DATA 621 Business Analytics and Data Mining

Homework #1 (Submitted by Group 1)

Calvin Wong, Sudhan Maharjan, Kevin Benson, Ravi Itwaru, Juanelle Marks

Contents

The data analyzed in this report is extracted from a dataset containing approximately 2200 records. Each record represents a professional baseball team from the years 1871 to 2006 inclusive. Each record in the data set has the performance of the team for the given year with all of the statistics adjusted to match the performance of the 162 games season. Below is a short description of the variables of interest in the data set.

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

Figure 1.1

This table will be used as reference throughout this report.

Data Exploration

In this analysis, the response variable is TARGET _WINS. Before modelling for prediction, it is important to explore the dataset so as to get a sense of features of the various variables in the dataset. This process helps in establishing a possible need for cleaning and transformations as well as it provides critical information for variable selection at the modelling stage.

Glimpse of the data set

See below for a glimpse of the actual data set (See Figure 1.2). Each baseball team is represented by an index number.

INDEX <int>	TARGET_WINS <int>	TEAM_BATTING_H <int>	TEAM_BATTING_2B <int>	TEAM_BATTING_3B <int>	TEAM_BATTING_HR <int>	TEAM_BATTING_BB <int>	TEAM_BATTING_SO <int>	TEAM_BASERUN_SB <int>	TEAM_BASERUN_CS <int>
1	39	1445	194	39	13	143	842	NA	NA
2	70	1339	219	22	190	685	1075	37	28
3	86	1377	232	35	137	602	917	46	27
4	70	1387	209	38	96	451	922	43	30
5	82	1297	186	27	102	472	920	49	39
6	75	1279	200	36	92	443	973	107	59

6 rows | 1-10 of 17 columns

Figure 1.2

Structure of the data set

A look at Figure 1.3 below shows that this dataset comprises 2276 records/observations and 17 variables. All of the variables are of the type, "int". Their values are continuous numerical values. There is evidence of missing values in the dataset. For example, the variable TEAM_BATTING_HBP, has some 2085 missing values (See Figure1.4). Since these will affect modelling, they will be addressed at the data preparation stage.

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

Figure 1.3

Descriptive statistics

Below is a summary of descriptive statistics (See Figure 1.4).

```
     INDEX         TARGET_WINS     TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB TEAM_BASERUN_CS
 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   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   1st Qu.: 38.0  
 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   Median : 49.0  
 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   Mean   : 52.8  
 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   3rd Qu.: 62.0  
 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   Max.   :201.0  
                                                                                                                    NA's   :102      NA's   :131     NA's   :772    
 TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E  TEAM_FIELDING_DP
 Min.   :29.00    Min.   : 1137   Min.   :  0.0    Min.   :   0.0   Min.   :    0.0   Min.   :  65.0   Min.   : 52.0   
 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 :58.00    Median : 1518   Median :107.0    Median : 536.5   Median :  813.5   Median : 159.0   Median :149.0   
 Mean   :59.36    Mean   : 1779   Mean   :105.7    Mean   : 553.0   Mean   :  817.7   Mean   : 246.5   Mean   :146.4   
 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.   :95.00    Max.   :30132   Max.   :343.0    Max.   :3645.0   Max.   :19278.0   Max.   :1898.0   Max.   :228.0   
 NA's   :2085                                                       NA's   :102                        NA's   :286     
```

Figure 1.4

It is evident from these statistics that missing values in the dataset range from approximately 5 - 92% of the data provided for the respective variables. The descriptive statistics also give evidence that there might be cases where there is skewness in some of the distributions since there are some huge differences between the 3rd quartile value and maximum value of some variables. The histograms in Figure 1.5 below helps us visualize some of these cases. The possible outliers will have to be addressed at the data preparation stage.

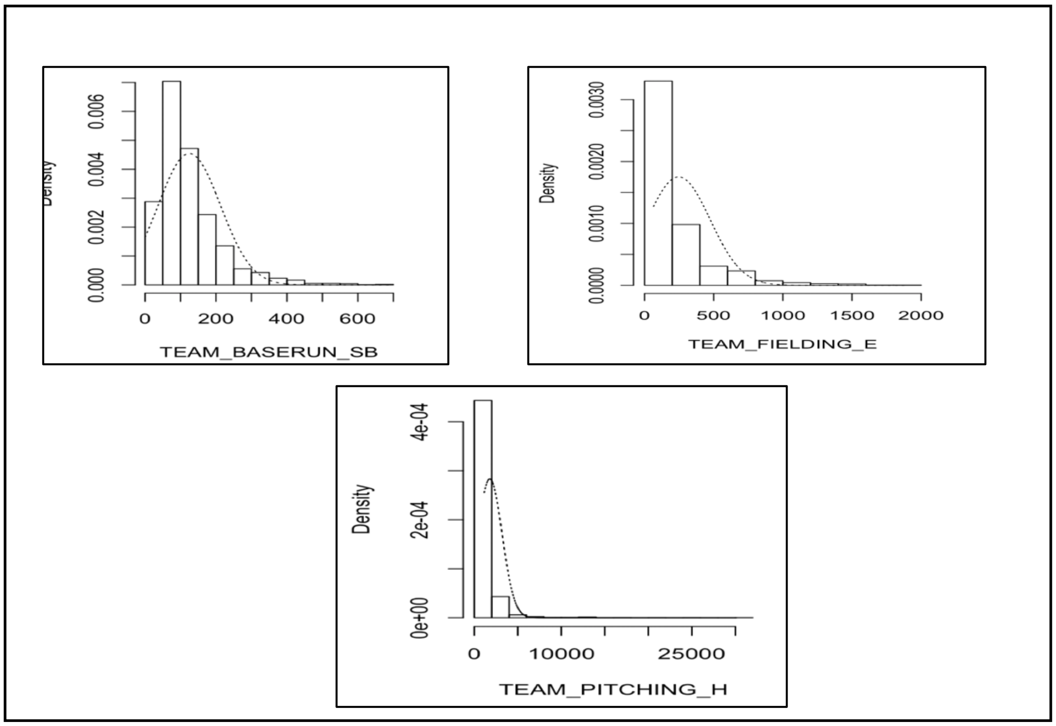

Figure 1.5

Pairwise relationships

Knowledge of the type and strength of the relationship between the response and predictor variables helps in determining which variables to include in the model. Figure 1.6, Figure 1.7 and Figure 1.8 below is an image of pairwise graphs that visualizes this information.

