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 equal causation it is suggested that a focus on some of the variables such as a focus on either 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.

.....To add more here....

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.

```
## Rows: 2,276
## Columns: 17
## $ INDEX            <int> 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 15, 16, 17, 18...
## $ TARGET_WINS      <int> 39, 70, 86, 70, 82, 75, 80, 85, 86, 76, 78, 68, 72...
## $ TEAM_BATTING_H   <int> 1445, 1339, 1377, 1387, 1297, 1279, 1244, 1273, 13...
## $ 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, 27, 31, 41...
## $ TEAM_BATTING_HR  <int> 13, 190, 137, 96, 102, 92, 122, 115, 114, 96, 82, ...
## $ 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, 82...
## $ TEAM_BASERUN_SB  <int> NA, 37, 46, 43, 49, 107, 80, 40, 69, 72, 60, 119, ...
## $ TEAM_BASERUN_CS  <int> NA, 28, 27, 30, 39, 59, 54, 36, 27, 34, 39, 79, 10...
## $ TEAM_BATTING_HBP <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ TEAM_PITCHING_H  <int> 9364, 1347, 1377, 1396, 1297, 1279, 1244, 1281, 13...
## $ TEAM_PITCHING_HR <int> 84, 191, 137, 97, 102, 92, 122, 116, 114, 96, 86, ...
## $ 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, 8...
## $ 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, 1...
```

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

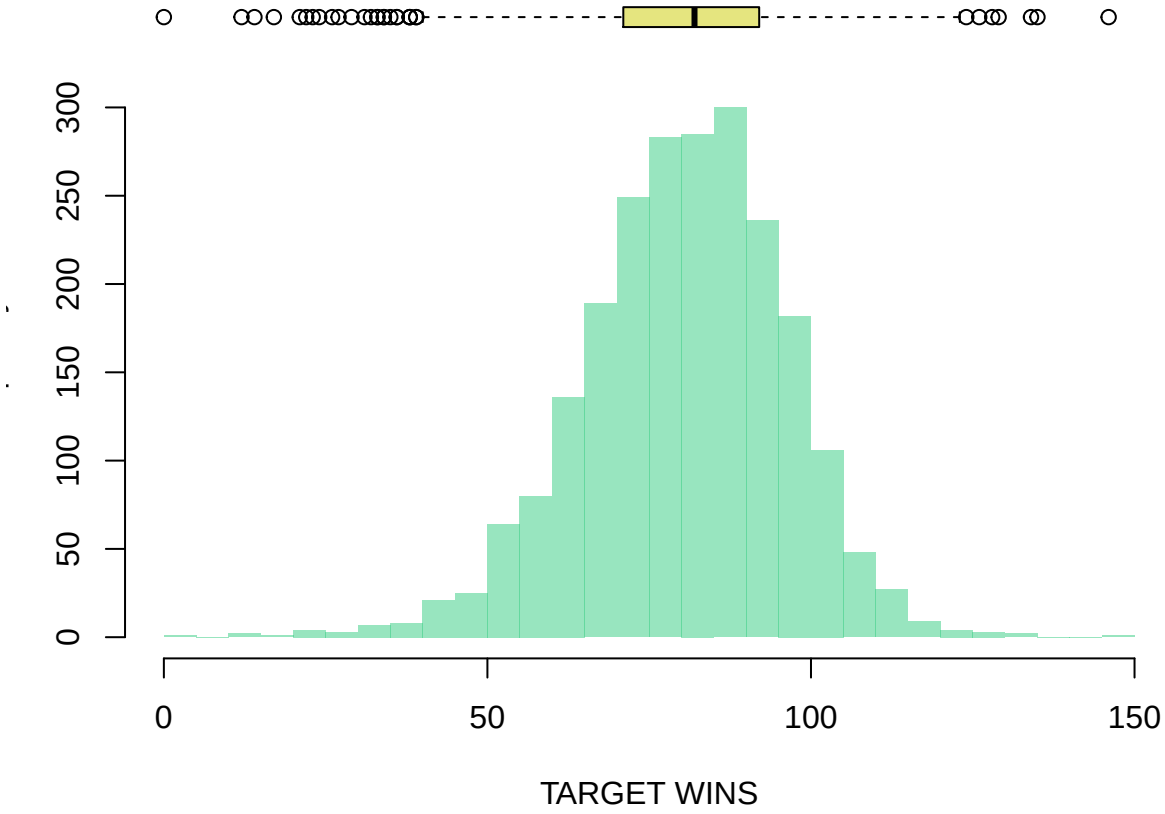
Mean (SD) Median Minimum Maximum

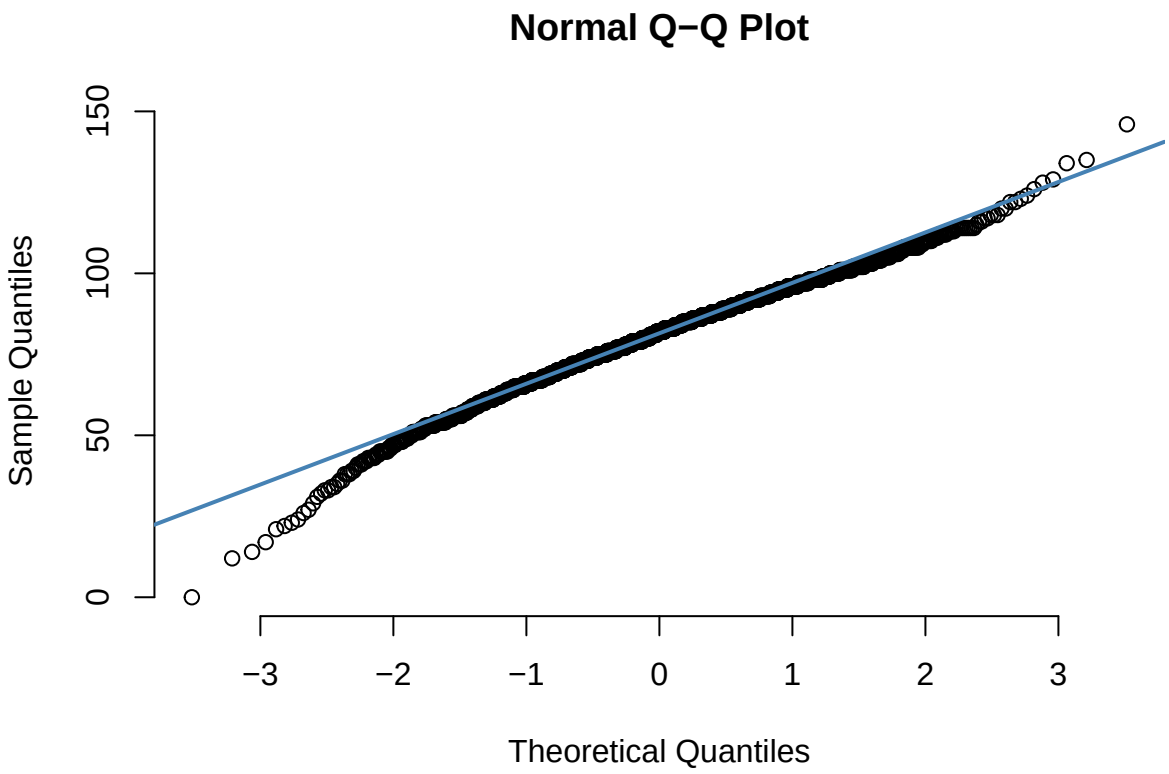
By examining the target wins variable in detail, there is a clear guideline of how many wins each team should approximately win. Most teams will likely win the average number of games (81), but there will be some variability from the average with some teams winning more or less than 81 games.

