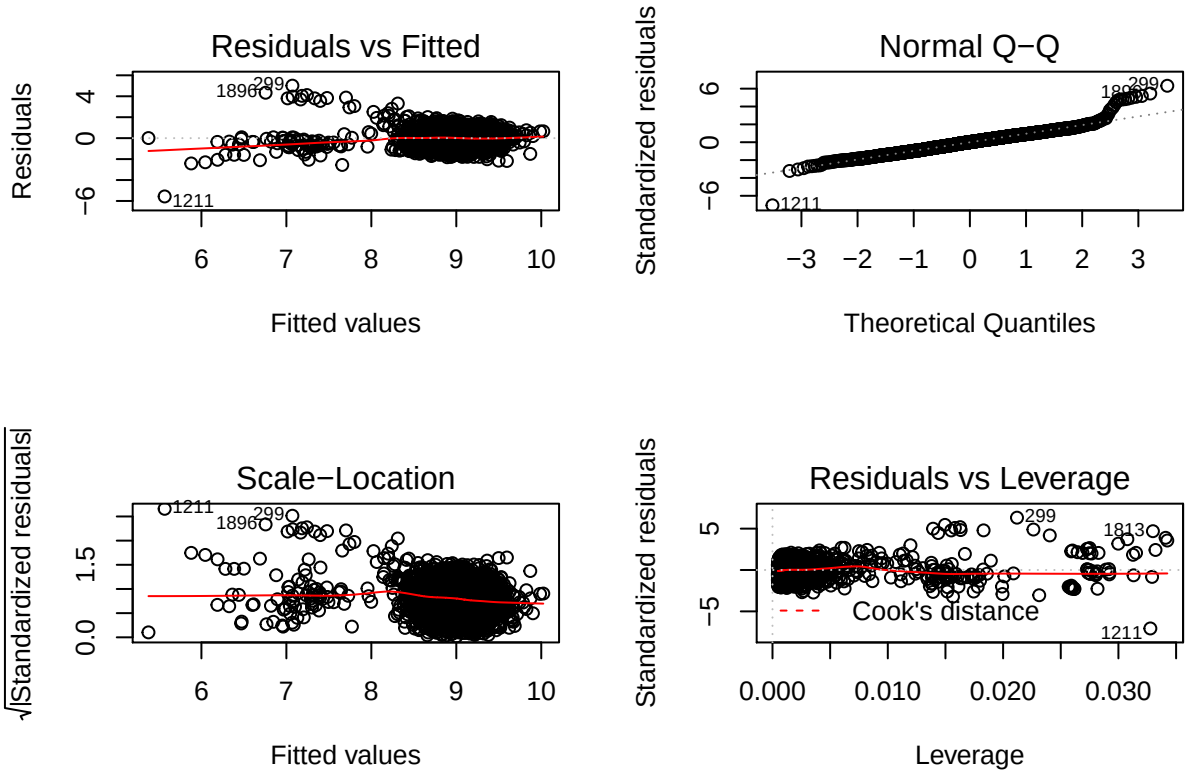
Let's remove those predictor variables and see what happens.

```
exclude <- c('TEAM_FIELDING_E', 'TEAM_FIELDING_E_Outlier', 'TEAM_BATTING_BB_Outlier')
newvars <- names( reduced.train) %in% exclude
reduced.train <- reduced.train[!newvars]
```

```
model6.lm <- lm(TARGET_WINS ~ . ,
             data = reduced.train)

summary(model6.lm)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = reduced.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5674 -0.5283  0.0070  0.5140  5.0121
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               5.406554   0.212296  25.467  < 2e-16 ***
## TEAM_BATTING_2B           0.158651   0.011596  13.681  < 2e-16 ***
## TEAM_BATTING_BB           0.031997   0.005799   5.518 3.82e-08 ***
## TEAM_BASERUN_SB           0.040678   0.004367   9.314  < 2e-16 ***
## TEAM_BATTING_2B_Outlier -0.161814   0.128341  -1.261    0.208
## TEAM_BASERUN_SB_Outlier -1.520733   0.133146 -11.422  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8012 on 2270 degrees of freedom
## Multiple R-squared:  0.2612, Adjusted R-squared:  0.2595
## F-statistic: 160.5 on 5 and 2270 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(model6.lm)
```



f) Linear model with extra TEAM_BATTING_2B_Outlier taken off

Since this model still seems to behave oddly, due to some variability in the data, I will remove **TEAM_BATTING_2B_Outlier** since it contradicts some of the hypothesis and rules of the game.