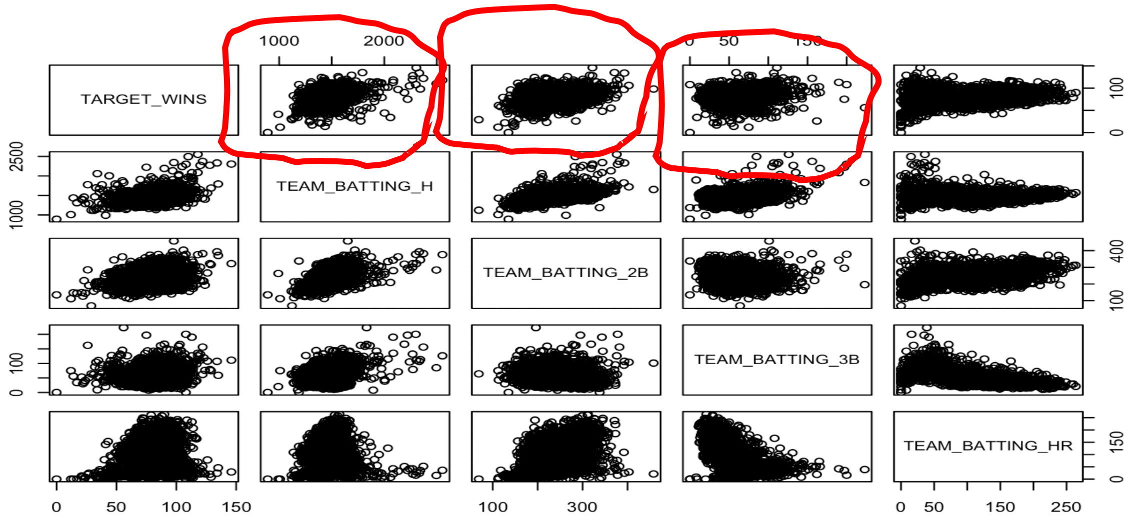

Figure1.6

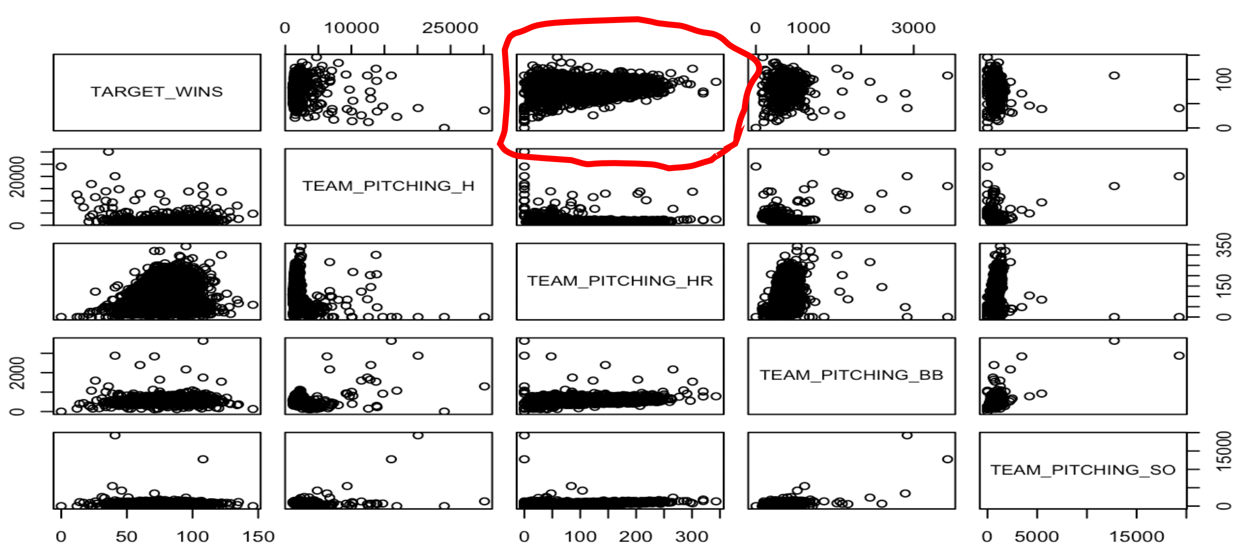

Figure 1.7

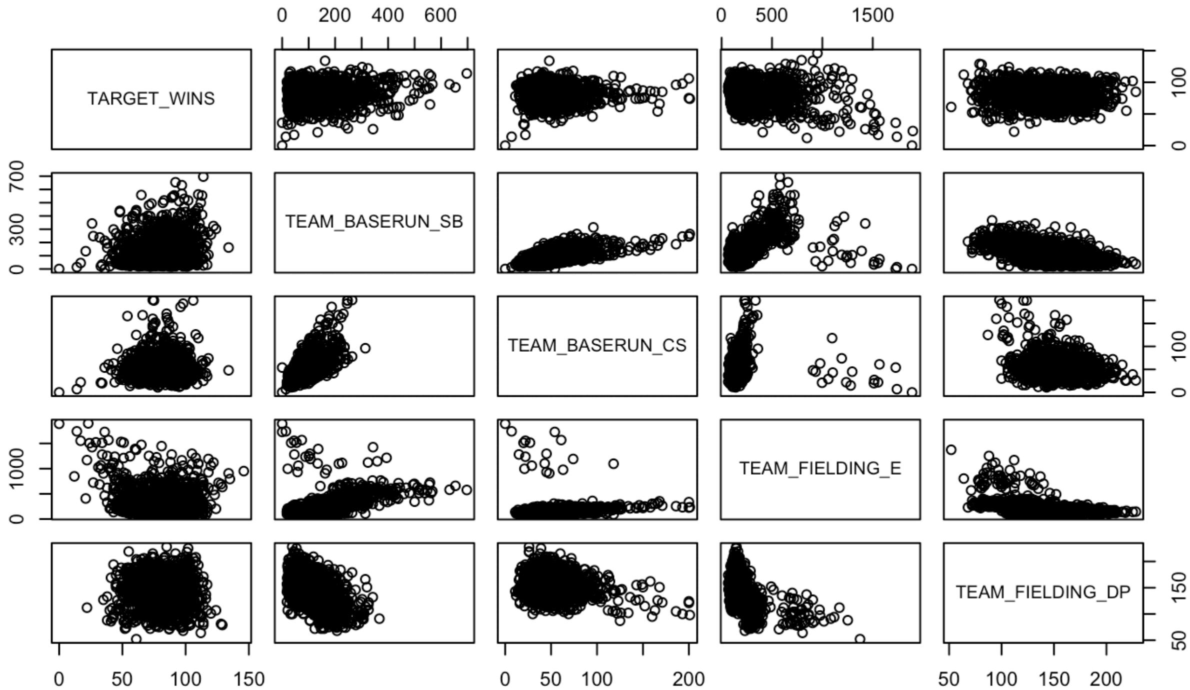

Figure 1.8

It is observed from the figures above that variables such as TEAM_BATTING_H, TEAM_BATTING_2B, TEAM _PITCHINGING_HR and TEAM_BATTING_BB have a very strong linear relationship with TARGET_WINS. This indicates that they would likely be good predictors in the model. It is also observed that variables such as TEAM_FIELDING_E and TEAM_BASERUN_CS have a weak relationship with the response variable.

In summary, there is a need for some data cleaning and transformations in this dataset before modelling is done. Some variables show strong indications of being suitable predictor variables in a model for predicting TARGET_WINS.

Data Preparation

Data preparation is a pre-processing step that involves cleansing, transforming, and consolidating data. Our first step in the data preparation process was to identify what variables needed to be manipulated within the dataset. We utilize the `funModeling` package for this purpose. This package contains a set of functions related to exploratory data analysis, data preparation, and model performance. `funModeling` is intimately related to the Data Science Live Book -Open Source- (2017). Here, `funModeling` df_status(), is being used to analyze the zeros, missing values (NA), infinity, data type, and number of unique values for a given dataset(See Figure 2.1 below).

```
##              variable q_zeros p_zeros q_na  p_na q_inf p_inf    type unique
## 11 TEAM_BATTING_HBP       0    0.00 2085 91.61     0     0 integer     55
## 10  TEAM_BASERUN_CS       1    0.04  772 33.92     0     0 integer    128
## 17 TEAM_FIELDING_DP       0    0.00  286 12.57     0     0 integer    144
## 9   TEAM_BASERUN_SB       2    0.09  131  5.76     0     0 integer    348
## 8   TEAM_BATTING_SO      20    0.88  102  4.48     0     0 integer    822
## 15 TEAM_PITCHING_SO      20    0.88  102  4.48     0     0 integer    823
## 1             INDEX       0    0.00    0  0.00     0     0 integer   2276
## 2       TARGET_WINS       1    0.04    0  0.00     0     0 integer    108
## 3    TEAM_BATTING_H       0    0.00    0  0.00     0     0 integer    569
## 4   TEAM_BATTING_2B       0    0.00    0  0.00     0     0 integer    240
## 5   TEAM_BATTING_3B       2    0.09    0  0.00     0     0 integer    144
## 6   TEAM_BATTING_HR      15    0.66    0  0.00     0     0 integer    243
## 7   TEAM_BATTING_BB       1    0.04    0  0.00     0     0 integer    533
## 12  TEAM_PITCHING_H       0    0.00    0  0.00     0     0 integer    843
## 13 TEAM_PITCHING_HR      15    0.66    0  0.00     0     0 integer    256
## 14 TEAM_PITCHING_BB       1    0.04    0  0.00     0     0 integer    535
## 16  TEAM_FIELDING_E       0    0.00    0  0.00     0     0 integer    549
```