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





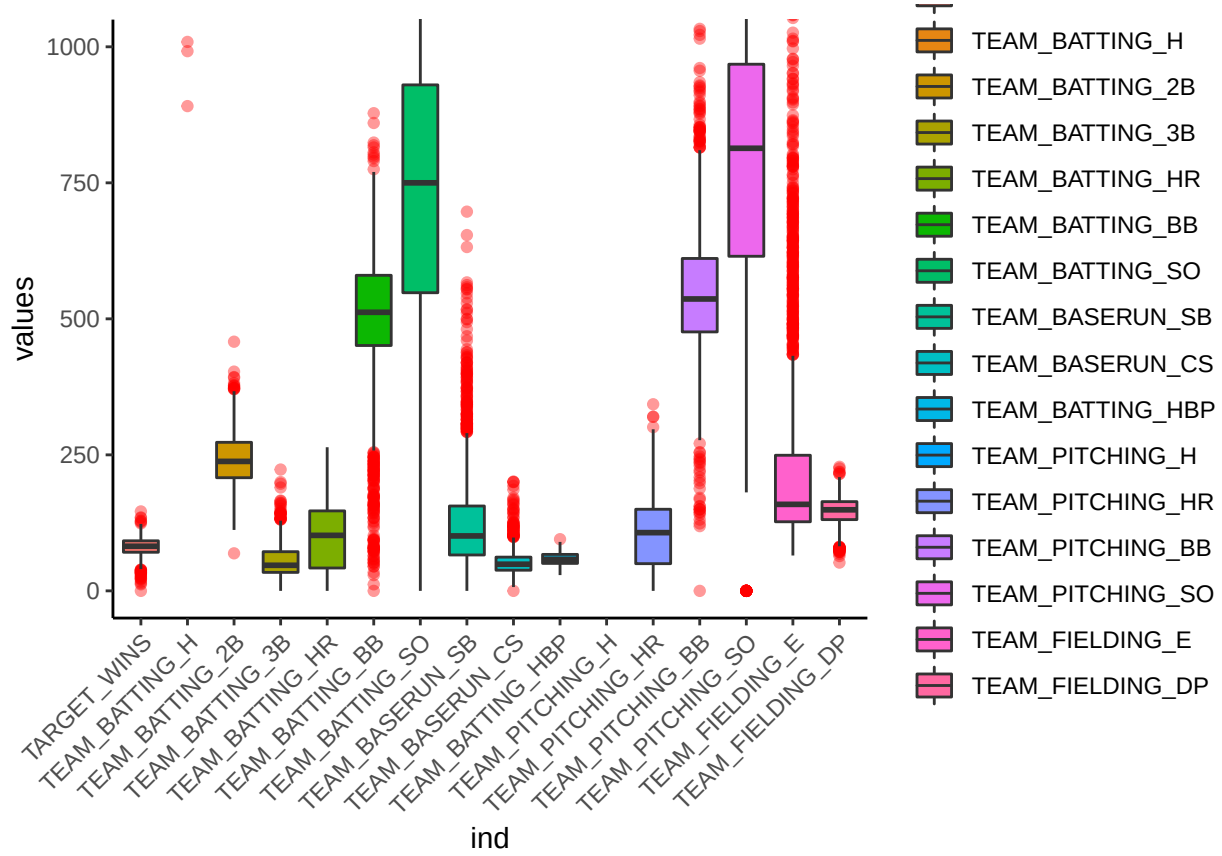
Now in order to build our models properly It's worth exploring for other columns with NA values.

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

```
#train_data[-c(1)] %>%
#  gather() %>%
#  ggplot(aes(value)) +
#    facet_wrap(~ key, scales = "free") +
#    geom_histogram(binwidth=1)
```

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.

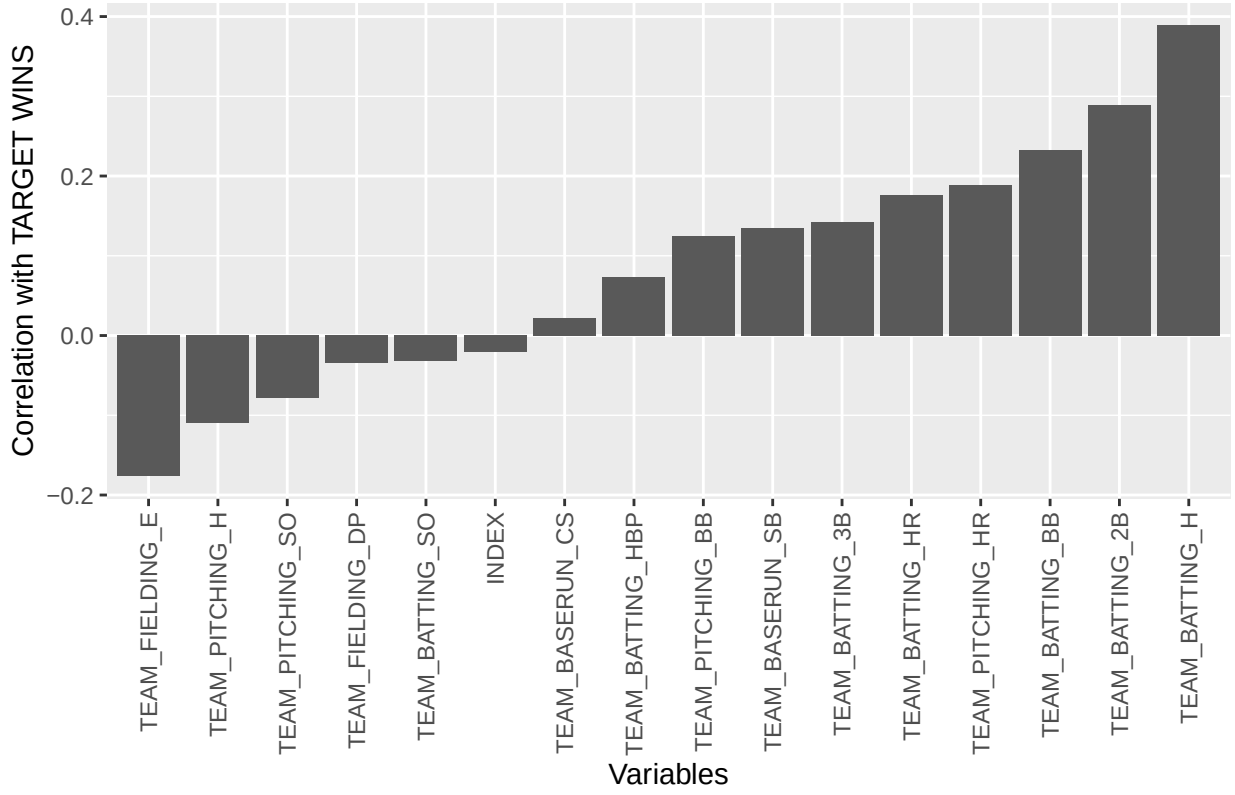
Correlations among predictors and Variable Selection

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

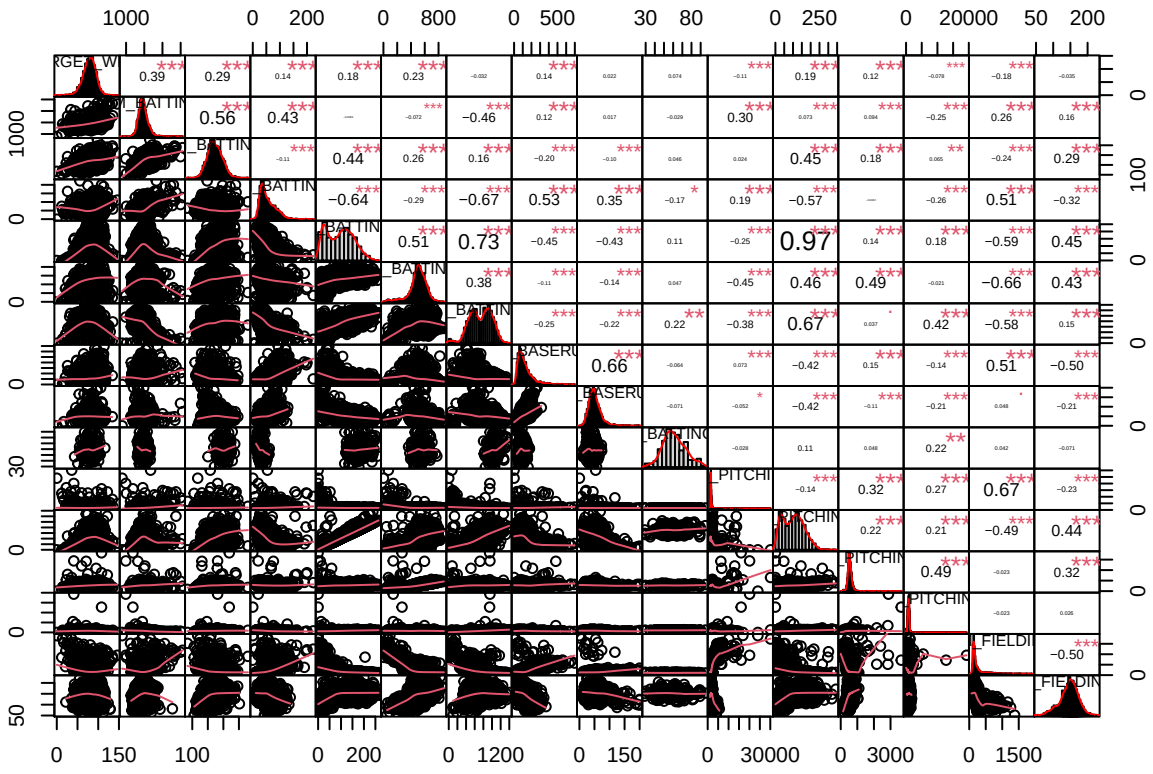
Figure 4: Correlation Against Target Win



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.

