Regression Analysis of Baseball Team Performance

Abdellah AitElmouden | Gabriel Abreu | Jered Ataky | Patrick Maloney

2/12/2021

Abstract

To see how regression will help us evaluate baseball team performance, this project is designed to explore whether a teams success in any given season can be predicted or explained by any number of statistics in that season. Our goal is to build a multiple linear regression model on the training data to predict the number of wins for the team. we will explore, analyze and model a historical baseball data set containing approximately 2200 records. Each record represents a professional baseball team from the years 1871 to 2006 inclusive, and the data include the performance of the team for the given year, with all of the statistics adjusted to match the performance of a 162 game season.

While correlation does not imply causation, it is suggested that a focus on some of the variables such as single hits or triple or more hits to the exclusion of doubles might be worth pursuing. Also the data suggests that a focus on home runs allowed may not be worth giving up a number of more normal hits.

Introduction

Because baseball is so numbers-heavy, there are many different statistics to consider when searching for the best predictors of team success. There are offensive statistics (offense meaning when a team is batting) and defensive statistics (defense meaning when a team is in the field). These categories can be broken up into many more subcategories. However, for the purpose of the this project we will use the available data to build a multiple linear regression model on the training data to predict the number of wins for the team.

To see how regression will help us predict the number of wins for the team, we actually don't need to understand all the details about the game of baseball, which has over 100 rules. Here, we distill the sport to the basic knowledge one needs to know how to effectively attack the data science problem. The goal of a baseball game is to score more runs (points) than the other team. Each team has 9 batters that have an opportunity to hit a ball with a bat in a predetermined order. After the 9th batter has had their turn, the first batter bats again, then the second, and so on. Each time a batter has an opportunity to bat, we call it a plate appearance (PA). At each PA, the other team's pitcher throws the ball and the batter tries to hit it. The PA ends with an binary outcome: the batter either makes an out (failure) and returns to the bench or the batter doesn't (success) and can run around the bases, and potentially score a run (reach all 4 bases). Each team gets nine tries, referred to as innings, to score runs and each inning ends after three outs (three failures).

Data Exploration

The dataset we will be using was provided in csv file. The files contain 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. The game statistics that will be used in this study are the following:

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
INDEX	Identification Variable (do not use)	None
TARGET_WINS	Number of wins	Outcome Variable
TEAM_BATTING_H	Base Hits by batters (1B,2B,3B,HR)	Positive Impact on Wins
TEAM_BATTING_2B	Doubles by batters (2B)	Positive Impact on Wins
TEAM_BATTING_3B	Triples by batters (3B)	Positive Impact on Wins
TEAM_BATTING_HR	Homeruns by batters (4B)	Positive Impact on Wins
TEAM_BATTING_BB	Walks by batters	Positive Impact on Wins
TEAM_BATTING_HBP	Batters hit by pitch (get a free base)	Positive Impact on Wins
TEAM_BATTING_SO	Strikeouts by batters	Negative Impact on Wins
TEAM_BASERUN_SB	Stolen bases	Positive Impact on Wins
TEAM_BASERUN_CS	Caught stealing	Negative Impact on Wins
TEAM_FIELDING_E	Errors	Negative Impact on Wins
TEAM_FIELDING_DP	Double Plays	Positive Impact on Wins
TEAM_PITCHING_BB	Walks allowed	Negative Impact on Wins
TEAM_PITCHING_H	Hits allowed	Negative Impact on Wins
TEAM_PITCHING_HR	Homeruns allowed	Negative Impact on Wins
TEAM_PITCHING_SO	Strikeouts by pitchers	Positive Impact on Wins

The initial steps are to download the data and take a quick glimpse of the columns, their data types, number of columns, and rows. Based on initial observations, the data contains 2276 teams with a variety of baseball performance statistics.

At first glance, the column BATTING_HBP has numerous NA values that will need to be addressed before building a model. Figure 1 show summary statistics of the target wins. The noteworthy statistics are the average number of wins in a season is 81 games, the median number of wins in a season is 82 games, and the standard deviation is 16 games.

Figure 1 : Summary Statistics

Characteristic	**N = 2,276**
TARGET_WINS	81 (16) 82 0 146
TEAM_BATTING_H	1,469 (145) 1,454 891 2,554
TEAM_BATTING_2B	241 (47) 238 69 458
TEAM_BATTING_3B	55 (28) 47 0 223
TEAM_BATTING_HR	100 (61) 102 0 264
TEAM_BATTING_BB	502 (123) 512 0 878
TEAM_BATTING_HBP	59 (13) 58 29 95
TEAM_BATTING_SO	736 (249) 750 0 1,399
TEAM_BASERUN_SB	125 (88) 101 0 697
TEAM_BASERUN_CS	53 (23) 49 0 201
TEAM_FIELDING_E	246 (228) 159 65 1,898
TEAM_FIELDING_DP	146 (26) 149 52 228
TEAM_PITCHING_BB	553 (166) 536 0 3,645
TEAM_PITCHING_SO	818 (553) 814 0 19,278

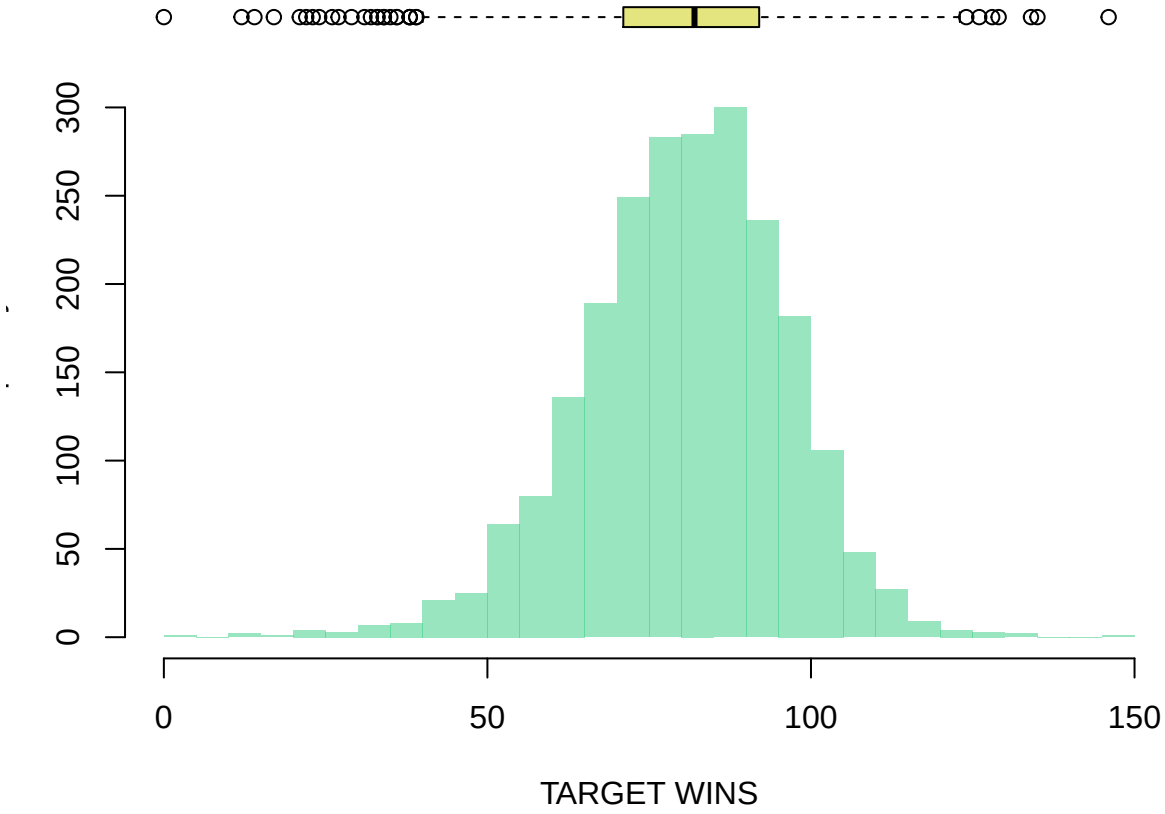
Mean (SD) Median Minimum Maximum

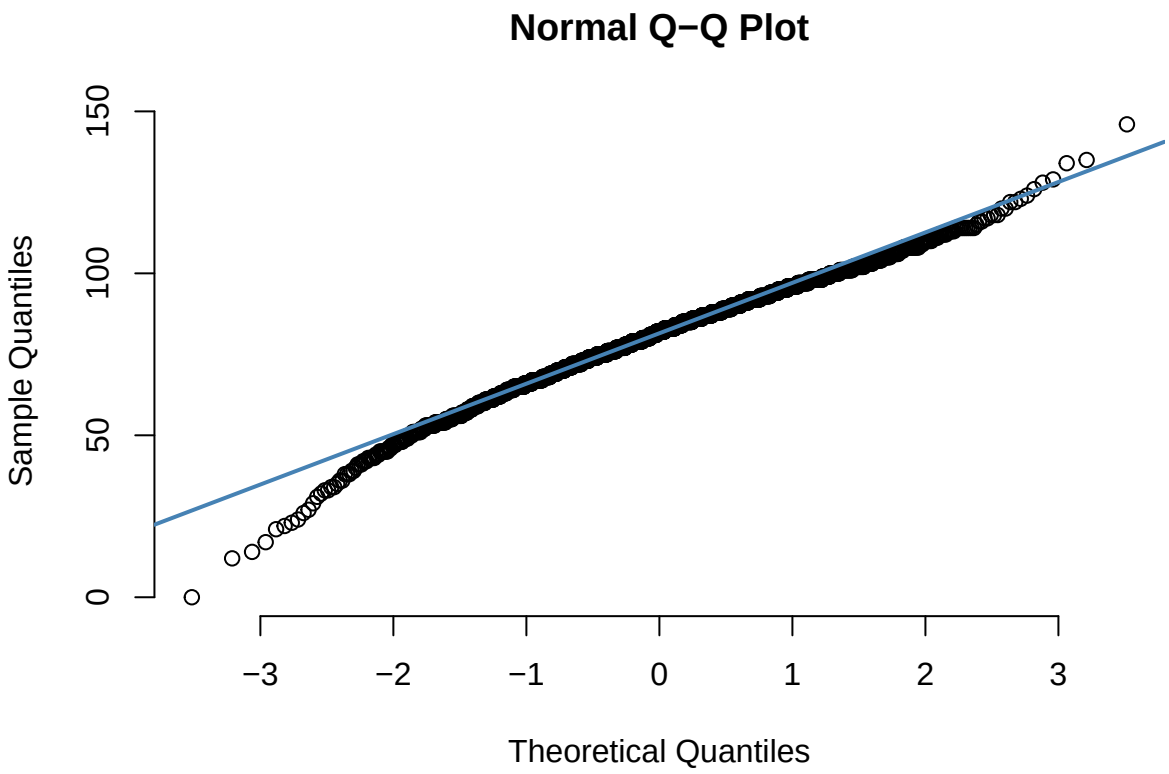
By examining the target wins variable in detail, there is a clear guideline of how many wins each team should approximately win. Most teams will likely win the average number of games (81), but there will be some variability from the average with some teams winning more or less than 81 games.

The other variables also play an important role in understanding the data. In Figure 1, summary statistics are presented for all the variables. it is sufficient in getting the gist of each variable's distribution. For example, the average Base Hits by batters per team is 1469 with the minimum base hits at 891 and maximum base hits at 2554. Remember that the dataset contains baseball statistics on 2276 teams. Missing values were excluded from the summary and they will be dealt with in the data preparation section of this report.

A quick look at Figure 2 will reveal the distribution of the target wins. The distribution is approximately normal with a majority of the target wins falling in the center of the distribution.The approximate normal distribution is confirmed by the QQ plot below the distribution plot. Most of the target wins fall on the line in the QQ plot with some data points diverging at the ends. This indicates possibility of outliers where some teams are winning more games or losing more games than what is expected in the normal range.In the boxplot, there are points that fall outside the whiskers which confirms our suspicions of outliers seen in the QQ plot.

Figure 2 : Distribution and Probability Plot for TARGET_WINS



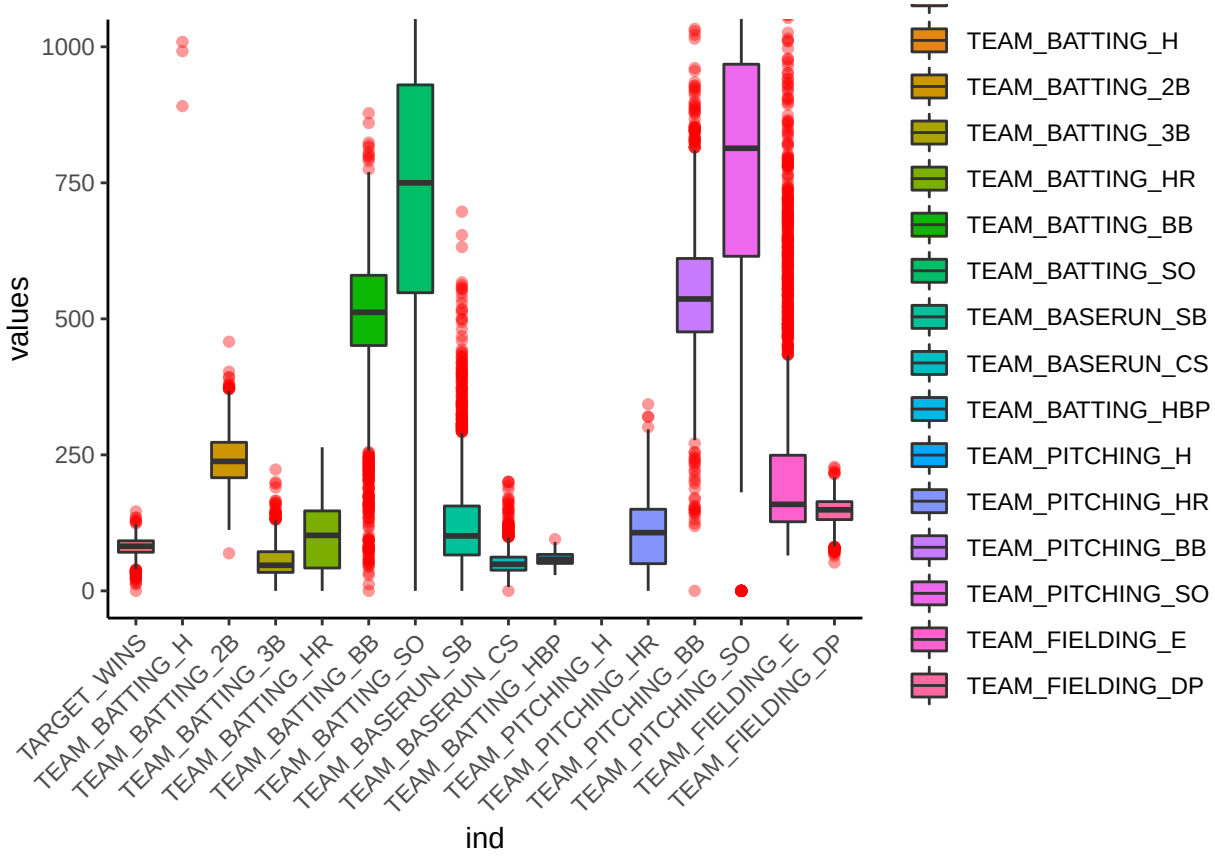


Now in order to build our models properly It's worth exploring for other columns with NA values.

Columns_w_NA	Percent_NA
team_batting_so	4.481547
team_baserun_sb	5.755712
team_baserun_cs	33.919156
team_batting_hbp	91.608084
team_pitching_so	4.481547
team_fielding_dp	12.565905

Outliers

The following diagram shows the outliers for all the variables, both dependent and independent.



As we can see from the graph only 4 of the 16 variables are normally or close to normally distributed. the other 12 variables have a significant skew. The response variable Target_wins seems to be normally distributed. Batting_Hr, Batting_SO and Pitching_HR are bi-modal. 10 of the 16 variables have a minimum value of 0. This is not a major concern as the total % of 0 in each column is less than 1%. The variables Batting_BB, Batting_CS, Baserun_SB, Pitching_BB and Fielding_E have a significant number of outliers.