Figure 2.1

Handling Missing Values

With this particular dataset, using df_status(), we identified our biggest challenge was to deal with NA's. There was a minimal amount of zero's which accounted for less than one percent of the dataset for any variable. Therefore, we decided to focus on NA's. We ordered the percentage of NA's and identified those variables to be transformed using different imputation methods to be discussed below:

Identified variables for transformations:

1. TEAM_BATTING_HBP
2. TEAM_BASERUN_CS
3. TEAM_FIELDING_DP
4. TEAM_BASERUN_SB
5. TEAM_BATTING_SO
6. TEAM_PITCHING_SO.

We decided to use mean imputation on all the above variables. However, for variables above the ten percent (10%) threshold, we included the use of a dummy variable to identify if an NA is present. The three variables with a dummy variable are:

1. TEAM_BATTING_HBP
2. TEAM_BASERUN_CS
3. TEAM_FIELDING_DP

Transformations followed this approach:

1. A dummy variable called HBP_missing was created. This triggers "1"" if TEAM_BATTING_HBP value is NA and "0" if not. All NA's were then imputed to the mean for this variable.
2. A dummy variable called CS_missing was created which triggers "1" if TEAM_BASERUN_CS value is NA and "0" if not All NA's were then imputed to the mean for this variable.
3. A dummy variable called DP_missing was created which triggers "1" if TEAM_FIELDING_DP value is NA and "0" if not. All NA's were then imputed to the mean for this variable.
4. Since the other three variables did not fall above the ten percent (10%) threshold, no dummy variables were utilized. Instead, just imputation of NA's to the mean was done to transform these variables.

Figure 2.2 below shows the status of our data frame after these transformations. As can be seen there are now zero missing data values for all variables. The 3 last rows represent dummy variables and now show "0" values.

```
##              variable q_zeros p_zeros q_na p_na q_inf p_inf    type unique
## 1               INDEX       0    0.00    0    0     0     0 integer   2276
## 2         TARGET_WINS       1    0.04    0    0     0     0 integer    108
## 3      TEAM_BATTING_H       0    0.00    0    0     0     0 integer    569
## 4     TEAM_BATTING_2B       0    0.00    0    0     0     0 integer    240
## 5     TEAM_BATTING_3B       2    0.09    0    0     0     0 integer    144
## 6     TEAM_BATTING_HR      15    0.66    0    0     0     0 integer    243
## 7     TEAM_BATTING_BB       1    0.04    0    0     0     0 integer    533
## 8     TEAM_BATTING_SO      20    0.88    0    0     0     0 numeric    823
## 9     TEAM_BASERUN_SB       2    0.09    0    0     0     0 numeric    349
## 10    TEAM_BASERUN_CS       1    0.04    0    0     0     0 numeric    129
## 11 TEAM_BATTING_HBP       0    0.00    0    0     0     0 numeric     56
## 12    TEAM_PITCHING_H       0    0.00    0    0     0     0 integer    843
## 13 TEAM_PITCHING_HR      15    0.66    0    0     0     0 integer    256
## 14 TEAM_PITCHING_BB       1    0.04    0    0     0     0 integer    535
## 15 TEAM_PITCHING_SO      20    0.88    0    0     0     0 numeric    824
## 16    TEAM_FIELDING_E       0    0.00    0    0     0     0 integer    549
## 17 TEAM_FIELDING_DP       0    0.00    0    0     0     0 numeric    145
## 18        HBP_missing     191    8.39    0    0     0     0 numeric      2
## 19         CS_missing    1504   66.08    0    0     0     0 numeric      2
## 20         DP_missing    1990   87.43    0    0     0     0 numeric      2
```

Figure 2.2

Outliers

Our next step is to deal with outliers identified within the data distribution section. We identified four variables which needed to be worked.

1. TEAM_PITCHING_SO
2. TEAM_PITCHING_H
3. TEAM_PITCHING_BB
4. TEAM_FIELDING_E.

We identified through boxplots which variables are impacted with outliers. Secondly, TEAM_FIELDING_E shows the potential of applying transformation to retain all data points (See Figure 2.3 to 2.6 below).