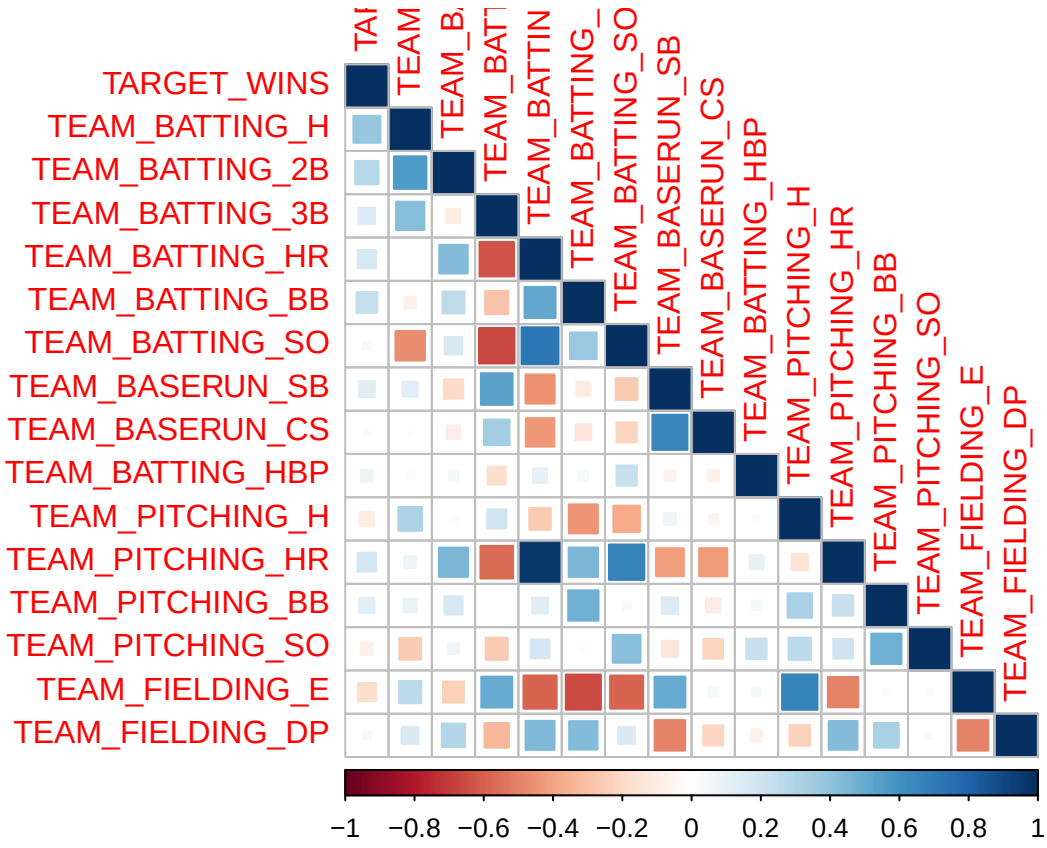
Comment : We need to decide if we can keep below graph

```
#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")
```




```
#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)
```

```
##                 row           column         cor            p
## 1       TARGET_WINS   TEAM_BATTING_H  0.388767521 0.000000e+00
## 2       TARGET_WINS  TEAM_BATTING_2B  0.289103645 0.000000e+00
## 3    TEAM_BATTING_H  TEAM_BATTING_2B  0.562849678 0.000000e+00
## 4       TARGET_WINS  TEAM_BATTING_3B  0.142608411 8.217427e-12
## 5    TEAM_BATTING_H  TEAM_BATTING_3B  0.427696575 0.000000e+00
## 6   TEAM_BATTING_2B  TEAM_BATTING_3B -0.107305824 2.877545e-07
## 7       TARGET_WINS  TEAM_BATTING_HR  0.176153200 0.000000e+00
## 8    TEAM_BATTING_H  TEAM_BATTING_HR -0.006544685 7.549934e-01
## 9   TEAM_BATTING_2B  TEAM_BATTING_HR  0.435397293 0.000000e+00
## 10  TEAM_BATTING_3B  TEAM_BATTING_HR -0.635566946 0.000000e+00
## 11      TARGET_WINS  TEAM_BATTING_BB  0.232559864 0.000000e+00
## 12   TEAM_BATTING_H  TEAM_BATTING_BB -0.072464013 5.407324e-04
## 13  TEAM_BATTING_2B  TEAM_BATTING_BB  0.255726103 0.000000e+00
## 14  TEAM_BATTING_3B  TEAM_BATTING_BB -0.287235841 0.000000e+00
## 15  TEAM_BATTING_HR  TEAM_BATTING_BB  0.513734810 0.000000e+00
## 16      TARGET_WINS  TEAM_BATTING_SO -0.031750708 1.388904e-01
## 17   TEAM_BATTING_H  TEAM_BATTING_SO -0.463853571 0.000000e+00
## 18  TEAM_BATTING_2B  TEAM_BATTING_SO  0.162685188 2.309264e-14
```

```
## 19   TEAM_BATTING_3B  TEAM_BATTING_SO -0.669781188 0.000000e+00
## 20   TEAM_BATTING_HR  TEAM_BATTING_SO  0.727069348 0.000000e+00
## 21   TEAM_BATTING_BB  TEAM_BATTING_SO  0.379750866 0.000000e+00
## 22       TARGET_WINS  TEAM_BASERUN_SB  0.135138921 3.298830e-10
## 23    TEAM_BATTING_H  TEAM_BASERUN_SB  0.123567797 9.377653e-09
## 24   TEAM_BATTING_2B  TEAM_BASERUN_SB -0.199757239 0.000000e+00
## 25   TEAM_BATTING_3B  TEAM_BASERUN_SB  0.533506448 0.000000e+00
## 26   TEAM_BATTING_HR  TEAM_BASERUN_SB -0.453578426 0.000000e+00
## 27   TEAM_BATTING_BB  TEAM_BASERUN_SB -0.105115643 1.066116e-06
## 28   TEAM_BATTING_SO  TEAM_BASERUN_SB -0.254489232 0.000000e+00
## 29       TARGET_WINS  TEAM_BASERUN_CS  0.022404069 3.852582e-01
## 30    TEAM_BATTING_H  TEAM_BASERUN_CS  0.016705668 5.173884e-01
## 31   TEAM_BATTING_2B  TEAM_BASERUN_CS -0.099814059 1.055784e-04
## 32   TEAM_BATTING_3B  TEAM_BASERUN_CS  0.348764919 0.000000e+00
## 33   TEAM_BATTING_HR  TEAM_BASERUN_CS -0.433793868 0.000000e+00
## 34   TEAM_BATTING_BB  TEAM_BASERUN_CS -0.136988371 9.641725e-08
## 35   TEAM_BATTING_SO  TEAM_BASERUN_CS -0.217881368 0.000000e+00
## 36   TEAM_BASERUN_SB  TEAM_BASERUN_CS  0.655244804 0.000000e+00
## 37       TARGET_WINS TEAM_BATTING_HBP  0.073504242 3.122327e-01
## 38    TEAM_BATTING_H TEAM_BATTING_HBP -0.029112176 6.893171e-01
## 39   TEAM_BATTING_2B TEAM_BATTING_HBP  0.046084753 5.266947e-01
## 40   TEAM_BATTING_3B TEAM_BATTING_HBP -0.174247154 1.591723e-02
## 41   TEAM_BATTING_HR TEAM_BATTING_HBP  0.106181160 1.437532e-01
## 42   TEAM_BATTING_BB TEAM_BATTING_HBP  0.047460067 5.144185e-01
## 43   TEAM_BATTING_SO TEAM_BATTING_HBP  0.220942194 2.129956e-03
## 44   TEAM_BASERUN_SB TEAM_BATTING_HBP -0.064004982 3.790423e-01
## 45   TEAM_BASERUN_CS TEAM_BATTING_HBP -0.070513896 3.323798e-01
## 46       TARGET_WINS  TEAM_PITCHING_H -0.109937054 1.457270e-07
## 47    TEAM_BATTING_H  TEAM_PITCHING_H  0.302693709 0.000000e+00
## 48   TEAM_BATTING_2B  TEAM_PITCHING_H  0.023692188 2.585473e-01
## 49   TEAM_BATTING_3B  TEAM_PITCHING_H  0.194879411 0.000000e+00
## 50   TEAM_BATTING_HR  TEAM_PITCHING_H -0.250145481 0.000000e+00
## 51   TEAM_BATTING_BB  TEAM_PITCHING_H -0.449777625 0.000000e+00
## 52   TEAM_BATTING_SO  TEAM_PITCHING_H -0.375686369 0.000000e+00
## 53   TEAM_BASERUN_SB  TEAM_PITCHING_H  0.073285050 6.819772e-04
## 54   TEAM_BASERUN_CS  TEAM_PITCHING_H -0.052007809 4.373461e-02
## 55  TEAM_BATTING_HBP  TEAM_PITCHING_H -0.027696995 7.036928e-01
## 56       TARGET_WINS TEAM_PITCHING_HR  0.189013735 0.000000e+00
## 57    TEAM_BATTING_H TEAM_PITCHING_HR  0.072853119 5.045119e-04
## 58   TEAM_BATTING_2B TEAM_PITCHING_HR  0.454550818 0.000000e+00