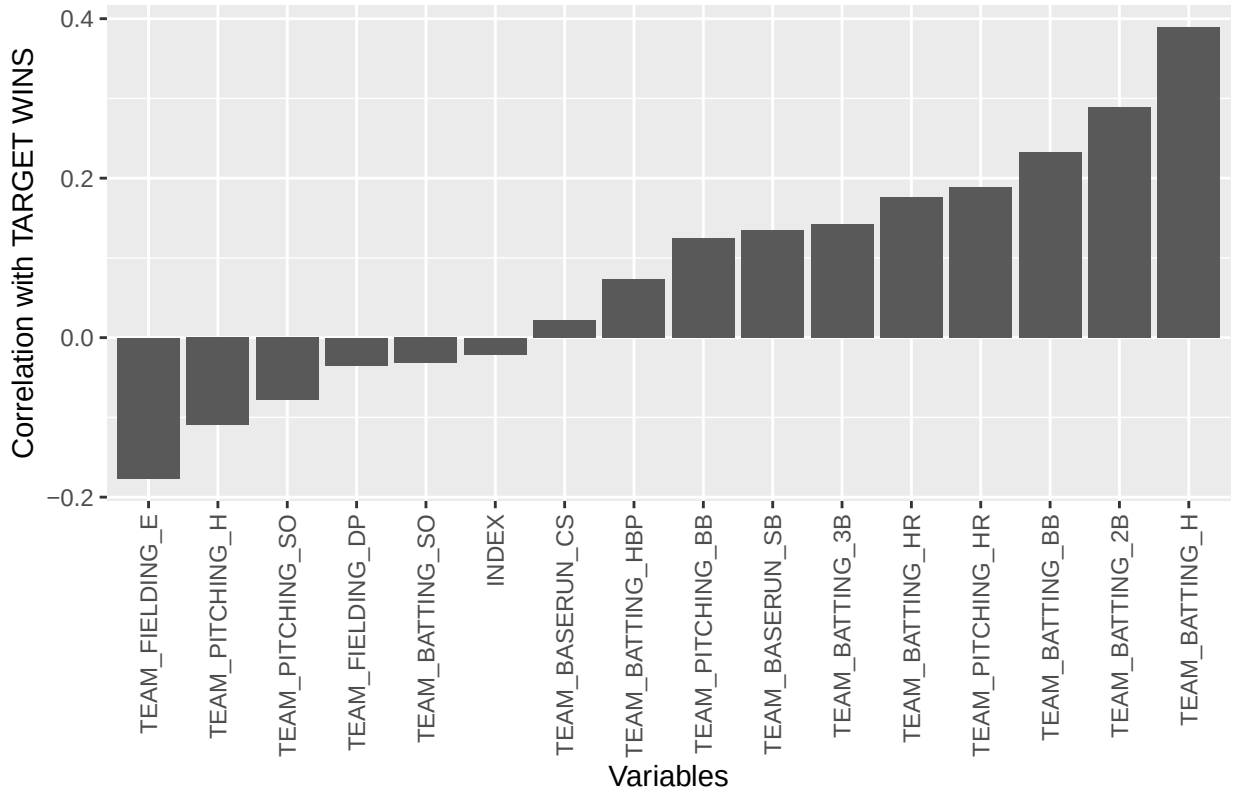
Correlations among predictors and Variable Selection

It is possible that not all variables will need to be used in creating an accurate model. In Figure 4, a correlation value is computed for each variable against target wins. Some variables are highly correlated with target wins, while other variables are not. For example, Base Hits by batters has a value of 0.38877 which is high while Caught stealing is barely correlated with target wins with a value of 0.0224.There is also a column for p-values which indicates whether the correlations are significant. We can use a decision rule of 95% meaning any variable with a p-value of less than 0.05 is significant. It appears that Strike-outs by batters (TEAM_BATTING_SO),Caught stealing(TEAM_BASERUN_CS), Batters hit by pitch (TEAM_BATTING_HBP), and Double plays (TEAM_FIELDING_DP)do not meet our decision rule and could be excluded from use.

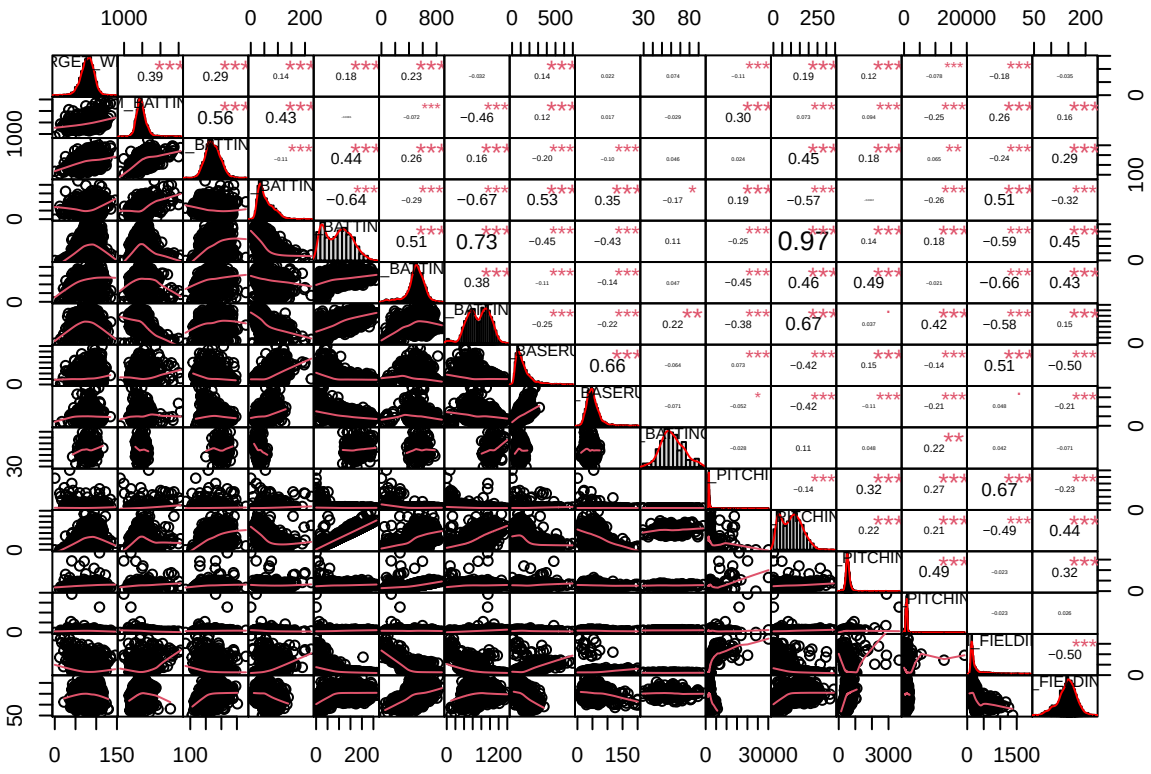
term	TARGET_WINS
INDEX	-0.02105643
TEAM_BATTING_H	0.38876752
TEAM_BATTING_2B	0.28910365
TEAM_BATTING_3B	0.14260841
TEAM_BATTING_HR	0.17615320
TEAM_BATTING_BB	0.23255986
TEAM_BATTING_SO	-0.03175071

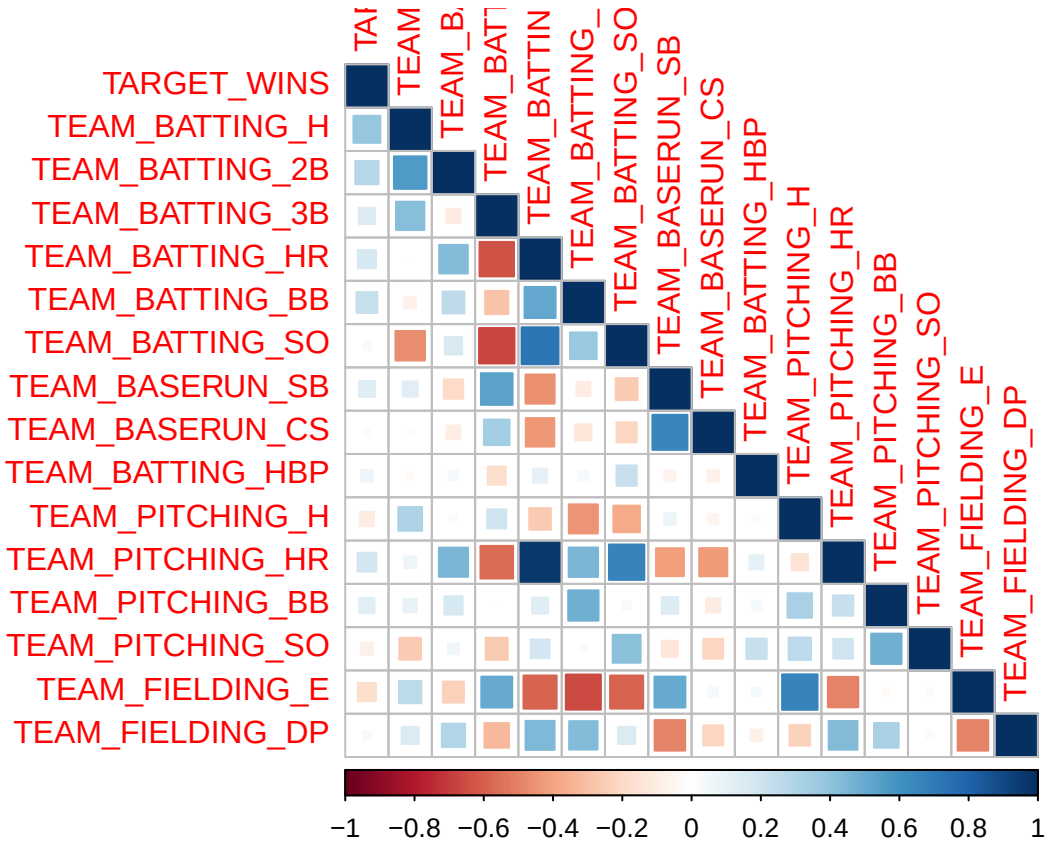
TEAM_BASERUN_SB	0.13513892
TEAM_BASERUN_CS	0.02240407
TEAM_BATTING_HBP	0.07350424
TEAM_PITCHING_H	-0.10993705
TEAM_PITCHING_HR	0.18901373
TEAM_PITCHING_BB	0.12417454
TEAM_PITCHING_SO	-0.07843609
TEAM_FIELDING_E	-0.17648476
TEAM_FIELDING_DP	-0.03485058

Figure 4: Correlation Against Target Win



Before entirely excluding variables, it is a good idea to transform the data by fixing missing values or combining variables and reexamine the viability of those variables for predicting wins.





From the table we can see that there are positive or negative correlations among the predictors. If we look at the numerical correlations with the response variable. We can see that the predictors Batting_H, Batting_HR, Batting_BB, Pitching_H, and Pitching_HR are more correlated and should be included in our regression.

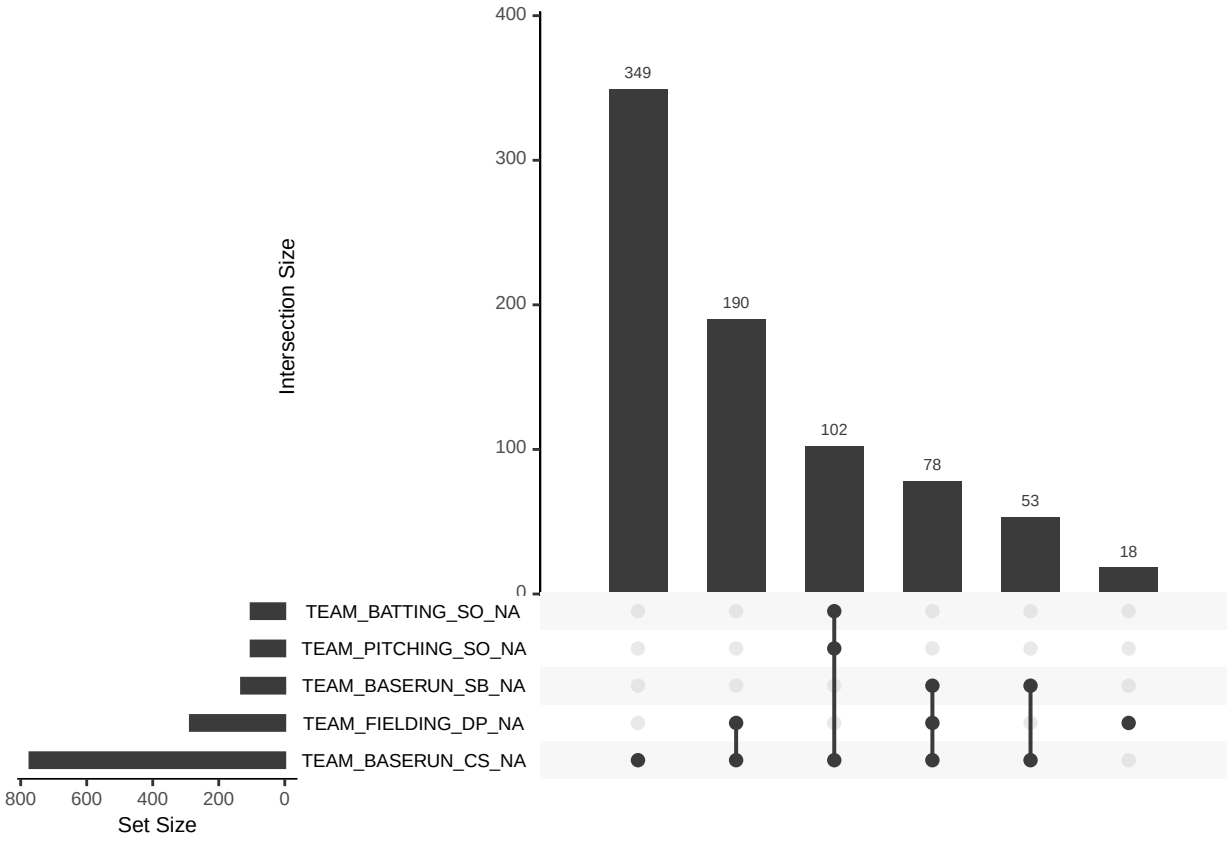
Also Examining significant correlations among the independent variables, we see that four of the pairs have a correlation close to 1. This can lead to multicollinearity issues in our analysis.

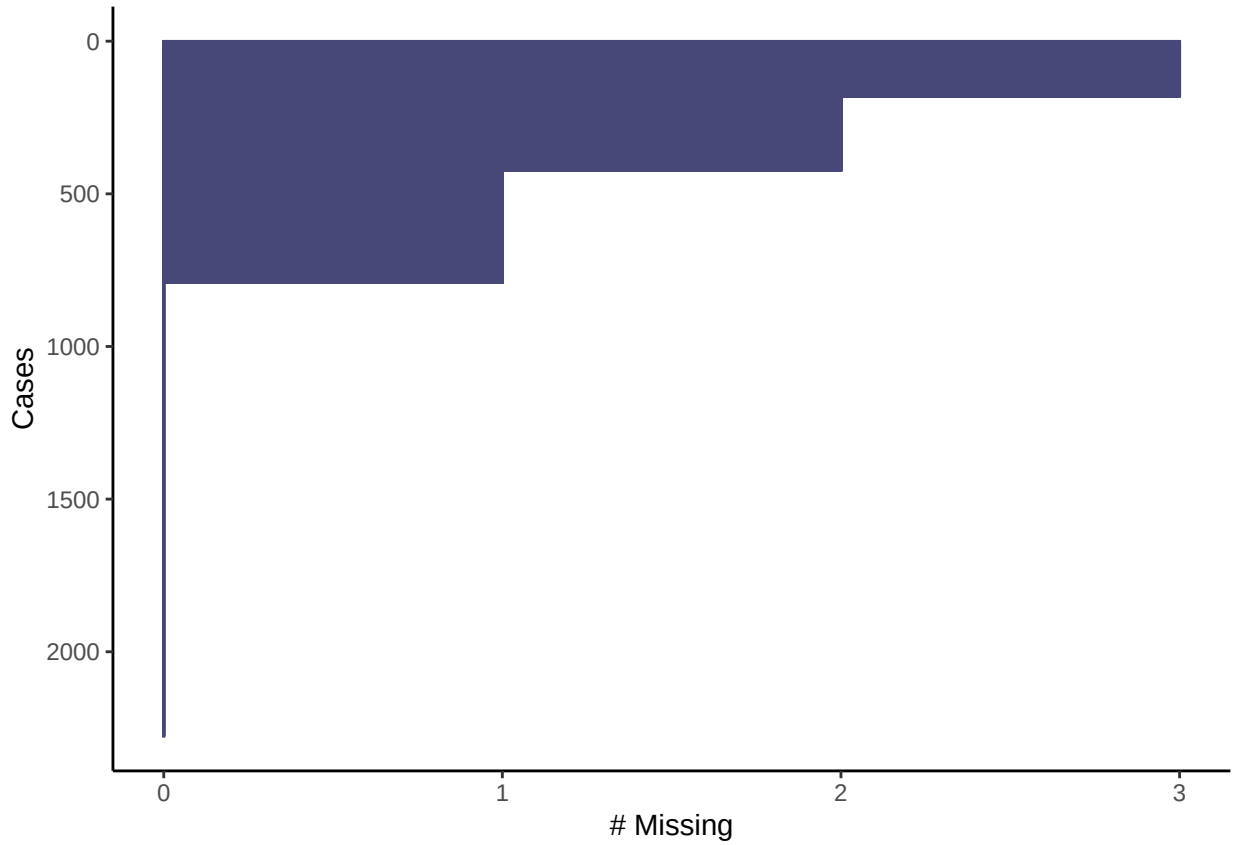
Data Preparation

Missing values need to be handled before building models.They can be handled by either dropping the records, dropping the entire variable, or imputation. In this case, it was determined that Batters hit by pitch variable should be dropped altogether prior to model building because it has too many missing values to properly impute.All other variables with missing values will be considered for the model because a majority of the records are not missing.These variables will be imputed.