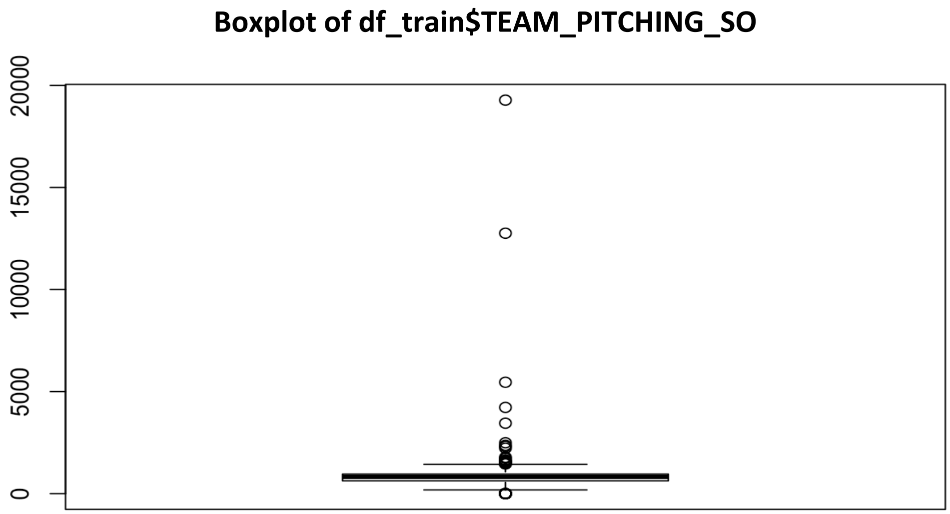

Figure 2.3

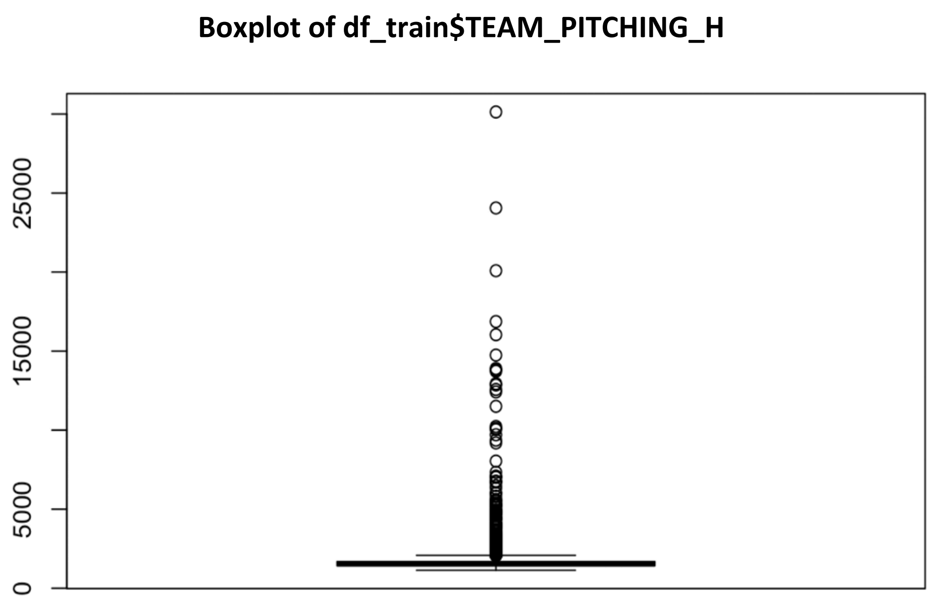

Figure 2.4

Boxplot of df_train$TEAM_PITCHING_BB

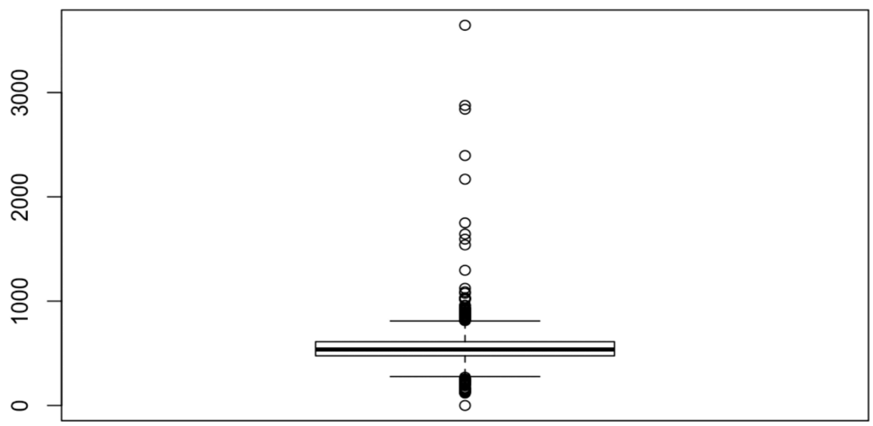

Figure 2.5

Histogram of df_train$TEAM_Fielding_E

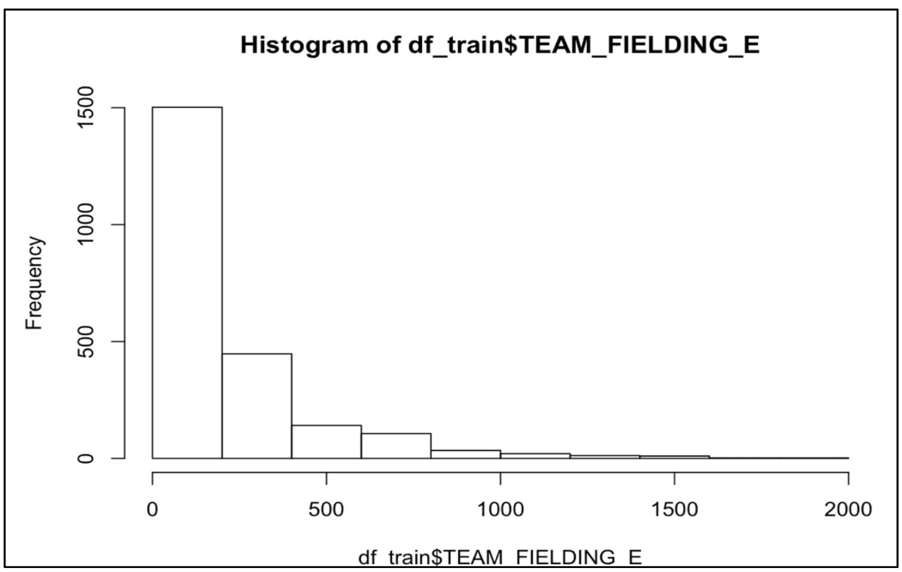

Figure 2.6

John Tukey invented the box-and-whisker plot in 1977 to display IQR values, he picked 1.5×IQR as the demarcation line for outliers.

Based on Tukey's outlier identifier, we will retain this approach and remove outliers for TEAM_PITCHING_SO, TEAM_PITCHING_H, and TEAM_PITCHING_BB using the above formula. We removed a total of 285 records which accounts for approximately 12.5% of the total dataset. The boxplots below show the three variables which were impacted by outliers are now normalized (See Figure 2.7 to 2.9 below).

TEAM_PITCHING_SO with outliers removed

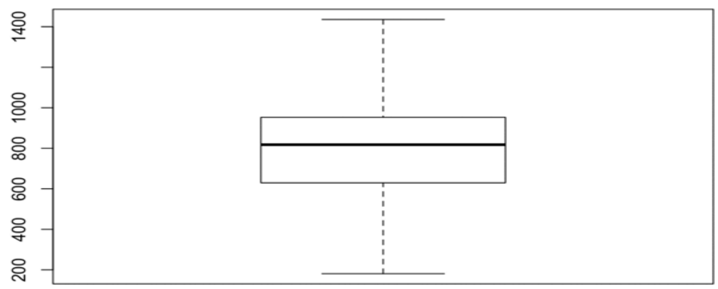

Figure 2.7

TEAM_PITCHING_H with outliers removed

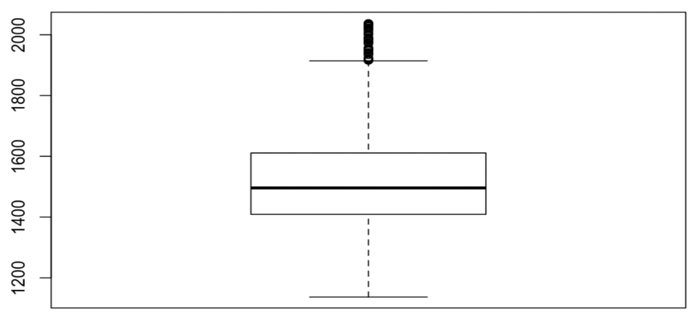

Figure 2.8

TEAM_PITCHING_BB with outliers removed

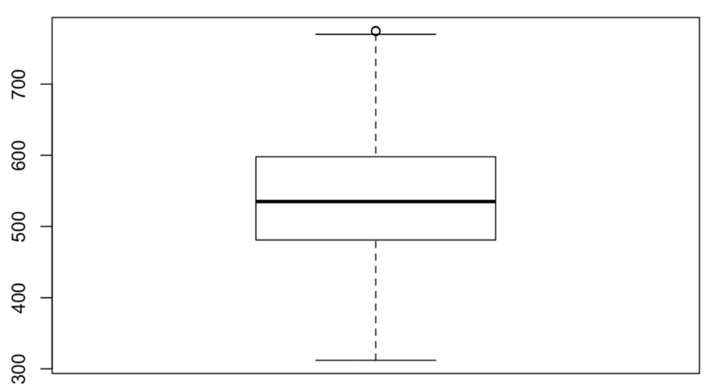

Figure 2.9

For the fourth variable, TEAM_FIELDING_E, we transformed the data using log 10 transformation and then re-plotted the data. Figure 2.10 below shows what the TEAM_FIELDING_E, distribution now looks like.

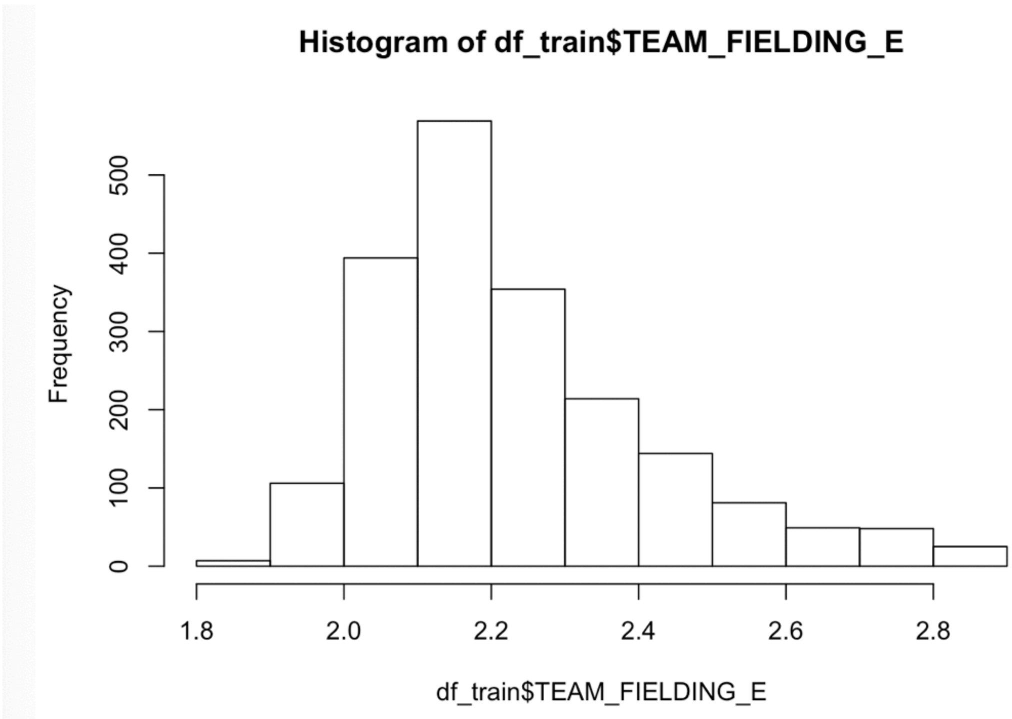

Figure 2.10

It was observed that the data was still somewhat skewed, however, shows much improvement from the original. We deduct that this transformation is adequate and will retain this method of transformation.