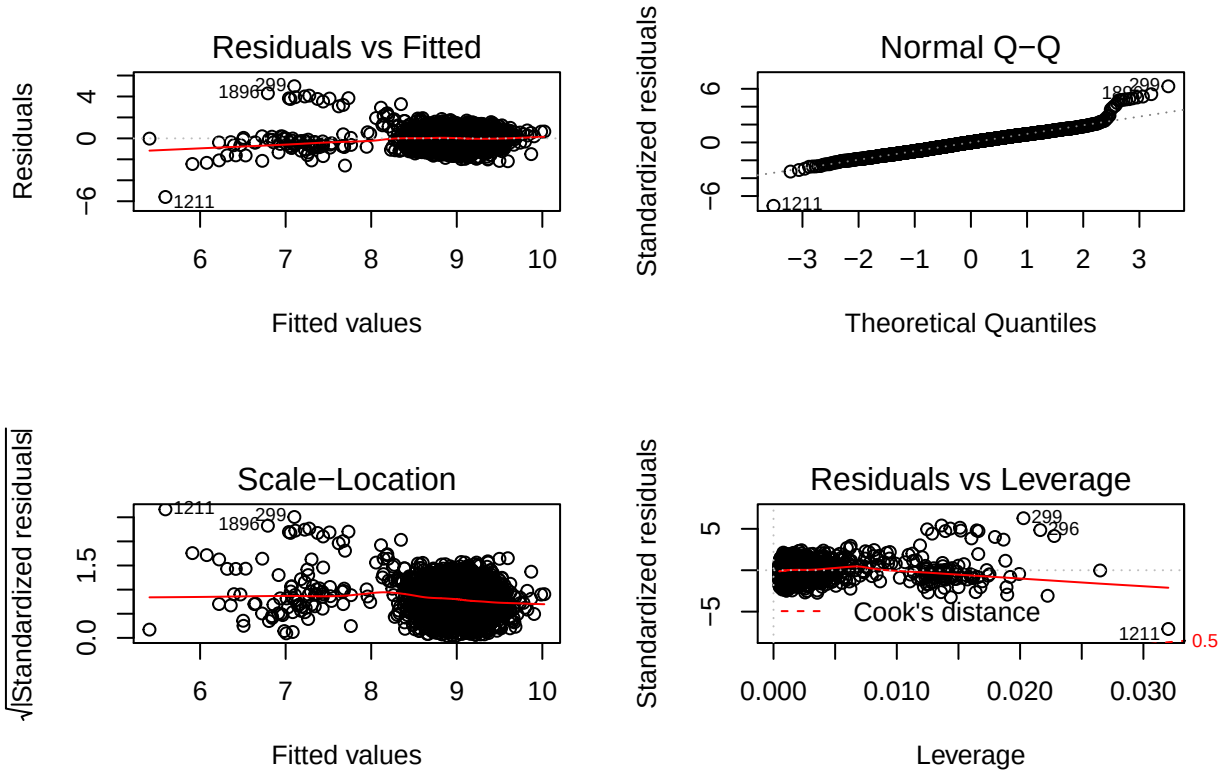
```
exclude <- c('TEAM_BATTING_2B_Outlier')
newvars <- names( reduced.train) %in% exclude
reduced.train <- reduced.train[!newvars]
```

```
model7.lm <- lm(TARGET_WINS ~ . ,
             data = reduced.train)

summary(model7.lm)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = reduced.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5950 -0.5275  0.0083  0.5151  4.9813
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               5.395563   0.212145  25.433  < 2e-16 ***
## TEAM_BATTING_2B           0.158419   0.011596  13.661  < 2e-16 ***
## TEAM_BATTING_BB           0.032539   0.005783   5.626 2.07e-08 ***
## TEAM_BASERUN_SB           0.040740   0.004368   9.328  < 2e-16 ***
## TEAM_BASERUN_SB_Outlier -1.641279   0.092677 -17.710  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8013 on 2271 degrees of freedom
## Multiple R-squared:  0.2607, Adjusted R-squared:  0.2594
## F-statistic: 200.2 on 4 and 2271 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(model7.lm)
```



Something interesting to visualize is the recurring outlier 1211; this value is off limits from the Cook's distance and also is far away from the fitted values plots.

Let's see what's going on with that specific record.

```
baseball.train[1211,]
```

```
##      INDEX TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## 1211  1347           0            891             135               0
##      TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## 1211               0               0               0               0
##      TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
## 1211               0               NA           24057               0
##      TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## 1211                0                0            1890              NA
```

By looking at that specific record, we noticed that some sort of odd behavior happened with that entry. Not sure if it is a mistake or if it was a penalization; either way, I believe that will be safe not to include this record into the training data set.

Let's see if there are more values related to **TARGET_WINS** with zero value.

```
baseball.train[which(baseball.train$TARGET_WINS == 0),]
```

```
##      INDEX TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## 1211  1347           0            891             135               0
##      TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## 1211               0               0               0               0
##      TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
## 1211               0               NA           24057                0
##      TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## 1211                0                0            1890               NA
```

From that, we have confirmed that this is a very odd entry and I will exclude that data entry from the training set.

```
reduced.train <- subset(reduced.train, TARGET_WINS > 0)
```

