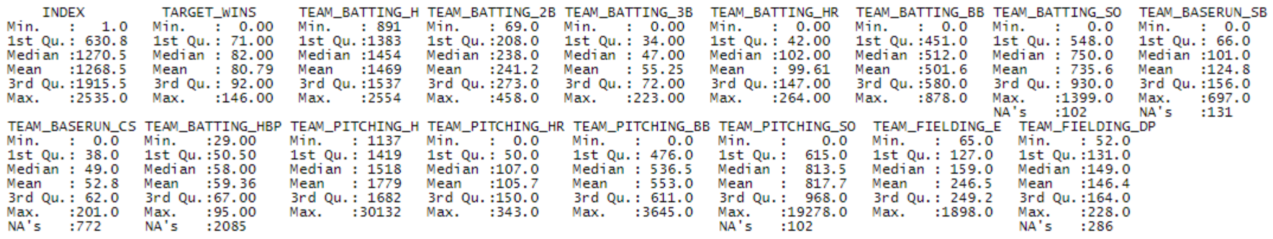
Data Set Overview:

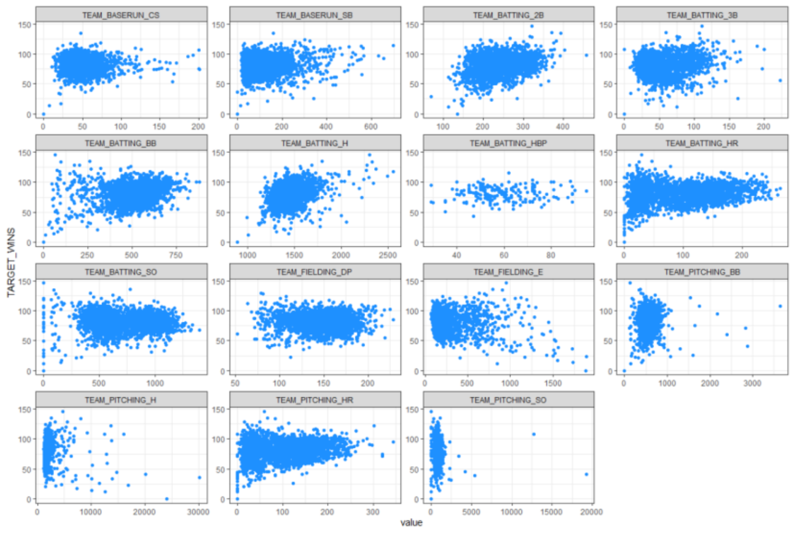
The "Money Ball" data set contains 2,276 observations of 17 variables. Each observation appears to be a collection of data for a baseball team's full season. None of observations are identified as any specific baseball team or any specific year. 15 of the variables capture season totals for various batting, base running, pitching and fielding totals for that unidentified team's season. The remaining two variables are Index, an integer identifier for each observation and "Target_Wins", which represents the number of wins that given observation produced.

Table 1 – Summary Statistics for Money Ball Data Set



```
     INDEX          TARGET_WINS      TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB
 Min.   :   1.0   Min.   :  0.00   Min.   : 891   Min.   : 69.0   Min.   :  0.00   Min.   :  0.00   Min.   :  0.0   Min.   :   0.0   Min.   :  0.0  
 1st Qu.: 630.8   1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0   1st Qu.: 34.00   1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 548.0   1st Qu.: 66.0  
 Median :1270.5   Median : 82.00   Median :1454   Median :238.0   Median : 47.00   Median :102.00   Median :512.0   Median : 750.0   Median :101.0  
 Mean   :1268.5   Mean   : 80.79   Mean   :1469   Mean   :241.2   Mean   : 55.25   Mean   : 99.61   Mean   :501.6   Mean   : 735.6   Mean   :124.8  
 3rd Qu.:1915.5   3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0   3rd Qu.: 72.00   3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 930.0   3rd Qu.:156.0  
 Max.   :2535.0   Max.   :146.00   Max.   :2554   Max.   :458.0   Max.   :223.00   Max.   :264.00   Max.   :878.0   Max.   :1399.0   Max.   :697.0  
                                                                                                                    NA's   :102      NA's   :131    
 TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E  TEAM_FIELDING_DP
 Min.   :  0.0   Min.   :29.00    Min.   : 1137   Min.   :  0.0    Min.   :   0.0   Min.   :    0.0   Min.   :  65.0   Min.   : 52.0   
 1st Qu.: 38.0   1st Qu.:50.50    1st Qu.: 1419   1st Qu.: 50.0    1st Qu.: 476.0   1st Qu.:  615.0   1st Qu.: 127.0   1st Qu.:131.0   
 Median : 49.0   Median :58.00    Median : 1518   Median :107.0    Median : 536.5   Median :  813.5   Median : 159.0   Median :149.0   
 Mean   : 52.8   Mean   :59.36    Mean   : 1779   Mean   :105.7    Mean   : 553.0   Mean   :  817.7   Mean   : 246.5   Mean   :146.4   
 3rd Qu.: 62.0   3rd Qu.:67.00    3rd Qu.: 1682   3rd Qu.:150.0    3rd Qu.: 611.0   3rd Qu.:  968.0   3rd Qu.: 249.2   3rd Qu.:164.0   
 Max.   :201.0   Max.   :95.00    Max.   :30132   Max.   :343.0    Max.   :3645.0   Max.   :19278.0   Max.   :1898.0   Max.   :228.0   
 NA's   :772     NA's   :2085                                                       NA's   :102                        NA's   :286     
```

Baseball on the surface is a well understood process. Score more runs than your opponent and tally a win. Tally the most wins in your division and you get to play in the post season for a chance to win the World Series. How a team accomplishes the goal of scoring more runs than their opponent in a given game, is not so straight forward. A run is scored when a batter is able to advance all the way around the base path. The ways a batter can advance around the base path are many. The simplest way is to hit a home run. One swing and the batter can adavnce all the way around the base path and score a run. An example of a complex way is for the batter to get hit by a pitch, steal 2 bases and then get walked home when the pitcher loads the bases. To prevent a team from scoring a run, the most simple path is for the pitching side to strike out every batter they face, an event that has never happened. However, our data set does not include run totals. What we do have are offensive and defensive data for an entire season. While it would seem simple to graph home runs and predict wins, or to graph pitching strike outs to predict wins, this is simply not the case. On the following page are scatter plots of the data points vs "Target_Wins". The plots presented are prior to any data preparation work. Initial EDA has demonstrated that work needs to be done on this data set to deal with missing values, outliers and likely erroneous or improbable values.



Research Questions:

- Do the of the teams with the highest target wins exhibit patterns in their variables that differentiate them from the rest of the population?
- If these patterns or anomalies exist, what do they tell us about the best performing teams if anything?
- Are the top teams defined by pitching or hitting?
- Do any hitting or pitching stats that stand out from the rest?
- Does base running or fielding factor in?

The top performers may be defined as top 5%, 10% or top quartile. 10% will be the starting case.

Research Methods:

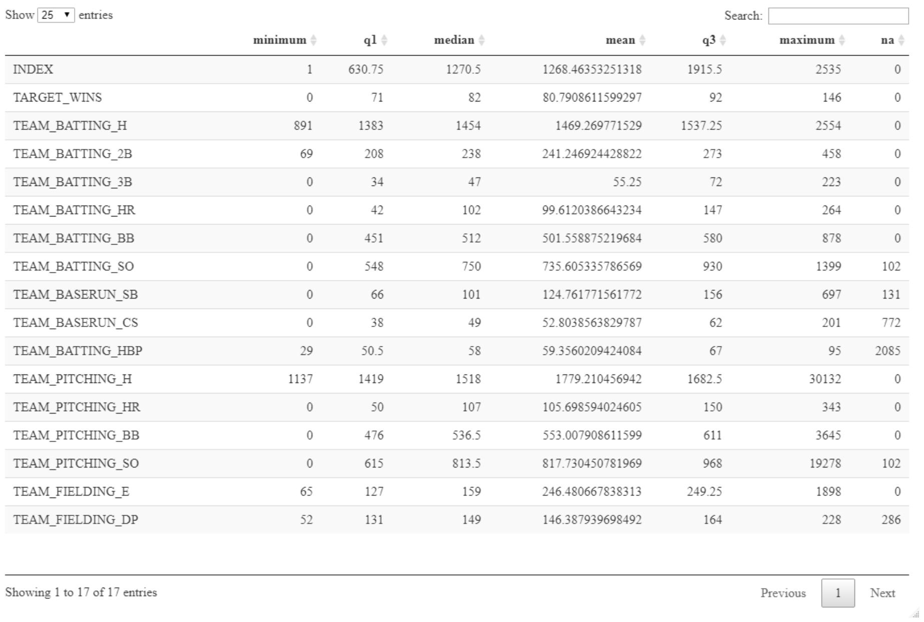
The application of Self Organizing Maps (SOM), will be applied to search for potential patterns and anomalies in the top performing observations as defined by the highest. To perform this analysis, the following SOMs will be investigated with levels for target wins of either top performing or not top performing.

- Batting metrics
- Base running metrics
- Combined offensive metrics (batting and base running)*
- Pitching metrics
- Fielding metrics
- Combined defensive metrics (pitching and fielding)*
- Combined offensive and defensive metrics*

Combined SOMs (*) may be of selected variables from each category dependent on the results of the previous analyses. Variable creation within the categories will likely play a part in developing the analysis. For example, the variable “1B-hits” may be created or doubles, triples and HRs may be combined into “extra base hits”.

EDA and Data Preparation:

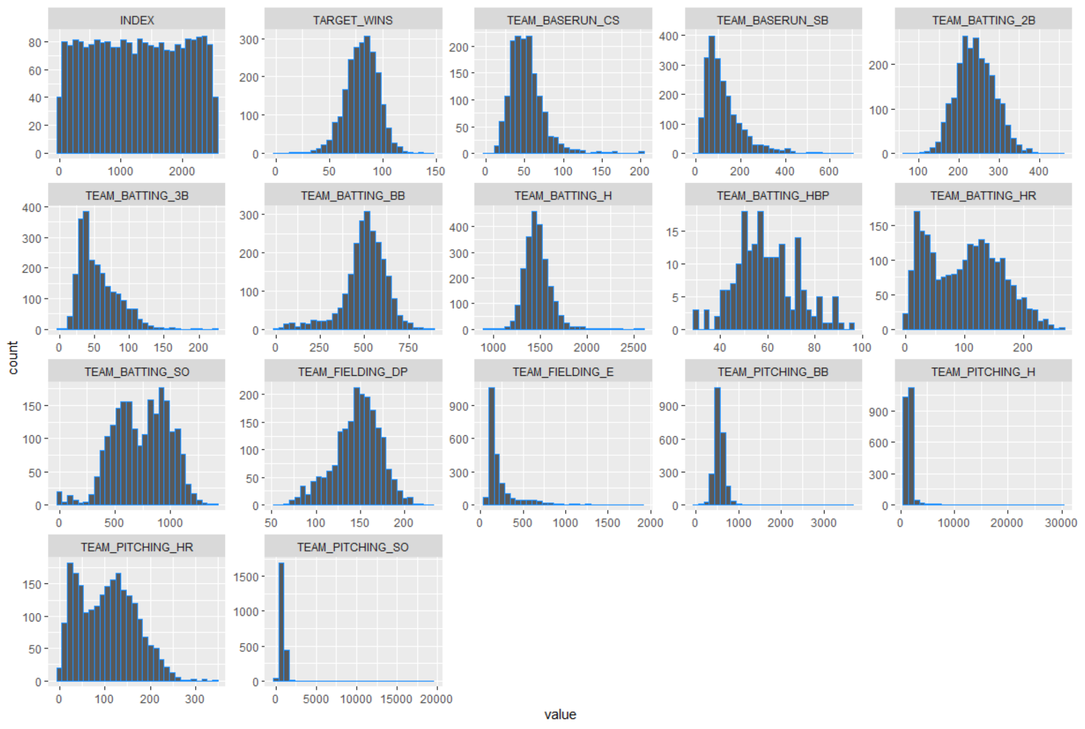
Table of Initial State



Show 25 entries Search:

	minimum	q1	median	mean	q3	maximum	na
INDEX	1	630.75	1270.5	1268.46353251318	1915.5	2535	0
TARGET_WINS	0	71	82	80.7908611599297	92	146	0
TEAM_BATTING_H	891	1383	1454	1469.269771529	1537.25	2554	0
TEAM_BATTING_2B	69	208	238	241.246924428822	273	458	0
TEAM_BATTING_3B	0	34	47	55.25	72	223	0
TEAM_BATTING_HR	0	42	102	99.6120386643234	147	264	0
TEAM_BATTING_BB	0	451	512	501.558875219684	580	878	0
TEAM_BATTING_SO	0	548	750	735.605335786569	930	1399	102
TEAM_BASERUN_SB	0	66	101	124.761771561772	156	697	131
TEAM_BASERUN_CS	0	38	49	52.8038563829787	62	201	772
TEAM_BATTING_HBP	29	50.5	58	59.3560209424084	67	95	2085
TEAM_PITCHING_H	1137	1419	1518	1779.210456942	1682.5	30132	0
TEAM_PITCHING_HR	0	50	107	105.698594024605	150	343	0
TEAM_PITCHING_BB	0	476	536.5	553.007908611599	611	3645	0
TEAM_PITCHING_SO	0	615	813.5	817.730450781969	968	19278	102
TEAM_FIELDING_E	65	127	159	246.480667838313	249.25	1898	0
TEAM_FIELDING_DP	52	131	149	146.387939698492	164	228	286

Showing 1 to 17 of 17 entries Previous 1 Next



The first order of business is to sense check the ranges of the variables and deal with missing and erroneous or improbable values. After the initial analysis, a redetermination may be made on whether to adjust the changes below.

