Data 621 Homework 1

Critical Thinking Group 3: Vyannna Hill, Jose Rodriguez, and Christian Uriostegui

2023-11-06

Introduction of the MoneyBall game statistics from 1871 to 2006

For this analysis a Multiple Linear Regression Model (MLR) will be built. The objective is to predict how many games are won, in a given season, based on baseball game event metrics. The baseball game events include batting, strikeouts, fielding, and pitching. Each variable can have a positive or a negative influence on the number of wins for the season. For more details on variables, see section **1.1, About the Dataset**.

The final report will include the following sections:
1. Data Exploration: high-level statistical information about the training data set. This includes, but is not limited to variable distributions, correlations, visualizations and completeness of the data.

2. Data Preparation: describes steps and techniques used to transform the data.

3. Building models: will report on several models, its accuracy, and steps taken to improve it.

4. Model selection: will outline the best model and why it was chosen among all the different alternatives.

1. Data Exploration

For building linear regression models, data exploration is usually the first step. Its a best practice which allows scientists to find discrepancies in the dataset before diving into model building. Moreover, the data quality can be quantified and visualized. The results are then used to formulate the model-building approach.

1.1 About the Dataset

The provided dataset contains two files in CSV format. One for training the MLR model, and one for generating predictions.

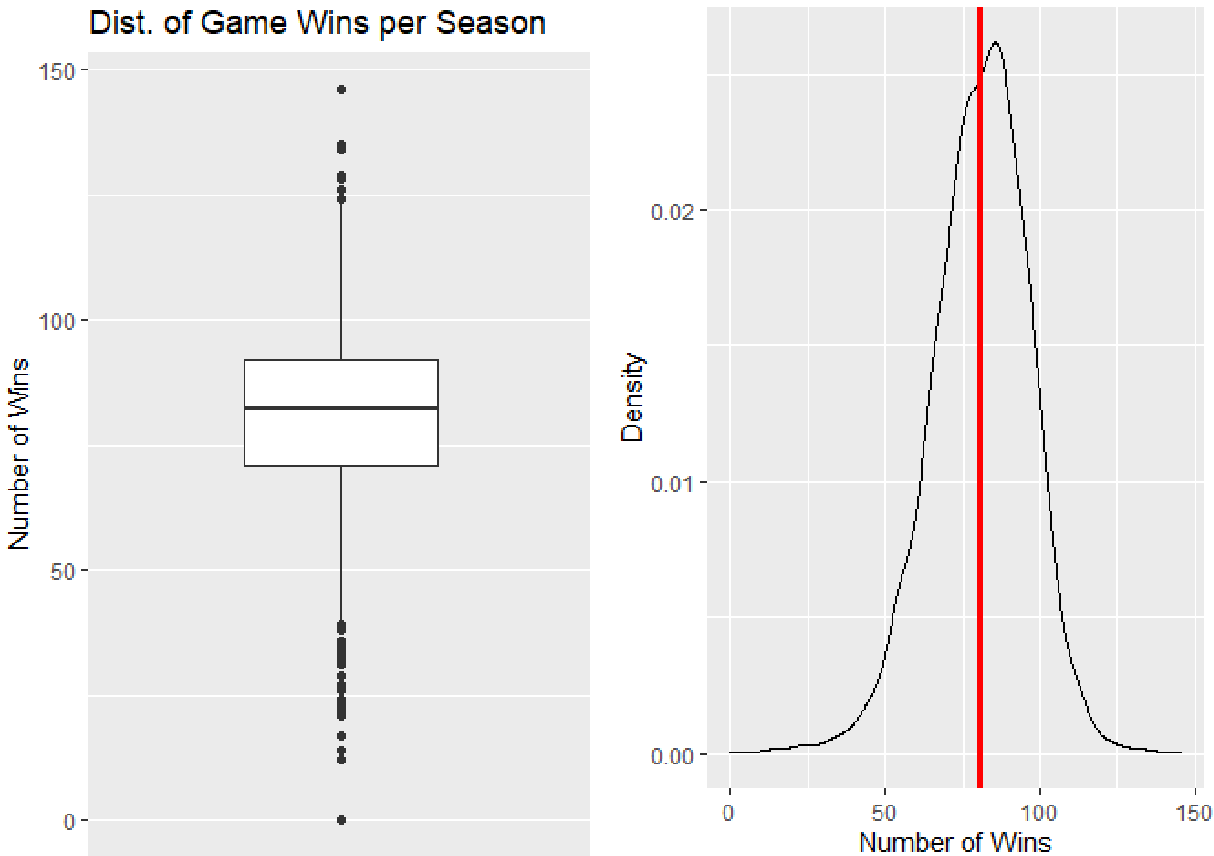
The following variables are found in the dataset:

Variable Name	Definition	Theoretical Effect
INDEX	Identification Variable (do not use)	
TARGET_WINS	Number of wins	Response 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

Variable Name	Definition	Theoretical Effect
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

1.2 Descriptive Statistical Analysis

On average, teams saw 80 wins in a given season. The most prolific season saw a high of 146, whereas the least saw a value of 0. One path of preemptive checks on normality is to view the binomial distribution of the response variable. The distribution for TARGET_WINS, the response variable for this analysis, appears to be normally distributed. It can be inferred that it is a good candidate for a linear regression model. It should be noted there is a slight dip in density distribution around 70 wins. This could reveal an issue with the unprocessed data set.



1.3 Data Wrangling Pre-inspection:

```
summary(training_set)
```

```
##      INDEX         TARGET_WINS     TEAM_BATTING_H TEAM_BATTING_2B
##  Min.   :   1.0   Min.   :  0.00   Min.   : 891   Min.   : 69.0  
##  1st Qu.: 630.8   1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0  
##  Median :1270.5   Median : 82.00   Median :1454   Median :238.0  
##  Mean   :1268.5   Mean   : 80.79   Mean   :1469   Mean   :241.2  
##  3rd Qu.:1915.5   3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0  
##  Max.   :2535.0   Max.   :146.00   Max.   :2554   Max.   :458.0  
##                                                                  
##  TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_S
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.0   Min.   :   0.0
##  1st Qu.: 34.00   1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 548.0
##  Median : 47.00   Median :102.00   Median :512.0   Median : 750.0
##  Mean   : 55.25   Mean   : 99.61   Mean   :501.6   Mean   : 735.6
##  3rd Qu.: 72.00   3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 930.0
##  Max.   :223.00   Max.   :264.00   Max.   :878.0   Max.   :1399.0
##                                                    NA's   :102   
##  TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H
```

```
##   Min.    :  0.0   Min.    :  0.0   Min.    :29.00    Min.    : 1137
##   1st Qu.: 66.0   1st Qu.: 38.0   1st Qu.:50.50    1st Qu.: 1419
##   Median :101.0   Median : 49.0   Median :58.00    Median : 1518
##   Mean   :124.8   Mean   : 52.8   Mean   :59.36    Mean   : 1779
##   3rd Qu.:156.0   3rd Qu.: 62.0   3rd Qu.:67.00    3rd Qu.: 1682
##   Max.   :697.0   Max.   :201.0   Max.   :95.00    Max.   :30132
##   NA's   :131     NA's   :772     NA's   :2085
##   TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDIN
##   Min.   :  0.0    Min.   :   0.0   Min.   :    0.0   Min.   :  65
##   1st Qu.: 50.0    1st Qu.: 476.0   1st Qu.:  615.0   1st Qu.: 127
##   Median :107.0    Median : 536.5   Median :  813.5   Median : 159
##   Mean   :105.7    Mean   : 553.0   Mean   :  817.7   Mean   : 246
##   3rd Qu.:150.0    3rd Qu.: 611.0   3rd Qu.:  968.0   3rd Qu.: 249
##   Max.   :343.0    Max.   :3645.0   Max.   :19278.0   Max.   :1898
##                                     NA's   :102
##   TEAM_FIELDING_DP
##   Min.   : 52.0
##   1st Qu.:131.0
##   Median :149.0
##   Mean   :146.4
##   3rd Qu.:164.0
##   Max.   :228.0
##   NA's   :286
```