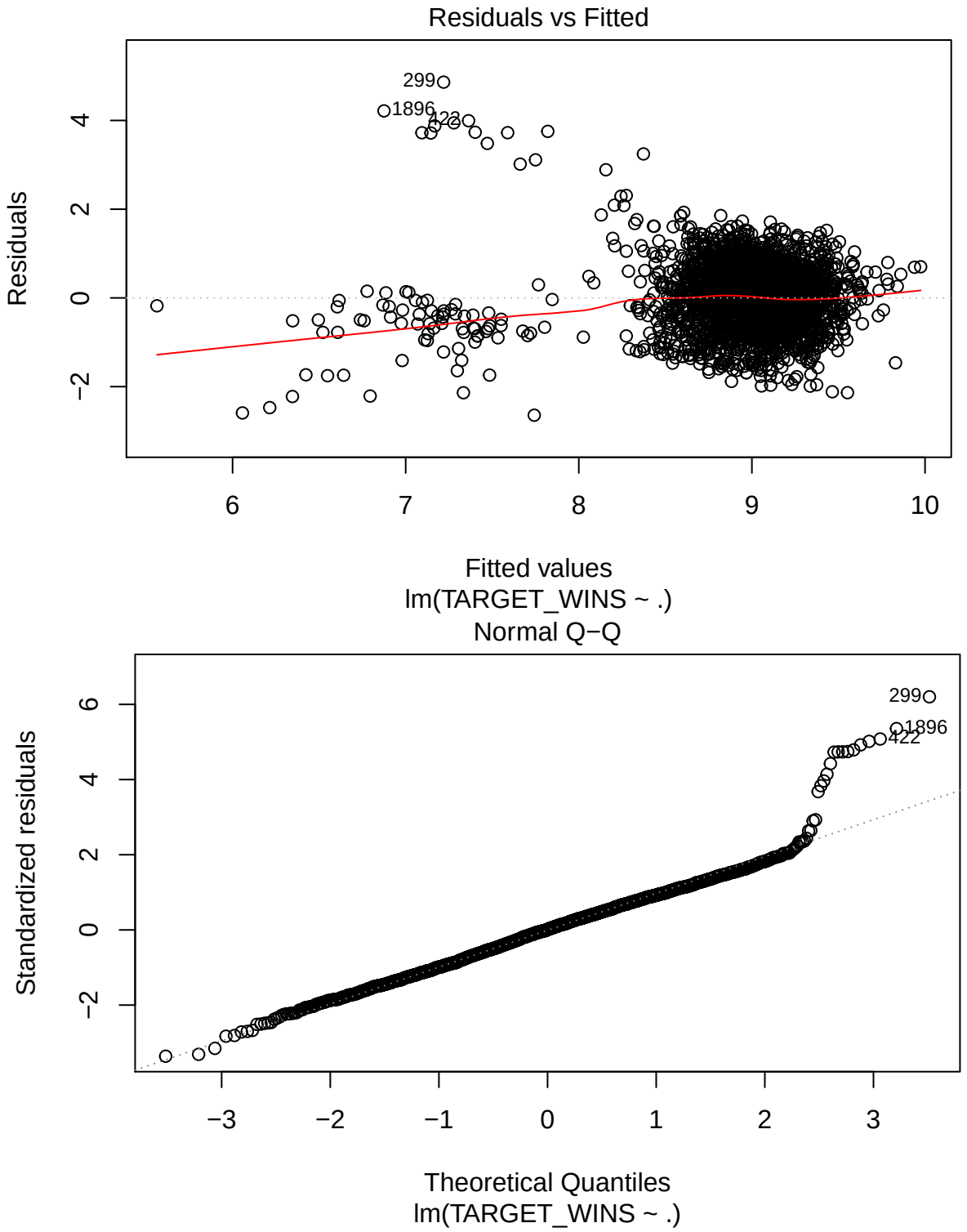
g) Linear model excluding outlier 1211

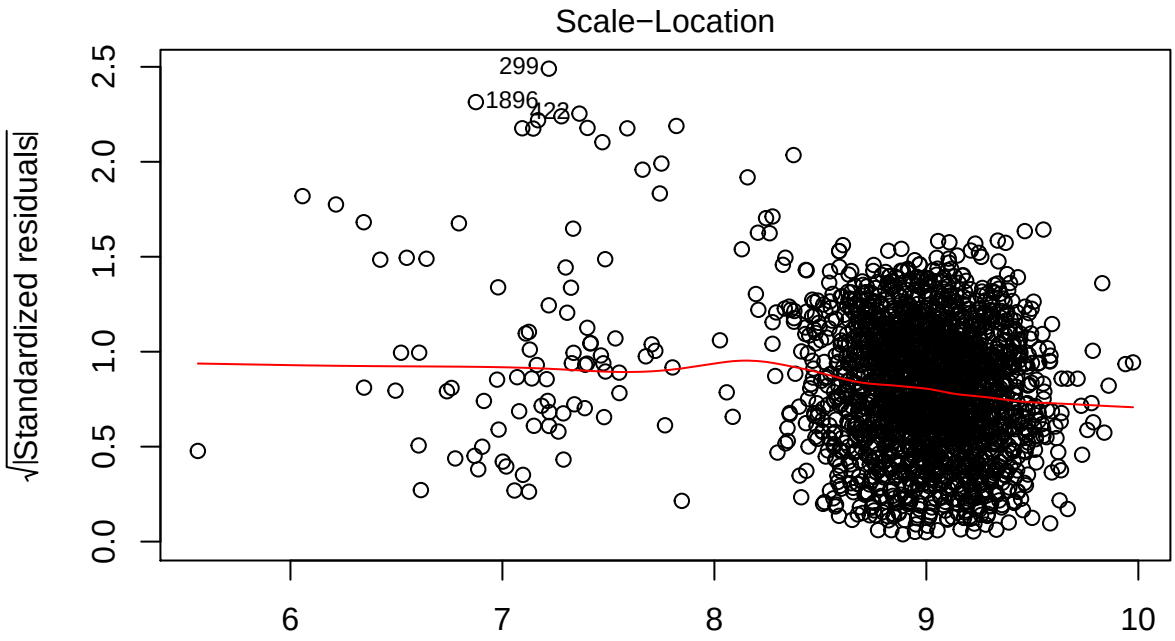
```
model8.lm <- lm(TARGET_WINS ~ . ,
             data = reduced.train)

summary(model8.lm)
```

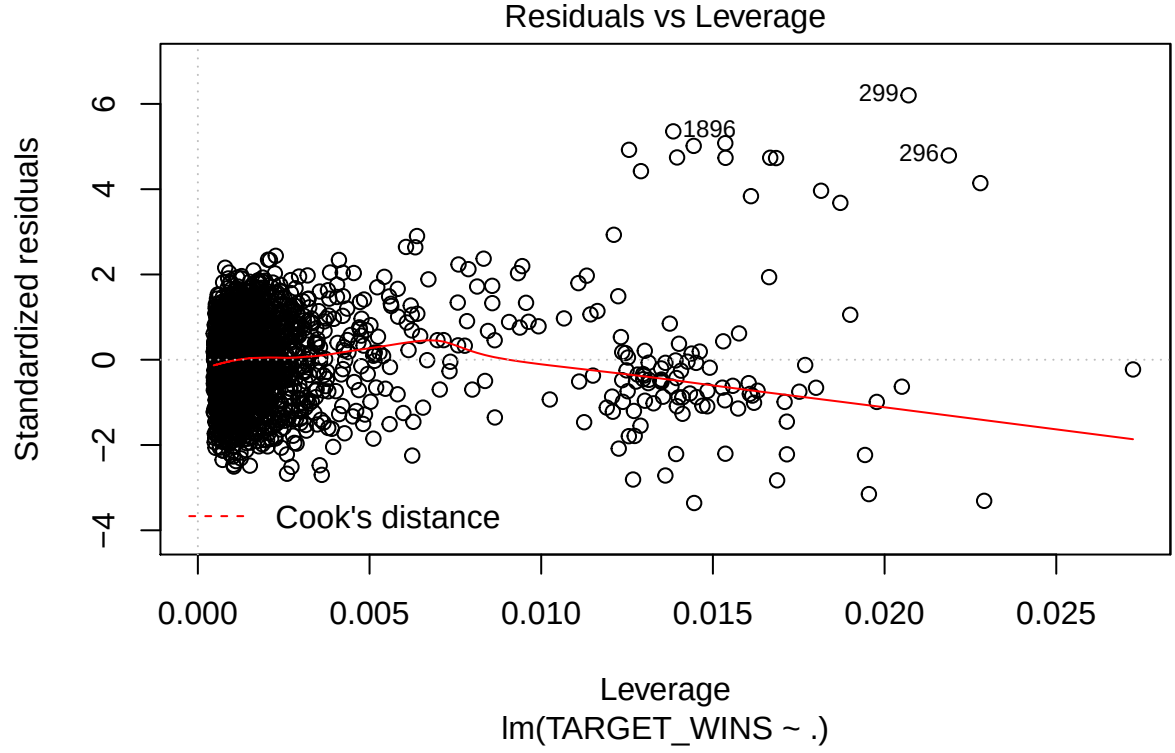
```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = reduced.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6437 -0.5306  0.0103  0.5167  4.8639
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                5.565010   0.211150  26.356  < 2e-16 ***
## TEAM_BATTING_2B            0.156413   0.011473  13.633  < 2e-16 ***
## TEAM_BATTING_BB            0.027591   0.005762   4.789 1.79e-06 ***
## TEAM_BASERUN_SB            0.038161   0.004335   8.803  < 2e-16 ***
## TEAM_BASERUN_SB_Outlier -1.602319   0.091825 -17.450  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7926 on 2270 degrees of freedom
## Multiple R-squared:  0.2465, Adjusted R-squared:  0.2452
## F-statistic: 185.7 on 4 and 2270 DF,  p-value: < 2.2e-16
```

Something significant to point is what I consider to be a low R^2, but based on the distribution of the data this might be expected since a lot of points are presenting a linear trend but not necessarily near the regression line; so that's understandable. In terms of the other indicators, it seems to be a good fit; and the reason is due to the low p-values, and low residual standard errors and the high number of degrees of freedom basically means that almost all data points were captured in order to create this linear model. Another indicator is the Median value which is near zero with the 1Q and 3Q centering zero; the only value that will be more concerning will be the Max value but this is expected due to outlier presence in the data set.

```
#par(mfrow=c(2,2))
plot(model8.lm)
```







4. SELECT MODELS

From the above models, the one that I will pick **is the last one (model8.lm)**; this is due to:

- it follows a linear model for most of the time, just deviating a few points towards the top right.