Initial Data Set Observations	=	2,276
Drop target wins =0	=	-1
Remove team with more than 1,783 hits	=	-51
Remove with 2Bs more than 373	=	-6
Remove teams with no Strike Outs or NAs	=	-114
Remove teams giving up more than 1,994 hits	=	-9
Remove teams giving up more than 870 walks	=	-23
Remove teams striking out more than 1,876 batters	=	0
Remove teams with more than 1,646 errors	=	-1
Remove teams with NAs for Caught Stealing	=	-598
Remove teams with NAs for Double Plays	=	-4
Total Reduction	=	-807
Sample Population	=	1469

*statistics used to prep data from fangraphs.com

Cleaning Discussion:

Target Wins

No team has ever had a winless season. The 1899 Cleveland spiders won 20 games. The one record with zero wins will be eliminated. No other issues are detected.

Team_Batting_Hits

The 1930 NY Giants hold the single season record for team hits as far as I can find at 1783 hits. There are 51 records that fall above the record for hits.

Team_Batting_SO

No teams have ever recorded zero strikeouts at the plate. Some consideration was giving to imputation of strikeouts but this was dropped for 2 reasons. First, dropping 120 observations from 2276 should not affect our ability to analyze the data and second, we don't know how significant strikeouts may be as a predictor.

Team_Batting_SB

Stolen bases present a small problem due to rule changes in the early years of the sport. In 1892, a steal included if the runner advanced on a flyout. The 1976 Athletics had the highest SB total of the modern era (1900+) of 341. For now, the stats will b

Team_Pitching_H

No team in history has given up more than 1993 hits (1930 Phillies).

Team_Pitching_BB

No team has given up more than 869 walks in an adjusted 162 game season.

Team_Pitching_SO

No team has struck out more batters than 1876 on an adjusted season total. In addition, no teams have ever recorded zero strikeouts in pitching. Some consideration was giving to imputation of strikeouts but this was dropped for 2 reasons. First, dropping 120 observations from 2276 should not affect our ability to analyze the data and second, we don't know how significant strikeouts may be as a predictor.

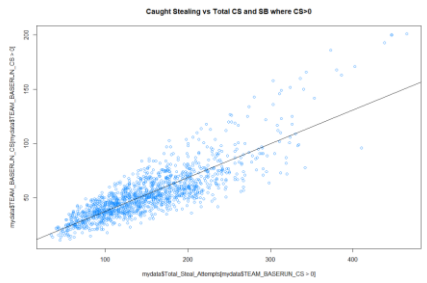
Deal with missing values (NA):

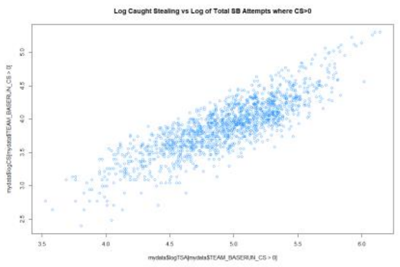
Team_Baserun_CS

There are still 598 teams that have no data for caught stealing. Rather than impute the average, a quick look was done at the relationship between ‘Caught Stealing’ (CS) and the total attempted stolen bases which is CS + SB (stolen base). The linear model is:

CS = 6.743+0.3107*SB

There may be some issues with using this method. It appears the number of caught stealing may be biased above the linear model for total SB+CS greater than 300.

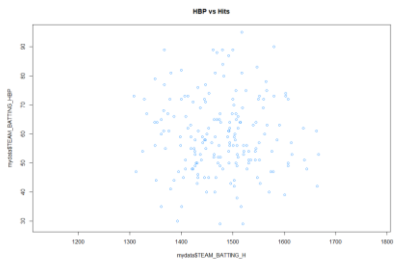


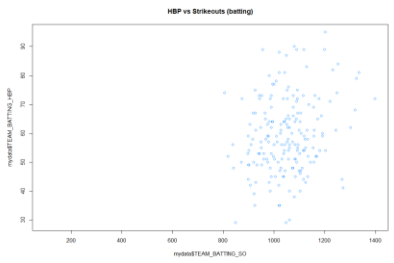


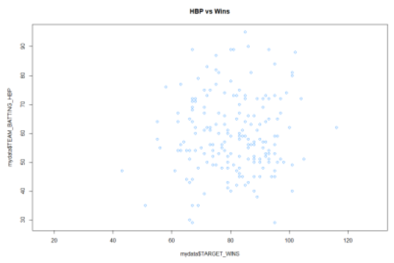
Based on the review above the decision was made to drop the observations missing the caught stealing variable.

Team Batting HBP (Hit by Pitch)

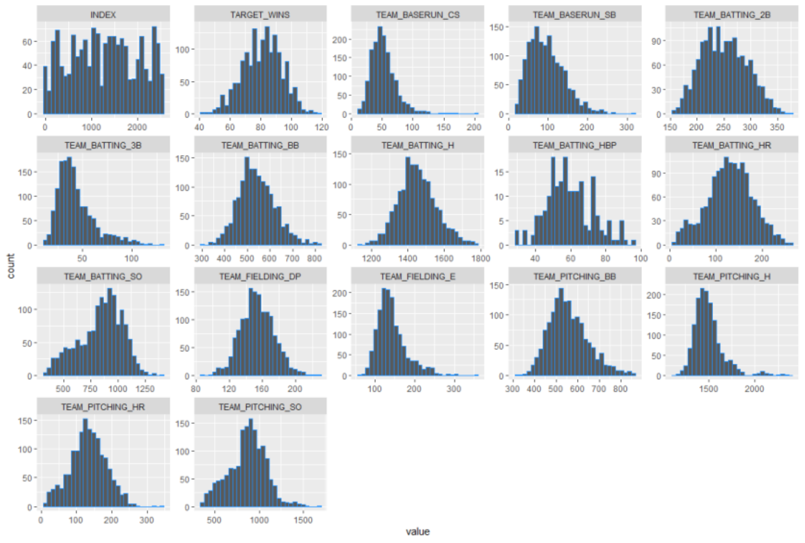
There are 1881 observations with no value for HBP. A look at whether HBP correlated to Batting Hits, Target Wins or Batting Strikeouts was performed. There is no correlation and the spread is quite wide for each of the relationships. I feel it would be unwise to impute the mean or assign a random value for this many observations and generating a relationship does not seem possible given the data set. At this time, this will be a variable that is held out of the analysis.







Data after cleaning:



Feature Engineering:

Top Winning Teams (user created factor)

Of the selected data set. Below are the cutoffs for wins for the top winning teams. (1469 Teams)

Percentile	# of Wins	Number of Teams
Top 5%	101	80
Top 10%	97	167
Top 15%	95	223
Top 20%	92	313
Top 25%	90	377
Top 30%	88	463

Team Batting – Hits Not HRs

This variable was created to capture the true value of any hit not a HR.

Bat_nonHR_Hits = H-HR

Batting – On Base Percentage

This is a proxy for how often a team gets on base for each batter. Each team gets 4,374 outs in a 162 game season with all games going 9 innings. This is not a perfect metric due to extra inning games.

Bat OBP = (H + BB) / (H + BB + 4,374)

Fielding – Defensive Quality

Defense = Double Plays – Errors

BaseRun – Base Running Success Rate

Base Run Success = SB / (SB+CS)

Assumptions:

A few significant assumptions are built into this exercise.

The first is that baseball has been played exactly the same way for the entire history of the data set from the late 1800s. The rules have not changed and that the general strategy has not changed. This is a very large assumption and some of the issues with this assumption will be discussed later.

Second, that there may be something that separates the top 10% of teams from the remaining 90% of teams. This is a huge assumption as well. Is there likely to be a significant or detectable difference in a team that wins 97 games vs a team in the top 20% that wins 92 games? A 5 game difference in a 162 game season is the difference in winning 3.1% more games. The difference between a cut off for teams in the top 10% and the top 30% of target wins is only 9 games or 5.6% of the season games. This a significant fact and should be kept in mind when viewing the maps that follow.

Modeling Procedure:

All maps are trained with 2000 epochs unless more are needed or less are indicated.

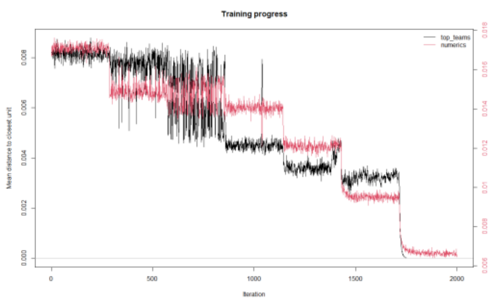
Map size is determined mathematically as follows unless empty cells are detected:

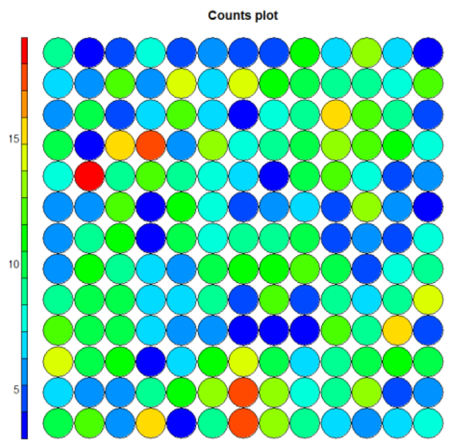
$$\text{Grid size } = floor\ of \sqrt{(\sqrt{N} * 5)} \qquad where\ N = number\ of\ observations$$

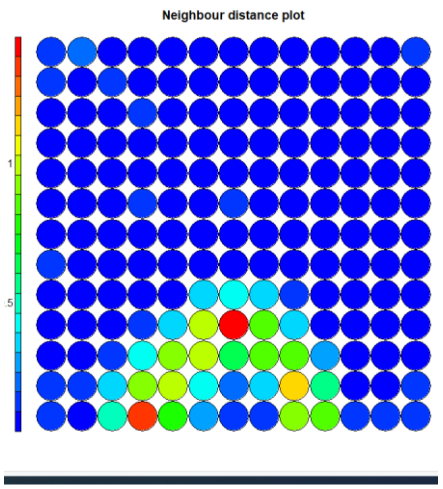
Map size would be adjusted if empty cells were found in the counts plot until no empty cells exist.

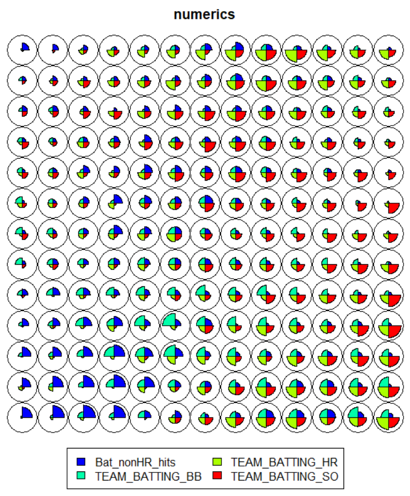
Batting Map

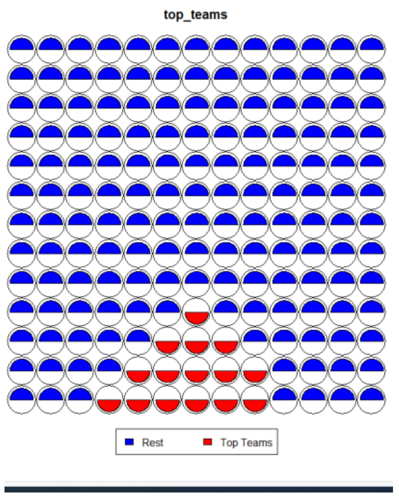
This map was created with the following variables, all are from the “Team_Batting” category and one user created: Non HR Hits, HR, SO, BB