## 59   TEAM_BATTING_3B TEAM_PITCHING_HR -0.567836679 0.000000e+00
## 60   TEAM_BATTING_HR TEAM_PITCHING_HR  0.969371396 0.000000e+00
## 61   TEAM_BATTING_BB TEAM_PITCHING_HR  0.459552072 0.000000e+00
## 62   TEAM_BATTING_SO TEAM_PITCHING_HR  0.667178892 0.000000e+00
## 63   TEAM_BASERUN_SB TEAM_PITCHING_HR -0.416510723 0.000000e+00
## 64   TEAM_BASERUN_CS TEAM_PITCHING_HR -0.422566046 0.000000e+00
## 65  TEAM_BATTING_HBP TEAM_PITCHING_HR  0.106758780 1.415740e-01
## 66   TEAM_PITCHING_H TEAM_PITCHING_HR -0.141612759 1.148881e-11
## 67       TARGET_WINS TEAM_PITCHING_BB  0.124174536 2.784686e-09
## 68    TEAM_BATTING_H TEAM_PITCHING_BB  0.094193027 6.755492e-06
## 69   TEAM_BATTING_2B TEAM_PITCHING_BB  0.178054204 0.000000e+00
## 70   TEAM_BATTING_3B TEAM_PITCHING_BB -0.002224148 9.155425e-01
## 71   TEAM_BATTING_HR TEAM_PITCHING_BB  0.136927564 5.388223e-11
## 72   TEAM_BATTING_BB TEAM_PITCHING_BB  0.489361263 0.000000e+00
```

```
## 73   TEAM_BATTING_SO TEAM_PITCHING_BB  0.037005141 8.452629e-02
## 74   TEAM_BASERUN_SB TEAM_PITCHING_BB  0.146415134 9.499512e-12
## 75   TEAM_BASERUN_CS TEAM_PITCHING_BB -0.106961236 3.230317e-05
## 76  TEAM_BATTING_HBP TEAM_PITCHING_BB  0.047851371 5.109529e-01
## 77   TEAM_PITCHING_H TEAM_PITCHING_BB  0.320676162 0.000000e+00
## 78  TEAM_PITCHING_HR TEAM_PITCHING_BB  0.221937505 0.000000e+00
## 79       TARGET_WINS TEAM_PITCHING_SO -0.078436090 2.515153e-04
## 80    TEAM_BATTING_H TEAM_PITCHING_SO -0.252656790 0.000000e+00
## 81   TEAM_BATTING_2B TEAM_PITCHING_SO  0.064792315 2.507323e-03
## 82   TEAM_BATTING_3B TEAM_PITCHING_SO -0.258818931 0.000000e+00
## 83   TEAM_BATTING_HR TEAM_PITCHING_SO  0.184707564 0.000000e+00
## 84   TEAM_BATTING_BB TEAM_PITCHING_SO -0.020756822 3.333647e-01
## 85   TEAM_BATTING_SO TEAM_PITCHING_SO  0.416233300 0.000000e+00
## 86   TEAM_BASERUN_SB TEAM_PITCHING_SO -0.137128609 4.853151e-10
## 87   TEAM_BASERUN_CS TEAM_PITCHING_SO -0.210222735 2.220446e-16
## 88  TEAM_BATTING_HBP TEAM_PITCHING_SO  0.221573754 2.066596e-03
## 89   TEAM_PITCHING_H TEAM_PITCHING_SO  0.267248074 0.000000e+00
## 90  TEAM_PITCHING_HR TEAM_PITCHING_SO  0.205880529 0.000000e+00
## 91  TEAM_PITCHING_BB TEAM_PITCHING_SO  0.488498653 0.000000e+00
## 92       TARGET_WINS  TEAM_FIELDING_E -0.176484759 0.000000e+00
## 93    TEAM_BATTING_H  TEAM_FIELDING_E  0.264902478 0.000000e+00
## 94   TEAM_BATTING_2B  TEAM_FIELDING_E -0.235150986 0.000000e+00
## 95   TEAM_BATTING_3B  TEAM_FIELDING_E  0.509778447 0.000000e+00
## 96   TEAM_BATTING_HR  TEAM_FIELDING_E -0.587339098 0.000000e+00
## 97   TEAM_BATTING_BB  TEAM_FIELDING_E -0.655970815 0.000000e+00
## 98   TEAM_BATTING_SO  TEAM_FIELDING_E -0.584664436 0.000000e+00
## 99   TEAM_BASERUN_SB  TEAM_FIELDING_E  0.509630902 0.000000e+00
## 100  TEAM_BASERUN_CS  TEAM_FIELDING_E  0.048321894 6.099538e-02
## 101 TEAM_BATTING_HBP  TEAM_FIELDING_E  0.041789712 5.659644e-01
## 102  TEAM_PITCHING_H  TEAM_FIELDING_E  0.667759010 0.000000e+00
## 103 TEAM_PITCHING_HR  TEAM_FIELDING_E -0.493144466 0.000000e+00
## 104 TEAM_PITCHING_BB  TEAM_FIELDING_E -0.022837561 2.761252e-01
## 105 TEAM_PITCHING_SO  TEAM_FIELDING_E -0.023291783 2.776873e-01
## 106      TARGET_WINS TEAM_FIELDING_DP -0.034850584 1.201464e-01
## 107   TEAM_BATTING_H TEAM_FIELDING_DP  0.155383321 3.179013e-12
## 108  TEAM_BATTING_2B TEAM_FIELDING_DP  0.290879978 0.000000e+00
## 109  TEAM_BATTING_3B TEAM_FIELDING_DP -0.323074847 0.000000e+00
## 110  TEAM_BATTING_HR TEAM_FIELDING_DP  0.448985348 0.000000e+00
## 111  TEAM_BATTING_BB TEAM_FIELDING_DP  0.430876747 0.000000e+00
## 112  TEAM_BATTING_SO TEAM_FIELDING_DP  0.154889392 1.319034e-11
## 113  TEAM_BASERUN_SB TEAM_FIELDING_DP -0.497077627 0.000000e+00
## 114  TEAM_BASERUN_CS TEAM_FIELDING_DP -0.214248008 0.000000e+00
## 115 TEAM_BATTING_HBP TEAM_FIELDING_DP -0.071208241 3.276290e-01
## 116  TEAM_PITCHING_H TEAM_FIELDING_DP -0.228650592 0.000000e+00
## 117 TEAM_PITCHING_HR TEAM_FIELDING_DP  0.439170397 0.000000e+00
## 118 TEAM_PITCHING_BB TEAM_FIELDING_DP  0.324457226 0.000000e+00
## 119 TEAM_PITCHING_SO TEAM_FIELDING_DP  0.026158043 2.559407e-01
## 120  TEAM_FIELDING_E TEAM_FIELDING_DP -0.497684954 0.000000e+00
```