- also the outlier 1211 was removed and not included.
- this model offers a good understanding of what happen if an excessive action is performed in the game; it might actually hurt the team by loosing games, this is the case of **TEAM_BASERUN_SB_Outlier**.

I believe that the mix of predictors including a dummy outlier, is a great indicator that need to be included along the other variables and transformations.

As I mentioned before, my only concerns will be the R^2; but based on the quantity of data points and linearity seeing in the plot along the transformation; it seems that this could be explained as to the distribution of points not being near the regression line.

Another great indicator is the low median value, even though other models offered a lower value but the high p-value made me not accept it.

Predict

First trial

In order to predict our raw data, I need to prepare the data as follows:

```
myvars <- c("INDEX", "TEAM_BATTING_2B", "TEAM_BATTING_BB", "TEAM_BASERUN_SB")
reduce.predict <- baseball.eval[myvars]

reduce.predict$TEAM_BATTING_2B <- sqrt(reduce.predict$TEAM_BATTING_2B)
reduce.predict$TEAM_BATTING_BB <- sqrt(reduce.predict$TEAM_BATTING_BB)
reduce.predict$TEAM_BASERUN_SB <- sqrt(reduce.predict$TEAM_BASERUN_SB)
```

Since there's no way to indicate if the new values are considered outliers or not, I will assign a zero instead; then I will re-evaluate with the given values.

```
reduce.predict$TEAM_BASERUN_SB_Outlier <- 0
```

Let's predict our **PREDICTED_WINS** and see the results. Keep in mind that I am elevating to a quadratic due to previously applying the square root to the **TARGET_WINS** variable; hence, we need to apply the inverse function in order to obtain an approximation to our predicted value. No decimals are placed due to round counts.

```
reduce.predict$PREDICTED_WINS <- round((predict(model8.lm, newdata=reduce.predict))^2,0)
summary(reduce.predict$PREDICTED_WINS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   45.00   78.00   81.00   81.36   85.00   96.00      13
```

Something interesting to note is the presence of **NA's**; this is something to be on the look out. Why is there a presence of NAs in our predicted values?

Let's compare our training data set with our given data set, in particular, let's focus on the **TEAM_BASERUN_SB** predictor variable.

Let's see our evaluation data set.

```
summary(reduce.predict[2:4])
```

```
##  TEAM_BATTING_2B  TEAM_BATTING_BB  TEAM_BASERUN_SB
##  Min.   : 6.633   Min.   : 3.873   Min.   : 0.000
##  1st Qu.:14.491   1st Qu.:20.893   1st Qu.: 7.681
##  Median :15.460   Median :22.561   Median : 9.592
##  Mean   :15.444   Mean   :22.112   Mean   :10.466
##  3rd Qu.:16.688   3rd Qu.:23.780   3rd Qu.:12.319
```

```
##  Max.   :19.391   Max.   :28.142   Max.   :24.083
##                                    NA's   :13
```

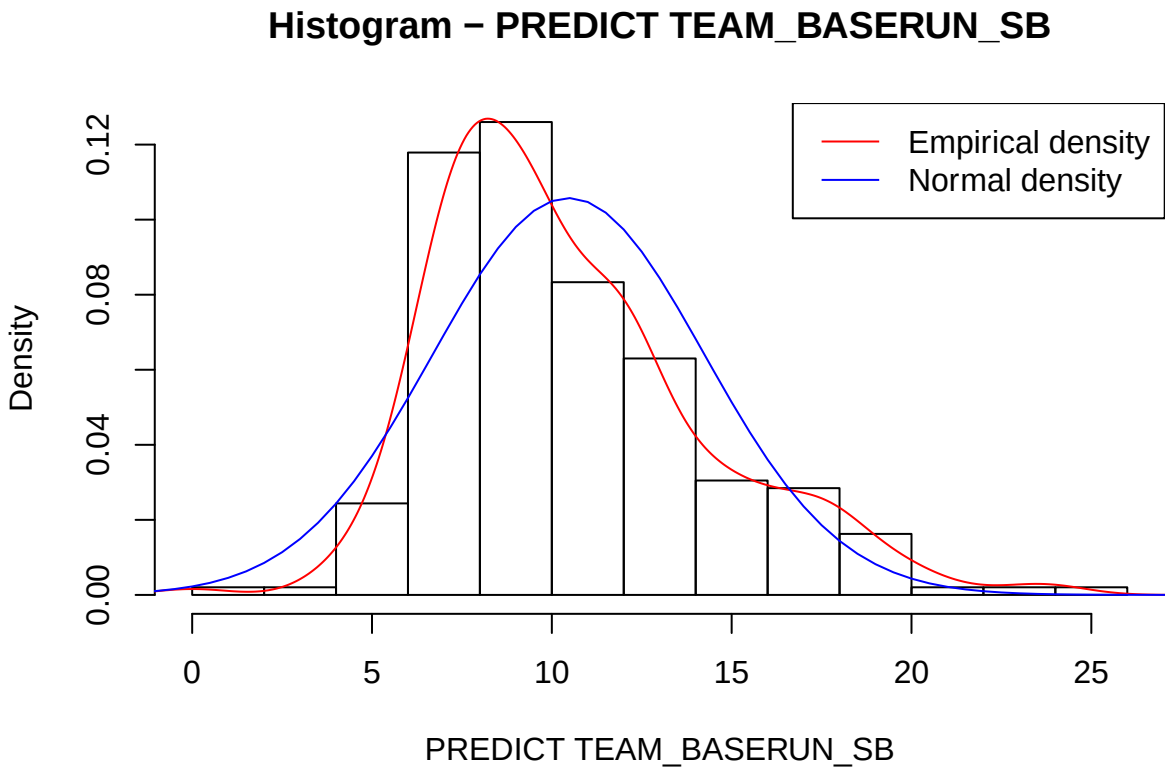
Let's see our training data set.

```
summary(reduced.train[2:4])
```