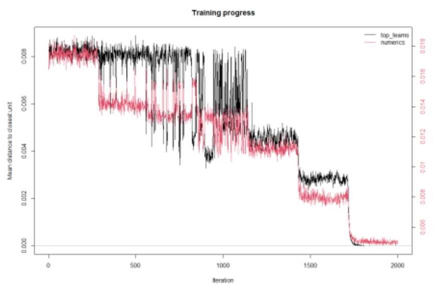


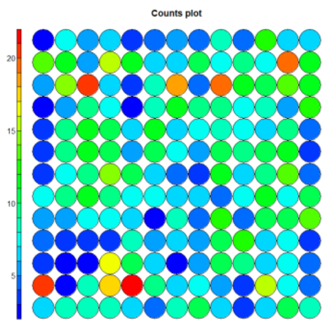


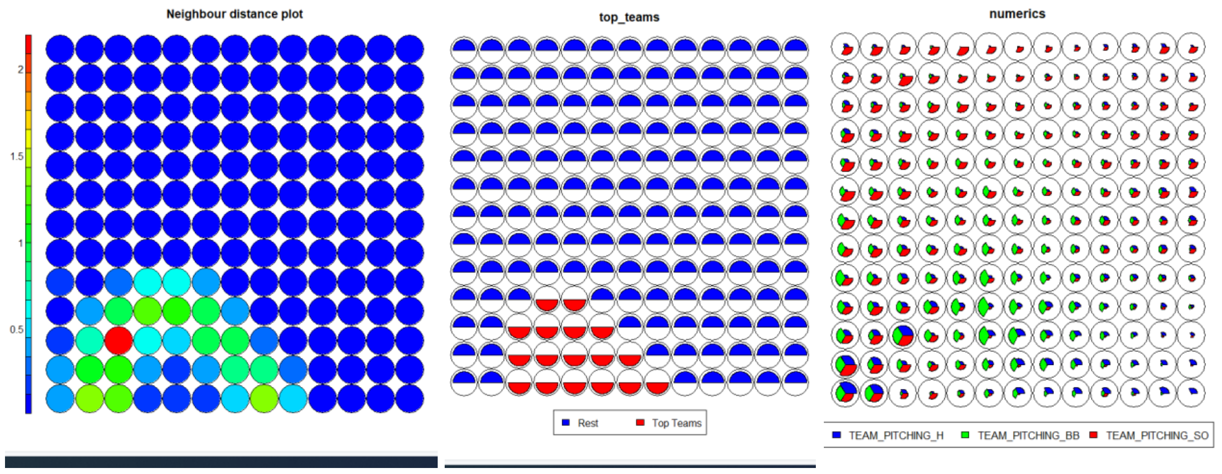
This map appears to demonstrate that batting walks are significant. This is the variable that stands out the most in and near the “Top Teams” grouping. Non HR hits might be significant. This variable appears some in the grouping and a quite strongly near the grouping. Strikes and HRs seem to have no strong bearing. This mapping suggests that the top teams are more likely to get walks during their plate appearances.

Pitching Map

This map was created with the following variables, all are from "Team_Pitching" category:
H, BB, SO.



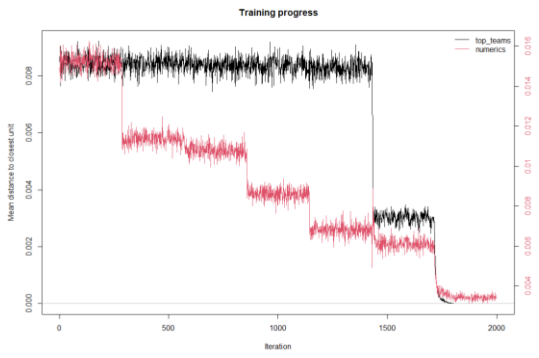


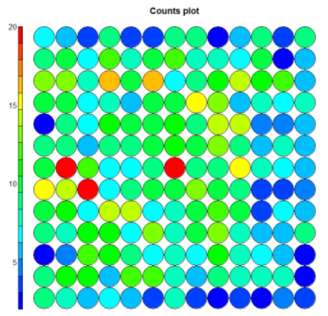


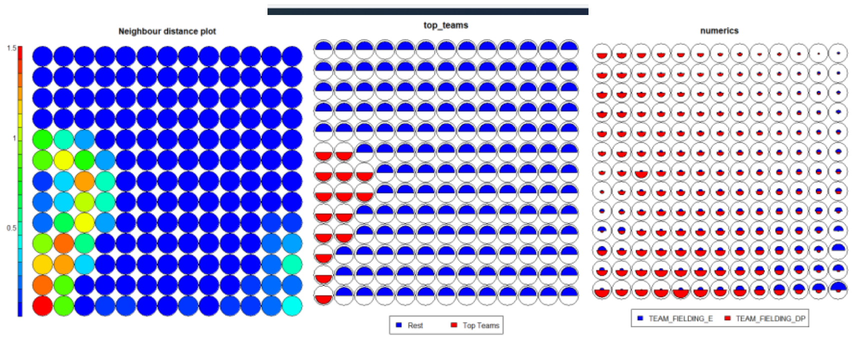
This map is really interesting and somewhat counter intuitive. This again indicates that walks are the important variable for the top teams. This is pitching walks, not batting walks. So, do the top teams walk more batters and walk more? Once again, strike outs have little bearing and in fact appear to be a negative predictor of top performance. This map also suggests giving up hits may not be a poor predictor of being a top team at first glance, however, the cells with stronger indications of hits also have low population counts.

Fielding Map

This map was created using the two Fielding variables.



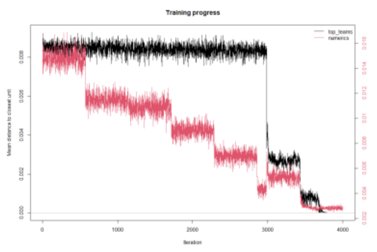


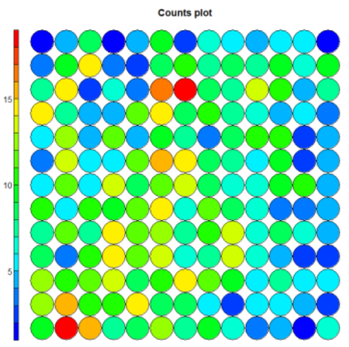


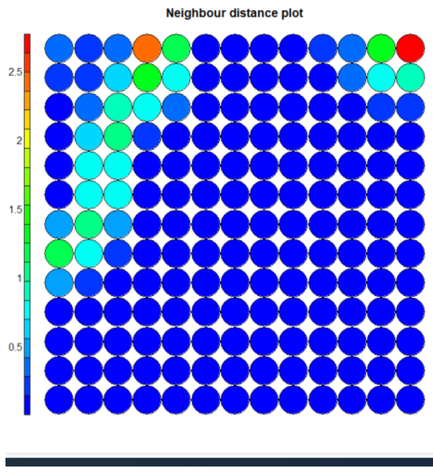
This mapping does not show any indication that fielding errors or double plays have any impact on whether a team is a top 10% performer or not.

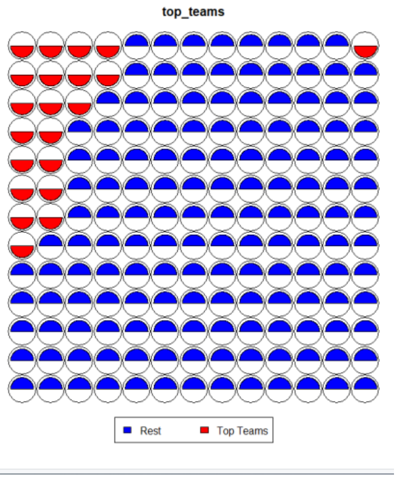
Base Running Map

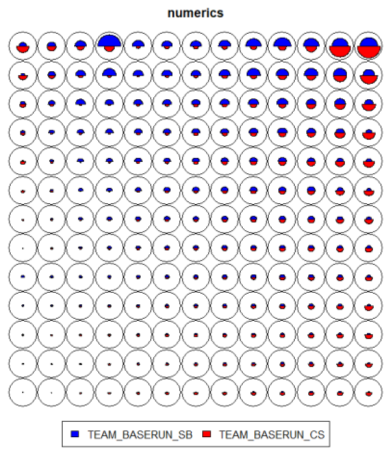
This map was created using the two base running variables.







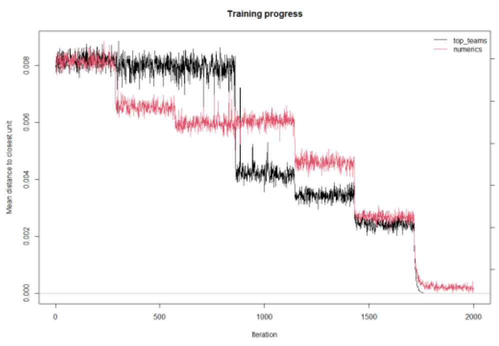


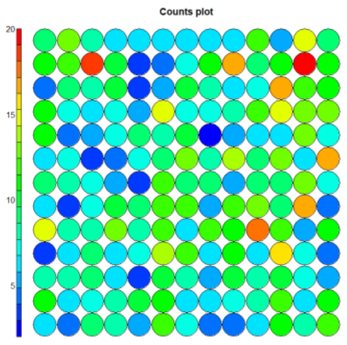


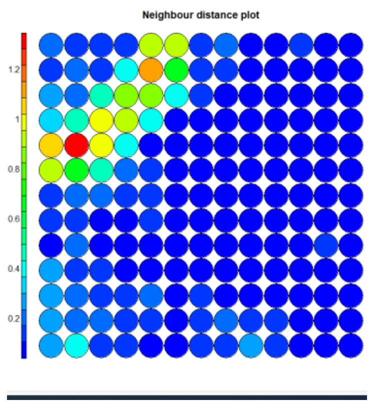
This map indicates that base running has no bearing on whether a team is a top peformer or not. In fact there are two cells that show inverse relations between the two variables, though their population count I slow. In the higher population cells for top teams, it is impossible to make any determination.

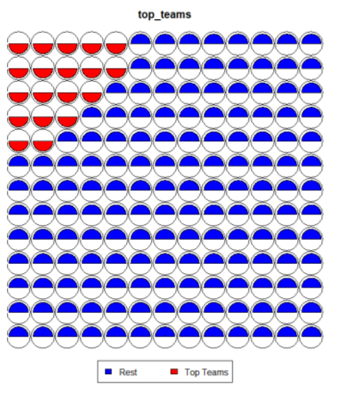
SOM Combined with User Created Vars

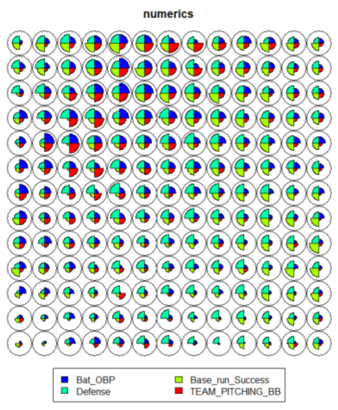
This map has been created to see how the SOM algorithm would perform with on base percentage for hitters, a traditional measure in baseball now, pitching walks and user created variables for defensive quality, base running success rate





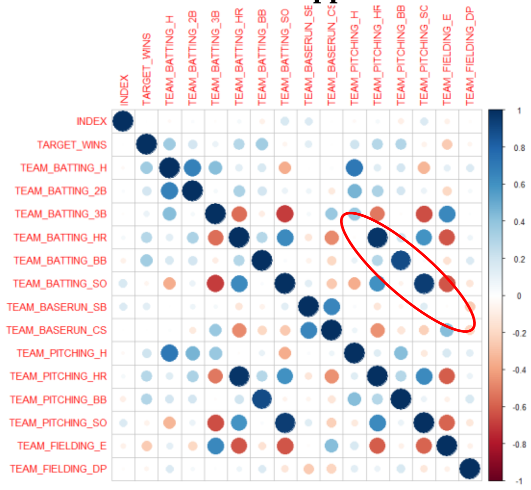




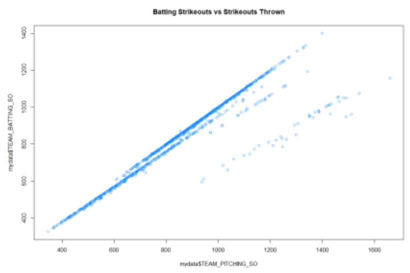


This map was created to see how on base percentage and two user summaries of fielding and base running with the most important variable from pitching, would perform. This map clearly demonstrates that getting runners on base and walking opponents batters are determinant of top performing teams. Walks are an integral part of on base percentage. There is no clear correlation for base running and defensive quality. There are some issues I encountered with generating a combined category SOM. On the following page, you can see there is significant, cross category correlations in this data set.

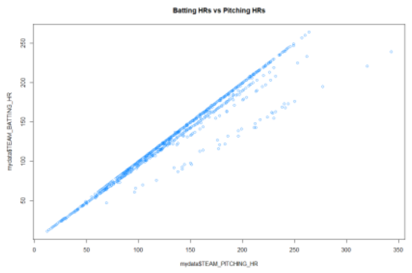
Correlation Plot of Prepped Data Set



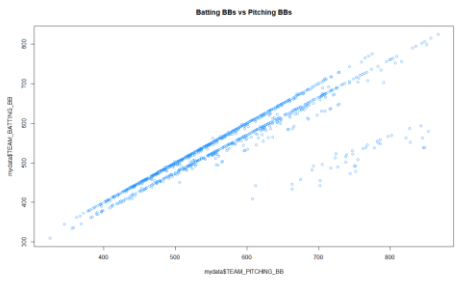
Batting Strikeouts vs Pitching Strikeouts



Batting HRs vs Pitching HRs



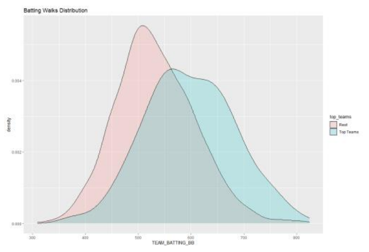
Batting BBs vs Pitching BBs

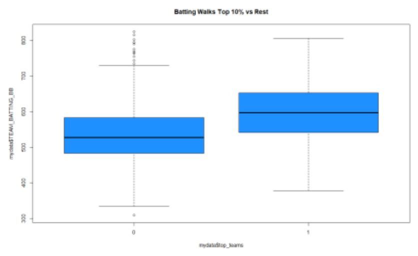


As you can see, there is significant correlation between pitching and hitting variables. There also appear to possibly be different slopes and different groupings on the plots. One possible explanation for this is the impact of time, which we do not have data on. This could be showing different eras.