We are now ready to proceed with regression model building.

Model Creation

We start building our models by reviewing statistical output from our kitchen sink regression model.

Model lm1

Model lm1 is our kitchen sink regression. This model basically has all the predictor variables (excluding the index) from our training dataset which includes our dummy variables. All missing values (NA's) in this model were replaced with the mean value of that associated predictor. The reasoning behind building such a model is so we can find some sort of statistical pattern in the regression output. This output from this model will assist us with building our additional models. Additionally, this model has a multiple R^2 of 0.4189, an adjusted R^2 of 0.4135, an F-statistic of 78.96, and a p-value of < 2.2e-16.

Figure 3.1 below shows the output for model lm1

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - INDEX, data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.349  -7.393   0.052   7.229  31.955
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      144.816321  10.042037  14.421  < 2e-16 ***
## TEAM_BATTING_H     0.013998   0.015374   0.910 0.362684
## TEAM_BATTING_2B   -0.039174   0.009300  -4.212 2.64e-05 ***
## TEAM_BATTING_3B    0.201772   0.018302  11.025  < 2e-16 ***
## TEAM_BATTING_HR    0.517821   0.109255   4.740 2.29e-06 ***
## TEAM_BATTING_BB    0.191804   0.045310   4.233 2.41e-05 ***
## TEAM_BATTING_SO   -0.137496   0.018767  -7.326 3.43e-13 ***
## TEAM_BASERUN_SB    0.065496   0.005125  12.781  < 2e-16 ***
## TEAM_BASERUN_CS   -0.020723   0.015113  -1.371 0.170465
## TEAM_BATTING_HBP   0.116458   0.060419   1.928 0.054061 .
## TEAM_PITCHING_H    0.017751   0.013957   1.272 0.203607
## TEAM_PITCHING_HR  -0.426622   0.104808  -4.071 4.88e-05 ***
## TEAM_PITCHING_BB  -0.148864   0.042794  -3.479 0.000515 ***
## TEAM_PITCHING_SO   0.117022   0.017553   6.667 3.38e-11 ***
## TEAM_FIELDING_E  -58.601672   3.181131 -18.422  < 2e-16 ***
## TEAM_FIELDING_DP  -0.110638   0.012786  -8.653  < 2e-16 ***
## HBP_missing        4.917800   1.033487   4.758 2.09e-06 ***
## CS_missing         4.426312   0.880186   5.029 5.38e-07 ***
## DP_missing         0.795946   1.708117   0.466 0.641282
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.77 on 1972 degrees of freedom
## Multiple R-squared:  0.4189, Adjusted R-squared:  0.4135
## F-statistic: 78.96 on 18 and 1972 DF,  p-value: < 2.2e-16
```

Figure 3.1

Model lm2

Model lm2 has 5 predictor variables. All missing values (NA's) were replaced with the mean value of that associated predictor. The selection of these predictors was based on creating a model that had 4 predictors with a positive impact on wins (TARGET_WINS) and 1 predictor with a negative impact on wins. This linear regression model fitted with these predictors produced 3 variables (TEAM_BATTING_H, TEAM_BATTING_BB, TEAM_BASERUN_SB) that are statistically significant. The coefficients in this model are mostly positive except for TEAM_BASERUN_CS which makes sense since caught stealing will cause your team to lose points. Additionally, this model has a multiple R^2 of 0.2068, an adjusted R^2 of 0.2049, an F-statistic of 103.5, and a p-value of <2.2e-16.

Figure 3.2 below shows the output for model lm2

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB +
##     TEAM_BATTING_HBP + TEAM_BASERUN_SB + TEAM_BASERUN_CS, data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.793  -8.297   0.363   8.713  41.698
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -7.262937   5.767452  -1.259    0.208
## TEAM_BATTING_H    0.040091   0.002541  15.777  < 2e-16 ***
## TEAM_BATTING_BB   0.044035   0.003358  13.114  < 2e-16 ***
## TEAM_BATTING_HBP  0.067026   0.070174   0.955    0.340
## TEAM_BASERUN_SB   0.029909   0.003836   7.796 1.02e-14 ***
## TEAM_BASERUN_CS  -0.013240   0.015273  -0.867    0.386
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.54 on 1985 degrees of freedom
## Multiple R-squared:  0.2068, Adjusted R-squared:  0.2049
## F-statistic: 103.5 on 5 and 1985 DF,  p-value: < 2.2e-16
```

Figure 3.2

Model lm3

Model lm3 has 7 predictor variables. All missing values (NA's) were replaced with the mean value of that associated predictor. The selection of these predictors was based on creating a model with predictors that had to do with batting performance. This linear regression model fitted with these predictors produced 5 variables (TEAM_BATTING_H, TEAM_BATTING_2B, TEAM_BATTING_3B, TEAM_BATTING_HR, TEAM_BATTING_BB) that are statistically significant. The coefficients in this model are mostly positive except for TEAM_BATTING_2B which doesn't really make sense since advancing to second base shouldn't have a negative impact on wins. This model is worth keeping since it is statistically sound. Additionally, this model has a multiple R^2 of 0.2204, an adjusted R^2 of 0.2176, an F-statistic of 80.07, and a p-value of <2.2e-16.

Figure 3.3 below shows the output for model lm3.

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_HBP +
##     TEAM_BATTING_SO, data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.515  -8.078   0.411   8.454  46.511
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -3.825044   7.404586  -0.517  0.60551
## TEAM_BATTING_H    0.035775   0.004766   7.506 9.15e-14 ***
## TEAM_BATTING_2B  -0.033091   0.009961  -3.322  0.00091 ***
## TEAM_BATTING_3B   0.155224   0.019128   8.115 8.43e-16 ***
## TEAM_BATTING_HR   0.058803   0.009631   6.106 1.23e-09 ***
## TEAM_BATTING_BB   0.038070   0.003682  10.339  < 2e-16 ***
## TEAM_BATTING_HBP  0.064747   0.069641   0.930  0.35262
## TEAM_BATTING_SO   0.003363   0.002354   1.428  0.15335
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.44 on 1983 degrees of freedom
## Multiple R-squared:  0.2204, Adjusted R-squared:  0.2176
## F-statistic: 80.07 on 7 and 1983 DF,  p-value: < 2.2e-16
```

Figure 3.3

Model lm4

Model lm4 has 9 predictor variables. All missing values (NA's) were replaced with the mean value of that associated predictor. This model has 4 predictors with a positive impact on wins (TARGET_WINS) and 5 predictors with a negative impact on wins. The reasoning behind this model was to see how well the combination of the batting and fielding stats explained our dependent variable. This linear regression model generated 6 statistically significant variables. Additionally, this model has a multiple R^2 of 0.2971, an adjusted R^2 of 0.2939, an F-statistic of 93.04, and a p-value of <2.2e-16.

Figure 3.4 below shows the output for model lm4

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR +
##     TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_FIELDING_E +
##     TEAM_PITCHING_BB + TEAM_PITCHING_H + TEAM_PITCHING_HR, data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.849  -8.151  -0.115   7.837  38.767
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      109.239564   9.095093  12.011  < 2e-16 ***
## TEAM_BATTING_H    -0.032970   0.014857  -2.219 0.026585 *
## TEAM_BATTING_HR   -0.047136   0.101481  -0.464 0.642352
## TEAM_BATTING_BB    0.210064   0.049206   4.269 2.06e-05 ***
## TEAM_BATTING_SO   -0.019203   0.002427  -7.912 4.16e-15 ***
## TEAM_BASERUN_SB    0.077516   0.004737  16.365  < 2e-16 ***
## TEAM_FIELDING_E  -36.618465   2.873953 -12.741  < 2e-16 ***
## TEAM_PITCHING_BB  -0.173785   0.046488  -3.738 0.000191 ***
## TEAM_PITCHING_H    0.058246   0.013824   4.214 2.63e-05 ***
## TEAM_PITCHING_HR   0.073878   0.098097   0.753 0.451474
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.82 on 1981 degrees of freedom
## Multiple R-squared:  0.2971, Adjusted R-squared:  0.2939
## F-statistic: 93.04 on 9 and 1981 DF,  p-value: < 2.2e-16
```