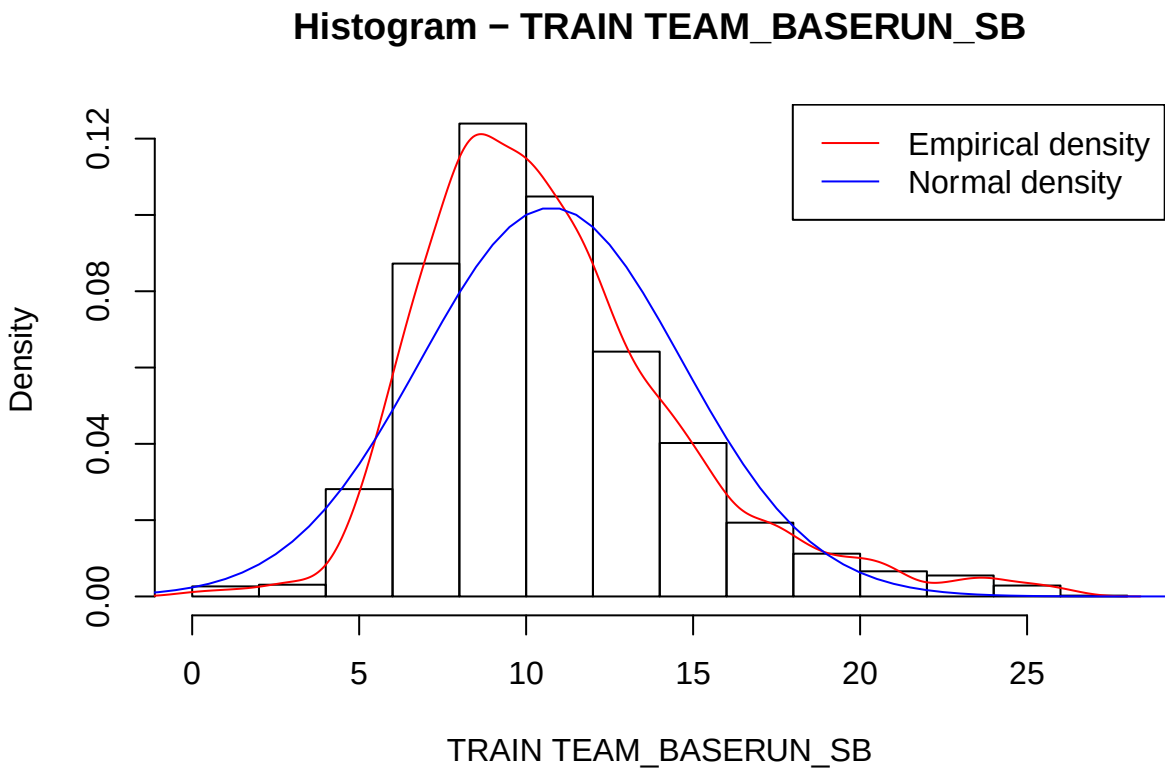
```
##  TEAM_BATTING_2B  TEAM_BATTING_BB  TEAM_BASERUN_SB
##  Min.   : 8.307   Min.   : 3.464   Min.   : 0.000
##  1st Qu.:14.422   1st Qu.:21.237   1st Qu.: 8.062
##  Median :15.427   Median :22.627   Median :10.100
##  Mean   :15.460   Mean   :22.177   Mean   :10.750
##  3rd Qu.:16.523   3rd Qu.:24.083   3rd Qu.:12.708
##  Max.   :21.401   Max.   :29.631   Max.   :26.401
```

If we look at the previous results, we noticed how the **TEAM_BASERUN_SB** present a total of 13 NAs. Let's plot this data and see it's behavior and compare it to our training data set.

```
hist(reduce.predict$TEAM_BASERUN_SB, freq = FALSE,
   main = paste('Histogram - PREDICT TEAM_BASERUN_SB'),
   xlab = 'PREDICT TEAM_BASERUN_SB',
   ylab = 'Density')
lines(density(reduce.predict$TEAM_BASERUN_SB, na.rm = TRUE), col="red")
lines(seq(-400, 500, by=.5),
    dnorm(seq(-400, 500, by=.5),
          mean(reduce.predict$TEAM_BASERUN_SB, na.rm = TRUE),
          sd(reduce.predict$TEAM_BASERUN_SB, na.rm = TRUE)),
    col="blue")
lnames <- c('Empirical density', 'Normal density')
legend('topright', lnames, col = c('red','blue'), lty = 1)
```



```
hist(reduced.train$TEAM_BASERUN_SB, freq = FALSE,
   main = paste('Histogram - TRAIN TEAM_BASERUN_SB'),
   xlab = 'TRAIN TEAM_BASERUN_SB',
   ylab = 'Density')
lines(density(reduced.train$TEAM_BASERUN_SB, na.rm = TRUE), col="red")
lines(seq(-400, 500, by=.5),
    dnorm(seq(-400, 500, by=.5),
          mean(reduced.train$TEAM_BASERUN_SB, na.rm = TRUE),
          sd(reduced.train$TEAM_BASERUN_SB, na.rm = TRUE)),
    col="blue")
lnames <- c('Empirical density', 'Normal density')
legend('topright', lnames, col = c('red','blue'), lty = 1)
```



From the above, we could conclude that the distributions follow some sort of similar normality; hence, I will replace the missing NA's that represent a total of 5.02 % of missing values with random generated values instead.

```
rand_values <- sample(0:max(reduce.predict$TEAM_BASERUN_SB, na.rm = TRUE),
                      size=sum(is.na(reduce.predict$TEAM_BASERUN_SB)), replace = TRUE)

reduce.predict$TEAM_BASERUN_SB[is.na(reduce.predict$TEAM_BASERUN_SB)] <- rand_values
```

It is noted, that previously this same procedure was performed while preparing our data in our training data set.

Now, that I have taken care of NA's, let's try another run.