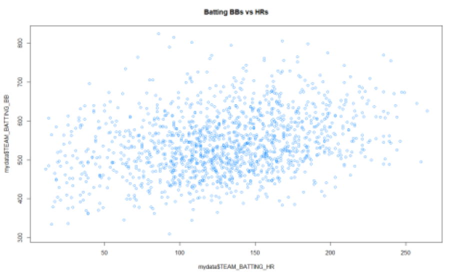
Findings and Commentary

Based on this analysis, the top 10% of teams in the data set have batters who walk more frequently. Below is a distribution plot of top teams and the rest.

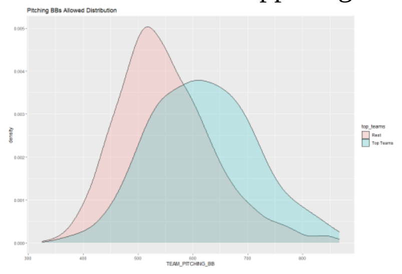


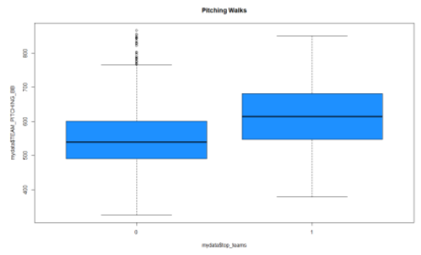


The question is what does this mean. Walks are not outs. A walk creates a base runner, a potential run scoring opportunity. Runners on base have the potential to score even if the current batter becomes an out with balls hit into play, for example, a sacrifice fly or fielders choice. A strikeout is an out and a batter who hits a ball that is not a HR, still has a significant chance of becoming an out instead of becoming a runner. Another way to look at it walking is the most assured way of generating a base runner. Many balls hit in fair territory can become outs. For example, according to fangraphs.com, in 2014, ground balls produced a batting average of .239, meaning they had about a 24% chance of producing a baserunner. Fly balls were worse, producing a baserunner 21% of the time. Line drives were the most successful at producing a baserunner with a 68.5% success rate. This does not even accout for fly ball outs in foul territory. The only other sure option of generating a base runner is a home run. As the plot below shows, walks are much more common than home runs.



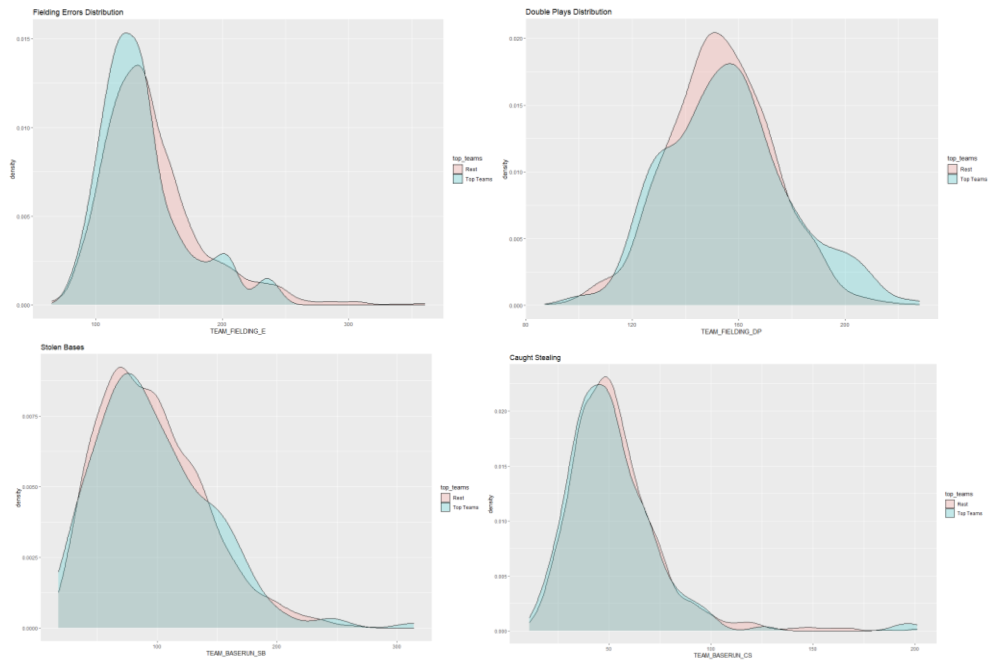
The top 10% of teams also walk more opposing batters.





How does this fit into being a winning team as it seems they are allowing other teams to ? Assuming that time is not a factor in this correlation, then good teams may pitch in a way that minimizes risk. This may be an optimization of risk/reward. This is a hypothesis as inidvidual player, game and situational data is not available.

Fielding and baserunning are not determinant of top teams. The SOMs did not show any correlation between a teams fileding or base running capabilities. Below are the distribution plots for the Fielding and Base Running category



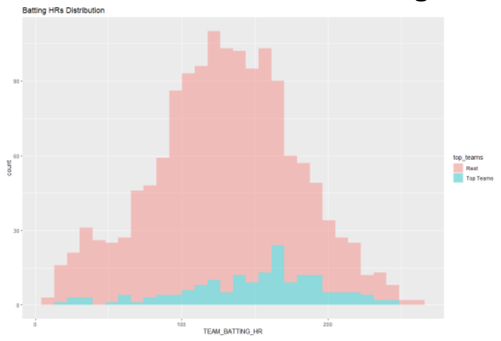
What must be remembered is the small difference between being a top 10% team and a top 20% or even top 30% team. It's a matter of winning an additional 3-5% of the games in a 162 game season. Small differences in the number of opportunities to score are what matter. This makes sense in the case of batting walks, where it gets fuzzy is in the case of pitching walks allowed. Beyond the previously presented hypothesis of risk management, more information would be required to study why these teams are "walking their way to victory" on both sides of the plate.

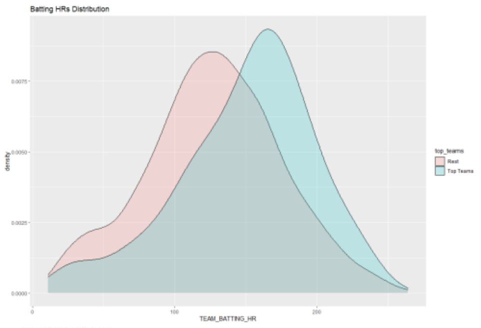
References:

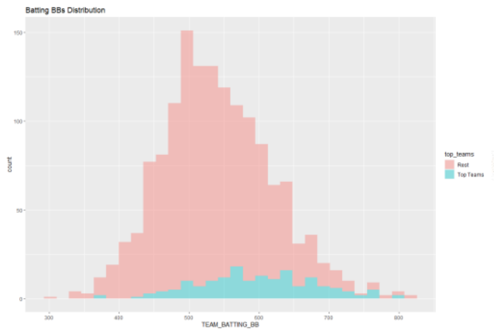
Fangraphs. (n.d.) Retrieved August 10, 2020 from https://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual=0&type=8&season=2020&month=0&season1=2020&ind=0&team=0,ts&rost=&age=&filter=&players=0

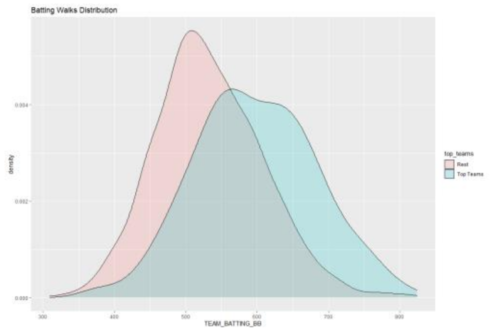
Appendix:

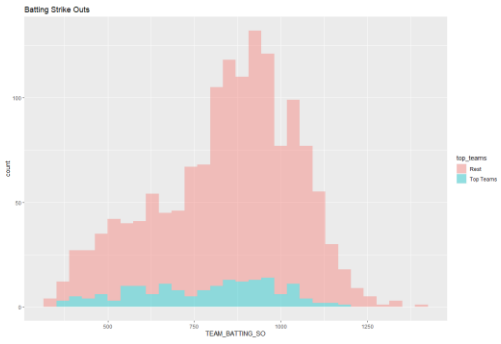
Additional Histograms and Distribution Plots

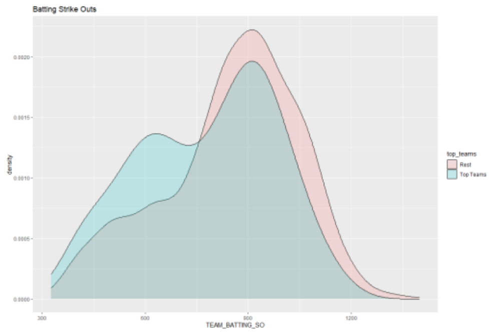


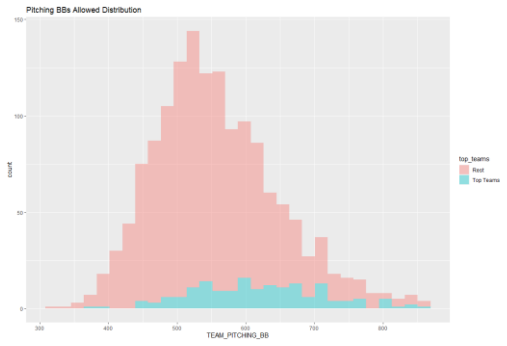


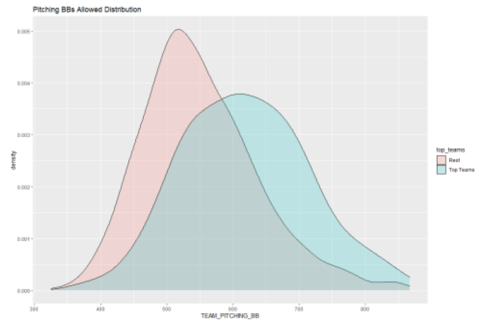


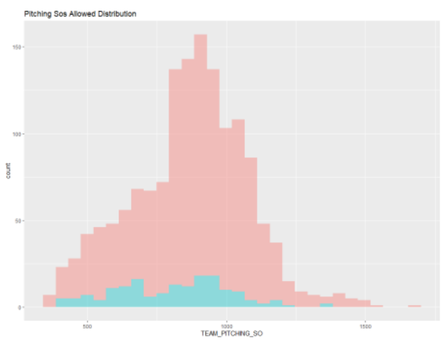


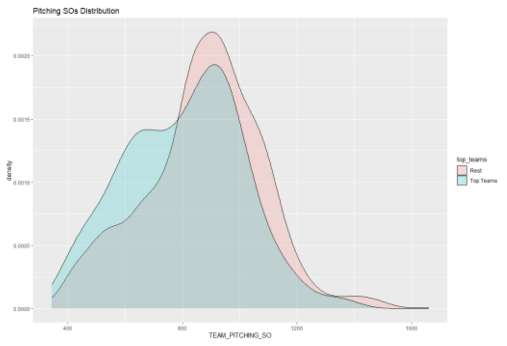


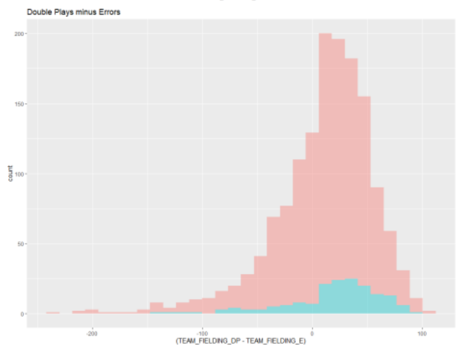


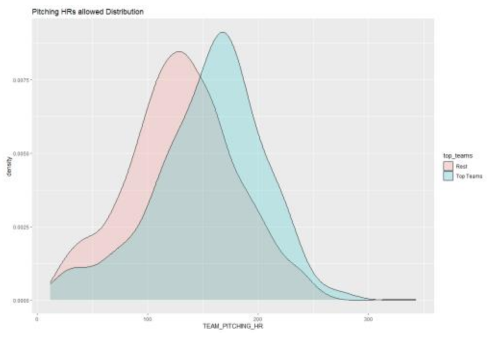


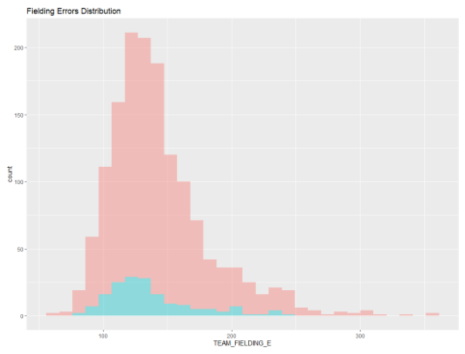


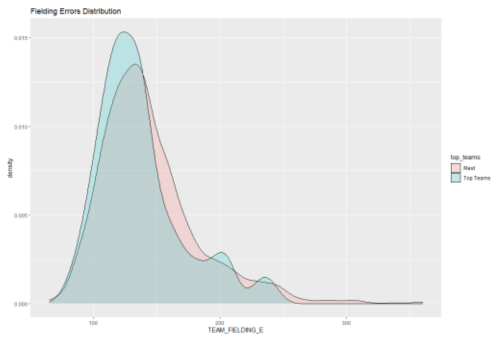

Pitching Sos Allowed Distribution
count
TEAM_PITCHING_SO
top_teams
Rest
Top Teams