Figure 3.4

Model lm5

In model lm5, we transform variables to produce combinations that attempt to capture key factors for winning teams, such as:

- Base hits + walks + hit-by-pitches: this is conceptually equivalent to on-base-percentage, i.e., the frequency of a team's batters to get on base. A team doesn't score runs unless its batters are able to get on base.
- Doubles + triples + homeruns: this is analogous to slugging percentage, i.e., the frequency of run-producing hits. Teams that produce higher numbers of these hits (as compared to singles or walks) will tend to produce more runs.
- Bases stolen - bases caught stealing: the net number of stolen bases should correlate to putting runners in scoring position, which should lead to more runs.
- Base hits allowed + walks allowed + homeruns allowed: the converse of the above on-base-percentage and slugging percentage, i.e., in favor of the opposing team. Figure 3.5 below shows the output for model lm5

```
## Call:
## lm(formula = TARGET_WINS ~ I(TEAM_BATTING_H + TEAM_BATTING_BB +
##     TEAM_BATTING_HBP) + I(TEAM_BATTING_2B + TEAM_BATTING_3B +
##     TEAM_BATTING_HR) + TEAM_BATTING_SO + I(TEAM_BASERUN_SB -
##     TEAM_BASERUN_CS) + I(TEAM_PITCHING_H + TEAM_PITCHING_BB +
##     TEAM_PITCHING_HR) + TEAM_PITCHING_SO + TEAM_FIELDING_E +
##     TEAM_FIELDING_DP + HBP_missing + CS_missing + DP_missing,
##     data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.729  -7.670  -0.075   7.548  48.687
##
## Coefficients:
##                                                         Estimate
## (Intercept)                                           130.545643
## I(TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_BATTING_HBP)    0.041242
## I(TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR)    0.027116
## TEAM_BATTING_SO                                        -0.062166
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                    0.067107
## I(TEAM_PITCHING_H + TEAM_PITCHING_BB + TEAM_PITCHING_HR) -0.008849
## TEAM_PITCHING_SO                                        0.047464
## TEAM_FIELDING_E                                       -48.697666
## TEAM_FIELDING_DP                                       -0.125512
## HBP_missing                                             6.836883
## CS_missing                                              3.641531
## DP_missing                                              1.759880
##                                                        Std. Error
## (Intercept)                                             9.637180
## I(TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_BATTING_HBP)    0.006533
## I(TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR)    0.006771
## TEAM_BATTING_SO                                         0.015156
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                    0.005176
## I(TEAM_PITCHING_H + TEAM_PITCHING_BB + TEAM_PITCHING_HR)  0.005614
## TEAM_PITCHING_SO                                        0.014728
## TEAM_FIELDING_E                                         3.056371
## TEAM_FIELDING_DP                                        0.013314
## HBP_missing                                             1.040754
## CS_missing                                              0.888313
## DP_missing                                              1.778641
##                                                        t value Pr(>|t|)
## (Intercept)                                             13.546  < 2e-16
## I(TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_BATTING_HBP)    6.313 3.37e-10
## I(TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR)    4.004 6.44e-05
## TEAM_BATTING_SO                                         -4.102 4.27e-05
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                    12.966  < 2e-16
## I(TEAM_PITCHING_H + TEAM_PITCHING_BB + TEAM_PITCHING_HR) -1.576  0.11514
## TEAM_PITCHING_SO                                         3.223  0.00129
## TEAM_FIELDING_E                                        -15.933  < 2e-16
## TEAM_FIELDING_DP                                        -9.427  < 2e-16
## HBP_missing                                              6.569 6.45e-11
## CS_missing                                               4.099 4.31e-05
## DP_missing                                               0.989  0.32256
##
## (Intercept)                                            ***
## I(TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_BATTING_HBP)   ***
## I(TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR)   ***
## TEAM_BATTING_SO                                        ***
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                   ***
## I(TEAM_PITCHING_H + TEAM_PITCHING_BB + TEAM_PITCHING_HR)
## TEAM_PITCHING_SO                                       **
## TEAM_FIELDING_E                                        ***
## TEAM_FIELDING_DP                                       ***
## HBP_missing                                            ***
## CS_missing                                             ***
## DP_missing
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.34 on 1979 degrees of freedom
## Multiple R-squared:  0.3536, Adjusted R-squared:   0.35
## F-statistic: 98.42 on 11 and 1979 DF,  p-value: < 2.2e-16
```

Figure 3.5

Model lm6

Finally, in model lm6, we try a different approach to combination variables, which attempts to capture the differential between a team and its opponents with respect to various dimensions of performance. Such factors include:

- Hits and walks: teams that produce more hits and walks than their opponents will tend to produce more runs.
- Homeruns: teams that hit more homeruns than their opponents will tend to score more runs.
- Strikeouts: teams that strike out more often than their opponents will tend to score fewer runs.

Figure 3.6 below shows the output for model lm6

```
##
## Call:
## lm(formula = TARGET_WINS ~ I(TEAM_BATTING_H + TEAM_BATTING_BB -
##     TEAM_PITCHING_H - TEAM_PITCHING_BB) + I(TEAM_BATTING_HR -
##     TEAM_PITCHING_HR) + I(TEAM_BATTING_SO - TEAM_PITCHING_SO) +
##     I(TEAM_BASERUN_SB - TEAM_BASERUN_CS) + TEAM_FIELDING_E +
##     TEAM_FIELDING_DP + HBP_missing + CS_missing + DP_missing,
##     data = df_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.364  -8.764   0.145   9.005  37.837
##
## Coefficients:
##                                                                       Estimate
## (Intercept)                                                         179.320452
## I(TEAM_BATTING_H + TEAM_BATTING_BB - TEAM_PITCHING_H - TEAM_PITCHING_BB)  -0.041643
## I(TEAM_BATTING_HR - TEAM_PITCHING_HR)                                -0.477483
## I(TEAM_BATTING_SO - TEAM_PITCHING_SO)                                 0.128068
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                                  0.063750
## TEAM_FIELDING_E                                                     -45.202763
## TEAM_FIELDING_DP                                                     -0.066685
## HBP_missing                                                           5.627750
## CS_missing                                                            4.950896
## DP_missing                                                           -1.202741
##                                                                     Std. Error
## (Intercept)                                                           7.228047
## I(TEAM_BATTING_H + TEAM_BATTING_BB - TEAM_PITCHING_H - TEAM_PITCHING_BB)   0.006531
## I(TEAM_BATTING_HR - TEAM_PITCHING_HR)                                 0.102895
## I(TEAM_BATTING_SO - TEAM_PITCHING_SO)                                 0.016309
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                                  0.005528
## TEAM_FIELDING_E                                                       3.159290
## TEAM_FIELDING_DP                                                      0.014785
## HBP_missing                                                           1.085682
## CS_missing                                                            0.958712
## DP_missing                                                            1.927763
##                                                                     t value
## (Intercept)                                                          24.809
## I(TEAM_BATTING_H + TEAM_BATTING_BB - TEAM_PITCHING_H - TEAM_PITCHING_BB)  -6.377
## I(TEAM_BATTING_HR - TEAM_PITCHING_HR)                                -4.641
## I(TEAM_BATTING_SO - TEAM_PITCHING_SO)                                 7.853
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                                 11.532
## TEAM_FIELDING_E                                                     -14.308
## TEAM_FIELDING_DP                                                     -4.510
## HBP_missing                                                           5.184
## CS_missing                                                            5.164
## DP_missing                                                           -0.624
##                                                                     Pr(>|t|)
## (Intercept)                                                          < 2e-16
## I(TEAM_BATTING_H + TEAM_BATTING_BB - TEAM_PITCHING_H - TEAM_PITCHING_BB) 2.25e-10
## I(TEAM_BATTING_HR - TEAM_PITCHING_HR)                               3.70e-06
## I(TEAM_BATTING_SO - TEAM_PITCHING_SO)                               6.62e-15
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                                 < 2e-16
## TEAM_FIELDING_E                                                      < 2e-16
## TEAM_FIELDING_DP                                                    6.86e-06
## HBP_missing                                                         2.40e-07
## CS_missing                                                          2.66e-07
## DP_missing                                                             0.533
##
## (Intercept)                                                         ***
## I(TEAM_BATTING_H + TEAM_BATTING_BB - TEAM_PITCHING_H - TEAM_PITCHING_BB) ***
## I(TEAM_BATTING_HR - TEAM_PITCHING_HR)                               ***
## I(TEAM_BATTING_SO - TEAM_PITCHING_SO)                               ***
## I(TEAM_BASERUN_SB - TEAM_BASERUN_CS)                                ***
## TEAM_FIELDING_E                                                     ***
## TEAM_FIELDING_DP                                                    ***
## HBP_missing                                                         ***
## CS_missing                                                          ***
## DP_missing
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.77 on 1981 degrees of freedom
## Multiple R-squared:  0.1797, Adjusted R-squared:  0.176
## F-statistic: 48.23 on 9 and 1981 DF,  p-value: < 2.2e-16
```