First we will remove Batting_HBP (Hit by Pitch) which has 92% missing values.

We will look at the patterns and intersections of missingness among the variables, using the naniar package. We can see that only 22 of the observations have all 5 variables missing, we will just delete these cases. The pattern suggests that the variables are Missing at Random (MAR)


Intersection Size
400
300
200
100
0
349
190
102
78
53
18
TEAM_BATTING_SO_NA
TEAM_PITCHING_SO_NA
TEAM_BASERUN_SB_NA
TEAM_FIELDING_DP_NA
TEAM_BASERUN_CS_NA
800
600
400
200
0
Set Size



By looking at the patterns and intersections of missing data among the variables. We can see that 5 variables have missing values, Team_BATTING has the most missing values so we are completely removing these observations. Overall, the pattern suggests that the variables are Missing at Random (MAR).

When it comes to fixing missing values,there are several methods at our disposal. The first technique is to fill the missing values with the mean values of each variable. We'll use the Hmisc R Package to fill the missing data with the mean, most of the time, mean imputation will lead to good results. The same procedure will be used for the other variables with missing values in Model 2 but by using the Median instead of the mean

The second technique for imputing missing values is to use a decision tree. This is slightly more involved, but will likely give the better results. A decision tree will be created for each variable with missing values. In mean imputation, a fixed value is used for missing values of an entire variable whereas in decision tree imputation, a value is used based on certain conditions.

Build Models

MODEL 1: MEAN FULL MODEL

This is a full model containing all the variables with the meanusred for missing values. This is a good starting model to determine how well each variable helps predict wins. The mean is generally an adequate guess for missing values. In this model, no selection technique is used. All variables are manually included.

To the that we used the Hmisc R package to imputes missing value using user defined statistical method (mean in our case)

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
```

```
##     TEAM_BASERUN_SB + TEAM_BASERUN_CS + TEAM_PITCHING_H + TEAM_PITCHING_HR +
##     TEAM_PITCHING_BB + TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP,
##     data = train_model1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.113  -7.633  -0.018   7.324  45.214
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      59.237509   6.113895   9.689  < 2e-16 ***
## TEAM_BATTING_H   -0.002070   0.005441  -0.380 0.703633
## TEAM_BATTING_2B  -0.023765   0.008808  -2.698 0.007034 **
## TEAM_BATTING_3B   0.168568   0.018723   9.003  < 2e-16 ***
## TEAM_BATTING_HR   0.321816   0.055324   5.817 6.98e-09 ***
## TEAM_BATTING_BB   0.093055   0.018164   5.123 3.30e-07 ***
## TEAM_BATTING_SO  -0.049966   0.008917  -5.603 2.40e-08 ***
## TEAM_BASERUN_SB   0.081187   0.006161  13.178  < 2e-16 ***
## TEAM_BASERUN_CS  -0.059350   0.014616  -4.061 5.09e-05 ***
## TEAM_PITCHING_H   0.025458   0.002263  11.251  < 2e-16 ***
## TEAM_PITCHING_HR -0.216833   0.052137  -4.159 3.33e-05 ***
## TEAM_PITCHING_BB -0.060526   0.016914  -3.578 0.000354 ***
## TEAM_PITCHING_SO  0.028051   0.007993   3.509 0.000459 ***
## TEAM_FIELDING_E  -0.084678   0.005292 -16.001  < 2e-16 ***
## TEAM_FIELDING_DP -0.120255   0.012485  -9.632  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.96 on 1975 degrees of freedom
##   (286 observations deleted due to missingness)
## Multiple R-squared:  0.3858, Adjusted R-squared:  0.3814
## F-statistic: 88.61 on 14 and 1975 DF,  p-value: < 2.2e-16
```

The overall p-value for Model 1is less than 0.0001,which indicates a significant model in predicting wins. The Adjusted R-Squared and Mean Square Error (MSE) will be the metrics used to determine the best model.A higher Adjusted R-Squared is better and a lower MSE is better.In this case, the Adjusted R-Squared is 0.3814 and MSE is 168.91, which will be the current benchmark.

The resulting equation for Model 1 is :

WINS =+ 25.06831
+0.0473146*Base Hits by batters
-0.0209806*Doubles by batters
+0.0692224*Triples by batters
+0.0680963*Homeruns by batters
+0.0108690*Walks by batters
-0.0081244*Strikeouts by batters
+0.0299345*Stolen bases
-0.01173*Caught stealing
-0.00073149*Hits allowed
+0.01481*Homeruns allowed
+0.00008066*Walks allowed
+0.0026600*Strikeouts by pitchers
-0.02118*Errors
-0.1208451*Double plays

Most of Model 1 makes senseas positive measures of success like Base Hits, Triples, Homeruns, Walks by batters, and Stolen bases are positive coefficients in the equation while negative measures of success like Strikeouts by batters, Caught stealing, Hits allowed, and Errors are negative coefficients in the equation.All these values make intuitive sense.On the other hand, Doubles and Double plays are shown as negative coefficients when they should have a positive impact on wins. Also, Homeruns allowed and Walks allowed are shown as positive coefficients when they should have negative impact on wins.These values are counter intuitive.The counter intuitive parts of the model may need to be further investigated if this model were to be chosen for deployment.However, for now, the model will be kept as a benchmark despite certain measures not making sense.

MODEL 2: MEDIAN WITH STEPWISE

Earlier in the exploration of the data, the analysis revealed the possibility of outliers present in the data. Because the mean is highly influenced by outliers, this model attempts to remedy that by using the median to impute missing values.Model 2 is a significant model based on the p-value of less than 0.0001. The Adjusted R-Squared is 0.3147 and MSE is 169.799.

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
##     TEAM_BASERUN_SB + TEAM_BASERUN_CS + TEAM_PITCHING_H + TEAM_PITCHING_HR +
##     TEAM_PITCHING_BB + TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP,
##     data = train_model1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.113  -7.633  -0.018   7.324  45.214
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      59.237509   6.113895   9.689  < 2e-16 ***
## TEAM_BATTING_H   -0.002070   0.005441  -0.380 0.703633
## TEAM_BATTING_2B  -0.023765   0.008808  -2.698 0.007034 **
## TEAM_BATTING_3B   0.168568   0.018723   9.003  < 2e-16 ***
## TEAM_BATTING_HR   0.321816   0.055324   5.817 6.98e-09 ***
## TEAM_BATTING_BB   0.093055   0.018164   5.123 3.30e-07 ***
## TEAM_BATTING_SO  -0.049966   0.008917  -5.603 2.40e-08 ***
## TEAM_BASERUN_SB   0.081187   0.006161  13.178  < 2e-16 ***
## TEAM_BASERUN_CS  -0.059350   0.014616  -4.061 5.09e-05 ***
## TEAM_PITCHING_H   0.025458   0.002263  11.251  < 2e-16 ***
## TEAM_PITCHING_HR -0.216833   0.052137  -4.159 3.33e-05 ***
## TEAM_PITCHING_BB -0.060526   0.016914  -3.578 0.000354 ***
## TEAM_PITCHING_SO  0.028051   0.007993   3.509 0.000459 ***
## TEAM_FIELDING_E  -0.084678   0.005292 -16.001  < 2e-16 ***
## TEAM_FIELDING_DP -0.120255   0.012485  -9.632  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.96 on 1975 degrees of freedom
##   (286 observations deleted due to missingness)
## Multiple R-squared:  0.3858, Adjusted R-squared:  0.3814
## F-statistic: 88.61 on 14 and 1975 DF,  p-value: < 2.2e-16
```

The resulting equation for Model 2 is :

WINS =+ 25.0602
+0.04824*Base Hits by batters
-0.02006*Doubles by batters
+0.06047*Triples by batters
+0.05299*Homeruns by batters
+0.01042*Walks by batters
-0.009349*Strikeouts by batters
+0.02949*Stolen bases
-0.01188*Caught stealing
-0.0007342*Hits allowed
+0.01480*Homeruns allowed
+0.00008891*Walks allowed
+0.002843*Strikeouts by pitchers
-0.02112*Errors
-0.1210*Double plays

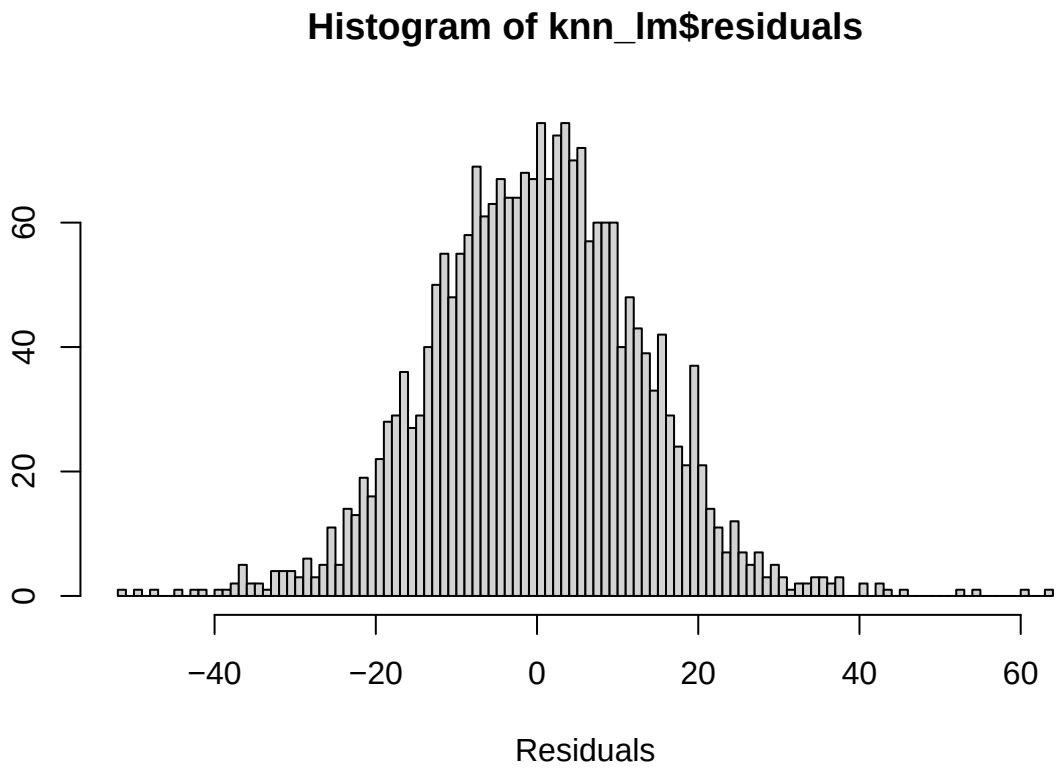
MODEL 3: knn Imputation

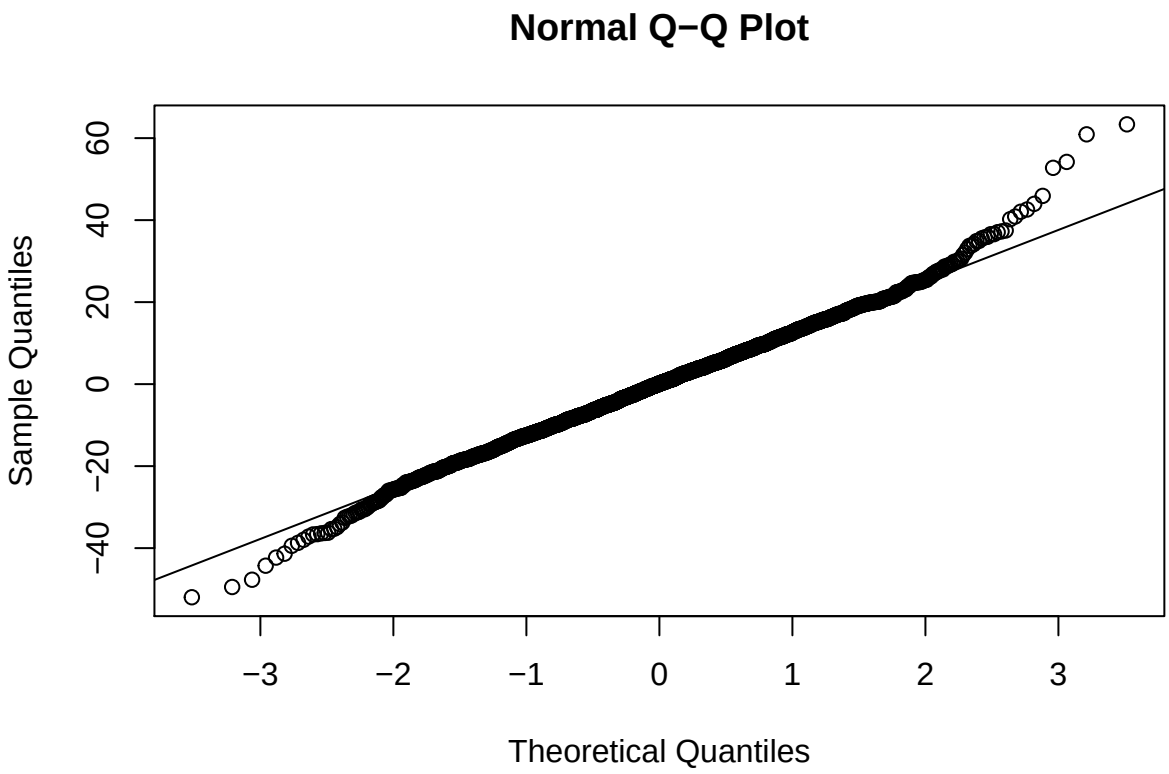
Building the stepwise regression model with knn imputed values