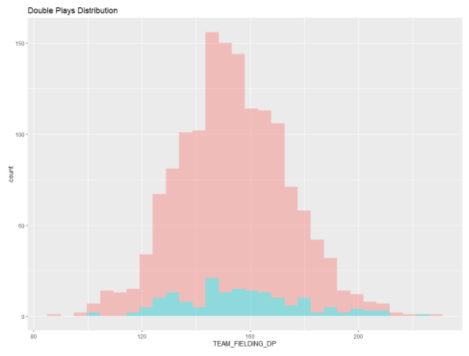

Pitching SOs Distribution
density
TEAM_PITCHING_SO
top_teams
Rest
Top Teams

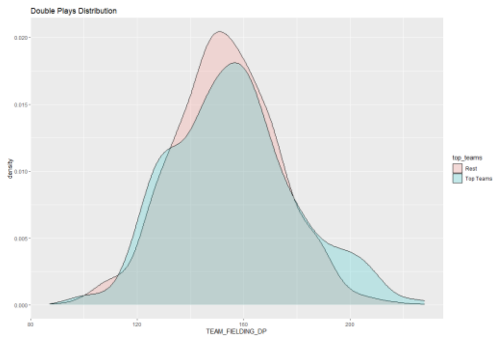

Double Plays minus Errors
count
(TEAM_FIELDING_DP - TEAM_FIELDING_E)
top_teams
Rest
Top Teams

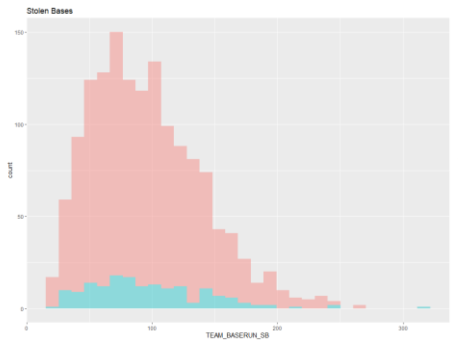

Pitching HRs allowed Distribution
density
TEAM_PITCHING_HR
top_teams
Rest
Top Teams

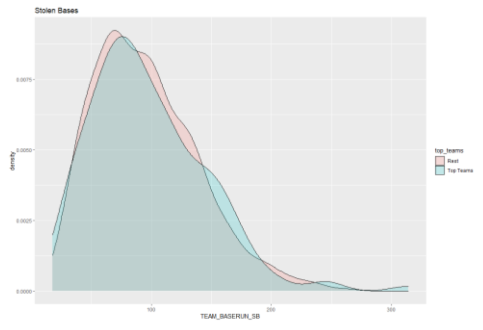

Fielding Errors Distribution
count
TEAM_FIELDING_E
top_teams
Rest
Top Teams


Fielding Errors Distribution
density
TEAM_FIELDING_E
top_teams
Rest
Top Teams

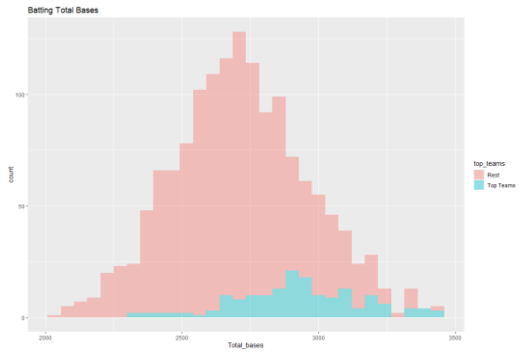

Double Plays Distribution
count
TEAM_FIELDING_DP
top_teams
Rest
Top Teams

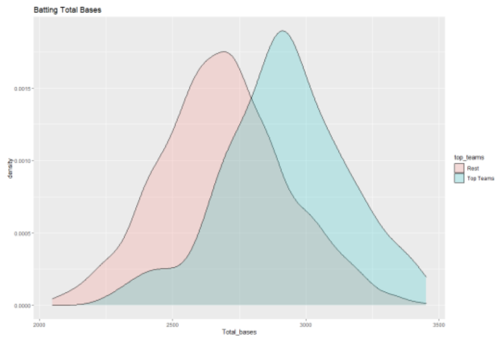

Double Plays Distribution
density
TEAM_FIELDING_DP
top_teams
Rest
Top Teams

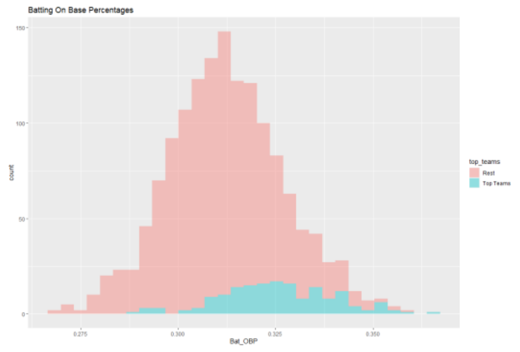

Stolen Bases
count
TEAM_BASERUN_SB
top_teams
Rest
Top Teams

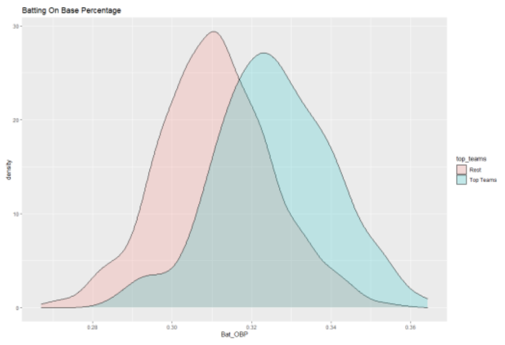

Stolen Bases
density
TEAM_BASERUN_SB
top_teams
Rest
Top Teams

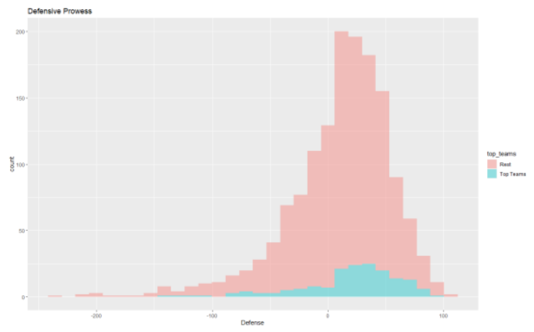
User Created

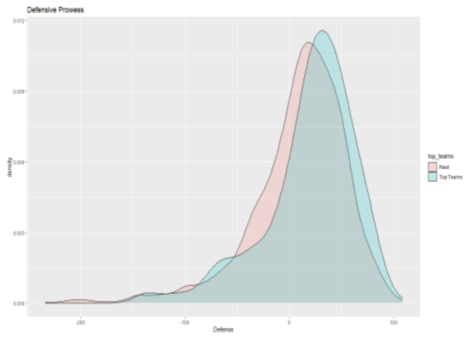


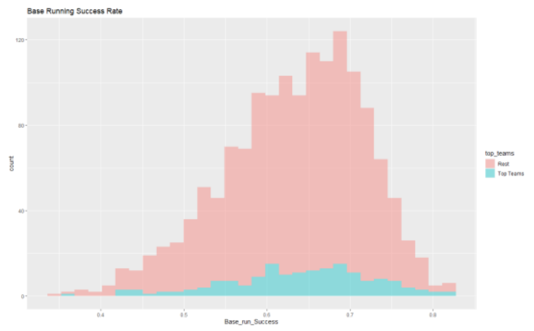


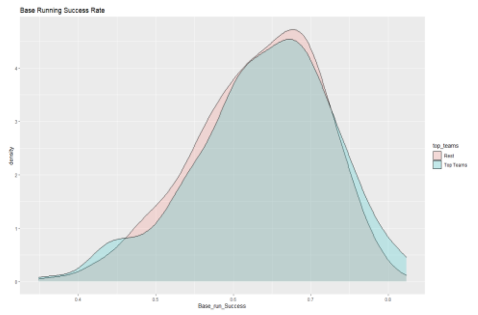












Additional Features Created

Team Batting - Singles (Team Batting 1B)
The variable Team Batting Hits is a total of all hits that a given team had during a season. This includes the doubles (Team Batting 2B), triples (Team Batting 3B) and home runs (Team Batting HR). To separate the stats, a variable Team Batting 1B will be created:

1B = H –2B –3B –HR

Team Batting – Total Base Runners
This variable was created to capture the true value of each hit and base runner that reached base. Not every hit is created equal and we should not treat them that way.

Total Base Runners = 1B + 2B + 3B + BB

Team Batting – Total Bases (Team Batting Total Bases)
This variable was created to capture the true value of each hit and base runner that reached base. Not every hit is created equal and we should not treat them that way.

Total Bases = 1B + (2 * 2B) + (3 * 3B) + (4 * HR) + BB + SB

Team Pitching - Base Runners Allowed
As HRs are included in the Hits total, let's create a variable to show how many hits besides HRs a pitching staff gave up for a season.