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.

Comment : We need to decide how to handle the missing values: Remove it or input the mean values

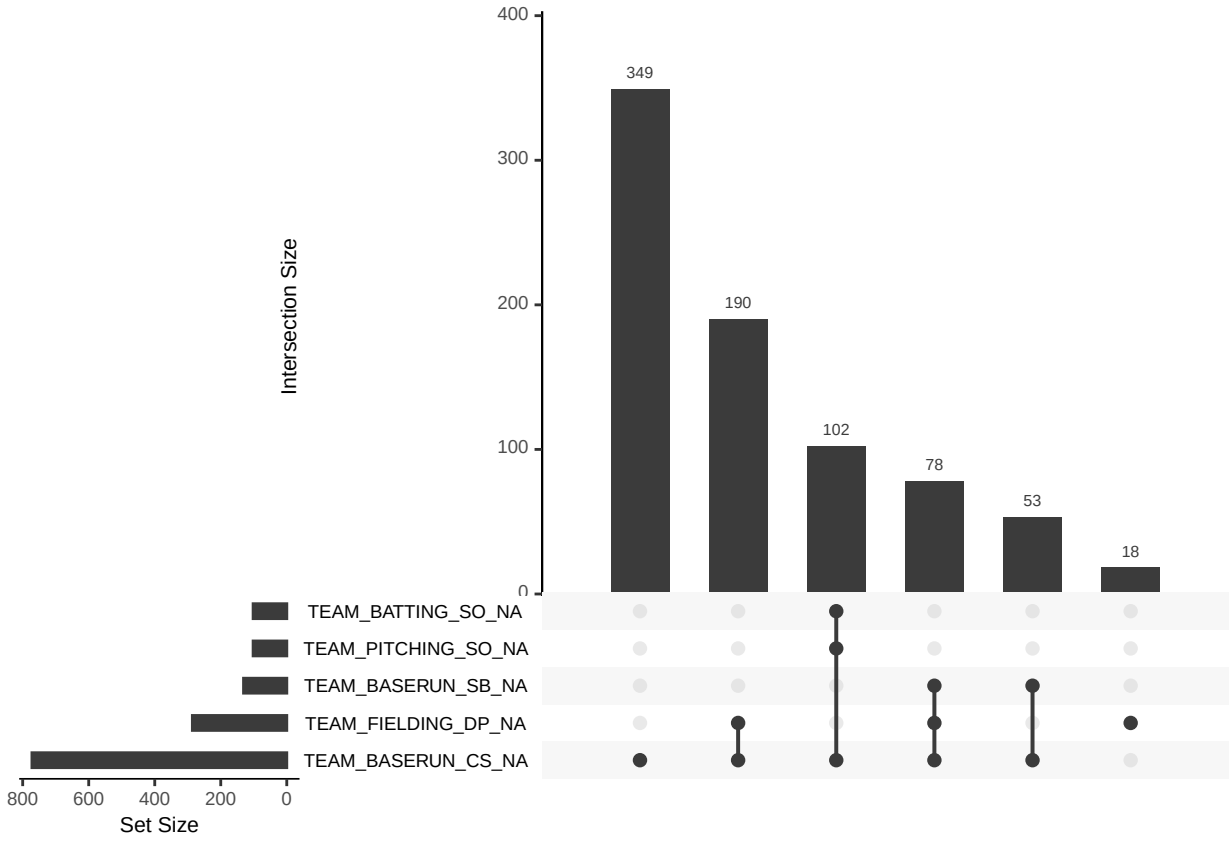
---

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

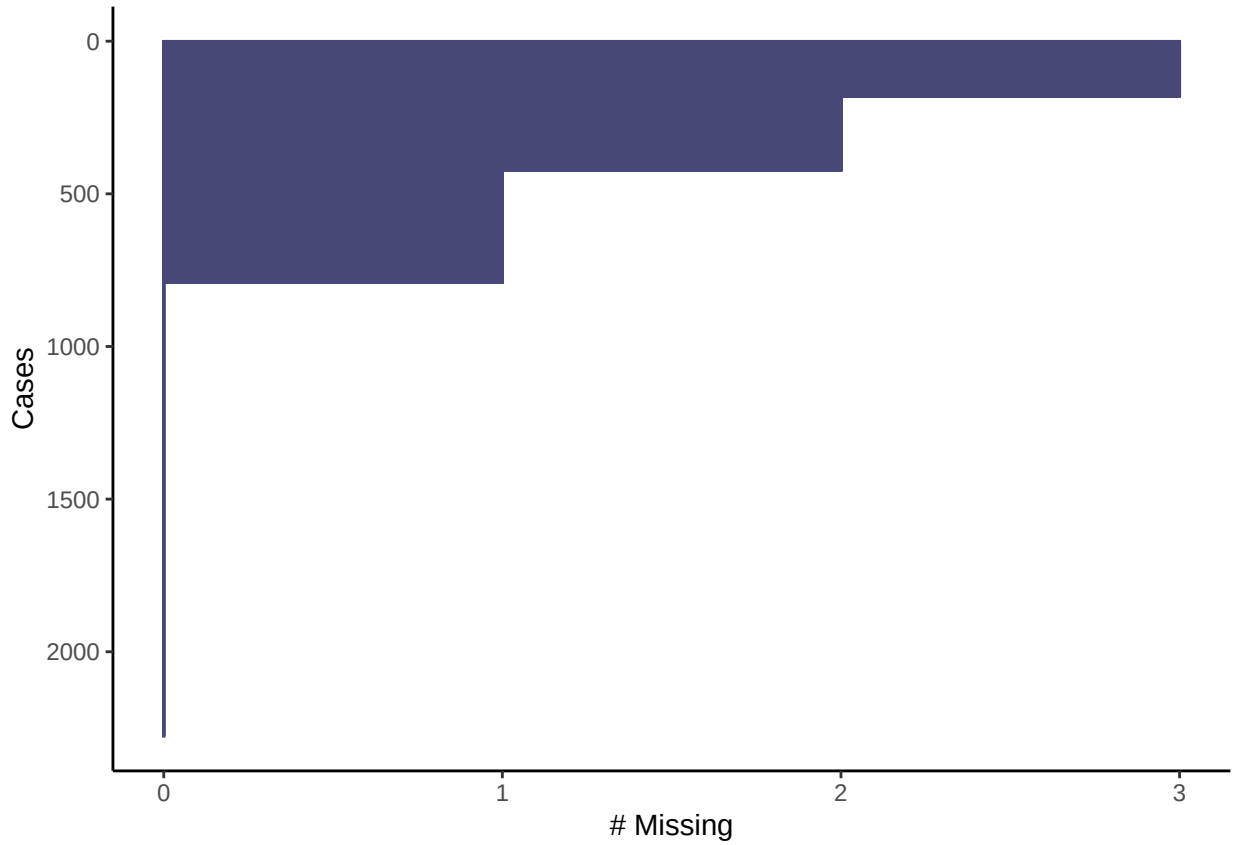
```
train_data <- train_data[-11]
```

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)

```
# Here is an example how to use the package https://naniar.njtierney.com/articles/naniar-visualisation.
library(naniar)
par(mfrow=c(1,2))
gg_miss_upset(train_data,
              nsets = 5,
              nintersects = NA)
```




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



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).

Build Models

MODEL 1: MEAN FULL MODEL

This is a full model containing all the variables with the meanused 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)

```
model1 <- lm(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_data)
summary(model1)
```

```
##
## 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_data)
##
## 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
```

Mean Square Error (MSE)

```
mean(model1$residuals^2)
```

```
## [1] 119.3236
```

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 0.3147 and MSE is 170.03,which will be the current benchmark. Here is the equation for predicting wins using Model 1.

WINS =+ 25.06831
+0.04824*Base Hits by batters
-0.02004*Doubles by batters
+0.06040*Triples by batters
+0.05298*Homeruns by batters
+0.01041*Walks by batters
-0.00935*Strikeouts by batters
+0.02946*Stolen bases
-0.01173*Caught stealing
-0.00073149*Hits allowed
+0.01481*Homeruns allowed
+0.00008066*Walks allowed
+0.00284*Strikeouts by pitchers
-0.02118*Errors
-0.12121*Double plays

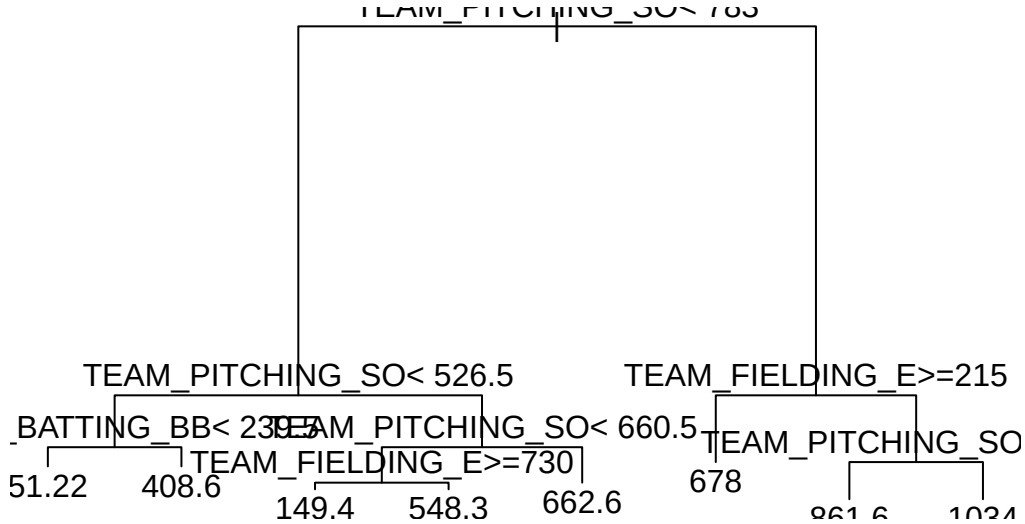
MODEL 2: Decision Tree

```
library(rpart)
train_data1 <-train_data[,-c(1,2,11)]
names(train_data1)
```

```
##  [1] "TEAM_BATTING_H"   "TEAM_BATTING_2B"  "TEAM_BATTING_3B"  "TEAM_BATTING_HR"
##  [5] "TEAM_BATTING_BB"  "TEAM_BATTING_SO"  "TEAM_BASERUN_SB"  "TEAM_BASERUN_CS"
##  [9] "TEAM_PITCHING_HR" "TEAM_PITCHING_BB" "TEAM_PITCHING_SO" "TEAM_FIELDING_E"
## [13] "TEAM_FIELDING_DP"
```

create a decision tree for each variable with missing values. cross validate to see if we can prune the tree to make simpler# TEAM_BATTING_SO, TEAM_BASERUN_CS, TEAM_BASERUN_SB, TEAM_PITCHING_SO, TEAM_FIELDING_DP