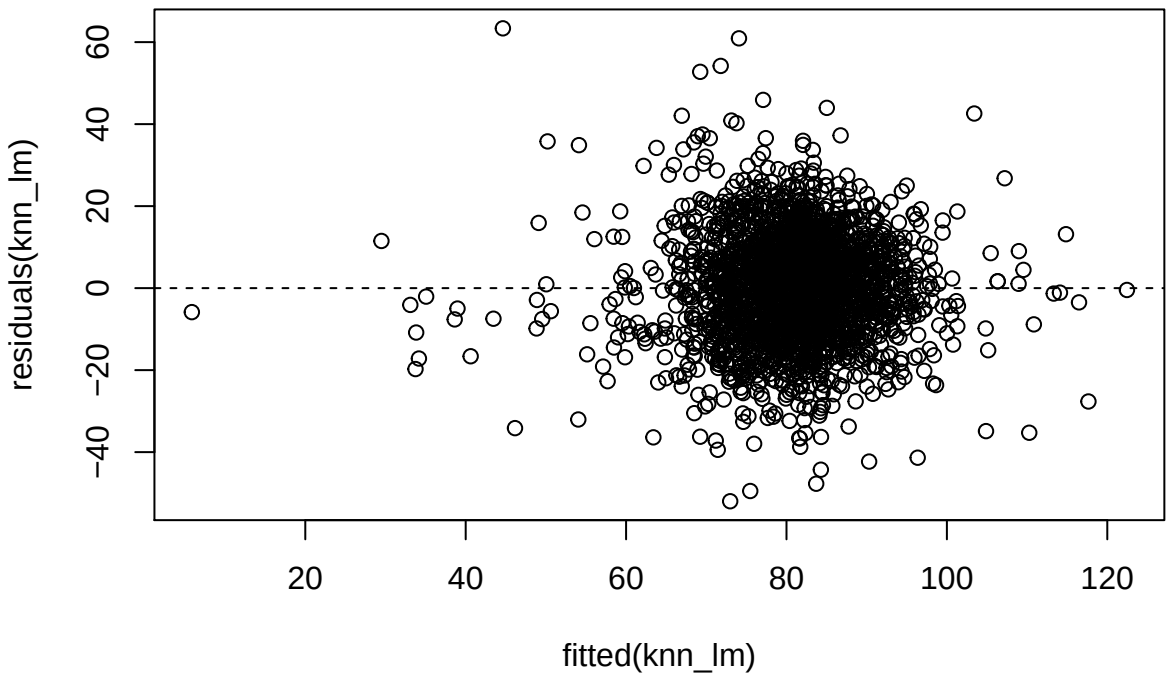
```
##    nvmax     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1      2 13.78932 0.2343893 10.88499 0.6614975 0.07029463 0.5327204
## 2      3 13.44196 0.2718753 10.66673 0.6533293 0.06768041 0.5672393
## 3      4 13.24900 0.2926471 10.52407 0.5980818 0.06413871 0.5609507
## 4      5 13.13180 0.3054890 10.41344 0.5694226 0.05939499 0.5481207
## 5      6 13.04114 0.3152270 10.30041 0.5724188 0.05945760 0.5268071
## 6      7 13.03826 0.3163928 10.25790 0.5631394 0.06009273 0.4779677
## 7      8 13.13591 0.3061616 10.26111 0.7867598 0.07393537 0.5471808
## 8      9 13.15876 0.3039878 10.31866 0.7998756 0.07291900 0.4994315
## 9     10 13.13067 0.3073919 10.25398 0.8163058 0.07716460 0.5208940
## 10    11 13.26476 0.2946175 10.40889 0.5152782 0.05234317 0.3645624
## 11    12 13.28846 0.2913946 10.35418 0.9590106 0.08530676 0.5822280
## 12    13 13.21327 0.2998969 10.30424 0.9759242 0.08878852 0.6051774
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_SO +
##     TEAM_BASERUN_SB + TEAM_PITCHING_HR + TEAM_FIELDING_E + TEAM_FIELDING_DP,
##     data = knn_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.980  -8.547   0.148   8.398  63.362
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      35.203403   4.656401   7.560 5.80e-14 ***
## TEAM_BATTING_H    0.044863   0.002599  17.263  < 2e-16 ***
## TEAM_BATTING_SO  -0.010323   0.002091  -4.937 8.52e-07 ***
## TEAM_BASERUN_SB   0.037616   0.003784   9.942  < 2e-16 ***
## TEAM_PITCHING_HR  0.059940   0.007971   7.520 7.85e-14 ***
## TEAM_FIELDING_E  -0.027353   0.001582 -17.289  < 2e-16 ***
## TEAM_FIELDING_DP -0.117420   0.013006  -9.028  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 13.13 on 2269 degrees of freedom
## Multiple R-squared:  0.3069, Adjusted R-squared:  0.3051
## F-statistic: 167.5 on 6 and 2269 DF,  p-value: < 2.2e-16
```




Normal Q−Q Plot
Sample Quantiles
−40
−20
0
20
40
60
−3
−2
−1
0
1
2
3
Theoretical Quantiles



Now lets run the model...

Select A Model

Let compare all the models based on with different measures stated in the table below:

Measures	Model 1	Model 2	Model 3
r_squared	0.3858	0.3858	0.3067
adj_rsquared	0.3814	0.3814	0.3049
rse	10.9600	10.9600	13.1300
f_stat	88.6100	88.6100	167.3000

Appendix

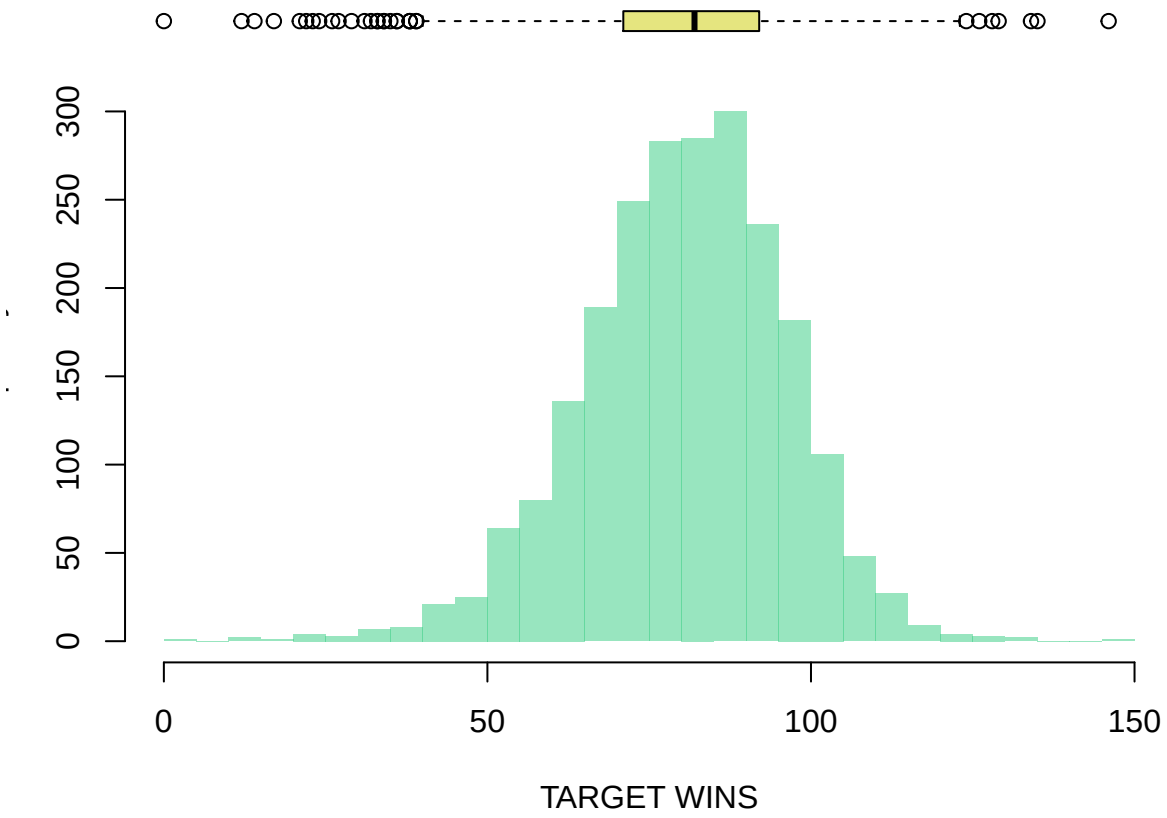
```
# 0.  Librairies

library(corrplot)
library(tidyverse)
library(Hmisc)
library(PerformanceAnalytics)
library(corrplot)
library(mice)
library(gt)
library(DMwR)
library(caret)
library(bnstruct)
```

```r
library(VIM)
library(corrr)
library(tidyverse)
library(gtsummary)
library(kableExtra)
```

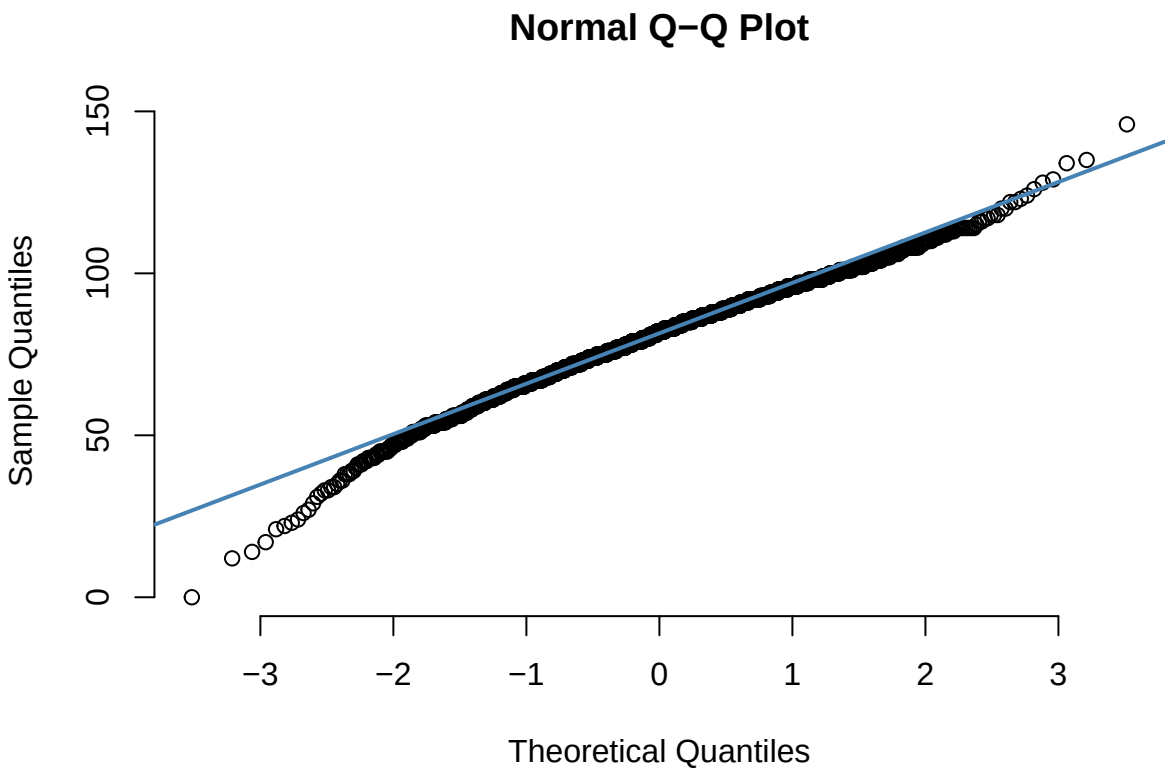
```r
# 1. Data Exploration

## Import data

train_data <- read.csv("https://raw.githubusercontent.com/aaitelmouden/DATA621/master/Project1/moneyball
glimpse(train_data)


## Summary table

wins <- train_data %>% select(TARGET_WINS, TEAM_BATTING_H, TEAM_BATTING_2B, TEAM_BATTING_3B, TEAM_BATTI
table1 <- tbl_summary(wins,
          statistic = list(all_continuous() ~ "{mean} ({sd}) {median} {min} {max}"), missing = "no")
table1


## Create more variables

singles <- train_data$TEAM_BATTING_H - (train_data$TEAM_BATTING_2B + train_data$TEAM_BATTING_3B + train_

train_data$TEAM_BATTING_SLG <- ((train_data$TEAM_BATTING_HR *4)+ (train_data$TEAM_BATTING_3B*3) + (trai
```

```r
## Plots

### Layout to split the screen
layout(mat = matrix(c(1,2),2,1, byrow=TRUE),  height = c(1,8))

####  Draw the boxplot and the histogram
par(mar=c(0, 3.1, 1.1, 2.1))
boxplot(train_data$TARGET_WINS ,main="Figure 2 : Distribution and Probability Plot for TARGET_WINS",cex
par(mar=c(4, 3.1, 1.1, 2.1))
hist(train_data$TARGET_WINS , breaks=40 , col=rgb(0.2,0.8,0.5,0.5) , border=F , main="" , xlab="TARGET
```

Figure 2 : Distribution and Probability Plot for TARGET_WINS




```
### qq plots

qqnorm(train_data$TARGET_WINS, pch = 1, frame = FALSE)
qqline(train_data$TARGET_WINS, col = "steelblue", lwd = 2)
```

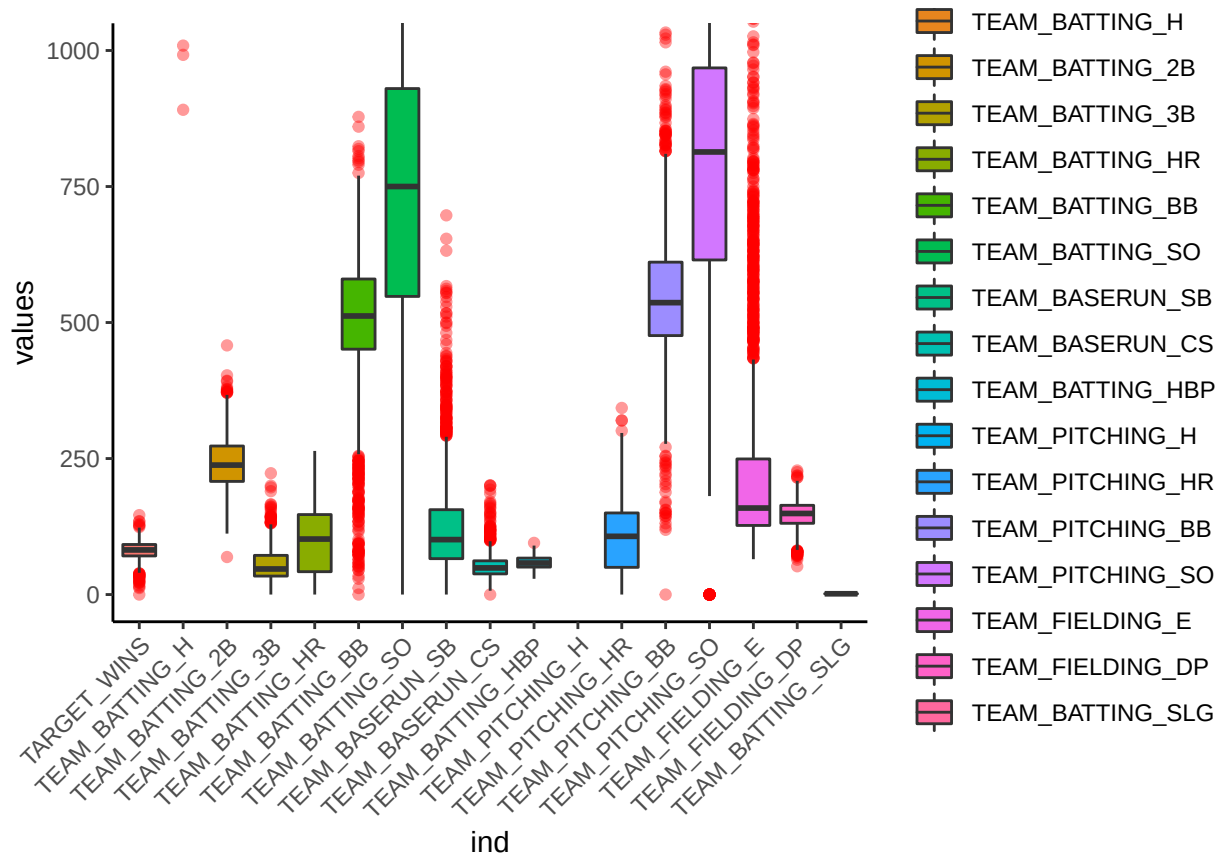



```
max_obs <- 2276
batting_so_na <- ((102/max_obs) * 100)
baserun_sb_na <- (131/max_obs) * 100
baserun_cs_na <- (772/max_obs) * 100
batting_hbp_na <- (2085/max_obs) * 100
pitching_so_na <- (102/max_obs) * 100
fielding_dp_na <- (286/max_obs) * 100

df_percent_na <- data.frame(Columns_w_NA = c("team_batting_so", "team_baserun_sb", "team_baserun_cs", "
```

```
### the largest islands in the world

gt_tbl <- gt(data = df_percent_na, )


### Show the gt Table

gt_tbl
```