1.3.1 Inspecting for null values:

While examining the dataset it was found that several variables have a large count of NA values. The largest unaccounted amount of values fall with batters hit by the pitchers. We do not have insight from the survey team to deduce if these NAs reflect no values recorded or a human imputation error. This will have to be sorted for the analysis.

1. Are there missing values in the dataset?

```
## [1] TRUE
```

2. How many? What is the proportion of missing values?

```
## [1] 3478
```

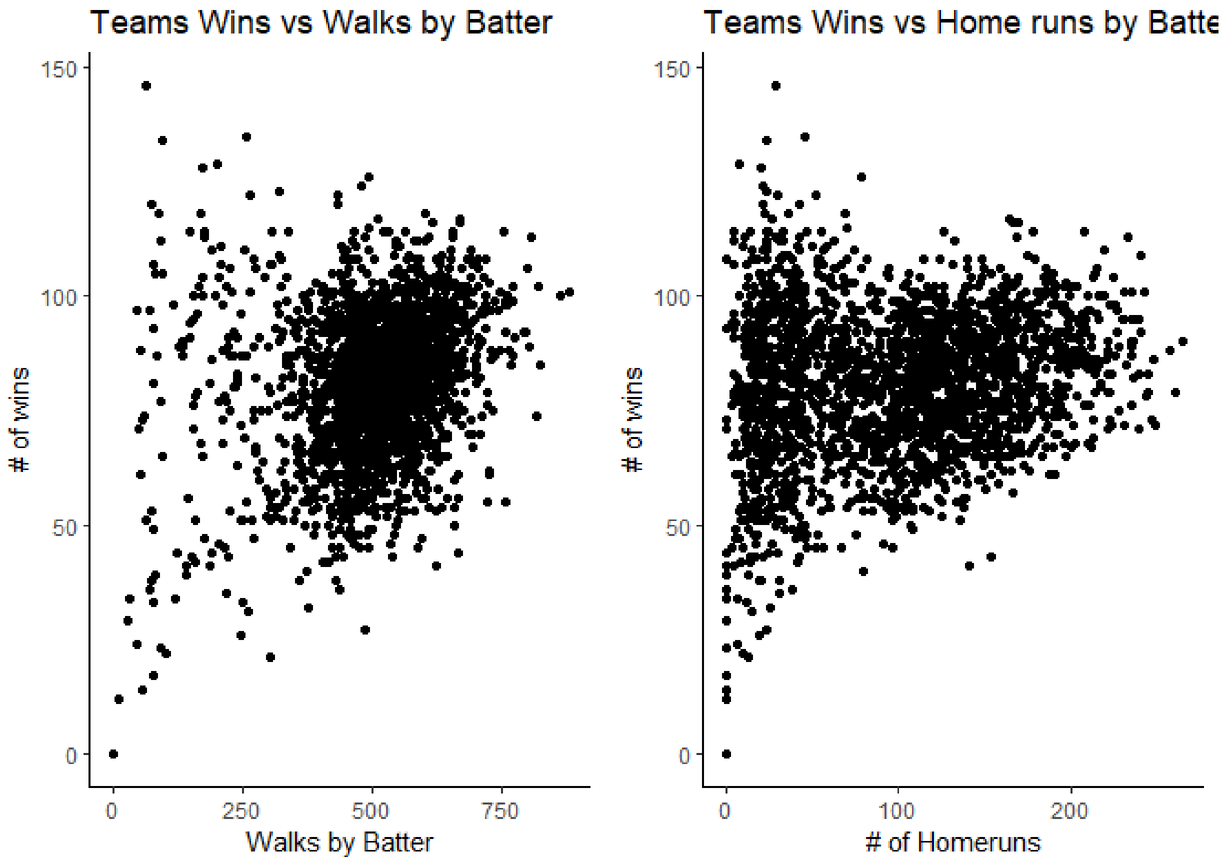
```
## [1] 0.08988938
```

3. Which variables are affected? Which contain the most missing values?

```
##            INDEX      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_
##                0                0                0
##  TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_
##                0                0                0              1
##  TEAM_BASERUN_SB  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_
##              131              772             2085
## TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_
##                0                0              102
## TEAM_FIELDING_DP
##              286
```

1.4 Investigating Relationships

Two possible predictor values were selected to plot against TARGET_WINS. TEAM_BATTING_BB (Walks by Batter), and TEAM_BATTING_HR (Home runs by Batter). There is concern for both variables as they do not appear to have a linear relationship with TARGET_WINS. Ultimately, this can be influenced by another variables.??



When visualizing correlation strength between target wins and the other predictor variables, we can see that there aren't many significant relationships. The batting variables all have positive correlations with Target Wins. Some of the pitching and fielding variables have negative correlations. A trend we're seeing is that the offense variables, which include batting, lead to more wins while some of the defensive stats can negatively affect wins. When creating our models, it may be worth creating them with this in mind. Some of the standout pairings are listed below.