Base Runners Allowed = Team_Pitching_H + Team_pitching_BB

Code:

```
####### MSDS 411 Sec 55, Summer 2020 ################################
#                                          #
#       Final Project                          #
#       Money Ball                              #
#                                          #
######################################################################


library(ggplot2)
library(gridExtra)
library(knitr)
library(RColorBrewer)
library(kableExtra)
library(lessR)
library(car)
library(psych)
library(plyr)
library(corrplot)
library(tidyr)
library(purrr)
library(broom)
library(tidyverse)



MoneyBall <- data.frame(MoneyBall)








MoneyBall %>%
  gather(-TARGET_WINS, -INDEX, key = "var", value = "value") %>%
  ggplot(aes(x = value, y = TARGET_WINS)) +
  geom_point(col="dodgerblue") +
  facet_wrap(~ var, scales = "free") +
  theme_bw()

MoneyBall %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free") +
  geom_histogram(col = "dodgerblue")


#par(mfrow = c(1,1))
boxplot(MoneyBall$TEAM_BATTING_H, main = " Hits set 3.0 IQR", col = "green",
        horizontal = TRUE, range = 3.0, notch = TRUE, xlab = "hits")
boxplot(MoneyBall$TEAM_BATTING_2B, main = " Doubles set 3.0 IQR", col = "green",
        horizontal = TRUE, range = 3.0, notch = TRUE, xlab = "hits")
boxplot(MoneyBall$TEAM_BATTING_3B, main = " Triples set 3.0 IQR", col = "green",
        horizontal = TRUE, range = 3.0, notch = TRUE, xlab = "hits")
boxplot(MoneyBall$TEAM_BATTING_HR , main = "HRs Full data set 1.5 IQR",
        col = "green",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "HRs")
boxplot(MoneyBall$TEAM_BATTING_BB, main = "Batter Walks  1.5 IQR",
        col = "green",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Batter Walks")
boxplot(MoneyBall$TEAM_BASERUN_SB, main = "Stolen Bases 1.5 IQR", col = "green",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Stolen Bases ")
boxplot(MoneyBall$TEAM_BATTING_BB, main = "Hitters Walked 1.5 IQR",
        col = "green",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Walks Earned")
boxplot(MoneyBall$TEAM_BATTING_SO , main = "Batter SOs 1.5 IQR",
        col = "red",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Batter Strike Outs")
boxplot(MoneyBall$TEAM_BASERUN_SB , main = "Stolen Base 1.5 IQR",
        col = "Green",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Stolen Bases")
boxplot(MoneyBall$TEAM_BASERUN_CS, main = "Caught Steeling 1.5 IQR",
        col = "red",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Batter Strike Outs")
boxplot(MoneyBall$TEAM_PITCHING_SO , main = "Pitcher SOs  1.5 IQR",
        col = "dodgerblue",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Pitcher Strike Outs")
boxplot(MoneyBall$TEAM_PITCHING_H , main = "Hits allowed  1.5 IQR", col = "orange",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Hits allowed")
boxplot(MoneyBall$TEAM_PITCHING_H , main = "Walks allowed  1.5 IQR",
        col = "orange",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Walks allowed")
boxplot(MoneyBall$TEAM_PITCHING_HR , main = "HR's allowed  1.5 IQR",
        col = "orange",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "HR's allowed")
boxplot(MoneyBall$TEAM_FIELDING_DP , main = "Double Plays Turned  1.5 IQR",
        col = "dodgerblue",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Double Plays")
boxplot(MoneyBall$TEAM_FIELDING_DP , main = "Errors 1.5 IQR", col = "orange",
        horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Errors")
```

```
add_na_col <- function(x){
  mutate(x, na = 0)
}

has_n_col <- function(x, n = 6){
  return(ncol(x) == n)
}


MoneyBall %>%
  select_if(is.numeric)  %>%
  map(~tidy(summary(.x))) %>%  # compute tidy summary of each var
  map_if(., has_n_col, add_na_col) %>%   # add na-col if missing
  do.call(rbind, .) -> MoneyBall_summary  # bind list elements into df

MoneyBall_summary




library(DT)
datatable(MoneyBall_summary)


mydata<- MoneyBall



#Eliminate record with zero wins
df2 <- subset(MoneyBall, mydata$TARGET_WINS>0)

# How many teams had hits more than

df3 <- subset(df2, df2$TEAM_BATTING_H<1784)

# How many teams had hits more than

df4 <- subset(df3, df3$TEAM_BATTING_2B<373)

# How many teams had no strikeouts


df5 <- subset(df4, df4$TEAM_BATTING_SO>0)
```

```
# How many teams have given up more than 1993 hits/

df6 <- subset(df5, df5$TEAM_PITCHING_SO < 1687)



# How many teams have given up more than 870 walks

df7 <- subset(df6, df6$TEAM_PITCHING_BB < 870)


# How many teams have struck out more than 1876 batters

df8 <- subset(df7, df7$TEAM_PITCHING_SO < 1876)

# How many teams have more than 1647 errors

df9 <- subset(df8, df8$TEAM_FIELDING_E< 1647)


# Prior to eliminating missing CS and missing DP
# mydata <- df9

df10 <- subset(df9, df9$TEAM_BASERUN_CS != 'na')
df11 <- subset(df10, df10$TEAM_FIELDING_DP != 'na')
df12 <- subset(df11, df11$TEAM_PITCHING_H < 1994)

mydata <- df12

View(mydata)
quantile(mydata$TARGET_WINS, c(0.95, 0.9, 0.85, 0.80, 0.75, 0.7))

length(mydata$TARGET_WINS[mydata$TARGET_WINS >= 97])

#### Use top 10% First ###

mydata$top_teams <- ifelse(mydata$TARGET_WINS >=97, 1,0)


View(mydata)

mydata %>%
  select_if(is.numeric)  %>%
  map(~tidy(summary(.x))) %>%  # compute tidy summary of each var
  map_if(., has_n_col, add_na_col) %>%   # add na-col if missing
  do.call(rbind, .) -> mydata_summary  # bind list elements into df
```

```
mydata_summary
datatable(mydata_summary)

summary(mydata)


####################################################################

#                  Correlation plot after removals

####################################################################

Cor.df <- mydata[,-c(11)]

baseball.cor <- cor(Cor.df)
corrplot(baseball.cor)






################# NEED TO DEAL WITH NA's ###########################

# Investigate Caught Stealing

# Create variable for total stolen base attempts
#   is a sum of caught stealing and stolen bases

mydata$Total_Steal_Attempts = mydata$TEAM_BASERUN_CS+mydata$TEAM_BASERUN_SB

mydata$logTSA <- log(mydata$Total_Steal_Attempts)
mydata$logCS <- log(mydata$TEAM_BASERUN_CS)

plot(mydata$TEAM_BASERUN_CS~mydata$TEAM_BASERUN_SB, col = 'dodgerblue',
    main = "Caught Stealing vs Stolen Bases")


plot(mydata$logCS[mydata$TEAM_BASERUN_CS>0]~
      mydata$logTSA[mydata$TEAM_BASERUN_CS>0],
    col = 'dodgerblue', main = "Log Caught Stealing vs Log of Total SB Attempts where CS>0")

# Create a linear model of to predict caught stealing vs total attempts

lm_cs <- lm(mydata$TEAM_BASERUN_CS[mydata$TEAM_BASERUN_CS>0]~
            mydata$Total_Steal_Attempts[mydata$TEAM_BASERUN_CS>0])
```

```
lm_cs


# Create a linear model of loglog to predict caught stealing vs total attempts

lm_logcs <- lm(mydata$logCS[mydata$TEAM_BASERUN_CS>0]~
            mydata$logTSA[mydata$TEAM_BASERUN_CS>0])
lm_logcs

FakeCS <-

plot(mydata$logCS[mydata$TEAM_BASERUN_CS>0]~
       mydata$logTSA[mydata$TEAM_BASERUN_CS>0],
     col = 'dodgerblue', main = "Caught Stealing vs Total CS and SB where CS>0")
abline(lm_logcs)

# HBP NAs

mydata$TEAM_BATTING_HBP[is.na(mydata$TEAM_BATTING_HBP)]

mean(mydata$TEAM_BATTING_HBP)

# Create plots of HBP vs various stats

plot(mydata$TEAM_BATTING_HBP~mydata$TEAM_BATTING_H, col = 'dodgerblue',
     main = "HBP vs Hits")
plot(mydata$TEAM_BATTING_HBP~mydata$TARGET_WINS, col = 'dodgerblue',
     main = "HBP vs Wins")
plot(mydata$TEAM_BATTING_HBP~mydata$TEAM_BATTING_SO, col = 'dodgerblue',
     main = "HBP vs Strikeouts (batting)")


# Try to model HBP

df_HBP <- subset(mydata, mydata$TEAM_BATTING_HBP>0)

lm_HBP1 <- lm(df_HBP$TEAM_BATTING_HBP ~ df_HBP$TEAM_BATTING_HR)
lm_HBP1
plot(df_HBP$TEAM_BATTING_HBP~df_HBP$TEAM_BATTING_HR)
abline(lm_HBP1)




# Double Plays
plot(mydata$TEAM_FIELDING_DP[mydata$TEAM_FIELDING_DP>0]~mydata$TARGET_WINS[my
data$TEAM_FIELDING_DP>0], col = 'dodgerblue',
    main = "Double Plays vs Wins")

plot(mydata$TEAM_FIELDING_DP[mydata$TEAM_FIELDING_DP>0]~mydata$TEAM_FIELDING_
E[mydata$TEAM_FIELDING_DP>0], col = 'dodgerblue',
    main = "Double Plays vs Errors")

plot(mydata$TEAM_FIELDING_DP[mydata$TEAM_FIELDING_DP>0]~mydata$TEAM_PITCHING_
BB[mydata$TEAM_FIELDING_DP>0], col = 'dodgerblue',
    main = "Double Plays vs Walks Allowed")




# Team Batting vs Pitching Strikeouts

plot(mydata$TEAM_BATTING_SO~mydata$TEAM_PITCHING_SO, col = 'dodgerblue',
    main = "Batting Strikeouts vs Strikeouts Thrown")




# Index vs SOs - does index relate to year?

plot(mydata$TEAM_BATTING_SO~mydata$INDEX, col = 'dodgerblue',
    main = "Batting Strikeouts vs Index")

# Batting HRs vs Pitching HRs

plot(mydata$TEAM_BATTING_HR~mydata$TEAM_PITCHING_HR, col = 'dodgerblue',
    main = "Batting HRs vs Pitching HRs")

# Batting HRs vs Index

plot(mydata$TEAM_BATTING_HR~mydata$INDEX, col = 'dodgerblue',
    main = "Batting HRs vs Index")

# Batting Walks vs pitching walks

plot(mydata$TEAM_PITCHING_BB,mydata$TEAM_BATTING_BB, col = 'dodgerblue',
    main = "Batting BBs vs Pitching BBs")
points(mydata$TEAM_PITCHING_BB[mydata$top_teams==1],mydata$TEAM_BATTING_BB[mydata
$top_teams==1], col = "red1", bg="red1", pch=19)

hist((mydata$TEAM_PITCHING_SO-mydata$TEAM_BATTING_SO), main =  "Batting BBs vs
Pitching BBs", xlab = "Difference")
```

```
View(mydata)


# Pitching HRs vs pitching BBs

plot(mydata$TEAM_PITCHING_BB,mydata$TEAM_PITCHING_HR, col = 'dodgerblue',
    main = "Pitching HRs vs Pitching BBs")
points(mydata$TEAM_PITCHING_BB[mydata$top_teams==1],mydata$TEAM_PITCHING_HR[mydat
a$top_teams==1], col = "red1", bg="red1", pch=19)

plot(mydata$TEAM_PITCHING_SO,mydata$TEAM_PITCHING_HR, col = 'dodgerblue',
    main = "Pitching HRs vs Pitching SOs")
points(mydata$TEAM_PITCHING_SO[mydata$top_teams==1],mydata$TEAM_PITCHING_HR[mydat
a$top_teams==1], col = "red1", bg="red1", pch=19)

plot(mydata$TEAM_PITCHING_H,mydata$TEAM_PITCHING_, col = 'dodgerblue',
    main = "Pitching HRs vs Pitching SOs")
points(mydata$TEAM_PITCHING_H[mydata$top_teams==1],mydata$TEAM_PITCHING_BB[mydata
$top_teams==1], col = "red1", bg="red1", pch=19)

hist((mydata$TEAM_PITCHING_BB-mydata$TEAM_PITCHING_HR), main =  "Pitching HRs vs
Pitching BBs", xlab = "Difference")

# Batting HRs vs Batting SOs

plot(mydata$TEAM_BATTING_SO,mydata$TEAM_BATTING_HR, col = 'dodgerblue',
    main = "Batting HRs to Batting SO")
points(mydata$TEAM_BATTING_SO[mydata$top_teams==1],mydata$TEAM_BATTING_HR[mydata
$top_teams==1], col = "red1", bg="red1", pch=19)

hist((mydata$TEAM_PITCHING_BB-mydata$TEAM_PITCHING_HR), main =  "Pitching HRs vs
Pitching BBs", xlab = "Difference")


# Batting OBP vs Walks

plot(mydata$Bat_OBP~mydata$TEAM_BATTING_BB, col = 'dodgerblue',
    main = "OBP vs  BBs")

# Batting OBP vs HIts

plot(mydata$Bat_OBP~mydata$TEAM_BATTING_H, col = 'dodgerblue',
    main = "OBP vs Hits")

# Batting HR vs Walks
```

```
plot(mydata$TEAM_BATTING_HR,mydata$TEAM_BATTING_BB, col = 'dodgerblue',
    main = "Batting BBs vs HRs")


plot((mydata$TEAM_BATTING_BB-
mydata$TEAM_PITCHING_BB),mydata$TEAM_PITCHING_BB, col = 'dodgerblue',
    main = "Pitching BBs vs  Batting BBs")
points((mydata$TEAM_BATTING_BB[mydata$top_teams==1]-
mydata$TEAM_PITCHING_BB[mydata$top_teams==1]),mydata$TEAM_PITCHING_BB[mydata$top_
teams==1], col = "red1", bg="red1", pch=19)

# Batting HR vs Walks

plot(MoneyBall$TEAM_BATTING_BB~MoneyBall$TARGET_WINS, col = 'dodgerblue',
    main = "Batting BBs vs Target wins - Prior to Cleaning")

plot(mydata$TEAM_BATTING_BB~mydata$TARGET_WINS, col = 'dodgerblue',
    main = "Batting BBs vs Target wins - Post Cleaning")


###############################################################################

#                   Distributions Check                                #

###############################################################################


mydata %>%
  gather(-TARGET_WINS, -INDEX, key = "var", value = "value") %>%
  ggplot(aes(x = value, y = TARGET_WINS)) +
  geom_point(col="dodgerblue") +
  facet_wrap(~ var, scales = "free") +
  theme_bw()

mydata %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free") +
  geom_histogram(col = "dodgerblue")



boxplot(mydata$TEAM_PITCHING_H , main = "Pitching Hits Allowed 3.0 IQR", col = "dodgerblue",
       horizontal = TRUE, range = 3.0, notch = TRUE, xlab = "Hits allowed")

boxplot(mydata$TEAM_BASERUN_CS , main = "Caught Steeling 3.0 IQR", col = "dodgerblue",
	horizontal = TRUE, range = 3.0, notch = TRUE, xlab = "Caught Stealing")


boxplot(mydata$TEAM_PITCHING_SO , main = "Pitching Strikeouts 1.5 IQR", col = "dodgerblue",
	horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Pitching Strikeouts")


boxplot(mydata$Bat_HRSO_rat , main = "Pitching Strikeouts 1.5 IQR", col = "dodgerblue",
	horizontal = TRUE, range = 1.5, notch = TRUE, xlab = "Pitching Strikeouts")

################################################################################
#                                                    #
#                Feature Engineering                          #
#                                                    #
################################################################################

# Create variable for singles

mydata$Sinlges <- mydata$TEAM_BATTING_H-mydata$TEAM_BATTING_2B-
mydata$TEAM_BATTING_3B-mydata$TEAM_BATTING_HR


# Create variable for base runners allowed

mydata$Runners_Allowed <- mydata$TEAM_PITCHING_H+mydata$TEAM_PITCHING_BB

# Create a variable for total base runners

mydata$Base_Runners <- mydata$TEAM_BATTING_H+mydata$TEAM_BATTING_BB-
mydata$TEAM_BATTING_HR - mydata$TEAM_BASERUN_CS

# Create a variable for Extra Base Hits

mydata$EBHs<- mydata$TEAM_BATTING_2B+mydata$TEAM_BATTING_3B


# Create pitching variable for all hits that are not HRs
mydata$Pitching_Hits_NoHR <- mydata$TEAM_PITCHING_H-mydata$TEAM_PITCHING_HR

# Create a variable for total bases
mydata$Total_bases <- mydata$Sinlges + (2*mydata$TEAM_BATTING_2B) +
(3*mydata$TEAM_BATTING_3B) + (4*mydata$TEAM_BATTING_HR) +
(mydata$TEAM_BATTING_BB)+mydata$TEAM_BASERUN_SB



# Create varible for Batting on Base Percentage
```

```
mydata$Bat_OBP<- (mydata$TEAM_BATTING_H +
mydata$TEAM_BATTING_BB)/(mydata$TEAM_BATTING_H+mydata$TEAM_BATTING_BB+4374
)