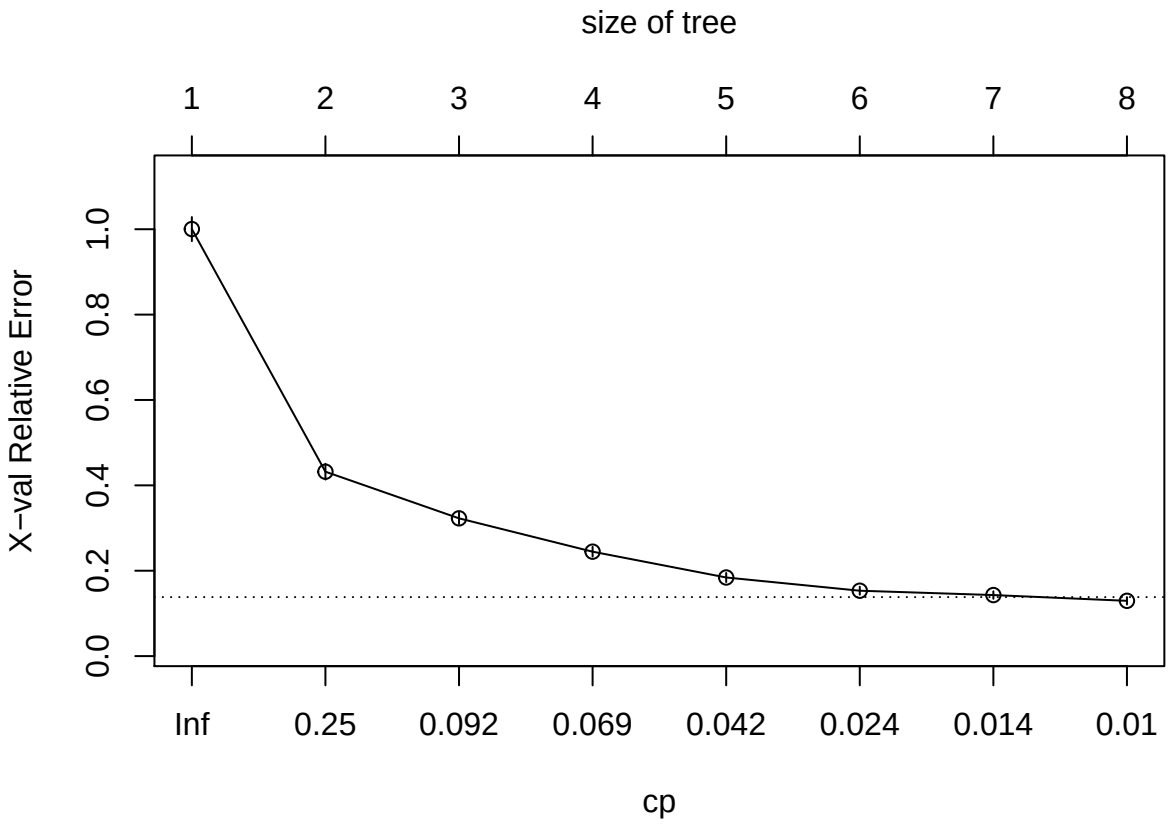
```
batting_so_tree <-rpart(TEAM_BATTING_SO~.,data=train_data1)
plot(batting_so_tree)
text(batting_so_tree)
```



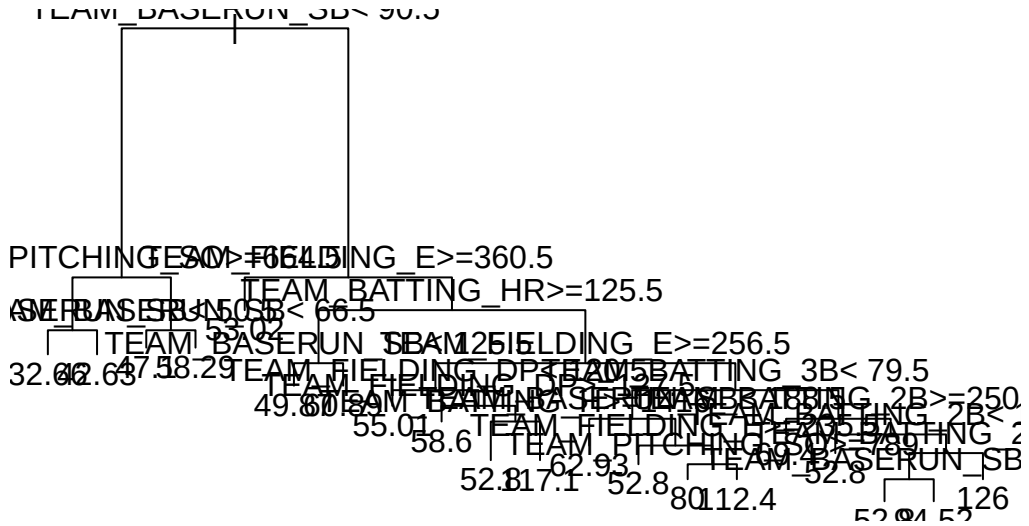

```
printcp(batting_so_tree)
```

```
##
## Regression tree:
## rpart(formula = TEAM_BATTING_SO ~ ., data = train_data1)
##
## Variables actually used in tree construction:
## [1] TEAM_BATTING_BB  TEAM_FIELDING_E  TEAM_PITCHING_SO
##
## Root node error: 134216171/2276 = 58970
##
## n= 2276
##
##         CP nsplit rel error  xerror      xstd
## 1 0.576763      0   1.00000 1.00035 0.0281871
## 2 0.104339      1   0.42324 0.43198 0.0185313
## 3 0.081054      2   0.31890 0.32268 0.0135313
## 4 0.058086      3   0.23784 0.24462 0.0114142
## 5 0.030039      4   0.17976 0.18412 0.0106853
## 6 0.019771      5   0.14972 0.15318 0.0094294
## 7 0.010425      6   0.12995 0.14300 0.0087633
## 8 0.010000      7   0.11952 0.12950 0.0087559
```

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




```
baserun_cs <-rpart(TEAM_BASERUN_CS~.,data=train_data1)
plot(baserun_cs)
text(baserun_cs)
```