1. Target Wins & Team Batting Hits TARGET_WINS & TEAM_BATTING_H have a moderately positive correlation of 0.39 which suggests that teams with more batting hits have more wins

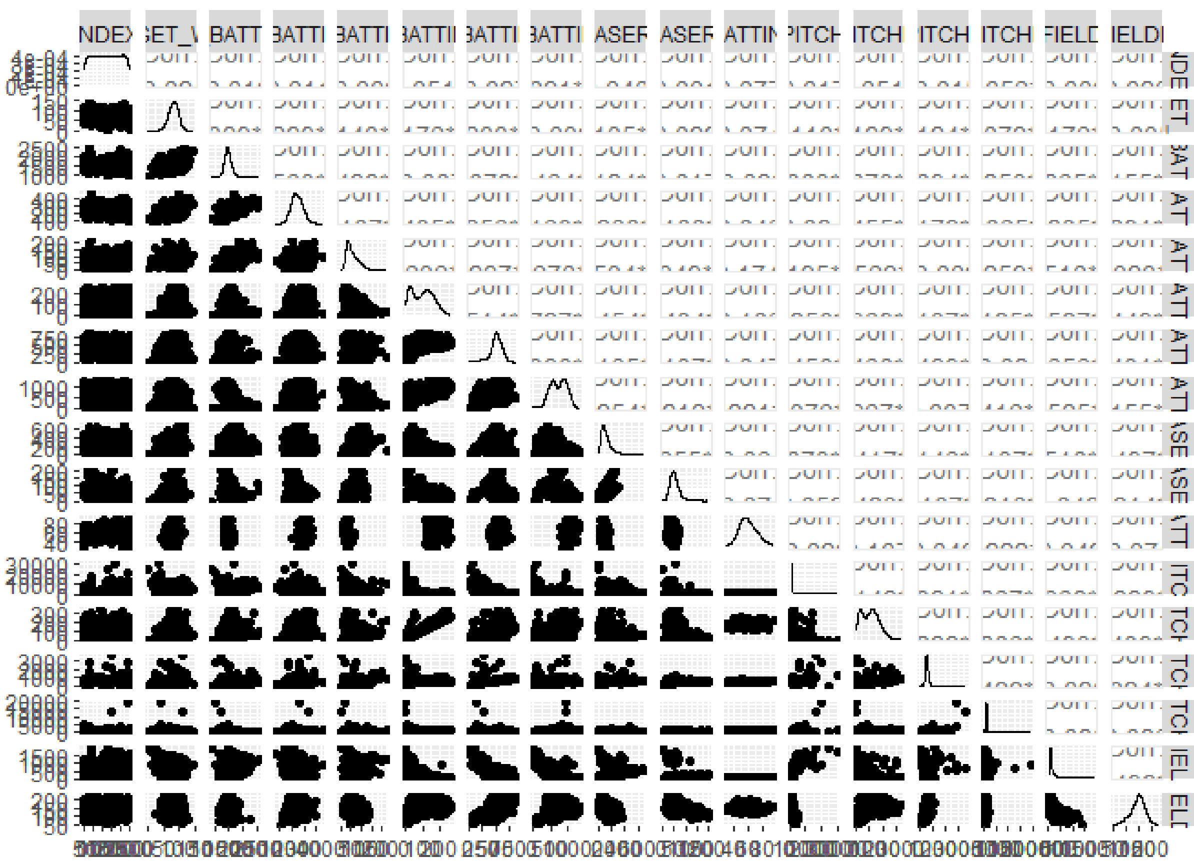
2. Target Wins & Team Batting Doubles TARGET_WINS & TEAM_BATTING_2B have a weak positive relationship with a correlation of 0.29. Teams with more batting doubles will have slightly more wins

3. Target Wins & Team Batting Walks TARGET_WINS & TEAM_BATTING_BB have a weak positive correlation of 0.23. Teams

with more batting walks have slightly more wins

4. Target Wins & Team Errors TARGET_WINS & TEAM_FIELDING_E have a weak negative correlation of 0.18. Teams with more fielding errors will have slightly less wins.

5. Target Wins & Team Pitching H TARGET WINS & TEAM_PITCHING_H has a weak negative correlation of -0.11. Teams with more hits allowed will have slightly less wins.



2. Data Preparation

From the previous section, there is no indication that NAs reflect zeros in the dataset. As such, they will be assumed to be missing values. Missing values can be handled with two common techniques.

The first technique consists of completely removing all data points containing any amount of NAs. In other words, this method filters to rows that are complete. This method includes all the original predictors at the risk of removing more than 90% of the dataset. A reduction from 2276 to 191 data points. This technique is not ideal as the reduced number of observations can negatively influence the model’s regression.

The alternative path consists of utilizing imputation to assign synthetic values. Imputation can take many different forms, but they all accomplish the same goal: to assume a value based on the distribution of the data. For this particular analysis, MICE is used to predict the missing values in the dataset. It involves removing predictors that pass a given threshold of randomness.

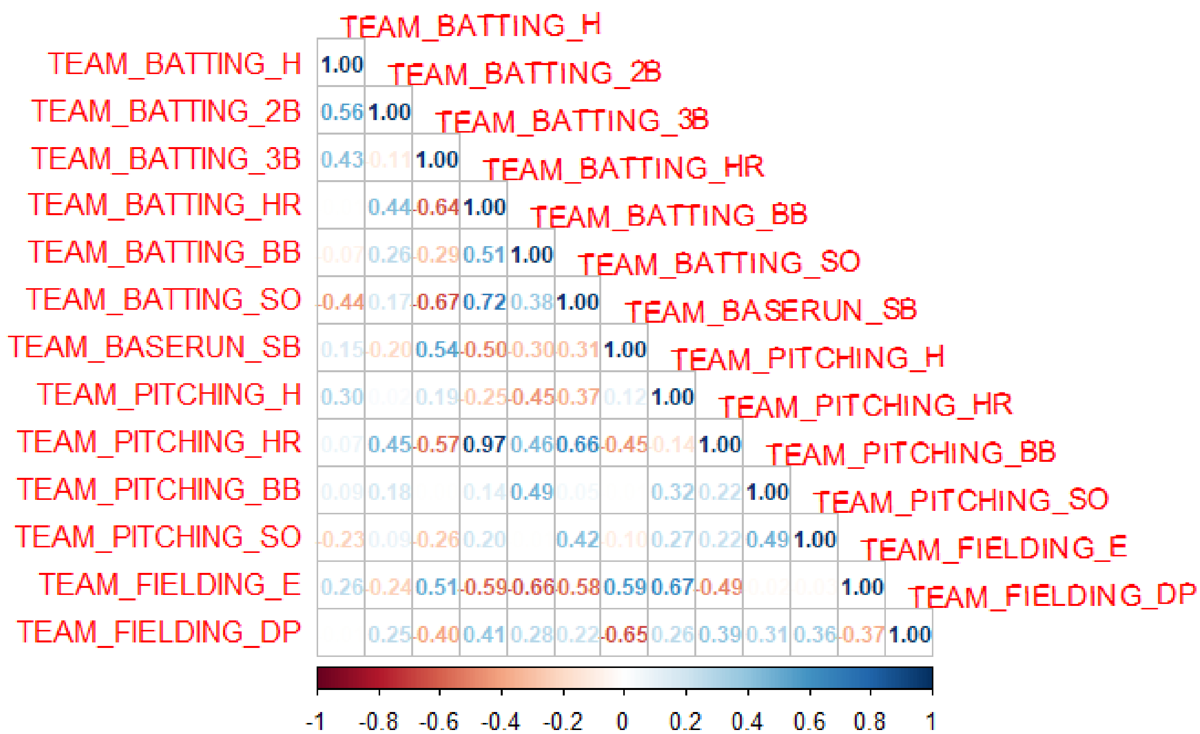
During the data exploration stage, it was found that variables “TEAM_BASERUN_CS” and “TEAM_BATTING_HBP” have the highest count of missing values. This is a strong indication that its NAs are not missing at random. It is best to remove these predictors before running the MICE model on the data set. After its removal, a model can be trained.