# Create variable non HR hits Batting

mydata$Bat_nonHR_hits <- mydata$TEAM_BATTING_H-mydata$TEAM_BATTING_HR

# Create variable pitching OBP
mydata$Pitch_OBP <-
(mydata$TEAM_PITCHING_H+mydata$TEAM_PITCHING_BB)/(mydata$TEAM_PITCHING_H+my
data$TEAM_PITCHING_BB+mydata$TEAM_FIELDING_E+4374)


# Create variable non SO outs for batters (contact outs)

mydata$bat_non_SO_outs <- 4374-mydata$TEAM_BATTING_SO


# Walk to strikeout ratio

mydata$Bat_BB_SO_Rat <- mydata$TEAM_BATTING_BB/mydata$TEAM_BATTING_SO

# Walk + HR to strikeout ratio

mydata$Bat_BBHR_SO_Rat <-
(mydata$TEAM_BATTING_BB+mydata$TEAM_BATTING_HR)/mydata$TEAM_BATTING_SO

# Walk + Hit to strikeout ratio

mydata$Bat_BBHit_SO_Rat <-
(mydata$TEAM_BATTING_BB+mydata$TEAM_BATTING_HR)/mydata$TEAM_BATTING_SO

#Create Variable walk per AB ratio

mydata$BB_AB_RAT <-
mydata$TEAM_BATTING_BB/(mydata$TEAM_BATTING_H+mydata$TEAM_BATTING_BB+4374)

# Create variable for SLG
mydata$Bases_AB <-
mydata$Total_bases/(mydata$TEAM_BATTING_H+mydata$TEAM_BATTING_BB+4374)


# Create variable Better Defensive

mydata$Defense <-mydata$TEAM_FIELDING_DP-mydata$TEAM_FIELDING_E
```

```
# Create variable Base Running SUccess

mydata$Base_run_Success <-
mydata$TEAM_BASERUN_SB/(mydata$TEAM_BASERUN_SB+mydata$TEAM_BASERUN_CS)


# Create variable strikeout ratio

mydata$so_rat <- mydata$TEAM_PITCHING_SO/mydata$TEAM_BATTING_SO

# Create variable strikeout ratio

mydata$Bat_HRSO_rat <- mydata$TEAM_BATTING_HR/mydata$TEAM_BATTING_SO

# Create variable strikeout ratio

mydata$Pitch_HRSO_rat <- mydata$TEAM_PITCHING_HR/mydata$TEAM_PITCHING_SO


# Create variable strikeout ratio

mydata$Bat_HRBB_rat <- mydata$TEAM_BATTING_HR/mydata$TEAM_BATTING_BB

# Create variable strikeout ratio

mydata$Pitch_HRBB_rat <- mydata$TEAM_PITCHING_HR/mydata$TEAM_PITCHING_BB


# Create variable for delta strike outs
mydata$SO_Delta <- mydata$TEAM_PITCHING_SO-mydata$TEAM_BATTING_SO

################################################################################
#                                                    #
#            Define Top Teams (10%, 20%, 30%)                    #
#                                                    #
################################################################################



quantile(mydata$TARGET_WINS, c(0.95, 0.9, 0.85, 0.80, 0.75, 0.7))

length(mydata$TARGET_WINS[mydata$TARGET_WINS >= 88])

#### Use top 10% First ###
```

```
mydata$top_teams <- ifelse(mydata$TARGET_WINS >=97, 1,0)


length(mydata$TEAM_BATTING_HR[mydata$TARGET_WINS == 0])



#############################################################################
#                                                     #
#                Variable elimination                 #
#                                                     #
#############################################################################

View(mydata)


#############################################################################
#
#               SOMs
#
##############################################################################

######################################
# Helper functions for all SOMS
########################################

# find the graph node number by the coordinates
find_node_by_coordinates <- function(x, y, grid_width) {
  return(((y * grid_width) + x) - grid_width)
}

# return the number of observations represented by each node
get_node_counts <- function(x) {
  df <- data.frame(node = x)
  counts <- df %>%
    group_by(node) %>%
    summarize(observations = n())
}

# guideline for grid size = 5 * sqrt(N)
# where N is the number of observations in the data set
find_grid_size <- function(N) {
  return(floor(sqrt(sqrt(N) * 5)))
}

# Shane Lynn 14-01-2014 used to define the palette
coolBlueHotRed <- function(n, alpha = 1) {
  rainbow(n, end=4/6, alpha=alpha)[n:1]
}

###########################################

library(magrittr)






##############################################################################
#                                                       #
#               change top teams var to string and then facor                #
#                                                       #
##############################################################################

mydata$top_teams <- if_else(mydata$top_teams==0, 'Rest', 'Top Teams')


mydata$top_teams <- as.factor(mydata$top_teams)







#############################################################################

         # Batting 1

#############################################################################
require(kohonen)

View(mydata)

Batting_1 <- mydata[,c(20,7,6,8,18)]


numerics_b1 <- Batting_1 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_b1 <- Batting_1 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_b1){
  data_list[[fac]] = kohonen::classvec2classmat( Batting_1[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Batting_1[,numerics_b1])




# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

map_dimension = find_grid_size(dim(Batting_1)[1])
#map_dimension = 11

epochs = 2000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_b1, 'numerics')
                  ,dist.fcts = distances
```

```
         ,keep.data = TRUE)

plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)


################################################################################

# Base Running

################################################################################
View(mydata)

BaseRun_1 <- mydata[,c(9,10,18)]


numerics_br1 <- BaseRun_1 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_br1 <- BaseRun_1 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_br1){
  data_list[[fac]] = kohonen::classvec2classmat(BaseRun_1[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(BaseRun_1[,numerics_br1])
```

```
# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

map_dimension = find_grid_size(dim(BaseRun_1)[1])


epochs = 4000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_br1, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)

plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)




################################################################################
```

```
# Pitching 1

################################################################################
View(mydata)

Pitching_1 <- mydata[,c(12,14,15,18)]


numerics_p1 <- Pitching_1 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_p1 <- Pitching_1 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_p1){
  data_list[[fac]] = kohonen::classvec2classmat( Pitching_1[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Pitching_1[,numerics_p1])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

map_dimension = find_grid_size(dim(Pitching_1)[1])
#map_dimension = 12

epochs = 2000
set.seed(123)
```

```
# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_p1, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)


plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)



##############################################################################

# Pitching 2 - HRs allowed in place of SOs

##############################################################################
View(mydata)

Pitching_2 <- mydata[,c(12,14,13,26)]


numerics_p2 <- Pitching_2 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_p2 <- Pitching_2 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_p2){
  data_list[[fac]] = kohonen::classvec2classmat( Pitching_2[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Pitching_2[,numerics_p2])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

map_dimension = find_grid_size(dim(Pitching_2[1]))


epochs = 2000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_p2, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)
```

```
plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)









################################################################################

# Fielding 1

################################################################################
View(mydata)

Fielding_1 <- mydata[,c(16,17,26)]


numerics_f1 <- Fielding_1 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_f1 <- Fielding_1 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_f1){
  data_list[[fac]] = kohonen::classvec2classmat(Fielding_1[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Fielding_1[,numerics_f1])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

map_dimension = find_grid_size(dim(Fielding_1)[1])


epochs = 2000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_f1, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)


plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)
```

```
plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)
```

```
################################################################################

# Offensive Combined 1
#  hits, BBs, SBs
################################################################################

Combined_1 <- mydata[,c(3,7,9,18)]


numerics_c1 <- Combined_1 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_c1 <- Combined_1 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_c1){
  data_list[[fac]] = kohonen::classvec2classmat(Combined_1[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Combined_1[,numerics_c1])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

#map_dimension = find_grid_size(dim(Combined_1)[1])
map_dimension = 8

epochs = 2000
set.seed(123)

# create a grid onto which the  som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_c1, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)

plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)
plot(cc_som, type = "codes", palette.name = coolBlueHotRed)






################################################################################

# Combined 2 Pitching and Fielding
# BBs, Errors, DP

################################################################################
View(mydata)

Combined_2 <- mydata[,c(14,15,16,17,26)]


numerics_c2 <- Combined_2 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_c2 <- Combined_2 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_c2){
  data_list[[fac]] = kohonen::classvec2classmat(Combined_2[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Combined_2[,numerics_c2])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')
```

```
# Determine map dimensions

#map_dimension = find_grid_size(dim(Combined_2)[1])
map_dimension = 9

epochs = 2000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_c2, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)

plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)




################################################################################

# Combined 3
# batting BBs, batting HRs, SBs, Pitching BBs,

################################################################################
```

```
View(mydata)

Combined_3 <- mydata[,c(6,7,9,14,17, 26)]


numerics_c3 <- Combined_3 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_c3 <- Combined_3 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_c3){
  data_list[[fac]] = kohonen::classvec2classmat(Combined_3[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Combined_3[,numerics_c3])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

map_dimension = find_grid_size(dim(Combined_2)[1])
#map_dimension = 9

epochs = 2000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")
```

```
# train the SOM
cc_som = supersom(data_list
                ,grid = som_grid
                ,rlen = epochs
                ,alpha = c(0.1, 0.01)
                ,whatmap = c(factors_c3, 'numerics')
                ,dist.fcts = distances
                ,keep.data = TRUE)

plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)


###############################################################################

# Combined 4
# Pitching BBs, Good Defense, Strike out Delta, Pitching BBs,

###############################################################################


Combined_4 <- mydata[,c(7,33,14, 37,18)]


numerics_c4 <- Combined_4 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_c4 <- Combined_4 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()
```

```
# create a layer for each factor
for (fac in factors_c4){
  data_list[[fac]] = kohonen::classvec2classmat(Combined_4[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Combined_4[,numerics_c4])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

#map_dimension = find_grid_size(dim(Combined_4)[1])
map_dimension = 7

epochs = 2000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_c4, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)

plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
```

```
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)


###############################################################################

# Combined 5
# OBP Good Defense, Base Running Success, Pitching BBs,

###############################################################################


Combined_5 <- mydata[,c(21,26,24,15,18)]


numerics_c5 <- Combined_5 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_c5 <- Combined_5 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_c5){
  data_list[[fac]] = kohonen::classvec2classmat(Combined_5[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Combined_5[,numerics_c5])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')
```

```
# Determine map dimensions

map_dimension = find_grid_size(dim(Combined_5)[1])
#map_dimension = 12

epochs = 2000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_c5, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)

plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)




###################################################################################
```

```
# Combined 6
# OBP, defense, base run success, pitching SO

################################################################################


Combined_6 <- mydata[,c(21,26,24,15,18)]


numerics_c6 <- Combined_6 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_c6 <- Combined_6 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_c6){
  data_list[[fac]] = kohonen::classvec2classmat(Combined_6[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Combined_6[,numerics_c6])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