Second trial

Let's predict our **PREDICTED_WINS** and see the results. Keep in mind that I am elevating to a quadratic due to previously applying the square root to the **TARGET_WINS** variable; hence, we need to apply the inverse function in order to obtain an approximation to our predicted value. No decimals are placed due to round counts.

```
reduce.predict$PREDICTED_WINS <- round((predict(model8.lm, newdata=reduce.predict))^2,0)
summary(reduce.predict$PREDICTED_WINS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   45.00   78.00   81.00   81.06   85.00   96.00
```

Since previously; I entered the Outlier value as zero, I will rerun once again in order to identify some possible outliers and hopefully refine our given prediction values.

```
reduce.predict1 <- reduce.predict[c('INDEX',
                                    'PREDICTED_WINS',
                                    'TEAM_BATTING_2B',
                                    'TEAM_BATTING_BB',
                                    'TEAM_BASERUN_SB')]

reduce.predict1$PREDICTED_WINS <- sqrt(reduce.predict1$PREDICTED_WINS)
```

```
reduce.predict1$TEAM_BASERUN_SB_Outlier <- IsOutlier(reduce.predict1$PREDICTED_WINS,
                                                     reduce.predict1$TEAM_BASERUN_SB)
```

After re-running our previous fit with the procedure to identify possible outliers, it was concluded that 5 outliers were identified; based on that, I will run the predictive model once again but this including the identified outliers.

Final prediction

```
reduce.predict1$FINAL_PREDICTED_WINS <- round((predict(model8.lm, newdata=reduce.predict1))^2,0)
final.answer <- reduce.predict1[1]
final.answer$PREDICTED_WINS_NO <- reduce.predict$PREDICTED_WINS
final.answer$PREDICTED_WINS_WO <- reduce.predict1$FINAL_PREDICTED_WINS
```

Compare final predictions

Let's see our training data set.

```
summary(baseball.train$TARGET_WINS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   71.00   82.00   80.79   92.00  146.00
```

Let's see our predicted values data set.

```
summary(final.answer[2:3])
```

```
##  PREDICTED_WINS_NO PREDICTED_WINS_WO
##  Min.   :45.00     Min.   :26.00
##  1st Qu.:78.00     1st Qu.:78.00
##  Median :81.00     Median :81.00
##  Mean   :81.06     Mean   :80.63
```

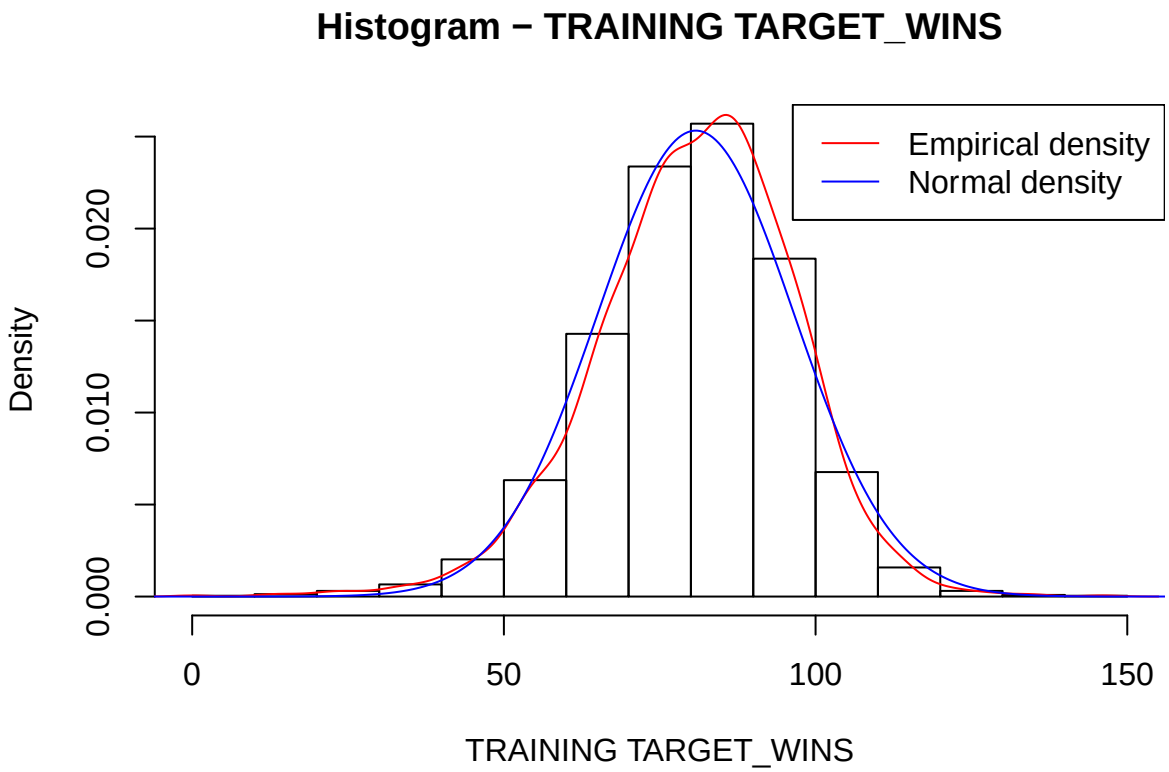
```
##  3rd Qu.:85.00     3rd Qu.:85.00
##  Max.   :96.00     Max.   :96.00
```

Please note that I am presenting two columns; one with ouliers re-evaluation and the other one with no consideration for outliers.

Let's visualize the results.

This first histogram is the repreentation of the training data set.