2.1 Using MICE Imputation

```
#Technique #2: Drop predictors HBP and CS and use MICE imputation
train.c1<-training_set %>%
  select(-c(INDEX,TEAM_BATTING_HBP,TEAM_BASERUN_CS)) #Filter dataset

train.mice<-complete(mice(train.c1,method = "lasso.norm",seed = 333)

summary(train.mice)
```



2.2 Calculating New Predictors

Although the number of observations were significantly reduced, it did open the possibility of new predictors. The MLB[1] and other baseball fanatic sites[2] provides a list of advanced statistics which expands the amount of available predictors. The new predictors introduced to the clean dataset are Strikeouts to walk ratio “STW Ratio” and Total Bases. Both predictors can be calculated with elementary arithmetic.

*New Baseball Variables +STW Ratio: The times a pitcher strikeouts over the times a batter walks to first base +Total Bases: Number of bases gain by batter by hits

```
# Found some baseball formulas here to add onto our analysis
#STW= pitchers strikeouts/ walks by batter
train.mice<-train.mice %>% mutate(STW_Ratio=TEAM_PITCHING_SO/TEAM_BA

#total bases=[H + 2B + (2 X 3B) + (3 X HR)].
train.mice<-train.mice%>%mutate(TB=TEAM_BATTING_H+TEAM_BATTING_2B+(2

#two observations had NAs
```

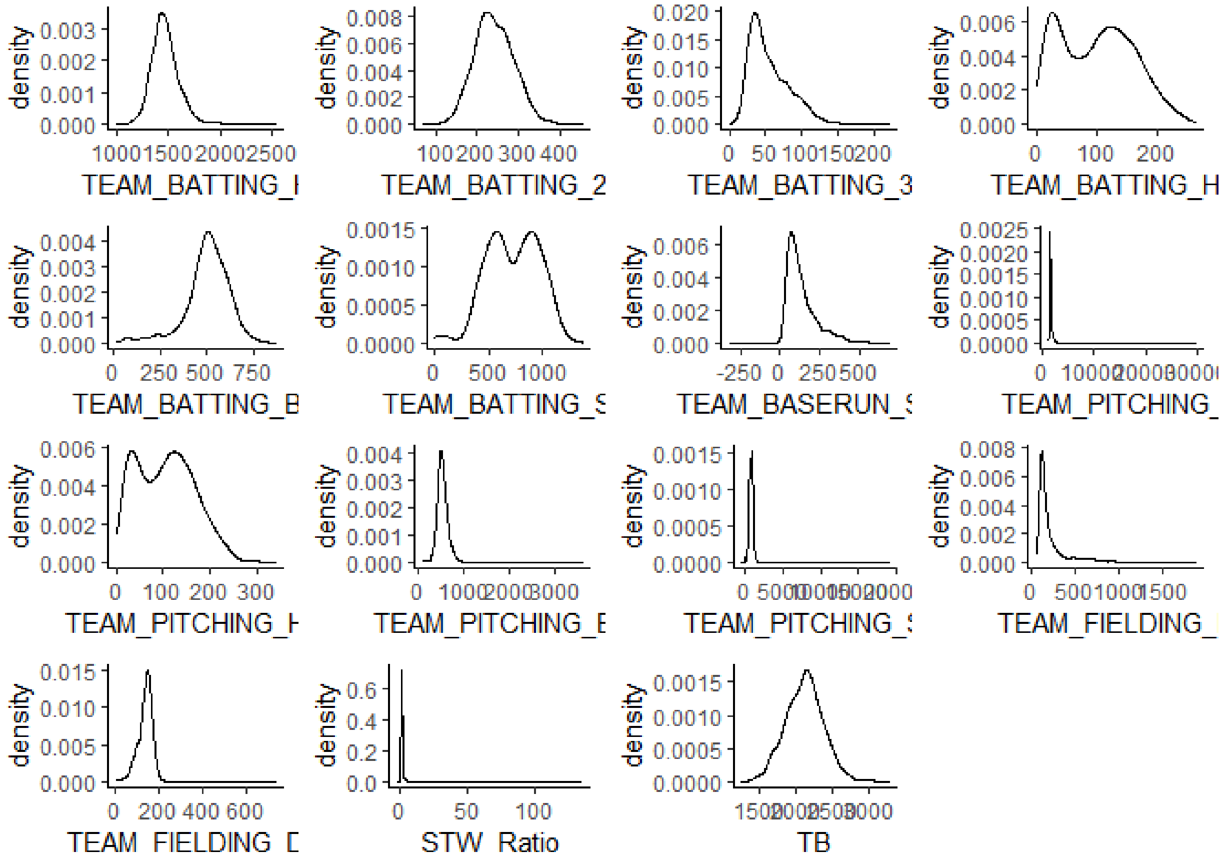
```r
train.mice<-na.omit(train.mice)

#review new columns
head(train.mice,3)
```

```
##   TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B TEAM_
## 1          39           1445             194              39
## 2          70           1339             219              22
## 3          86           1377             232              35
##   TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_PITCHING_H
## 1             143             842        56.62168            9364
## 2             685            1075        37.00000            1347
## 3             602             917        46.00000            1377
##   TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDIN
## 1               84              927             5456            1
## 2              191              689             1082
## 3              137              602              917
##   TEAM_FIELDING_DP STW_Ratio   TB
## 1         254.1154 38.153846 1756
## 2         155.0000  1.579562 2172
## 3         153.0000  1.523256 2090
```

2.3 Scaling the Data

Scaling the training set before analysis as the new variables are not on the same scale. Now, the training set is ready to be fitted!



3. Building Models

Now its time to create three linear models with distinct predictor variable subsets and compare the results.