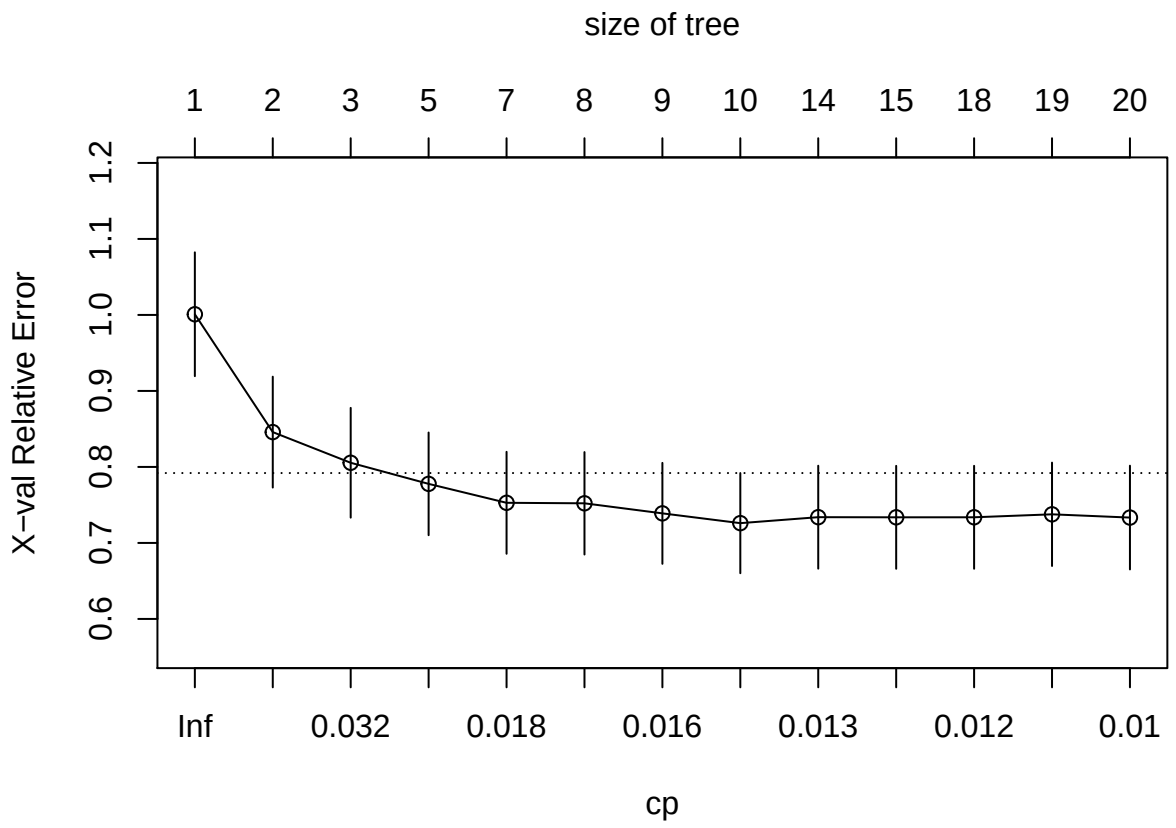



```
printcp(baserun_cs)
```

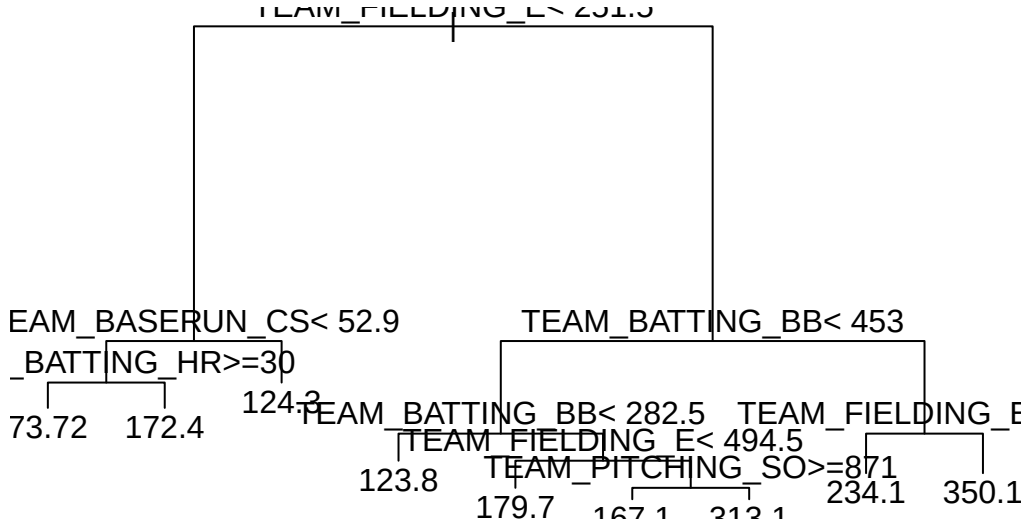
```
##
## Regression tree:
## rpart(formula = TEAM_BASERUN_CS ~ ., data = train_data1)
##
## Variables actually used in tree construction:
## [1] TEAM_BASERUN_SB  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_H
## [5] TEAM_BATTING_HR  TEAM_FIELDING_DP TEAM_FIELDING_E  TEAM_PITCHING_SO
##
## Root node error: 792071/2276 = 348.01
##
## n= 2276
##
##          CP nsplit rel error  xerror     xstd
## 1  0.166309      0   1.00000 1.00089 0.081497
## 2  0.035305      1   0.83369 0.84581 0.072973
## 3  0.028652      2   0.79839 0.80547 0.072237
## 4  0.018552      4   0.74108 0.77772 0.067568
## 5  0.017865      6   0.70398 0.75279 0.067096
## 6  0.016088      7   0.68611 0.75214 0.067413
## 7  0.015025      8   0.67002 0.73890 0.066415
## 8  0.012827      9   0.65500 0.72604 0.065906
## 9  0.012500     13   0.60183 0.73387 0.067739
## 10 0.012382     14   0.58933 0.73371 0.067737
```

```
## 11 0.011589     17   0.55218 0.73374 0.067745
## 12 0.010008     18   0.54059 0.73765 0.068009
## 13 0.010000     19   0.53058 0.73337 0.068276
```

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



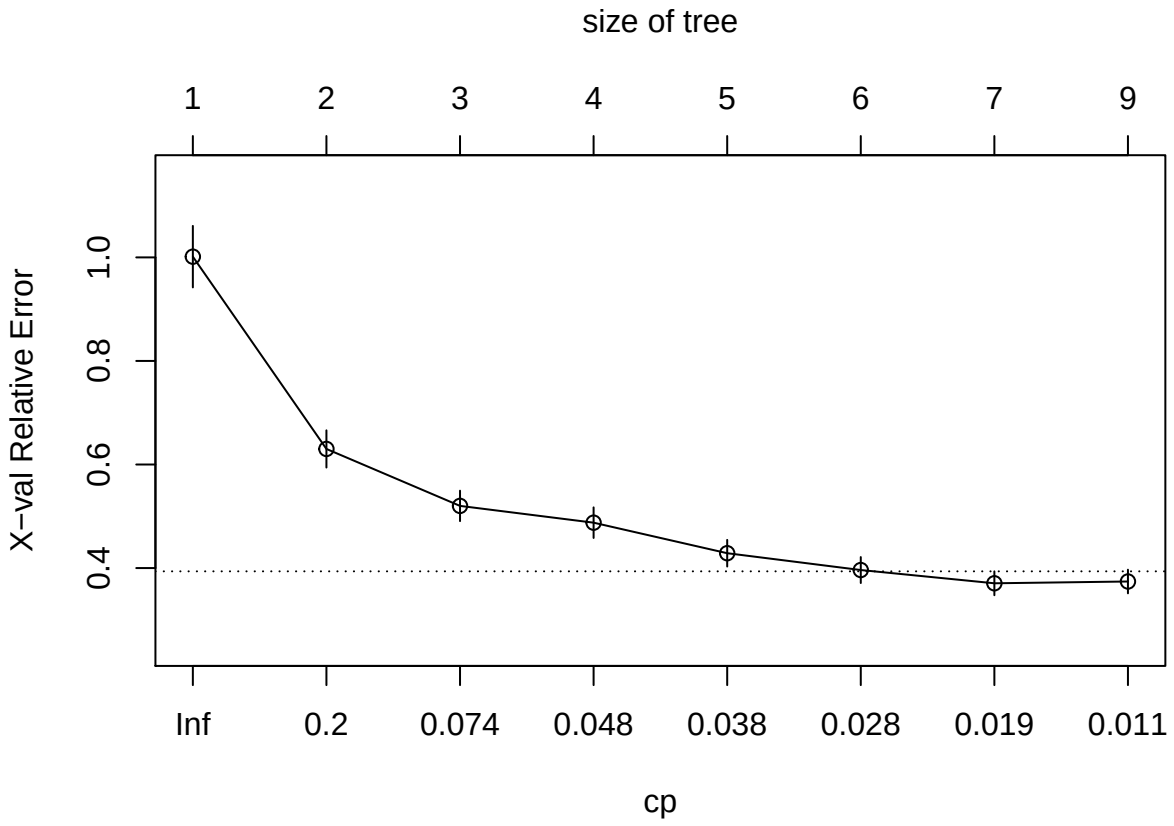

```
baserun_sb <-rpart(TEAM_BASERUN_SB~.,data=train_data1)
plot(baserun_sb)
text(baserun_sb)
```



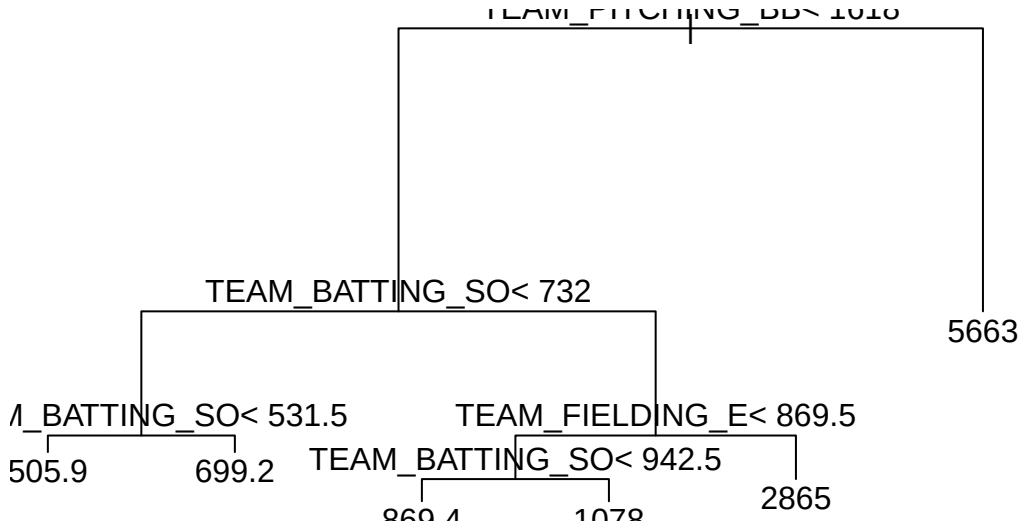

```
printcp(baserun_sb)
```

```
##
## Regression tree:
## rpart(formula = TEAM_BASERUN_SB ~ ., data = train_data1)
##
## Variables actually used in tree construction:
## [1] TEAM_BASERUN_CS  TEAM_BATTING_BB  TEAM_BATTING_HR  TEAM_FIELDING_E
## [5] TEAM_PITCHING_SO
##
## Root node error: 16524427/2276 = 7260.3
##
## n= 2276
##
##         CP nsplit rel error  xerror     xstd
## 1 0.375771      0   1.00000 1.00143 0.059419
## 2 0.110574      1   0.62423 0.63004 0.035949
## 3 0.049601      2   0.51365 0.52005 0.029400
## 4 0.047225      3   0.46405 0.48766 0.029539
## 5 0.030236      4   0.41683 0.42867 0.025746
## 6 0.026695      5   0.38659 0.39617 0.025108
## 7 0.013160      6   0.35990 0.37058 0.023074
## 8 0.010000      8   0.33358 0.37401 0.022849
```