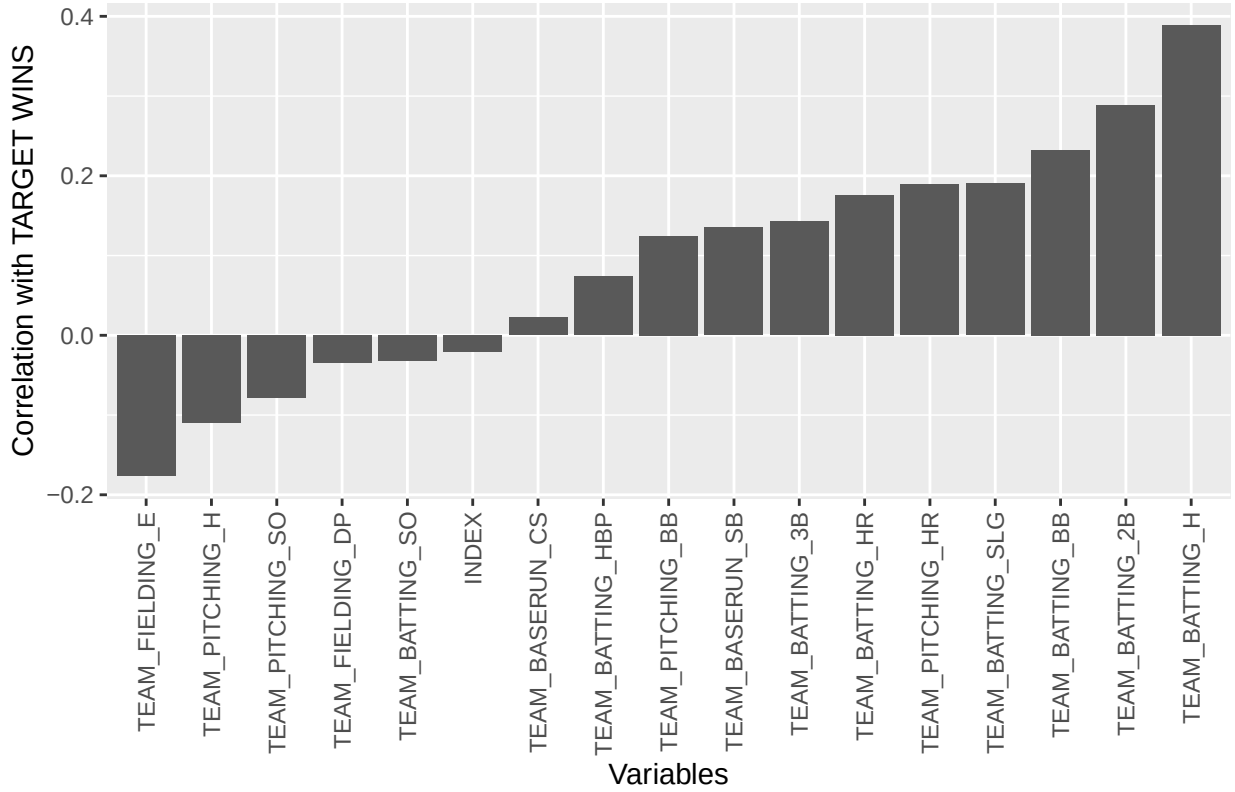
```
### Outlliers

ggplot(stack(train_data[,-1]), aes(x = ind, y = values, fill=ind)) +
  geom_boxplot(outlier.colour = "red",  outlier.alpha=.4) +
  coord_cartesian(ylim = c(0, 1000)) +
  theme_classic()+
```

```
  theme(axis.text.x=element_text(angle=45, hjust=1))
```

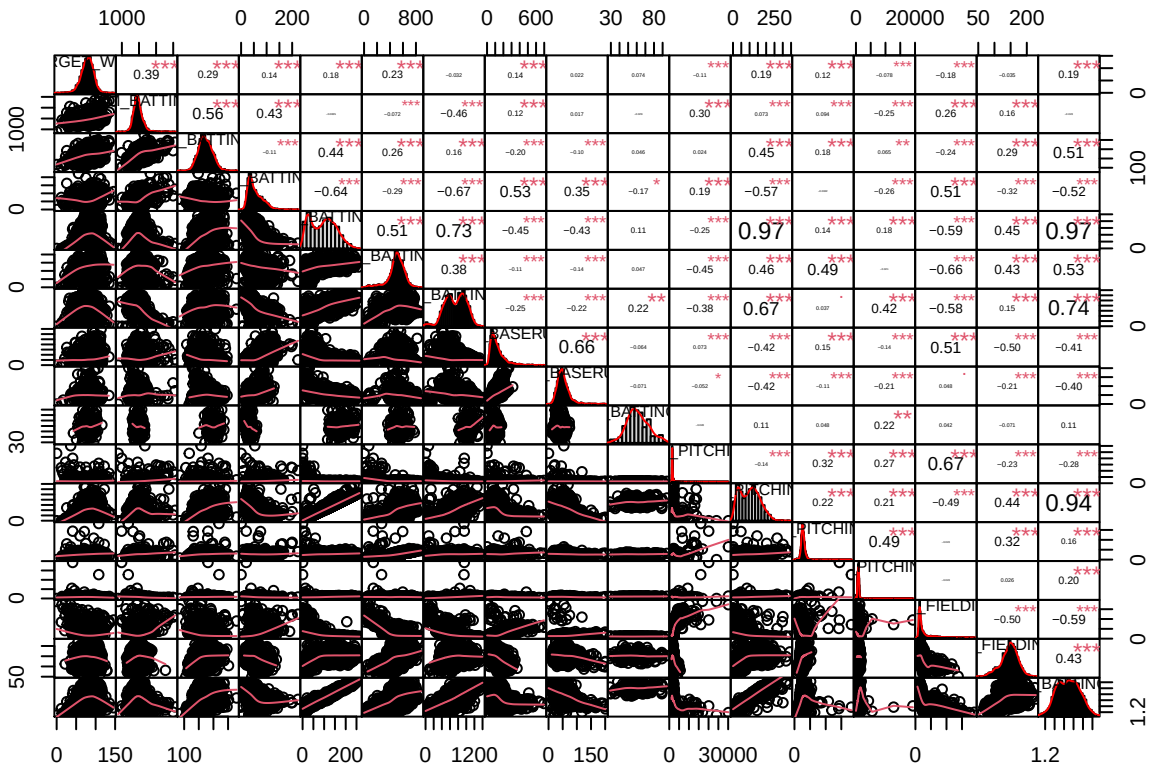



```
### Correlation

COR <- train_data %>%
  correlate() %>%
  focus(TARGET_WINS)
gt(COR)

COR %>%
  mutate(rowname = factor(term, levels = term[order(TARGET_WINS)])) %>%  # Order by correlation strengt
  ggplot(aes(x = rowname, y = TARGET_WINS)) +
    geom_bar(stat = "identity") +
    ylab("Correlation with TARGET WINS") +
    xlab("Variables") + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+ ggtitle("F
```

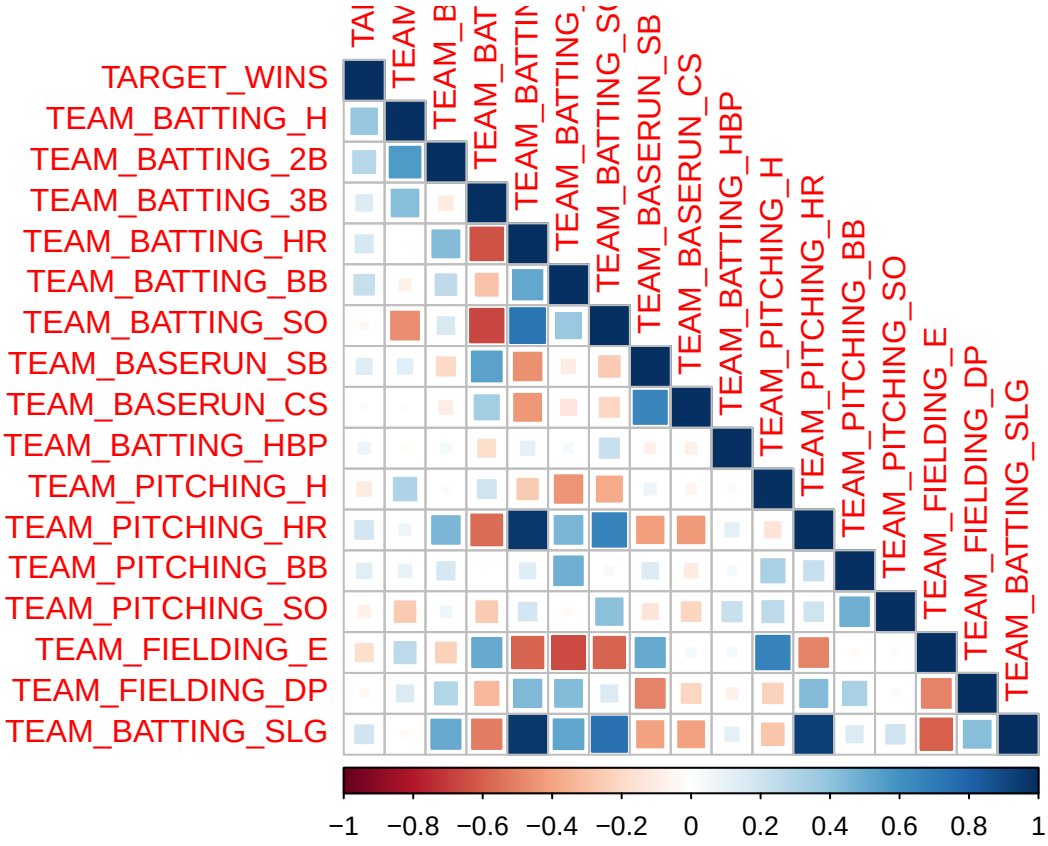
Figure 4: Correlation Against Target Win




```
#### pairwise.complete.obs ignores NA values and computes correlation on complete observations
#### we might have to run these corrplots again after we handle the NA values

chart.Correlation(train_data[-c(1)], histograme=TRUE, method= "pearson", use="pairwise.complete.obs")
```

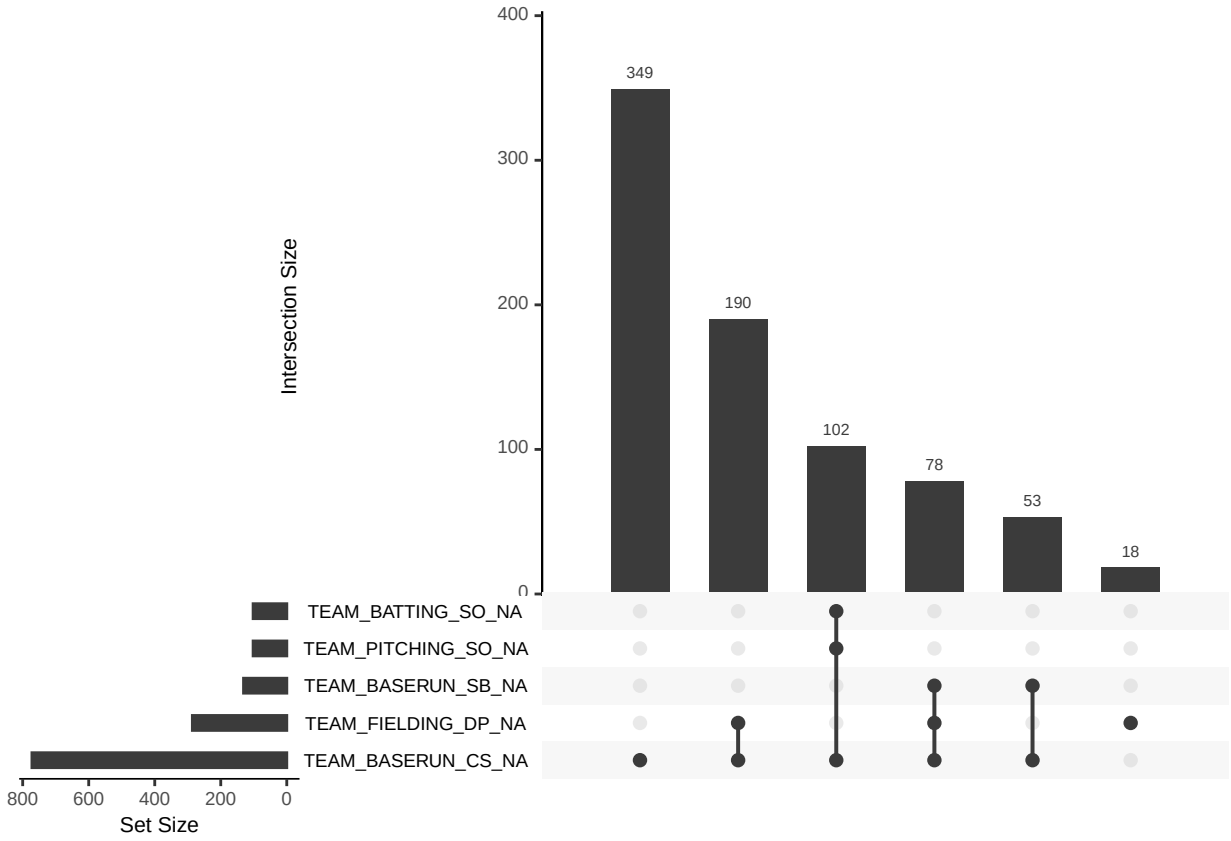


```
data.corr <- cor(train_data[-c(1)], use="pairwise.complete.obs")

corrplot(data.corr, type = "lower", method="square")
```

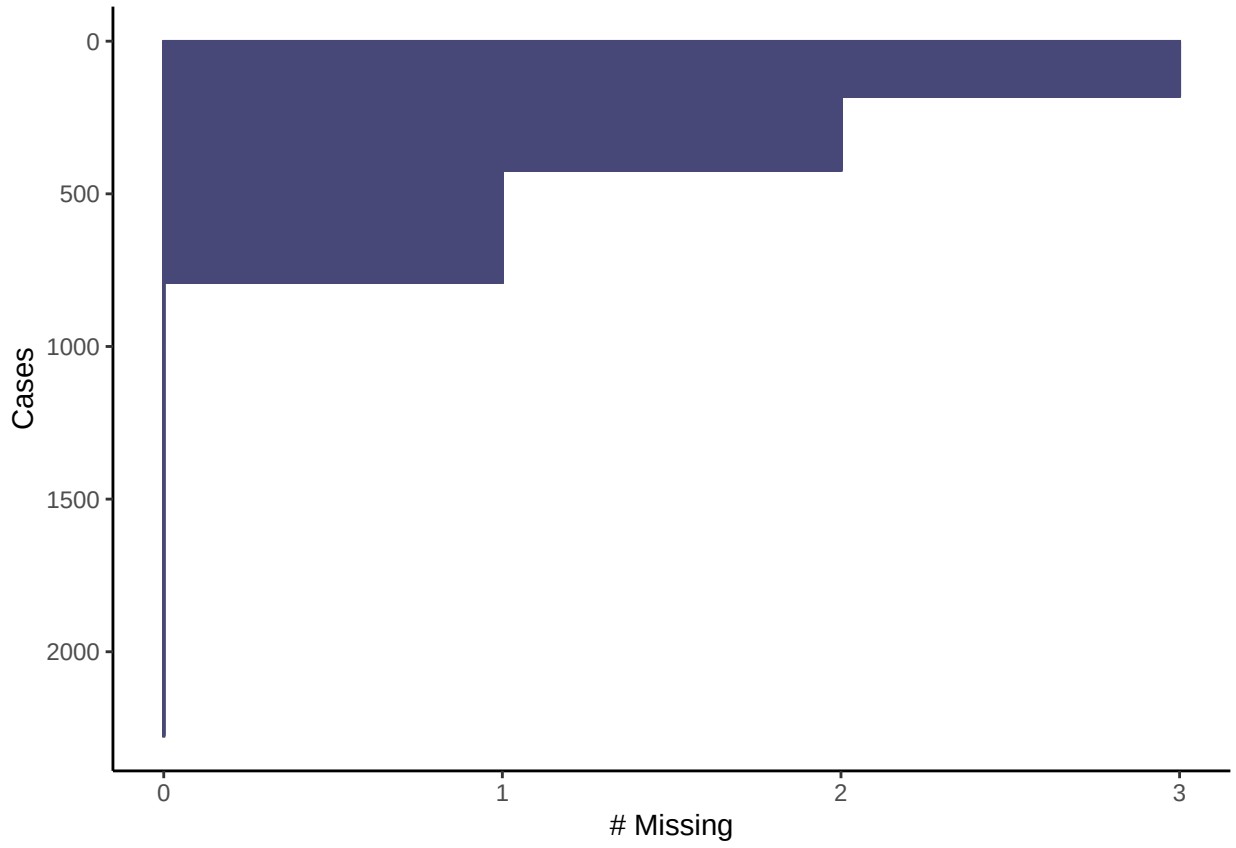



```
flattenCorrMatrix <- function(cormat, pmat) {
  ut <- upper.tri(cormat)
  data.frame(
    row = rownames(cormat)[row(cormat)[ut]],
    column = rownames(cormat)[col(cormat)[ut]],
    cor  =(cormat)[ut],
    p = pmat[ut]
    )
}

#### Eliminate INDEX from data frame
data_no_index <- train_data[-c(1)]

cor_matrix <- rcorr(as.matrix(data_no_index))

flattenCorrMatrix(cor_matrix$r, cor_matrix$P)
```