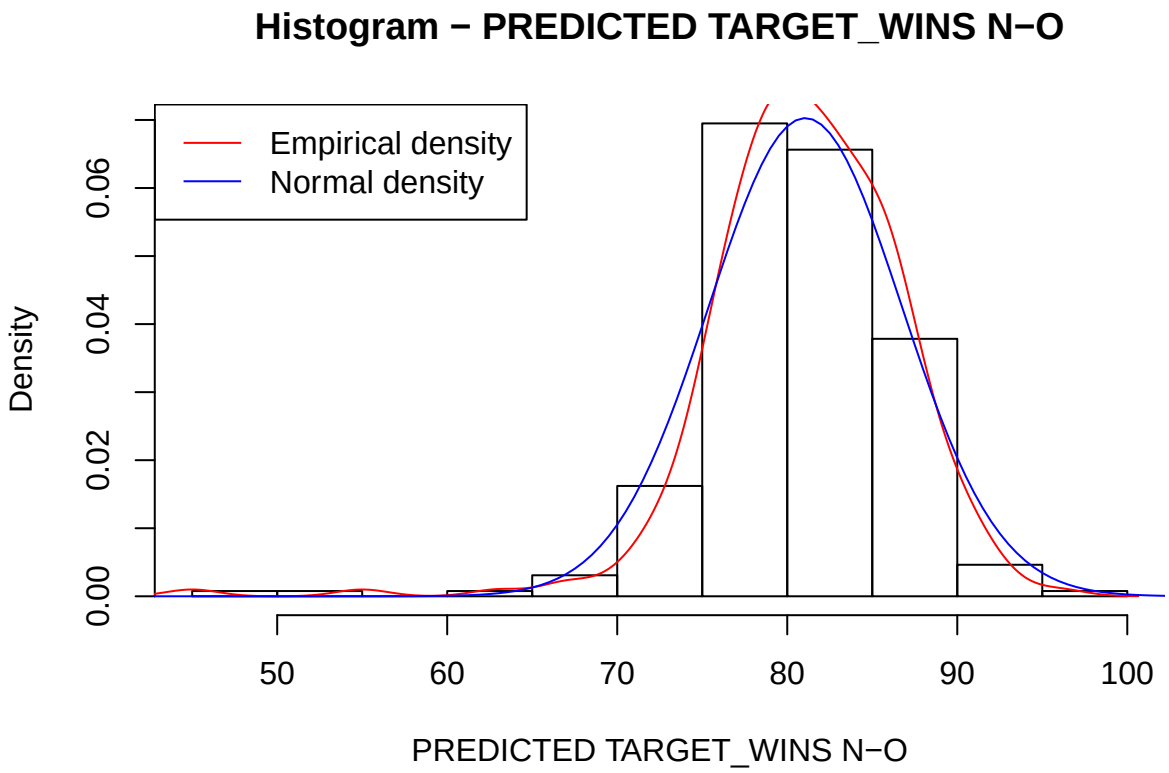
```
hist(baseball.train$TARGET_WINS, freq = FALSE,
   main = paste('Histogram - TRAINING TARGET_WINS'),
   xlab = 'TRAINING TARGET_WINS',
   ylab = 'Density')
lines(density(baseball.train$TARGET_WINS, na.rm = TRUE), col="red")
lines(seq(-400, 500, by=.5),
    dnorm(seq(-400, 500, by=.5),
          mean(baseball.train$TARGET_WINS, na.rm = TRUE),
          sd(baseball.train$TARGET_WINS, na.rm = TRUE)),
    col="blue")
lnames <- c('Empirical density', 'Normal density')
legend('topright', lnames, col = c('red','blue'), lty = 1)
```



This second histogram is the repreentation of the predicted wins data set with NO outliers identified.

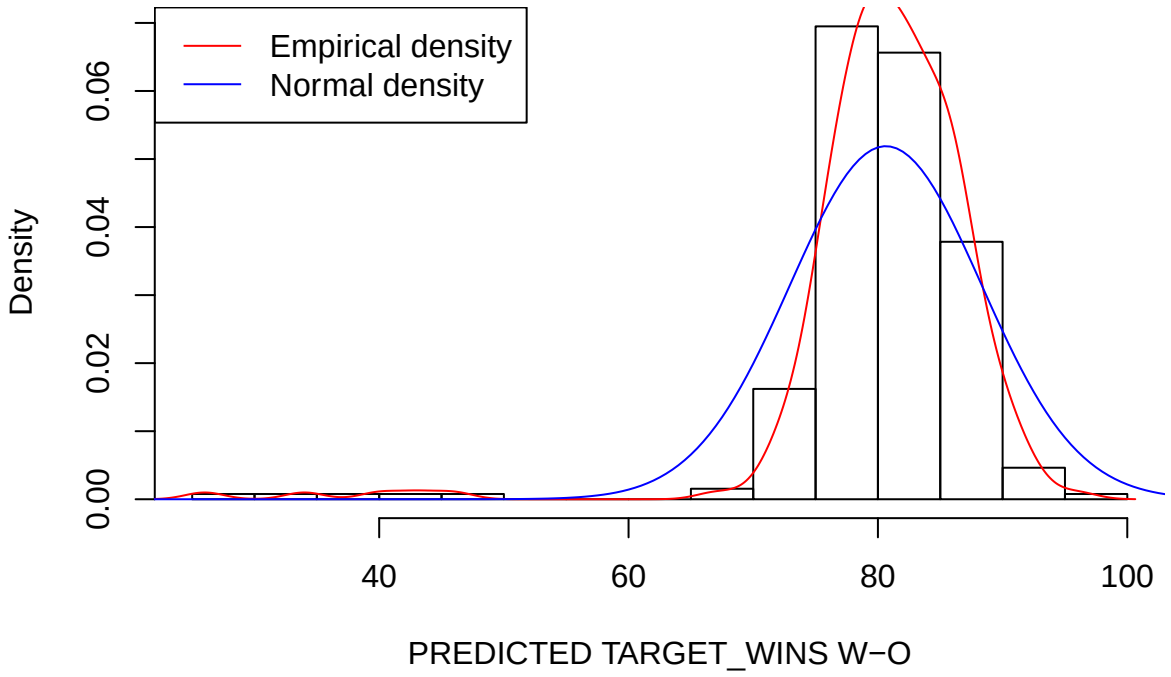
```
hist(final.answer$PREDICTED_WINS_NO, freq = FALSE,
   main = paste('Histogram - PREDICTED TARGET_WINS N-O'),
   xlab = 'PREDICTED TARGET_WINS N-O',
   ylab = 'Density')
lines(density(final.answer$PREDICTED_WINS_NO, na.rm = TRUE), col="red")
lines(seq(-400, 500, by=.5),
    dnorm(seq(-400, 500, by=.5),
```

```
          mean(final.answer$PREDICTED_WINS_NO, na.rm = TRUE),
          sd(final.answer$PREDICTED_WINS_NO, na.rm = TRUE)),
    col="blue")
lnames <- c('Empirical density', 'Normal density')
legend('topleft', lnames, col = c('red','blue'), lty = 1)
```



This second histogram is the repreentation of the predicted wins data set with outliers identified.

```
hist(final.answer$PREDICTED_WINS_WO, freq = FALSE,
   main = paste('Histogram - PREDICTED TARGET_WINS W-O'),
   xlab = 'PREDICTED TARGET_WINS W-O',
   ylab = 'Density')
lines(density(final.answer$PREDICTED_WINS_WO, na.rm = TRUE), col="red")
lines(seq(-400, 500, by=.5),
    dnorm(seq(-400, 500, by=.5),
          mean(final.answer$PREDICTED_WINS_WO, na.rm = TRUE),
          sd(final.answer$PREDICTED_WINS_WO, na.rm = TRUE)),
    col="blue")
lnames <- c('Empirical density', 'Normal density')
legend('topleft', lnames, col = c('red','blue'), lty = 1)
```