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



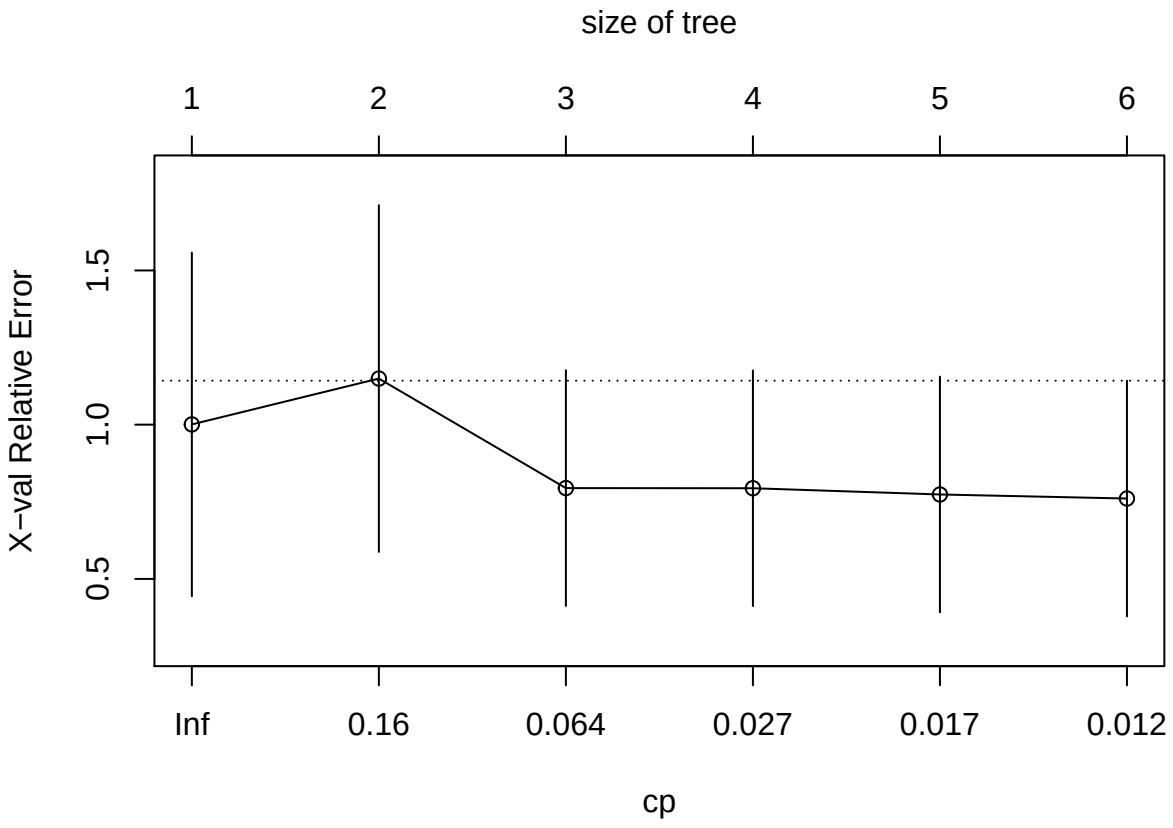

```
pitching_so <-rpart(TEAM_PITCHING_SO~.,data=train_data1)
plot(pitching_so)
text(pitching_so)
```



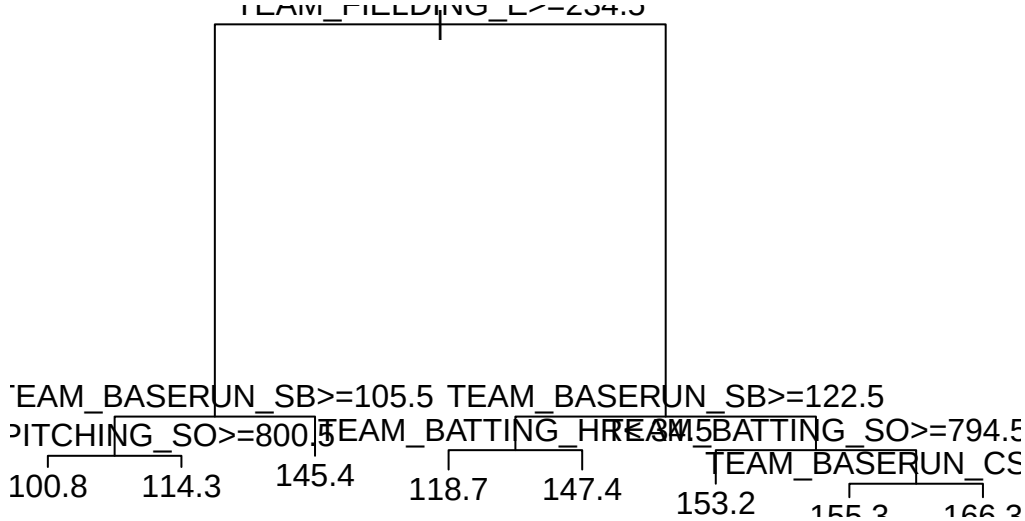

```
printcp(pitching_so)
```

```
##
## Regression tree:
## rpart(formula = TEAM_PITCHING_SO ~ ., data = train_data1)
##
## Variables actually used in tree construction:
## [1] TEAM_BATTING_SO  TEAM_FIELDING_E  TEAM_PITCHING_BB
##
## Root node error: 664727332/2276 = 292059
##
## n= 2276
##
##         CP nsplit rel error  xerror    xstd
## 1 0.248030      0   1.00000 1.00081 0.55716
## 2 0.108811      1   0.75197 1.14954 0.56202
## 3 0.038180      2   0.64316 0.79434 0.38206
## 4 0.019420      3   0.60498 0.79394 0.38206
## 5 0.014501      4   0.58556 0.77366 0.38212
## 6 0.010000      5   0.57106 0.76054 0.38213
```

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



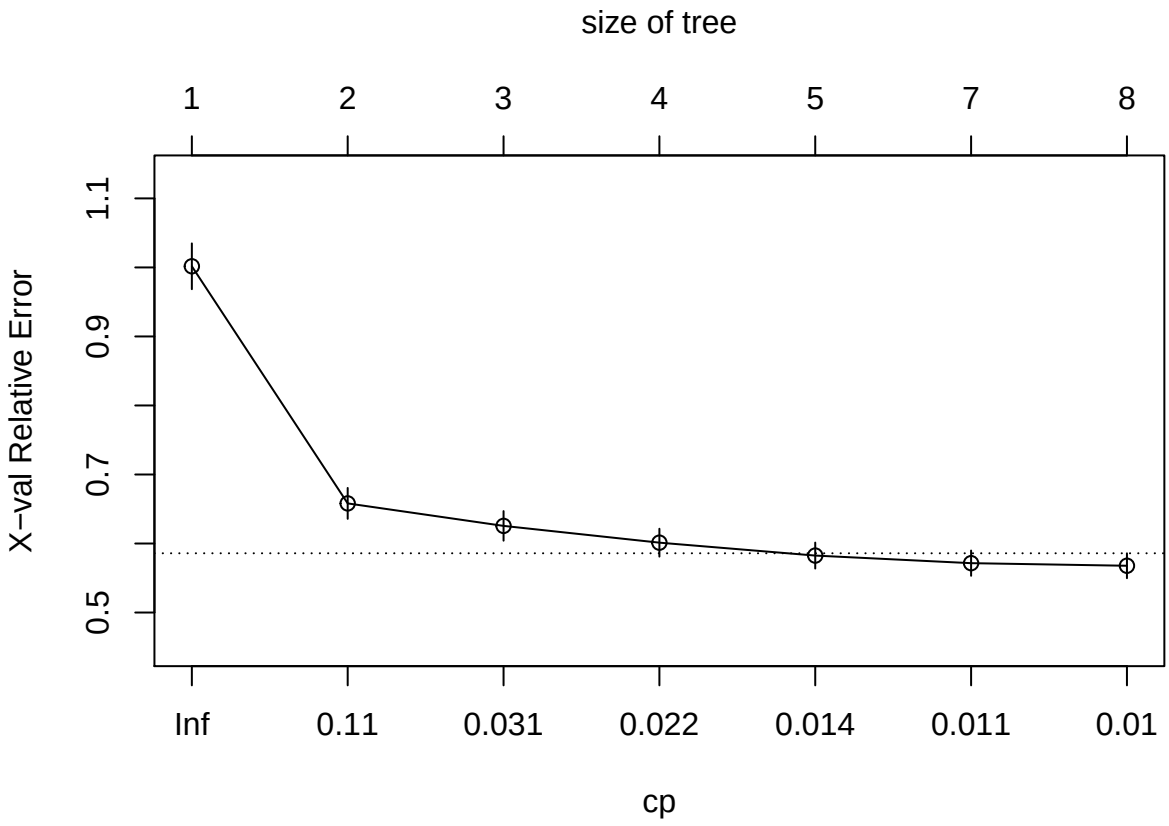

```
fielding_dp <-rpart(TEAM_FIELDING_DP~.,data=train_data1)
plot(fielding_dp)
text(fielding_dp)
```




```
printcp(fielding_dp)
```

```
##
## Regression tree:
## rpart(formula = TEAM_FIELDING_DP ~ ., data = train_data1)
##
## Variables actually used in tree construction:
## [1] TEAM_BASERUN_CS  TEAM_BASERUN_SB  TEAM_BATTING_HR  TEAM_BATTING_SO
## [5] TEAM_FIELDING_E  TEAM_PITCHING_SO
##
## Root node error: 1368081/1990 = 687.48
##
## n=1990 (286 observations deleted due to missingness)
##
##         CP nsplit rel error  xerror     xstd
## 1 0.358546      0   1.00000 1.00165 0.033189
## 2 0.035856      1   0.64145 0.65811 0.022397
## 3 0.027639      2   0.60560 0.62553 0.021428
## 4 0.017750      3   0.57796 0.60121 0.020177
## 5 0.011486      4   0.56021 0.58254 0.018807
## 6 0.010242      6   0.53724 0.57152 0.018302
## 7 0.010000      7   0.52699 0.56778 0.018061
```

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



MODEL 3

Log Transformation

MODEL 4

BoxCox Transformation

MODEL 5

kNN Imputation

Select A Model

Appendix

References