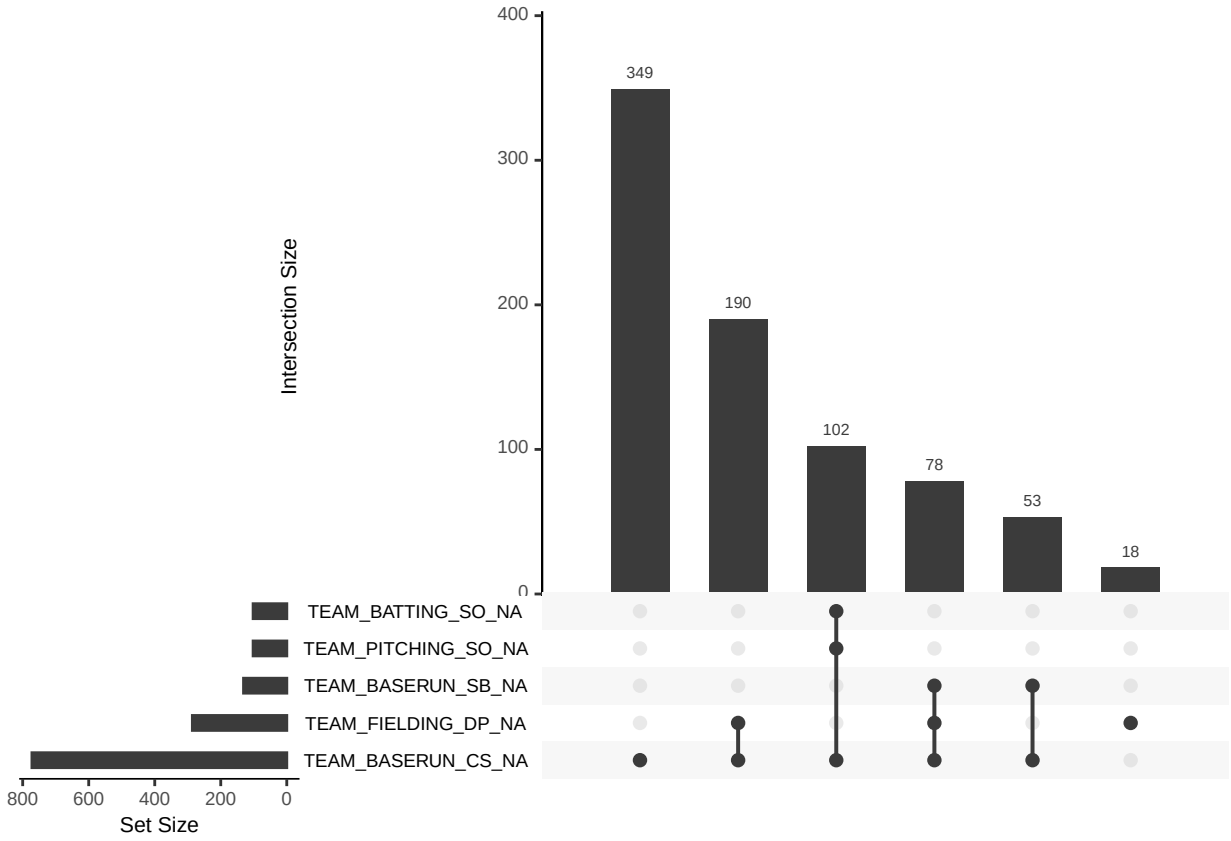
```
#. 2 Data preparation

train_data <- train_data[-11]


par(mfrow=c(1,2))
gg_miss_upset(train_data,
              nsets = 5,
              nintersects = NA)
```



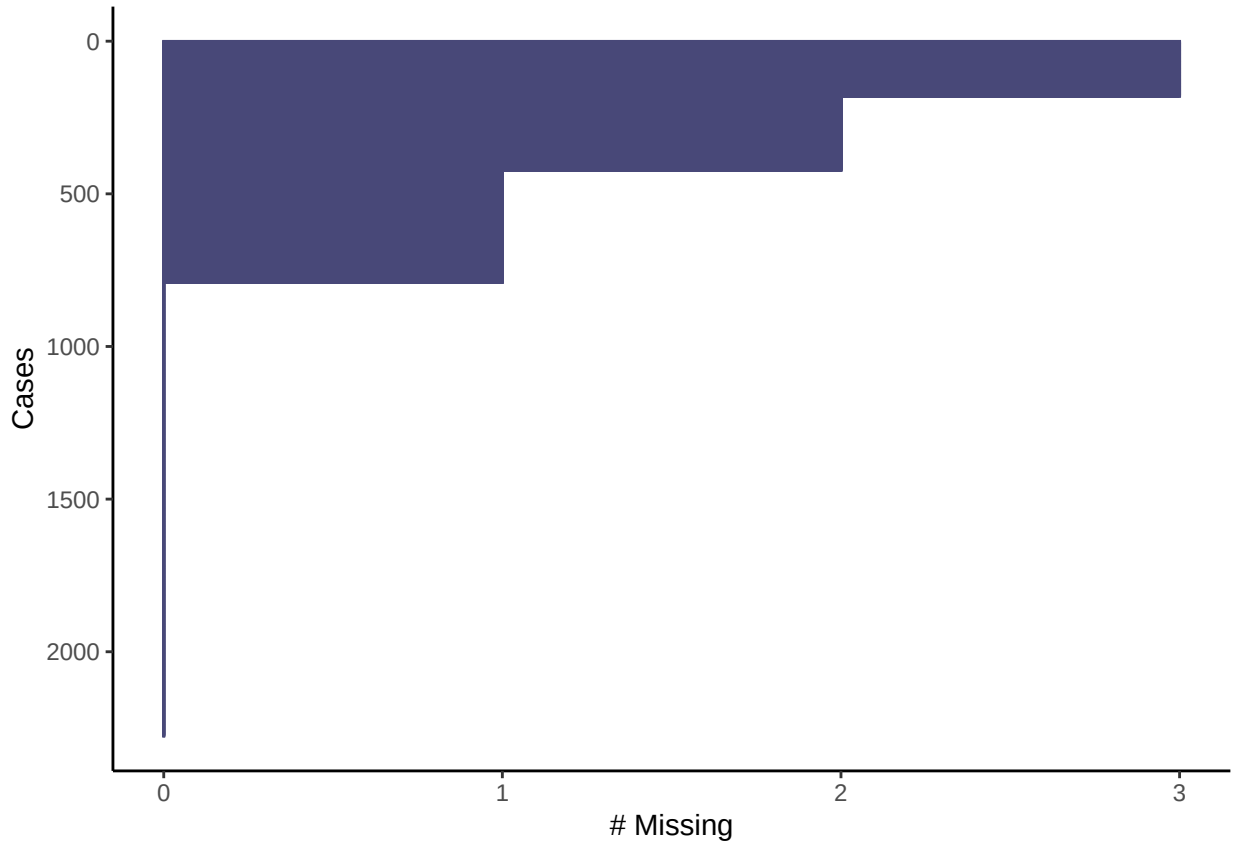

```
gg_miss_case(train_data)+
  theme_classic()
```




```
par(mfrow=c(1,2))
gg_miss_upset(train_data,
              nsets = 5,
              nintersects = NA)
```

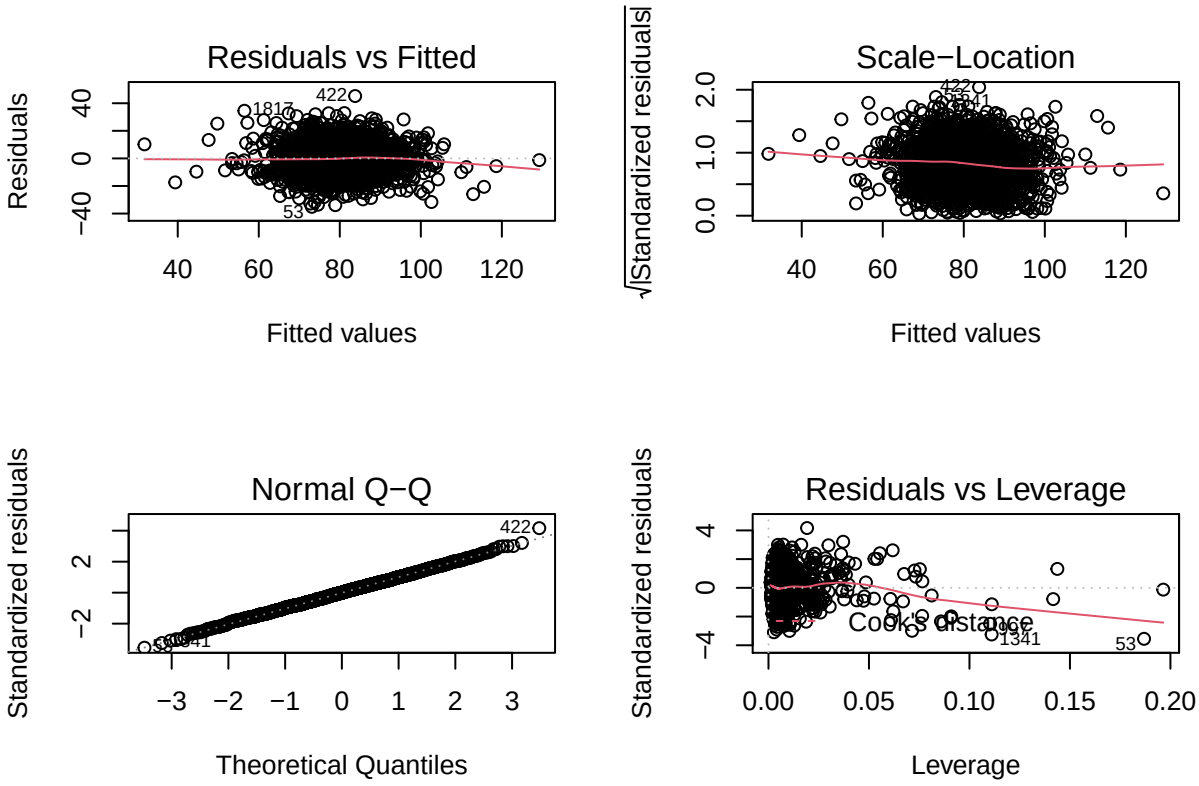



```
gg_miss_case(train_data)+
  theme_classic()
```



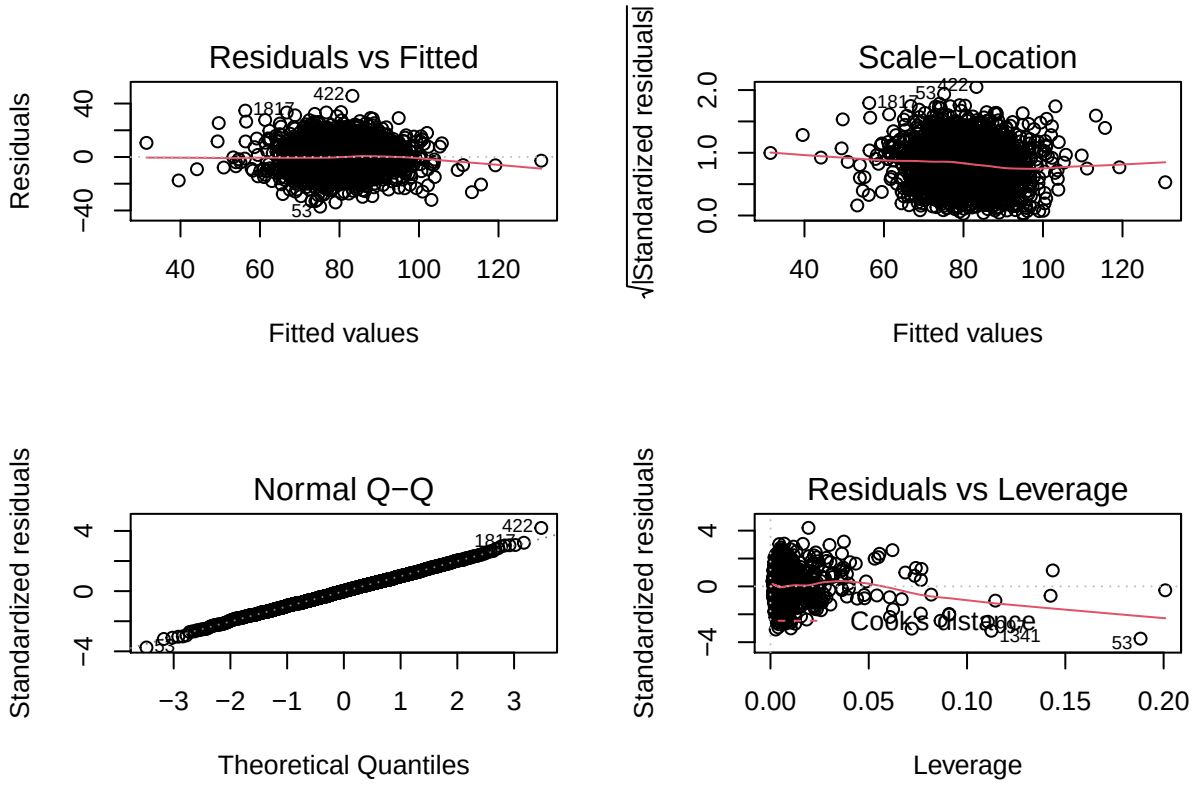

```
# 3. Build models

### MODEL 1: MEAN FULL MODEL

### Filling missing values with Mean using the impute package

train_model1 <- train_data
train_model1$TEAM_BATTING_SO[is.na(train_model1$TEAM_BATTING_SO)] = mean(train_model1$TEAM_BATTING_SO, r
train_model1$TEAM_BASERUN_SB[is.na(train_model1$TEAM_BASERUN_SB)] = mean(train_model1$TEAM_BASERUN_SB, r
train_model1$TEAM_BASERUN_CS[is.na(train_model1$TEAM_BASERUN_CS)] = mean(train_model1$TEAM_BASERUN_CS, r
train_model1$TEAM_PITCHING_SO[is.na(train_model1$TEAM_PITCHING_SO)] = mean(train_model1$TEAM_PITCHING_SO

model1 <- lm(TARGET_WINS ~
               TEAM_BATTING_H +   # Base Hits by batters (1B,2B,3B,HR)
               TEAM_BATTING_2B +  # Doubles by batters (2B)
               TEAM_BATTING_3B +  # Triples by batters (3B)
               TEAM_BATTING_HR +  # Homeruns by batters (4B)
               TEAM_BATTING_BB +  # Walks by batters
               TEAM_BATTING_SO +  # Strikeouts by batters
               TEAM_BASERUN_SB +  # Stolen bases
               TEAM_BASERUN_CS +  # Caught stealing
               TEAM_PITCHING_H +  # Hits allowed
               TEAM_PITCHING_HR + # Homeruns allowed
               TEAM_PITCHING_BB + # Walks allowed
               TEAM_PITCHING_SO + # Strikeouts by pitchers
               TEAM_FIELDING_E +  # Errors
```

```
                TEAM_FIELDING_DP,  # Double Plays
              data=train_model1)
summary(model1)

### Mean Square Error (MSE)

mean(model1$residuals^2)

layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(model1)
```