map_dimension = find_grid_size(dim(Combined_6)[1])
#map_dimension = 12

epochs = 2000
set.seed(123)
```

```
# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_c6, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)

plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)



################################################################################

# Combined
# Total Bases, Errors, Pitching BBs

################################################################################


Combined_7 <- mydata[,c(9,10,16,17,18)]


numerics_c7 <- Combined_7 %>%
  select_if(is.numeric) %>%
  names

# find all columns having factors
factors_c7 <- Combined_7 %>%
  select_if(is.factor) %>%
  names

data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors_c7){
  data_list[[fac]] = kohonen::classvec2classmat(Combined_7[[fac]] )
  distances = c(distances, 'tanimoto')
}


# Center and scale

data_list[['numerics']] = scale(Combined_7[,numerics_c7])
#View(data_list)


# calc distances
distances = c(distances, 'sumofsquares')



# Determine map dimensions

map_dimension = find_grid_size(dim(Combined_7)[1])
#map_dimension = 12

epochs = 2000
set.seed(123)

# create a grid onto which the Batting 1 som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")



# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors_c7, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE)
```

```
plot(cc_som, type = "changes")

plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

plot(cc_som, type = "codes", palette.name = coolBlueHotRed)









##################################################################################

#            Top 10% Histograms of Important Vars                        #

##################################################################################

p1 <- hist(subset(mydata$TEAM_BATTING_BB, mydata$top_teams == 1))
p2 <- hist(subset(mydata$TEAM_BATTING_BB, mydata$top_teams == 0))
plot(p2, col = "blue", main= "Batting Walks for Top 10 and Rest", xlab= "Batting BBs")
plot(p1, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)




p3 <- hist(subset(mydata$TEAM_BATTING_HR, mydata$top_teams == 1))
p4 <- hist(subset(mydata$TEAM_BATTING_HR, mydata$top_teams == 0))
plot(p4, col = "blue", main= "Batting HRs for Top 10 and Rest", xlab= "Batting HRs")
plot(p3, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)


p5 <- hist(subset(mydata$TEAM_FIELDING_E, mydata$top_teams == 1))
p6 <- hist(subset(mydata$TEAM_FIELDING_E, mydata$top_teams == 0))
plot(p6, col = "blue", main= "Fielding Errors for Top 10 and Rest", xlab= "Errors")
plot(p5, col= "red", add = T)
```

```
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)



p7 <- hist(subset(mydata$TEAM_FIELDING_DP, mydata$top_teams == 1))
p8 <- hist(subset(mydata$TEAM_FIELDING_DP, mydata$top_teams == 0))
plot(p8, col = "blue", main= "Fielding Double Plays for Top 10 and Rest", xlab= "Double Plays Turned")
plot(p7, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)



p9 <- hist(subset(mydata$TEAM_PITCHING_BB, mydata$top_teams == 1))
p10 <- hist(subset(mydata$TEAM_PITCHING_BB, mydata$top_teams == 0))
plot(p10, col = "blue", main= "Pitching Walks Issued for Top 10 and Rest", xlab= "Walks Issued")
plot(p9, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)


p11 <- hist(subset(mydata$TEAM_PITCHING_SO, mydata$top_teams == 1))
p12 <- hist(subset(mydata$TEAM_PITCHING_SO, mydata$top_teams == 0))
plot(p12, col = "blue", main= "Pitching Strikeouts for Top 10 and Rest", xlab= "Strikeouts Thrown")
plot(p11, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)

p13 <- hist(subset(mydata$Base_Runners, mydata$top_teams == 1))
p14 <- hist(subset(mydata$Base_Runners, mydata$top_teams == 0))
plot(p14, col = "blue", main= "Base Runners for Top 10 and Rest", xlab= "Base Runners")
plot(p13, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)



p15 <- hist(subset(mydata$Total_bases, mydata$top_teams == 1))
p16 <- hist(subset(mydata$Total_bases, mydata$top_teams == 0))
plot(p16, col = "blue", main= "Batter Total Bases Top 10 and Rest", xlab= "Total Bases")
plot(p15, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)




p17 <- hist(subset(mydata$Bat_OBP, mydata$top_teams == 1))
p18 <- hist(subset(mydata$Bat_OBP, mydata$top_teams == 0))
plot(p18, col = "blue", main= "On Base Percentage Top 10 and Rest Alt", xlab= "OBP")
plot(p17, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)
```

```
p19 <- hist(subset(mydata$Pitch_OBP, mydata$top_teams == 1))
p20 <- hist(subset(mydata$Pitch_OBP, mydata$top_teams == 0))
plot(p20, col = "blue", main= "Pitching On Base Percentage Top 10 and Rest Alt", xlab= "OBP")
plot(p19, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)



p21 <- hist(subset(mydata$bat_non_SO_outs, mydata$top_teams == 1))
p22 <- hist(subset(mydata$bat_non_SO_outs, mydata$top_teams == 0))
plot(p22, col = "blue", main= "Batting Non Strikeout Outs (contact outs)", xlab= "Outs other than SOs")
plot(p21, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)


p23 <- density(subset((mydata$EBHs+mydata$TEAM_BATTING_HR), mydata$top_teams == 1))
p24 <- density(subset((mydata$EBHs+mydata$TEAM_BATTING_HR), mydata$top_teams == 0))
plot(p24, col = "blue", main= "Batting Extra Base Hits", xlab= "Extra Base Hits")
plot(p23, col= "red", add = T)
legend("topright", legend = c("Top 10%", "Rest"), col=c("red", "blue"),pch=19)



library(sm)


sm.density.compare(mydata$TEAM_BATTING_HR, mydata$top_teams, xlab=" Batting HRs")
title(main="HRs")
colfill<-c(2:(2+length(levels(mydata$top_teams))))
legend(locator(1), levels(mydata$top_teams), fill=colfill)

#plot Batting Walks

ggplot(mydata, aes(x=TEAM_BATTING_BB,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting BBs Distribution')

ggplot(mydata, aes(x=TEAM_BATTING_BB,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting BBs Distribution')

#plot Batting HRs

ggplot(mydata, aes(x=TEAM_BATTING_HR,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting HRs Distribution')

ggplot(mydata, aes(x=TEAM_BATTING_HR,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting HRs Distribution')


#plot Batting Hits

ggplot(mydata, aes(x=TEAM_BATTING_H,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting Hits Distribution')

ggplot(mydata, aes(x=TEAM_BATTING_H,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting Hits Distribution')


#plot Caught Stealing

ggplot(mydata, aes(x=TEAM_BASERUN_CS,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Caught Stealing Distribution')

ggplot(mydata, aes(x=TEAM_BASERUN_CS,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Caught Stealing Distribution')



#plot Pitchin BBs

ggplot(mydata, aes(x=TEAM_PITCHING_BB,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Pitching BBs Allowed Distribution')

ggplot(mydata, aes(x=TEAM_PITCHING_BB,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Pitching BBs Allowed Distribution')

#plot Pitchin SOs

ggplot(mydata, aes(x=TEAM_PITCHING_SO,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Pitching SOs Distribution')

ggplot(mydata, aes(x=TEAM_PITCHING_SO,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Pitching Sos Allowed Distribution')

#plot Pitchin HRs

ggplot(mydata, aes(x=TEAM_PITCHING_HR,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Pitching HRs allowed Distribution')

ggplot(mydata, aes(x=TEAM_PITCHING_HR,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Pitching HRs Allowed Distribution')

#plot Pitchin Hits
```

```
ggplot(mydata, aes(x=TEAM_PITCHING_H,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Pitching Hits allowed Distribution')

ggplot(mydata, aes(x=TEAM_PITCHING_H,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Pitching Hits Allowed Distribution')



#plot Fielding  Errors

ggplot(mydata, aes(x=TEAM_FIELDING_E,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Fielding Errors Distribution')

ggplot(mydata, aes(x=TEAM_FIELDING_E,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Fielding Errors Distribution')


#plot DPs

ggplot(mydata, aes(x=TEAM_FIELDING_DP,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Double Plays Distribution')

ggplot(mydata, aes(x=TEAM_FIELDING_DP,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Double Plays Distribution')

#plot DPs- Es

ggplot(mydata, aes(x=(TEAM_FIELDING_DP-TEAM_FIELDING_E),  fill=top_teams)) +
geom_density(alpha = 0.2)+
  ggtitle('Double Plays minus Errors')

ggplot(mydata, aes(x=(TEAM_FIELDING_DP-TEAM_FIELDING_E),  fill=top_teams)) +
geom_histogram(alpha = 0.4)+
  ggtitle('Double Plays minus Errors')


#plot Total Bases

ggplot(mydata, aes(x=Total_bases,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting Total Bases')

ggplot(mydata, aes(x=Total_bases,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting Total Bases')
```

```
#plot OBP

ggplot(mydata, aes(x=Bat_OBP,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting On Base Percentage')

ggplot(mydata, aes(x=Bat_OBP,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting On Base Percentages')

#plot OBP

ggplot(mydata, aes(x=Bat_OBP,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting On Base Percentage')

ggplot(mydata, aes(x=Bat_OBP,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting On Base Percentages')

#plot Pitching OBP

ggplot(mydata, aes(x=Pitch_OBP,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Pitching On Base Percentage')

ggplot(mydata, aes(x=Pitch_OBP,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Pitching On Base Percentages')




#plot Non Strike Out Outs

ggplot(mydata, aes(x=bat_non_SO_outs,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting Non Strike Out Outs')

ggplot(mydata, aes(x=bat_non_SO_outs,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting Non Strike Out Outs')


#plot walk to so ratio

ggplot(mydata, aes(x=Bat_BB_SO_Rat,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting Walk to SO ratio')

ggplot(mydata, aes(x=Bat_BB_SO_Rat,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting Walk to SO ratio')

#plot walk and Hits to so ratio

ggplot(mydata, aes(x=Bat_BBHit_SO_Rat,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting Walk and HRs to SO ratio')

ggplot(mydata, aes(x=Bat_BBHit_SO_Rat,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting Walk and HRs to SO ratio')


#plot singles

ggplot(mydata, aes(x=Sinlges,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting Singles')

ggplot(mydata, aes(x=Sinlges,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Batting Singles')

#plot HRs + BB

ggplot(mydata, aes(x=(TEAM_BATTING_HR+TEAM_BATTING_BB),  fill=top_teams)) +
geom_density(alpha = 0.2)+
  ggtitle('Batting Singles')

ggplot(mydata, aes(x=(TEAM_BATTING_HR+TEAM_BATTING_BB),  fill=top_teams)) +
geom_histogram(alpha = 0.4)+
  ggtitle('Batting Singles')



#plot HRs + BB

ggplot(mydata, aes(x=TEAM_BATTING_HR+TEAM_BATTING_BB,  fill=top_teams)) +
geom_density(alpha = 0.2)+
  ggtitle('Walks plus HRs')

ggplot(mydata, aes(x=TEAM_BATTING_HR+TEAM_BATTING_BB,  fill=top_teams)) +
geom_histogram(alpha = 0.4)+
  ggtitle('Walks plus HRs')


#plot sbs

ggplot(mydata, aes(x=TEAM_BASERUN_SB,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Stolen Bases')

ggplot(mydata, aes(x=TEAM_BASERUN_SB,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Stolen Bases')
```

```
#plot sbs

ggplot(mydata, aes(x=TEAM_BASERUN_CS,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Caught Stealing')

ggplot(mydata, aes(x=TEAM_BASERUN_SB,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Stolen Bases')


#plot SLG

ggplot(mydata, aes(x=(TEAM_BATTING_BB+TEAM_BATTING_H-TEAM_BATTING_SO),
fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Walks + H - SO')

ggplot(mydata, aes(x=(TEAM_BATTING_BB+TEAM_BATTING_H-TEAM_BATTING_SO),
fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Walks + H - SO')


#plot SO less HR

ggplot(mydata, aes(x=Base_run_Success,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Base Running Success Rate')

ggplot(mydata, aes(x=Base_run_Success,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Base Running Success Rate')


#plot Defenst
ggplot(mydata, aes(x=Defense,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Defensive Prowess')

ggplot(mydata, aes(x=Defense,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Defensive Prowess')

#plot Delta SOs
ggplot(mydata, aes(x=SO_Delta,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Delta SOs')

ggplot(mydata, aes(x=SO_Delta,  fill=top_teams)) + geom_histogram(alpha = 0.4)+
  ggtitle('Delta SOs')

#plot SOs + HRs
ggplot(mydata, aes(x=Bat_OBP,  fill=top_teams)) + geom_density(alpha = 0.2)+
  ggtitle('Batting OBP')
```

```
ggplot(mydata, aes(x=(TEAM_BATTING_SO+TEAM_BATTING_HR),  fill=top_teams)) +
geom_histogram(alpha = 0.4)+
  ggtitle('Batting SOs + Batting HRs')



boxplot(mydata$TEAM_BATTING_BB~mydata$top_teams, col='dodgerblue', main ='Batting Walks
Top 10% vs Rest')

boxplot(mydata$TEAM_BATTING_HR~mydata$top_teams, col='dodgerblue', main ='Batting HR')

boxplot(mydata$Bat_OBP~mydata$top_teams, col='dodgerblue', main ='On Base Percentage')

boxplot(mydata$TEAM_PITCHING_SO~mydata$top_teams, col='dodgerblue', main ='Pitching Strike
Outs')

boxplot(mydata$TEAM_PITCHING_H~mydata$top_teams, col='dodgerblue', main ='Pitching Hits')

boxplot(mydata$TEAM_PITCHING_BB~mydata$top_teams, col='dodgerblue', main ='Pitching Walks')

boxplot(mydata$Bat_OBP~mydata$top_teams, col='dodgerblue', main =' Batting On Base Percentage')

boxplot(mydata$Pitch_OBP~mydata$top_teams, col='dodgerblue', main =' Pitching On Base Percentage')

boxplot(mydata$TEAM_PITCHING_HR~mydata$top_teams, col='dodgerblue', main =' Pitching HRs
Given Up')

boxplot(mydata$Bat_HRSO_rat~mydata$top_teams, col='dodgerblue', main =' Batting HR to SO ratio')

boxplot(mydata$Pitch_HRSO_rat~mydata$top_teams, col='dodgerblue', main =' Pitch HR to SO ratio')

boxplot(mydata$Bat_HRBB_rat~mydata$top_teams, col='dodgerblue', main =' Batting HR to BB ratio')

boxplot(mydata$Pitch_HRBB_rat~mydata$top_teams, col='dodgerblue', main =' Pitch HR to BB ratio')

plot(mydata$Total_bases~mydata$TEAM_BATTING_BB, col='dodgerblue', main ='Batting Total Bases
vs Walks')
```