Figure 3.6

Model Selection and Prediction

Model Selection

We start by reviewing summary statistics for each model, including:

* N_Vars: number of predictor variables

* Sigma: residual standard error

* R_Sq: multiple R^2

* Adj_R_Sq: adjusted R^2

* F_P_Val: p-value corresponding to the F-statistic

* MSE: mean squared error

* RMSE: root mean squared error.

These statistics are computed based on the training dataset.

Model	N_Vars	Sigma	R_Sq	Adj_R_Sq	F_Stat	F_P_Val	MSE	RMSE
lm1	19	10.770	0.419	0.414	78.961	0	114.882	10.718
lm2	6	12.541	0.207	0.205	103.536	0	156.791	12.522
lm3	8	12.439	0.220	0.218	80.074	0	154.118	12.414
lm4	10	11.817	0.297	0.294	93.045	0	138.947	11.788
lm5	12	11.338	0.354	0.350	98.424	0	127.778	11.304
lm6	10	12.766	0.180	0.176	48.233	0	162.150	12.734

Figure 4.1

Based on the summary statistics above, it appears that our kitchen sink model (`lm1`) has the best overall performance metrics: the lowest residual standard error (10.8), the highest adjusted R^2 (41.4%), and the lowest MSE / RMSE (114.8 / 10.7). We therefore select `lm1` as our champion model.

Model Prediction

Let's review the model diagnostics for our champion model to ensure that key model assumptions are satisfied:

* Linear relationship between the response and predictor variables

* Independence of errors

* Approximately constant variance of errors

* Approximately normal distribution of errors.

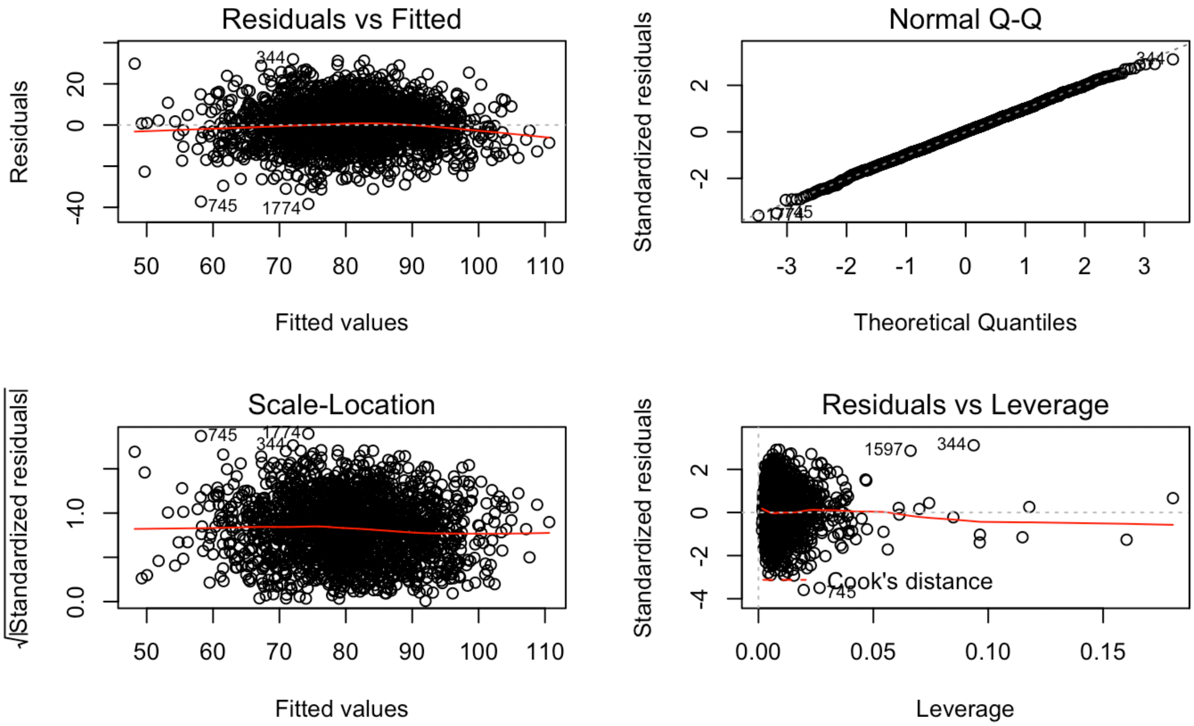

Figure 4.2

From the residual vs. fitted value chart, it appears that response and predictor variables follow a linear relationship. From the same chart as well as the square root absolute value residual vs. fitted value chart, it appears that the residuals have approximately constant variance. Finally, the normal Q-Q plot suggests that the residuals are approximately normally distributed. It is evident from the standardized residual vs. leverage chart that there are outliers in the dataset, which may have high leverage.

Finally, we plot the standardized residuals vs. each predictor variable in the champion model. Although the standardized residuals exhibit some structure with respect to certain variables (particularly for the variables like TEAM_BATTING_HBP where mean imputation was used for missing values), overall the standardized residuals are mostly consistent with our regression assumptions.