By comparing the above summaries and looking at the graphs, I decided that the best table of **TARGET_WINS** generated from the last model in which I recalculate values by considering some values considered as outliers, provide a "better"" table of results. Thus it only could possible affect 5 % of values from our predicted values which is manageable considering the low number of possible errors.

```
baseball.eval$TARGET_WINS <- final.answer$PREDICTED_WINS_WO

#write.csv(baseball.eval, file = "moneyball-my-evaluated-data.csv",row.names=FALSE)
```

APENDIX

Final table: **TARGET WINS**

```
baseball.eval[c('INDEX','TARGET_WINS')]
```

```
##      INDEX TARGET_WINS
## 1        9          72
## 2       10          70
## 3       14          74
## 4       47          88
## 5       60          79
## 6       63          81
## 7       74          86
## 8       83          73
## 9       98          76
## 10     120          76
## 11     123          80
## 12     135          78
## 13     138          76
```

```
## 14    140          77
## 15    151          75
## 16    153          74
## 17    171          77
## 18    184          84
## 19    193          73
## 20    213          88
## 21    217          86
## 22    226          85
## 23    230          81
## 24    241          76
## 25    291          83
## 26    294          85
## 27    300          74
## 28    348          80
## 29    350          88
## 30    357          84
## 31    367          89
## 32    368          83
## 33    372          85
## 34    382          83
## 35    388          80
## 36    396          81
## 37    398          80
## 38    403          82
## 39    407          80
## 40    410          80
## 41    412          86
## 42    414          87
## 43    436          40
## 44    440          89
## 45    476          82
## 46    479          87
## 47    481          84
## 48    501          81
## 49    503          78
## 50    506          80
## 51    519          81
## 52    522          84
## 53    550          80
## 54    554          74
## 55    566          81
## 56    578          77
## 57    596          84
## 58    599          79
## 59    605          46
## 60    607          78
## 61    614          84
## 62    644          80
## 63    692          84
## 64    699          84
## 65    700          82
## 66    716          90
## 67    721          78
```

```
## 68    722          78
## 69    729          79
## 70    731          82
## 71    746          80
## 72    763          79
## 73    774          79
## 74    776          77
## 75    788          81
## 76    789          78
## 77    792          82
## 78    811          82
## 79    835          78
## 80    837          79
## 81    861          89
## 82    862          88
## 83    863          91
## 84    871          83
## 85    879          79
## 86    887          78
## 87    892          77
## 88    904          79
## 89    909          83
## 90    925          92
## 91    940          78
## 92    951          91
## 93    976          75
## 94    981          77
## 95    983          83
## 96    984          81
## 97    989          85
## 98    995          86
## 99   1000          87
## 100  1001          89
## 101  1007          85
## 102  1016          78
## 103  1027          81
## 104  1033          76
## 105  1070          82
## 106  1081          83
## 107  1084          78
## 108  1098          87
## 109  1150          86
## 110  1160          80
## 111  1169          86
## 112  1172          90
## 113  1174          86
## 114  1176          86
## 115  1178          82
## 116  1184          88
## 117  1193          87
## 118  1196          86
## 119  1199          82
## 120  1207          82
## 121  1218          82
```

```
## 122  1223          76
## 123  1226          72
## 124  1227          43
## 125  1229          76
## 126  1241          79
## 127  1244          80
## 128  1246          77
## 129  1248          86
## 130  1249          86
## 131  1253          86
## 132  1261          85
## 133  1305          81
## 134  1314          80
## 135  1323          80
## 136  1328          82
## 137  1353          80
## 138  1363          77
## 139  1371          88
## 140  1372          83
## 141  1389          75
## 142  1393          72
## 143  1421          88
## 144  1431          82
## 145  1437          80
## 146  1442          76
## 147  1450          83
## 148  1463          84
## 149  1464          81
## 150  1470          83
## 151  1471          82
## 152  1484          86
## 153  1495          26
## 154  1507          72
## 155  1514          77
## 156  1526          75
## 157  1549          90
## 158  1552          83
## 159  1556          82
## 160  1564          74
## 161  1585          89
## 162  1586          90
## 163  1590          85
## 164  1591          91
## 165  1592          87
## 166  1603          81
## 167  1612          77
## 168  1634          79
## 169  1645          78
## 170  1647          84
## 171  1673          83
## 172  1674          85
## 173  1687          83
## 174  1688          87
## 175  1700          85
```

```
## 176  1708          79
## 177  1713          80
## 178  1717          76
## 179  1721          76
## 180  1730          79
## 181  1737          85
## 182  1748          85
## 183  1749          84
## 184  1763          83
## 185  1768          86
## 186  1778          93
## 187  1780          92
## 188  1782          82
## 189  1784          76
## 190  1794          96
## 191  1803          76
## 192  1804          79
## 193  1819          79
## 194  1832          80
## 195  1833          84
## 196  1844          76
## 197  1847          79
## 198  1854          79
## 199  1855          78
## 200  1857          81
## 201  1864          75
## 202  1865          78
## 203  1869          79
## 204  1880          84
## 205  1881          82
## 206  1882          81
## 207  1894          86
## 208  1896          85
## 209  1916          87
## 210  1918          79
## 211  1921          90
## 212  1926          80
## 213  1938          81
## 214  1979          74
## 215  1982          75
## 216  1987          79
## 217  1997          78
## 218  2004          88
## 219  2011          80
## 220  2015          82
## 221  2022          84
## 222  2025          84
## 223  2027          88
## 224  2031          81
## 225  2036          75
## 226  2066          77
## 227  2073          80
## 228  2087          86
## 229  2092          84
```

```
## 230  2125          77
## 231  2148          78
## 232  2162          83
## 233  2191          78
## 234  2203          79
## 235  2218          80
## 236  2221          77
## 237  2225          81
## 238  2232          78
## 239  2267          82
## 240  2291          76
## 241  2299          83
## 242  2317          83
## 243  2318          82
## 244  2353          86
## 245  2403          76
## 246  2411          83
## 247  2415          78
## 248  2424          85
## 249  2441          75
## 250  2464          86
## 251  2465          86
## 252  2472          67
## 253  2481          86
## 254  2487          34
## 255  2500          72
## 256  2501          77
## 257  2520          85
## 258  2521          87
## 259  2525          87
```