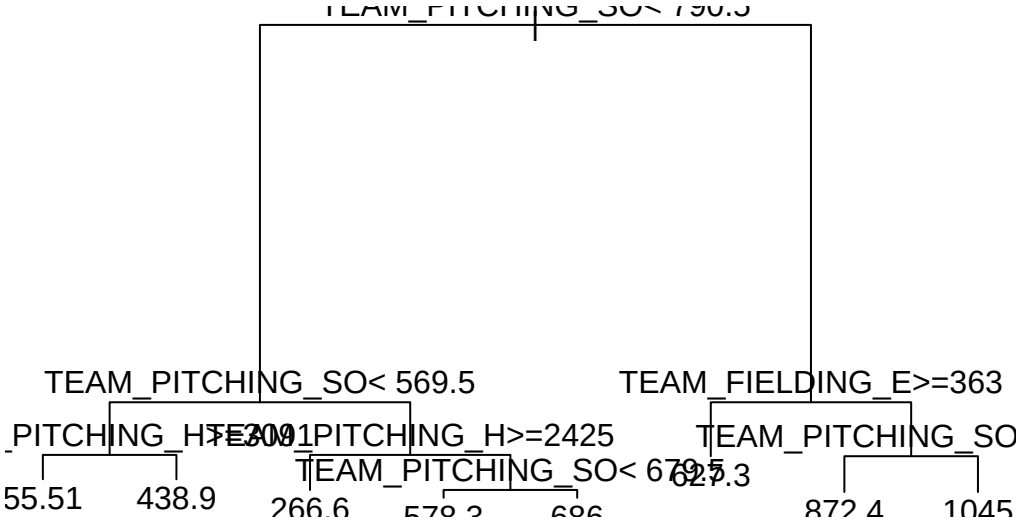

```
### MODEL 2: MEDIAN WITH STEPWISE

train_model2 <- train_data

#### Filling missing values with Mean using the impute package

train_model2$TEAM_BATTING_SO[is.na(train_model2$TEAM_BATTING_SO)] = median(train_model2$TEAM_BATTING_SO
train_model2$TEAM_BASERUN_SB[is.na(train_model2$TEAM_BASERUN_SB)] = median(train_model2$TEAM_BASERUN_SB
train_model2$TEAM_BASERUN_CS[is.na(train_model2$TEAM_BASERUN_CS)] = median(train_model2$TEAM_BASERUN_CS
train_model2$TEAM_PITCHING_SO[is.na(train_model2$TEAM_PITCHING_SO)] = median(train_model2$TEAM_PITCHING_


model2 <- lm(TARGET_WINS ~
               TEAM_BATTING_H +   # Base Hits by batters (1B,2B,3B,HR)
               TEAM_BATTING_2B +  # Doubles by batters (2B)
```

```
                TEAM_BATTING_3B +  # Triples by batters (3B)
                TEAM_BATTING_HR +  # Homeruns by batters (4B)
                TEAM_BATTING_BB +  # Walks by batters
                TEAM_BATTING_SO +  # Strikeouts by batters
                TEAM_BASERUN_SB +  # Stolen bases
                TEAM_BASERUN_CS +  # Caught stealing
                TEAM_PITCHING_H +  # Hits allowed
                TEAM_PITCHING_HR + # Homeruns allowed
                TEAM_PITCHING_BB + # Walks allowed
                TEAM_PITCHING_SO + # Strikeouts by pitchers
                TEAM_FIELDING_E +  # Errors
                TEAM_FIELDING_DP,  # Double Plays
              data=train_model2)
summary(model1)

mean(model2$residuals^2)

layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(model2)
```

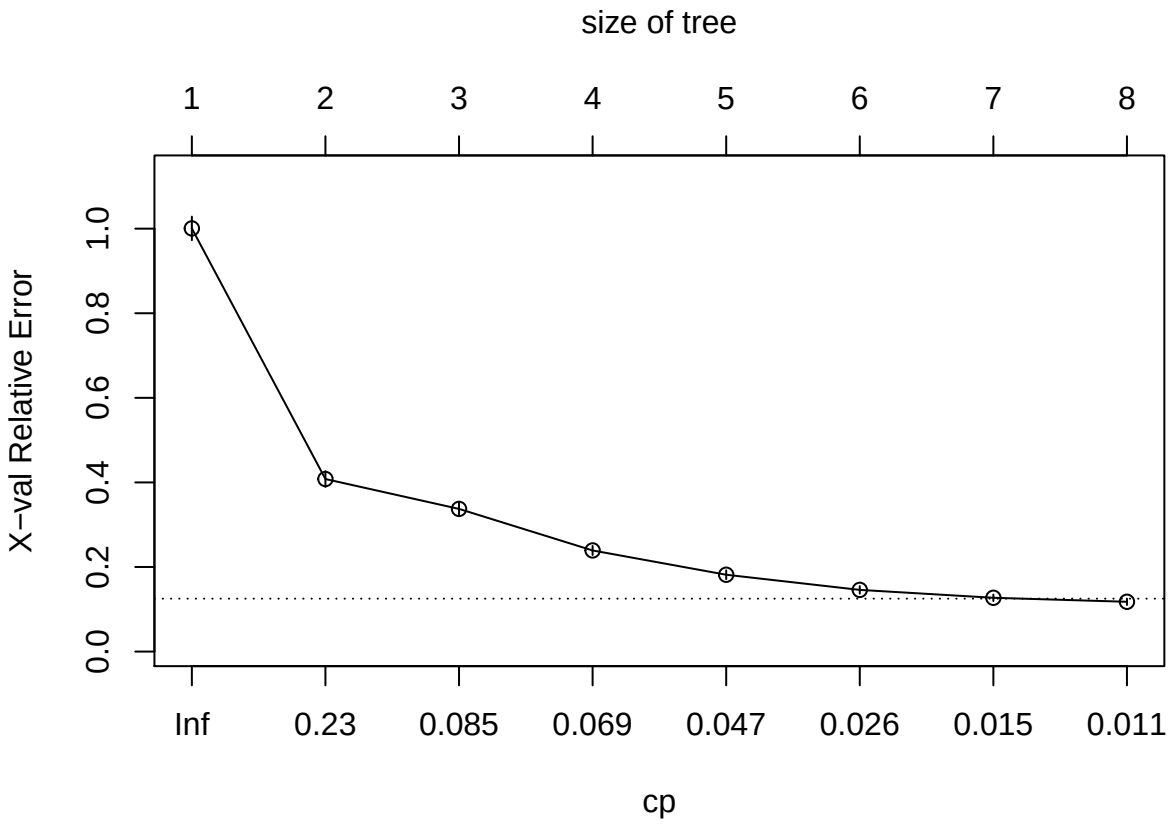



```
### MODEL 3: Decision Tree

train_model3 <-train_data[,-c(1)]
names(train_model3)

batting_so_tree <-rpart(TEAM_BATTING_SO~.,data=train_model3)
```

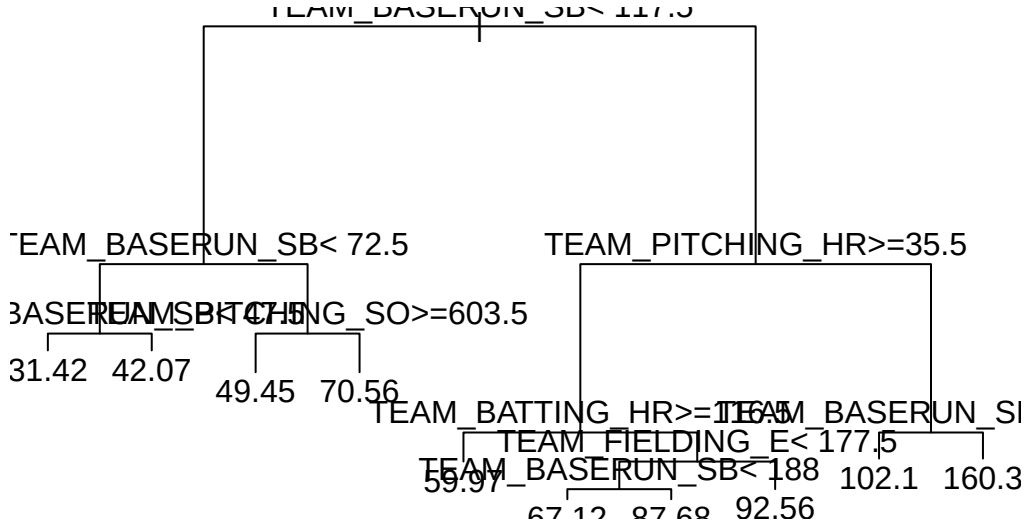
```
plot(batting_so_tree)
text(batting_so_tree)
```



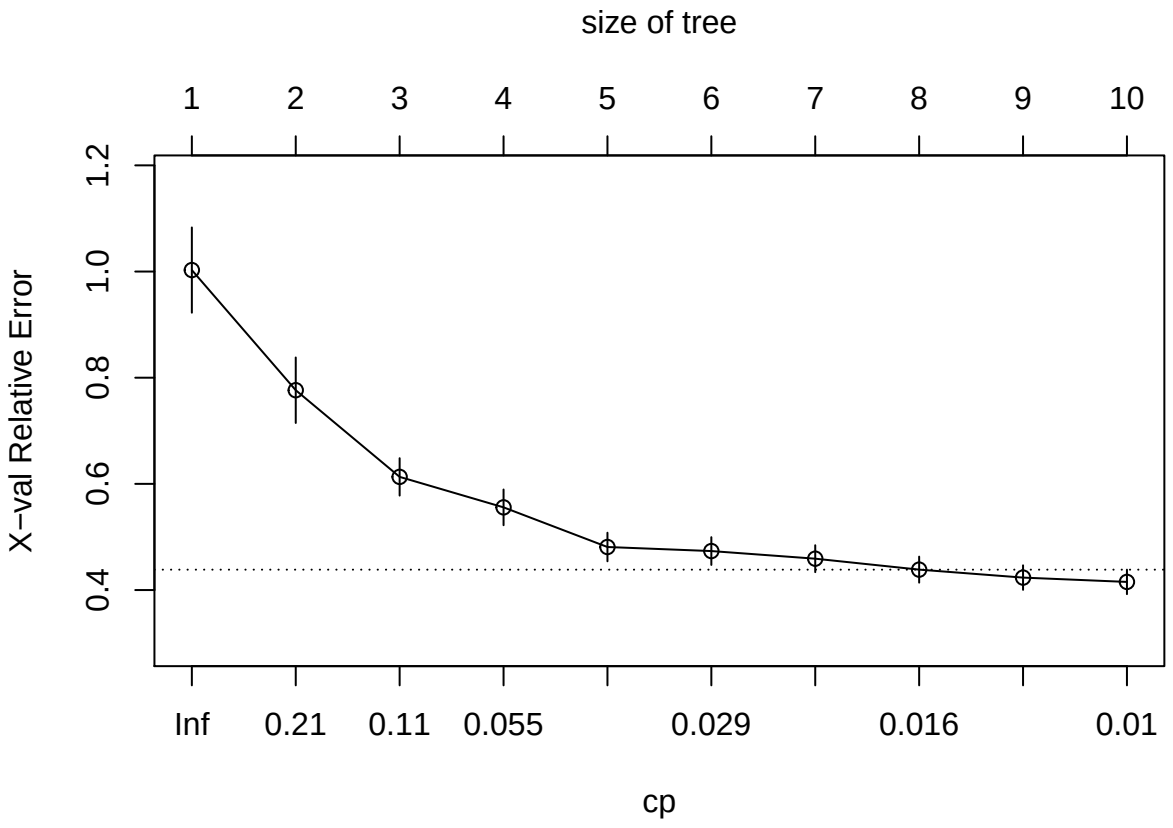

```
printcp(batting_so_tree)
plotcp(batting_so_tree) # keep all 8 branches
```



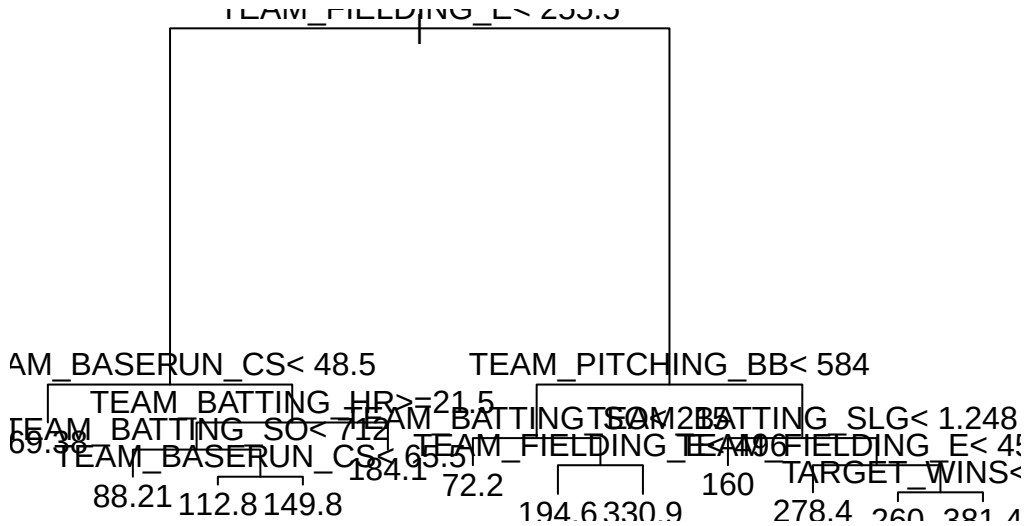

```
baserun_cs <-rpart(TEAM_BASERUN_CS~.,data=train_model3)
plot(baserun_cs)
text(baserun_cs)
```



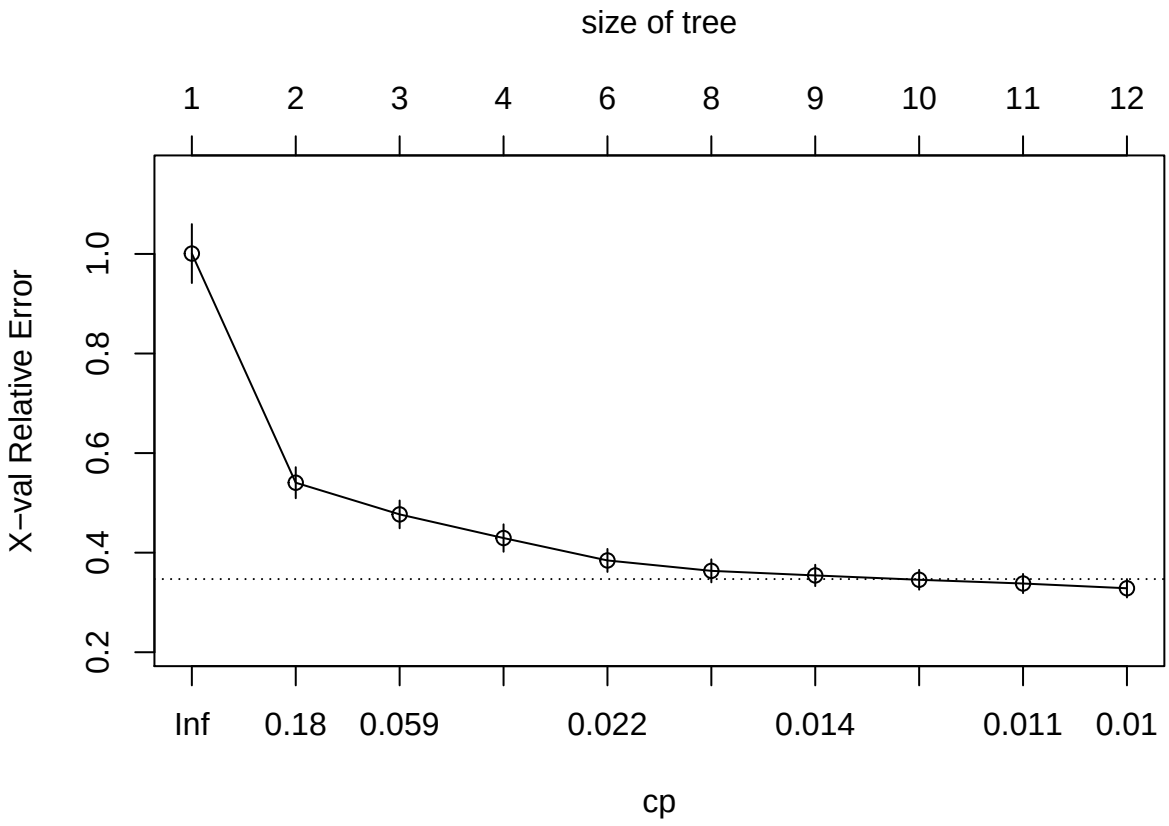

```
printcp(baserun_cs)
plotcp(baserun_cs) # keep all 10 branches
```