3.1 Model Fit 1: Full Model

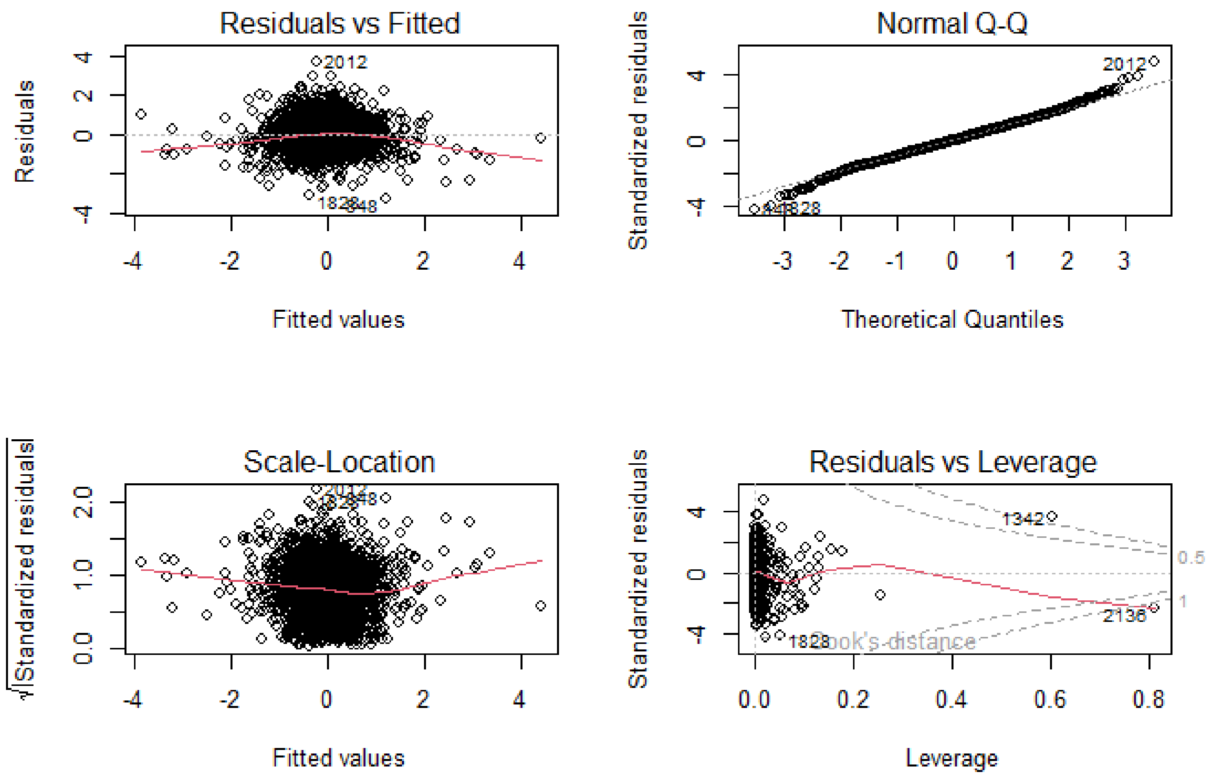
The first model will be a full model, meaning it will include all predictors against *TARGET_WINS*.

Looking at the coefficient numbers, we notice something odd. Batting variables that should theoretically have a positive effect on winning like *TEAM_BATTING_H*, *TEAM_BATTING_2B*, *TEAM_BATTING_3B*, and *TEAM_BATTING_HR* have a negative coefficient. This means for every increase in this stat, it decreases the number of wins. We see the inverse for some variables. *TEAM_PITCHING_H* or hits allowed, a stat which has a negative impact on wins, has a positive coefficient. We suspect this is due to the effect of having all the predictor values together.

This model has the most predictive values, contains 12 statistically significant variables, and has an R squared value of 0.3816.

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2718 -0.5006 -0.0076  0.5104  3.6955
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        2.250e-15  1.649e-02   0.000 1.000000
## TEAM_BATTING_H    -4.630e-01  1.752e-01  -2.643 0.008273 **
## TEAM_BATTING_2B   -3.663e-01  6.152e-02  -5.954 3.02e-09 ***
## TEAM_BATTING_3B   -1.181e-01  7.467e-02  -1.582 0.113778
## TEAM_BATTING_HR   -8.627e-01  2.289e-01  -3.768 0.000169 ***
## TEAM_BATTING_BB    2.770e-01  5.585e-02   4.960 7.56e-07 ***
## TEAM_BATTING_SO   -3.581e-01  4.185e-02  -8.557  < 2e-16 ***
## TEAM_BASERUN_SB    3.175e-01  2.871e-02  11.057  < 2e-16 ***
## TEAM_PITCHING_H    1.520e-01  4.917e-02   3.092 0.002010 **
## TEAM_PITCHING_HR   9.923e-04  9.470e-02   0.010 0.991640
## TEAM_PITCHING_BB  -1.660e-01  4.834e-02  -3.435 0.000604 ***
## TEAM_PITCHING_SO   2.819e-01  6.332e-02   4.453 8.89e-06 ***
## TEAM_FIELDING_E   -8.449e-01  4.654e-02 -18.156  < 2e-16 ***
## TEAM_FIELDING_DP  -2.941e-01  2.660e-02 -11.056  < 2e-16 ***
## STW_Ratio         -8.871e-02  5.601e-02  -1.584 0.113403
## TB                 1.475e+00  3.139e-01   4.701 2.75e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7864 on 2259 degrees of freedom
## Multiple R-squared:  0.3857, Adjusted R-squared:  0.3816
## F-statistic: 94.57 on 15 and 2259 DF,  p-value: < 2.2e-16
```

When investigating the residual plots, linearity and homoscedasticity is observed (variance is constant). In the Normal Q-Q plot it can be seen that most points adhere to the diagonal line indicating a normal distribution of residuals. In the leverage visual, we can notice a few deviated points in the model.



3.2 Model Fit 2: Modeling Batting Variables

The second model will only contain batting variables such as *TEAM_BATTING_H* and *TEAM_BATTING_2B* (offense variables). Batting variables involve scoring points for a team, therefore increasing the chances of a team winning. Theoretically, these should be strong predictors.

Unlike the first model where some of the batting variables had negative coefficients, most of them are positive here - which which aligns with what is theoretically expected.

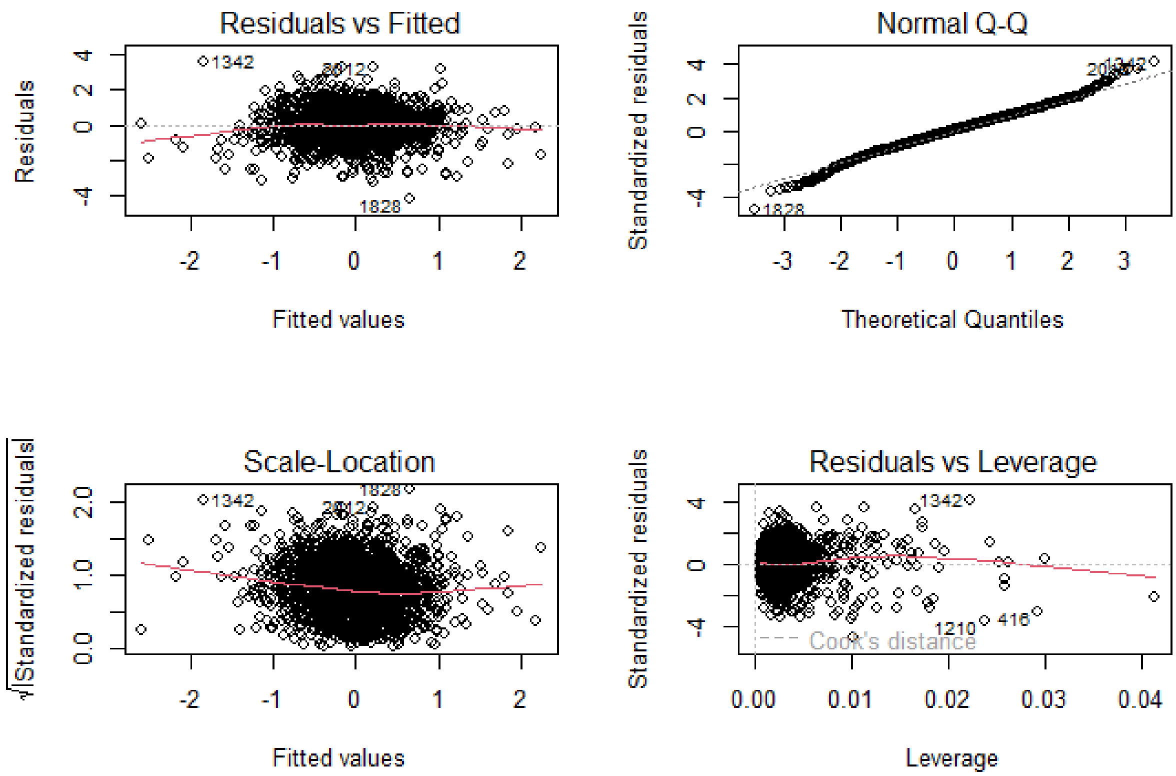
This model contains less predictor variables when compared to model 1, however it is comprised mainly of statistically significant variables. A lower R squared value is observed, meaning that fielding and pitching explains a large portion of variance.

```
#second model with just batting variables
fit2 <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BAT
```

```r
summary(fit2)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BA
##     data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1499 -0.5503  0.0316  0.5823  3.5768
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -7.608e-16  1.843e-02   0.000   1.0000
## TEAM_BATTING_H   3.750e-01  3.502e-02  10.709  < 2e-16 ***
## TEAM_BATTING_2B -3.871e-02  2.840e-02  -1.363   0.1731
## TEAM_BATTING_3B  1.735e-01  2.933e-02   5.916 3.80e-09 ***
## TEAM_BATTING_HR  1.464e-01  3.745e-02   3.910 9.52e-05 ***
## TEAM_BATTING_BB  2.068e-01  2.187e-02   9.456  < 2e-16 ***
## TEAM_BATTING_SO  5.904e-02  3.571e-02   1.653   0.0984 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.879 on 2268 degrees of freedom
## Multiple R-squared:  0.2294, Adjusted R-squared:  0.2273
## F-statistic: 112.5 on 6 and 2268 DF,  p-value: < 2.2e-16
```

Though this model meets linearity and homoscedasticity, the first model appears to be more linear. In the QQ plot, the residual points mostly fall on the diagonal line, with both tails slightly deviating away from it. We can observe a few outlier points which



3.3 Model Fit 3: Modeling Pitching and Fielding Variables

The third model will only contain pitching and fielding stats (defensive variables). Outside of batting offense, pitching is important because it can limit the scoring of the opposing team. Similarly, the less Fielding Errors - which can give up scoring opportunities - the higher chances of winning. Fielding Double Plays - which is the ability to achieve two outs in a defensive play - can also increase the chances of a win. This model can also potentially give us a high winning percentage.

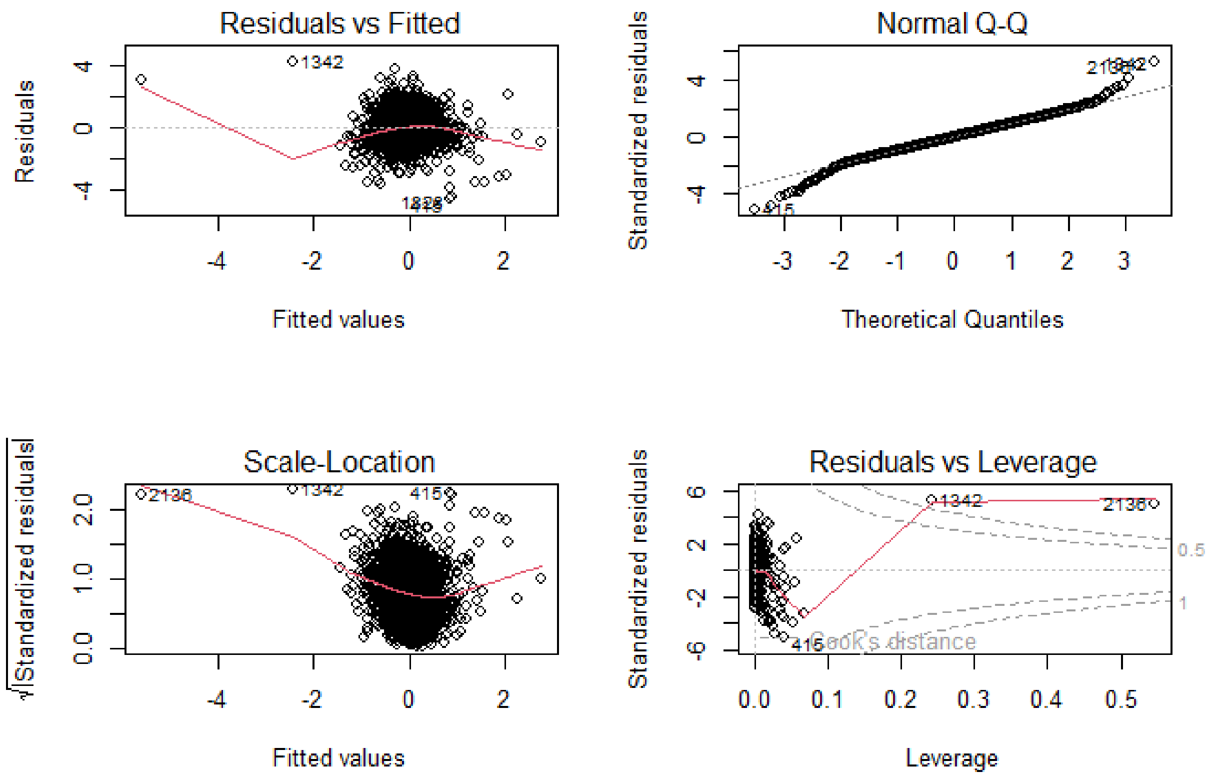
```
#third model with pitching and fielding variables
fit3 <- lm(TARGET_WINS ~ TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_P
summary(fit3)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_PITCHING_H + TEAM_PITCHING_HR +
##     TEAM_PITCHING_BB + TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_
##     data = train_set)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5424 -0.5741  0.0033  0.5928  4.1816
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.024e-15  1.921e-02   0.000    1.000
## TEAM_PITCHING_H   3.521e-01  3.189e-02  11.042   <2e-16 ***
## TEAM_PITCHING_HR  3.071e-02  2.917e-02   1.053    0.293
## TEAM_PITCHING_BB  1.881e-01  2.118e-02   8.883   <2e-16 ***
## TEAM_PITCHING_SO -1.822e-01  2.079e-02  -8.763   <2e-16 ***
## TEAM_FIELDING_E  -5.733e-01  4.598e-02 -12.467   <2e-16 ***
## TEAM_FIELDING_DP -3.896e-01  2.664e-02 -14.626   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9162 on 2268 degrees of freedom
## Multiple R-squared:  0.1628, Adjusted R-squared:  0.1606
## F-statistic: 73.51 on 6 and 2268 DF,  p-value: < 2.2e-16
```

This model does not appear linear based on the vertical shape of the data points, it is homoscedastic. Though the model is normally distributed, compared to the previous models, it has the most points that fall off the line on both ends of the tail. This model appears to be the weakest model of the three.