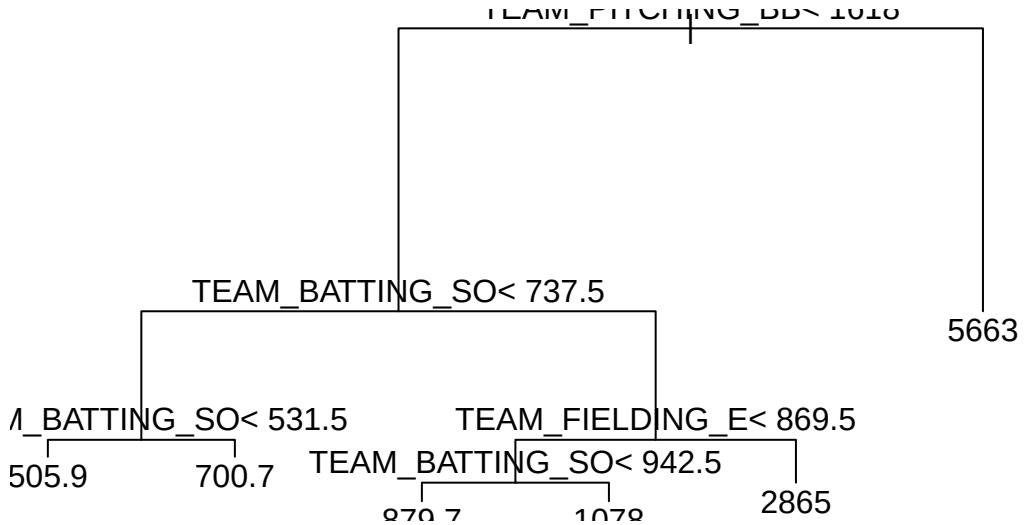

```
baserun_sb <-rpart(TEAM_BASERUN_SB~.,data=train_model3)
plot(baserun_sb)
text(baserun_sb)
```



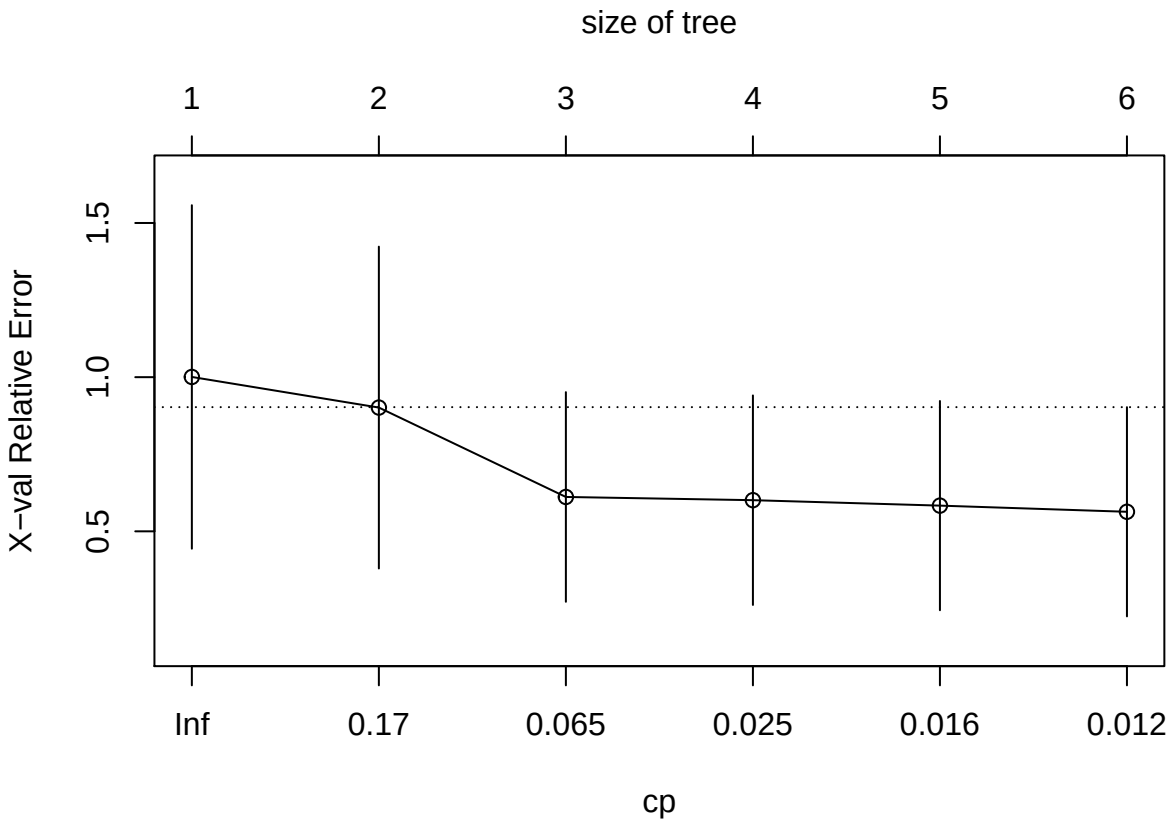

```
printcp(baserun_sb)
plotcp(baserun_sb) # keep all 10 branches (maybe 8)
```



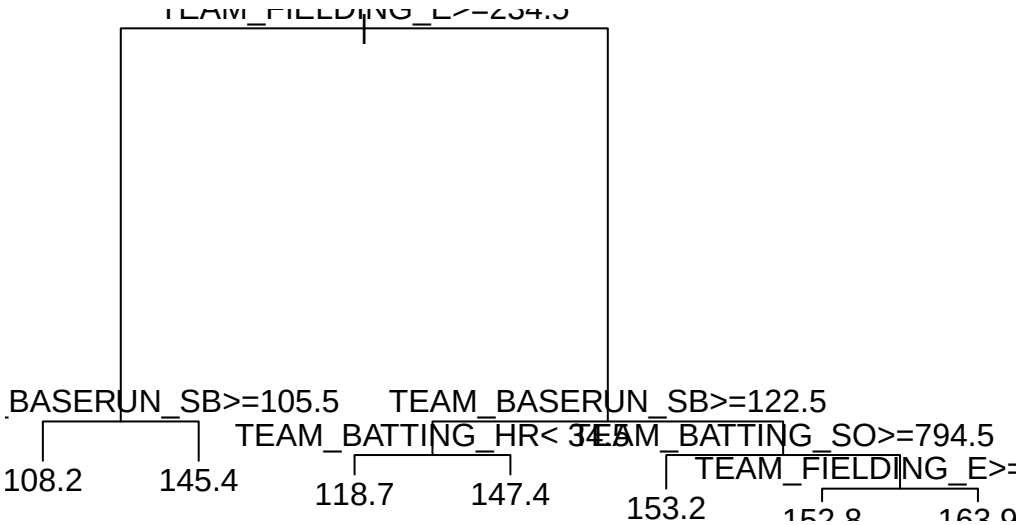

```
pitching_so <-rpart(TEAM_PITCHING_SO~.,data=train_model3)
plot(pitching_so)
text(pitching_so)
```



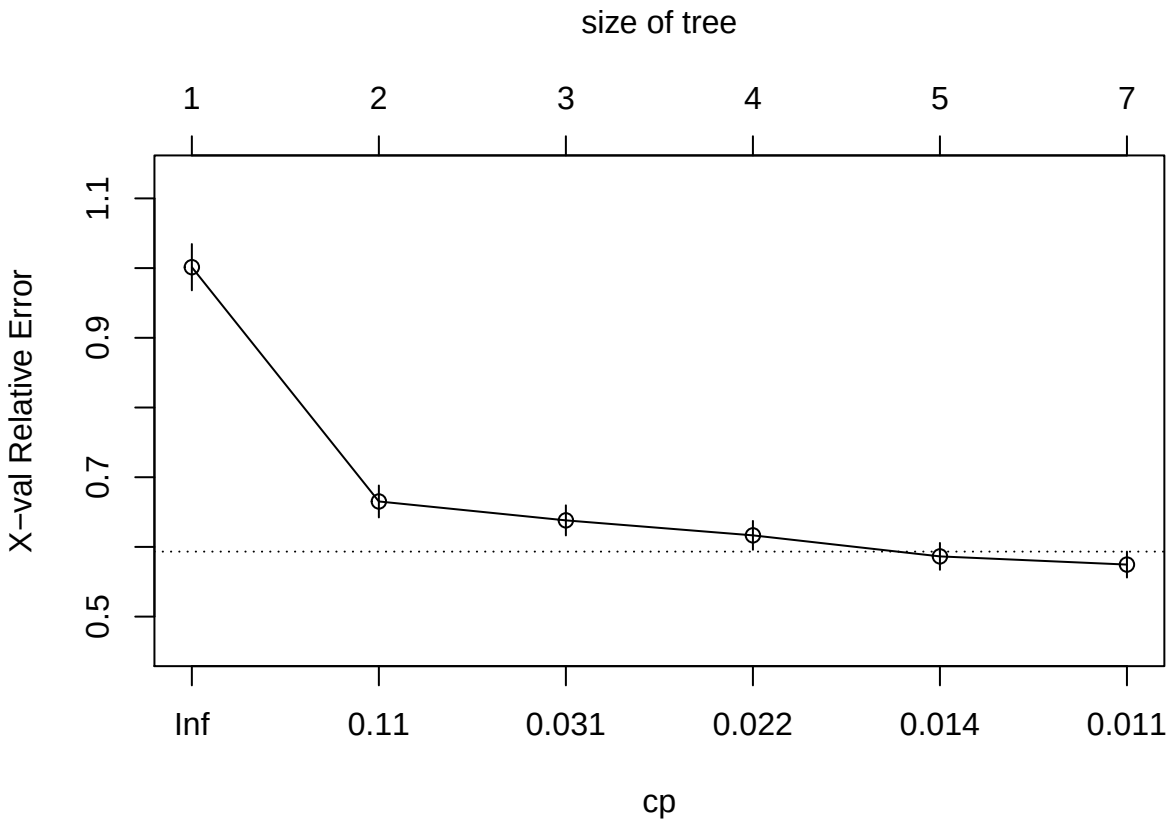

```
printcp(pitching_so)
plotcp(pitching_so) # keep all 6 branches (maybe 5)
```




```
fielding_dp <-rpart(TEAM_FIELDING_DP~.,data=train_model3)
plot(fielding_dp)
text(fielding_dp)
```

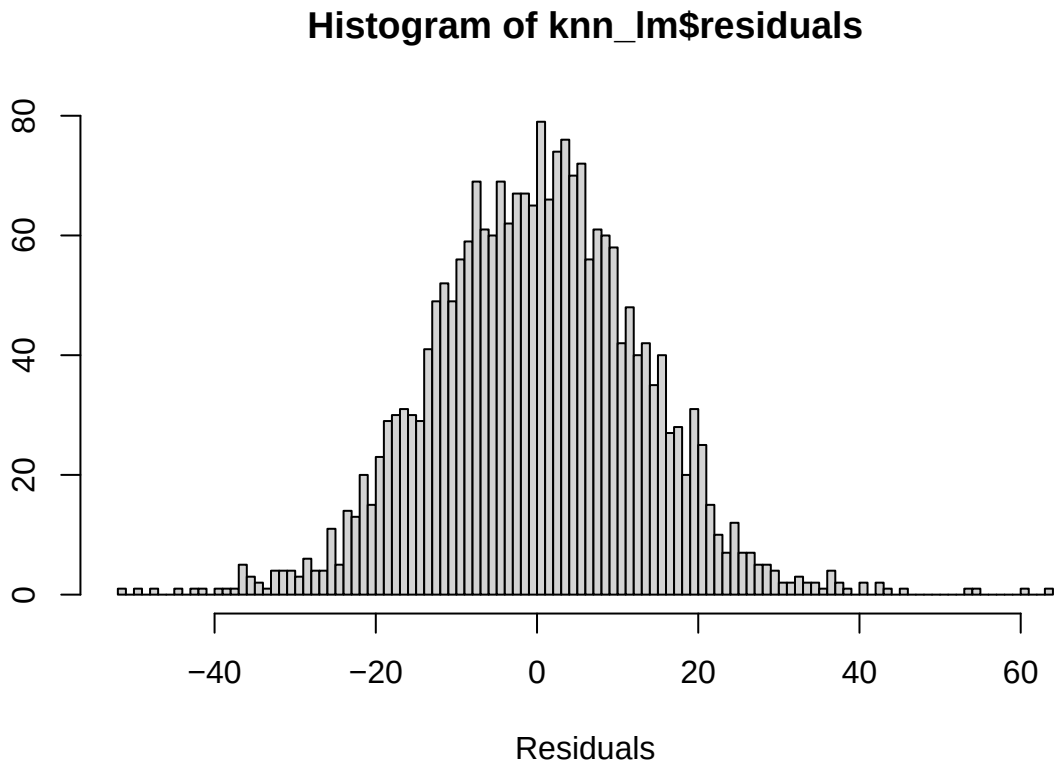



```
printcp(fielding_dp)
plotcp(fielding_dp) # keep all 6 branches
```

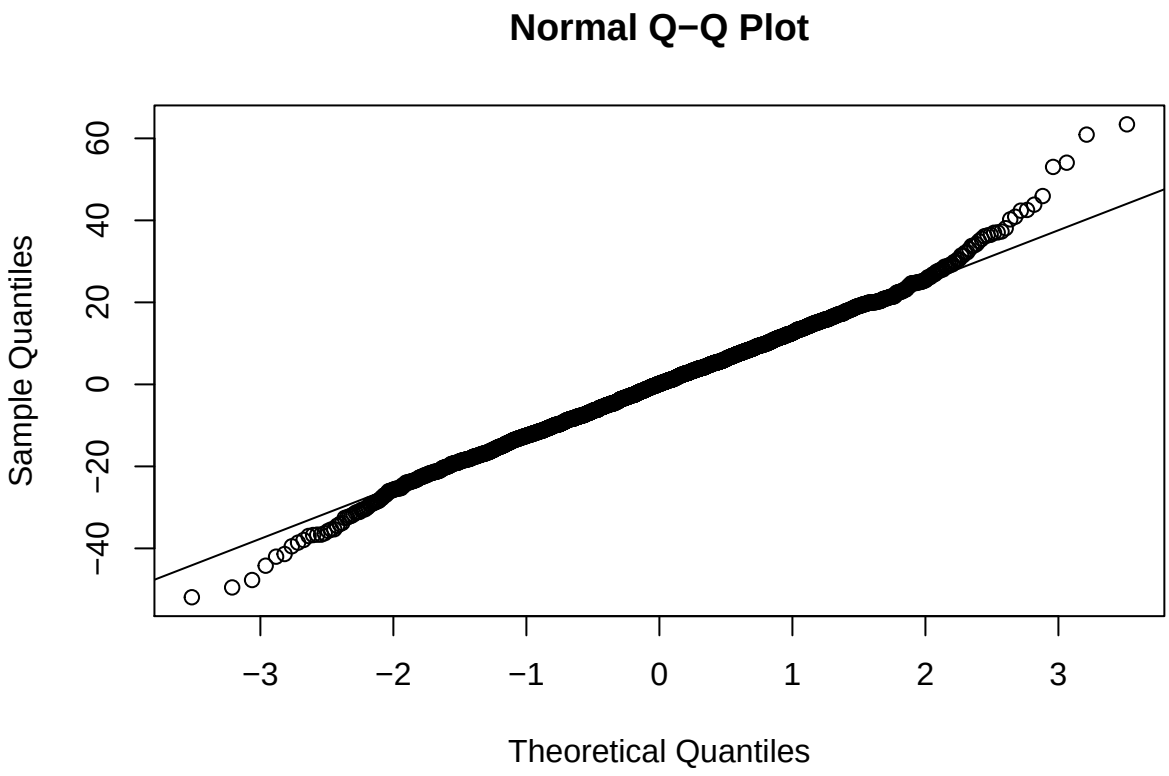



```
### MODEL 4: kNN Imputation

train_data_impute <- select(train_data, -c(INDEX))


#### using default values

knn_data <- knnImputation(train_data_impute)

summary(knn_data)

train.control <- trainControl(method ='cv', number=10)

step.model <- train(TARGET_WINS ~., data = knn_data,
                    method = "leapSeq",
                    tuneGrid = data.frame(nvmax = 2:13),
                    trControl = train.control
                    )

step.model$results

step.model$bestTune

summary(step.model$finalModel)

coef(step.model$finalModel, 6)
```

```
knn_lm <- lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_PITCHING_

summary(knn_lm)


hist(knn_lm$residuals, xlab = "Residuals", ylab = "", breaks=100)
```




```
qqnorm(knn_lm$residuals)
qqline(knn_lm$residuals)
```




```
plot(fitted(knn_lm), residuals(knn_lm))
abline(h=0, lty = 2)
```