```r
par(mfrow=c(2, 2))
plot(fit3)
```



The variables *TEAM_PITCHING_H*, *TEAM_PITCHING_HR*, *TEAM_PITCHING_BB* have positive coefficients which makes sense given that they have a positive effect on wins. *TEAM_FIELDING_E* and *TEAM_PITCHING_SO* are detrimental to a game and so their coefficient is negative. It's odd that *TEAM_FIELDING_DP* is negative because they are a positive occurrence in a game.

This model is comprised mostly of statistically variables, but has the lowest R squared compared to the other models.

4. Model Selection

The model of choice will be model 1. Based on the findings, it is evident that both offense and defense variable together explain the variance best. However, it should be noted that model 1 has 16 predictors. It would be a good idea to experiment using a subset of variables, perhaps by using a 'stepwise' or 'regsubsets' algorithm from the leaps library. Furthermore, it should be noted that R squared always increases with the addition of more predictors, therefore its best to abide by the Adjusted R squared instead.

Next, residuals were inspected to check model conformance. For model 1 and 2 there residuals seem to be normally distributed and display constant variance. However, model 3 display made it very poor candidate in terms of residual diagnostics and Adjusted R squared score.

Furthermore, it would be a good idea to explore removing TEAM_BATTING_HR as it has a high correlation to TEAM_BATTING_SO and TEAM_PITCHING_HR.

Predictions on test_set:

```
##
##  iter imp variable
##   1   1  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   1   2  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   1   3  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   1   4  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   1   5  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   2   1  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   2   2  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   2   3  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   2   4  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   2   5  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   3   1  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   3   2  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   3   3  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   3   4  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   3   5  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   4   1  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   4   2  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   4   3  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   4   4  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   4   5  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   5   1  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   5   2  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   5   3  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   5   4  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
##   5   5  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_SO  TEAM
```

```
##        1         2         3         4         5         6
##  66.77140  71.04229  75.78749  84.38106  66.27144  66.84261  79.7
##        9        10        11        12        13        14
##  72.91115  75.50288  71.02994  82.12591  81.71390  82.80009  88.1
##       17        18        19        20        21        22
##  71.15455  78.52890  75.47073  87.78210  87.64114  86.79996  81.7
##       25        26        27        28        29        30
##  84.23658  88.65174  53.78645  73.97883  84.95389  74.15989  89.0
##       33        34        35        36        37        38
##  86.29994  83.15900  80.44587  79.56313  76.67294  86.36835  86.4
##       41        42        43        44        45        46
##  82.70895  98.12180  47.11341 101.61824  94.31540  94.07337  94.1
##       49        50        51        52        53        54
##  67.97226  79.40646  79.00394  88.34526  72.73849  77.81022  71.7
##       57        58        59        60        61        62
##  86.60914  75.63829  60.50815  76.49365  90.59260  86.36401  86.2
##       65        66        67        68        69        70
##  85.02605  97.55045  66.57847  75.77033  73.49095  86.27999  93.2
##       73        74        75        76        77        78
##  80.66697  87.29189  79.43335  77.83774  86.15015  80.58530  64.0
##       81        82        83        84        85        86
##  84.42870  87.29227  95.07848  78.43919  83.35695  82.84563  84.1
##       89        90        91        92        93        94
##  93.49700  87.25368  77.30745  65.07671  69.76554  87.11775  82.9
##       97        98        99       100       101       102
##  88.67839 102.75324  90.67548  90.26988  82.73692  74.23017  86.6
##      105       106       107       108       109       110
##  71.83857  60.58636  70.57662  77.54853  87.14401  55.06371  85.0
##      113       114       115       116       117       118
##  94.72044  92.15125  82.84928  81.20067  86.57220  80.40547  74.7
##      121       122       123       124       125       126
##  89.69304  62.21416  69.35988  72.47921  69.57484  88.94260  91.0
##      129       130       131       132       133       134
##  90.13998  91.24440  85.00175  85.07337  72.35823  80.06601  92.8
##      137       138       139       140       141       142
##  77.74281  80.80925  96.57259  80.90864  61.14031  74.50273  94.2
##      145       146       147       148       149       150
##  80.47611  76.20614  72.36174  83.53889  84.20930  84.45616  80.8
##      153       154       155       156       157       158
##  24.57422  68.52798  82.14616  70.37849  86.92260  86.69848  86.5
##      161       162       163       164       165       166
```

```
##  99.46214 105.71581  95.64833 100.99398  94.29612  96.10413  86.2
##       169       170       171       172       173       174
##  72.99604  81.28457  84.37795  88.91873  82.79714  95.28431  78.6
##       177       178       179       180       181       182
##  83.45291  71.38542  77.56347  79.09162  87.04849  81.39238  87.5
##       185       186       187       188       189       190
##  68.71633  89.41641  78.90706       NaN  57.77629 105.55508  63.9
##       193       194       195       196       197       198
##  73.12799  77.93047  76.97805  66.22271  73.66815  90.46414  81.5
##       201       202       203       204       205       206
##  72.32330  83.90060  79.22275  92.88797  83.22815  82.74198  82.4
##       209       210       211       212       213       214
##  73.83665  68.44657 100.39224  87.99148  82.30745  65.73996  71.7
##       217       218       219       220       221       222
##  82.25456  90.14471  80.05481  81.05915  75.49500  72.96026  78.0
##       225       226       227       228       229       230
##  73.06563  79.81284  79.74706  75.32134  85.00478  97.99532  71.5
##       233       234       235       236       237       238
##  83.23795  84.46891  76.09208  76.80335  77.99903  81.60996  85.7
##       241       242       243       244       245       246
##  87.53042  94.56050  89.06868  85.61235  63.71742  87.39294  80.4
##       249       250       251       252       253       254
##  78.54440  84.54764  79.66732  56.41372  90.31944       NaN  73.2
##       257       258       259
##  83.13359  80.22078  72.22607
```

Appendix

```
library(tidyverse)
library(ggpubr)
library(corrplot)
library(mice)
library(NHANES)
library(naniar)
library(GGally)
library(faraway)

training_set<-read_csv("https://raw.githubusercontent.com/Vy4thewin/
test_set<-read_csv("https://raw.githubusercontent.com/Vy4thewin/crit
require(gridExtra)
plot1 <- ggplot() + # Boxplot of TARGET_WINS
```

```
  geom_boxplot(aes(y = training_set$TARGET_WINS)) +
  scale_x_discrete( ) +
  labs(title = "Dist. of Game Wins per Season",
       y = "Number of Wins")

# compute mean TARTGET_WINS
mean_tw <- training_set %>%
  pull(TARGET_WINS) %>%
  mean() %>%
  signif(6)

plot2 <- ggplot( #review distribution of the response variable, see
  data=training_set, aes(x=TARGET_WINS))+
  geom_density()+
  labs(y = "Density",
       x = "Number of Wins")+
  geom_vline(xintercept=mean_tw, size=1.2, color="red")


grid.arrange(plot1, plot2, ncol=2)
summary(training_set)
any_na(training_set)
n_miss(training_set)
prop_miss(training_set)
training_set %>% is.na() %>% colSums()
#See the number of NAs per columns. Noticed Batters hit per pitch ha
colSums(is.na(training_set))

#Visually checking for a positive linear trend with a singular predi
g<-ggplot(training_set,aes(x=TEAM_BATTING_BB,y=TARGET_WINS))+geom_po

# plotting the wins vs home runs
g1<-ggplot(training_set,aes(x=TEAM_BATTING_HR,y=TARGET_WINS))+geom_p

#Viewing multiple graphs with a selected predictor variable and its
ggarrange(g,g1,ncol = 2,nrow = 1)

ggpairs(training_set)
#Technique #1: removing rows with any NAs
#Noticed there are entries with NAs values that cannot be replaced w
training_set_na_rem <- na.omit(training_set)
```

```r
#remove the index column as it does not a have effect on the data
train.c2<-training_set_na_rem %>%
  select(-c(INDEX))

summary(train.c2)
#Technique #2: Drop predictors HBP and CS and use MICE imputation
train.c1<-training_set %>%
  select(-c(INDEX,TEAM_BATTING_HBP,TEAM_BASERUN_CS)) #Filter dataset

train.mice<-complete(mice(train.c1,method = "lasso.norm",seed = 333)

summary(train.mice)
#Seeing the correlation between non-NA predictors to indicate mutli-
corrplot(cor(train.mice[,2:14]),method = "number",type="lower", tl.s
# Found some baseball formulas here to add onto our analysis
#STW= pitchers strikeouts/ walks by batter
train.mice<-train.mice %>% mutate(STW_Ratio=TEAM_PITCHING_SO/TEAM_BA

#total bases=[H + 2B + (2 X 3B) + (3 X HR)].
train.mice<-train.mice%>%mutate(TB=TEAM_BATTING_H+TEAM_BATTING_2B+(2

#two observations had NAs
train.mice<-na.omit(train.mice)

#review new columns
head(train.mice,3)
#see any possible predictors for skeweness and apply transformations
g1<-ggplot(data=train.mice,aes(x=TEAM_BATTING_H))+geom_density()+the
g2<-ggplot(data=train.mice,aes(x=TEAM_BATTING_2B))+geom_density()+th
g3<-ggplot(data=train.mice,aes(x=TEAM_BATTING_3B))+geom_density()+th
g4<-ggplot(data=train.mice,aes(x=TEAM_BATTING_HR))+geom_density()+th
g5<-ggplot(data=train.mice,aes(x=TEAM_BATTING_BB))+geom_density()+th
g6<-ggplot(data=train.mice,aes(x=TEAM_BATTING_SO))+geom_density()+th
g7<-ggplot(data=train.mice,aes(x=TEAM_BASERUN_SB))+geom_density()+th
g8<-ggplot(data=train.mice,aes(x=TEAM_PITCHING_H))+geom_density()+th
g9<-ggplot(data=train.mice,aes(x=TEAM_PITCHING_HR))+geom_density()+t
g10<-ggplot(data=train.mice,aes(x=TEAM_PITCHING_BB))+geom_density()+
g11<-ggplot(data=train.mice,aes(x=TEAM_PITCHING_SO))+geom_density()+
g12<-ggplot(data=train.mice,aes(x=TEAM_FIELDING_E))+geom_density()+t
g13<-ggplot(data=train.mice,aes(x=TEAM_FIELDING_DP))+geom_density()+
g14<-ggplot(data=train.mice,aes(x=STW_Ratio))+geom_density()+theme_c
g16<-ggplot(data=train.mice,aes(x=TB))+geom_density()+theme_classic(
```

```
ggarrange(g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g16,ncol =

#only 2b appears to be normal, let's fixed the variables that are po
#columns 15,12,8 cannot be transformed as some instances have negati
train.mice<-train.mice%>%mutate_at(c(2,9,11,13,14),~log10(.))


#scale the training set so its easier for the system to process the
train_set<-data.frame(scale(train.mice))
#first model using all predictors
fit1 = lm(TARGET_WINS ~., data = train_set)
summary(fit1)
par(mfrow=c(2, 2))
plot(fit1)
plot(fitted(fit1),residuals(fit1),xlab="Fitted",ylab="Residuals")
abline(h=0)
#Removing the largest outlier point we have a very similar R squared
cook <- cooks.distance(fit1)
halfnorm(cook,2, ylab="Cook’s distances")
#first model using all predictors
fit1i = lm(TARGET_WINS ~., data = train_set, subset=(cook < max(cook
summary(fit1i)
#second model with just batting variables
fit2 <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BAT
summary(fit2)
par(mfrow=c(2, 2))
plot(fit2)
#third model with pitching and fielding variables
fit3 <- lm(TARGET_WINS ~ TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_P
summary(fit3)
par(mfrow=c(2, 2))
plot(fit3)
#data cleanup
test.c1 <-test_set %>%
  select(-c(INDEX,TEAM_BATTING_HBP,TEAM_BASERUN_CS)) #Filter dataset

test.mice<-complete(mice(test.c1,method = "lasso.norm",seed = 333))

test.mice<-test.mice %>%
  mutate(STW_Ratio=TEAM_PITCHING_SO/TEAM_BATTING_BB)

#total bases=[H + 2B + (2 X 3B) + (3 X HR)].
```

```
test.mice<-test.mice%>%
  mutate(TB=TEAM_BATTING_H+TEAM_BATTING_2B+(2*TEAM_BATTING_3B)+(3*TE

#X observations had NAs
test.mice<-na.omit(test.mice)

test.mice<-test.mice%>%
  mutate_at(c(1,8,10,12,13),~log10(.))

test_set<-data.frame(scale(test.mice))

#predict
s.pred <- predict(fit1,new=test_set)

# backtransform scale:
(pred <- s.pred * sd(train.mice$TARGET_WINS) + mean(train.mice$TARGE
```

1. https://www.mlb.com/glossary/advanced-stats↩
2. http://hosted.stats.com/mlb/stats.asp?file=glossary↩