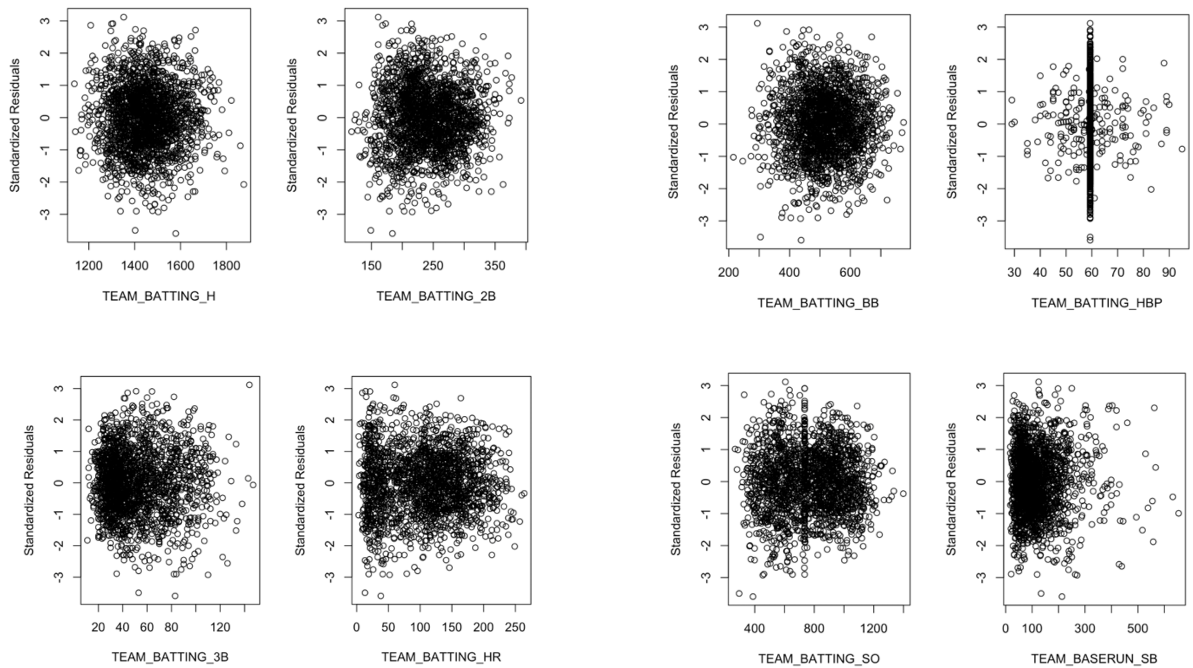

Figure 4.3

Several points relating to our model selection are worth highlighting:

Selection criteria for champion model: We chose our champion model on the basis of its predictive performance, primarily focusing on the residual standard error (σ), adjusted R^2, and the root mean squared error (RMSE) metrics as measured against the training dataset. The champion model has the lowest σ, highest R^2, and lowest RMSE. Although it doesn't have the highest F-statistic, the p-value associated with the F-statistic (roughly 0) is comparable to the p-values for the other models. Finally, the vast majority of the coefficients are statistically significant at the $\alpha = 0.05$ level.

Performance vs. reasonability: Reviewing the regression output above, the champion model has strong explanatory power, as most of the coefficient signs are consistent with intuition. For instance, teams that generate more runners on base (TEAM_BATTING_H, TEAM_BATTING_BB, TEAM_BATTING_HBP) or more run-producing hits (TEAM_BATTING_3B, TEAM_BATTING_HR) tend to win more games. Likewise, teams that allow their opponents to get on base or hit homeruns more often (TEAM_PITCHING_BB, TEAM_PITCHING_HR) tend to win fewer games. Some coefficients, however, had counter-intuitive signs, which likely results from idiosyncrasies with the data or likely multi-collinearity issues (see below). For instance, teams that produce more doubles (TEAM_BATTING_2B) should win more games on average (positive coefficient), and teams that allow more base hits (TEAM_PITCHING_H) should win fewer games on average (negative coefficient), but the estimated coefficient signs are reversed. As a side note, the coefficient for the fielding errors variable (TEAM_FIELDING_E) has a different order of magnitude compared to the other coefficients, but this is an artifact of the log variable transformation performed during data preparation.

Multi-collinearity: The champion model most likely has multi-collinearity issues, as some of the variables are related by definition. For instance, the total basehits variable (TEAM_BATTING_H) includes the numbers of doubles (TEAM_BATTING_2B), triples (TEAM_BATTING_3B), and homeruns (TEAM_BATTING_HR), which may explain why the basehits variable is not significant and why the doubles coefficient is negative. Likewise, the hits allowed variable (TEAM_PITCHING_H) includes the number of homeruns allowed (TEAM_PITCHING_HR), which again may explain why the hits allowed variable is not significant and has a counter-intuitive sign.

Inferences: Inferences from the model such as predicted mean values, confidence intervals, and prediction intervals for the target variable can be made using the `predict` function. One simply needs to input the relevant values of the predictor variables for which inferences are desired. For instance, in the next section, we use `predict` to generate predicted mean values for the target variable based on the evaluation dataset. If we were to specify the interval type ("confidence" or "prediction"), the function would also return the respective intervals.

Predicted Wins for the Evaluation Dataset

Now that we've chosen our champion model, we can use it to predict the number of wins for the evaluation dataset. First, we have to prepare the dataset using the same procedure followed above for the training dataset, in order to run it through the model. In particular, we use mean imputation to substitute for any NA values, create indicator variables for missing values, and use a log transform on the fielding errors variable. Then we use the champion model to predict the target values (number of wins) and save this to disk.

TEAM_PITCHING_SO <dbl>	TEAM_FIELDING_E <dbl>	TEAM_FIELDING_DP <dbl>	HBP_missing <dbl>	CS_missing <dbl>	DP_missing <dbl>	PREDICT_WINS <dbl>
1080	140	156	1	0	0	60.03075
929	135	164	1	0	0	66.60366
816	156	153	1	0	0	71.04223
914	124	154	0	0	0	77.76732
1123	616	130	1	1	0	136.63955
736	572	105	1	1	0	75.94654
569	490	NA	1	1	1	75.08456
715	328	104	1	1	0	70.49248
734	226	132	1	1	0	75.65300
622	184	145	1	1	0	72.15180

1-10 of 259 rows | 15-21 of 20 columns   Previous 1 2 3 4 5 6 ... 26 Next

Figure 4.4

Future Work

Ideas for further work include:

a) Investigate outliers: Some variables exhibited extreme outliers (see TEAM_PITCHING_SO); these may be data errors. Also, it would be prudent to investigate whether outliers in the data for key variables are influential leverage points, which might skew the fitted model.

b) Explore variable transformations: As seen in the exploratory data analysis, certain variables have moderately to strongly skewed distributions. In these cases, it might be fruitful to experiment with a variety of variable transformations, including the Box-Cox method.

c) Assess missing data indicators: As discussed in the data preparation, certain variables had a significant proportion of missing values. In general, we opted to use mean imputation to substitute for these missing values, and for certain variables with a high proportion of missing values (>10%), we also used indicator variables. It would be interesting to assess the usefulness of these missing indicator variables, by including them and then excluding them in the models and evaluating their impact on model performance.

Appendix

R CODE

```
library(tidyverse)
library(funModeling)

gh <- "https://raw.githubusercontent.com/kecbenson/DATA621/master/HW1/"
file_train <- paste0(gh, "moneyball-training-data.csv")
file_test <- paste0(gh, "moneyball-evaluation-data.csv")

df_train <- read_csv(file_train)
df_test <- read_csv(file_test)

head(df_train)
str(df_train)
summary(df_train)
attach(df_train)

par(mfrow = c(1, 2))
hist(df_train$TARGET_WINS)
qqnorm(df_train$TARGET_WINS)
qqline(df_train$TARGET_WINS)


#pairs(df_train)

for (j in 2:ncol(df_train)) {
   hist(df_train[[j]], main = paste0("Histogram of ", colnames(df_train)[j]),
       xlab = colnames(df_train)[j], freq = FALSE)
   minval <- min(df_train[[j]], na.rm = TRUE)
   maxval <- max(df_train[[j]], na.rm = TRUE)
   meanval <- mean(df_train[[j]], na.rm = TRUE)
   sdval <- sd(df_train[[j]], na.rm = TRUE)
   grid <- minval:maxval
   lines(grid, dnorm(grid, mean = meanval, sd = sdval), lty = 3)

}
```

```
# batting variables (hits through home runs)
pairs(df_train[2:6])
# batting variables (walks, strikeouts, hit by pitch)
pairs(df_train[c(2, 7:8, 11)])
# baserun and fielding variables
pairs(df_train[c(2, 9:10, 16:17)])
# pitching variables
pairs(df_train[c(2, 12:15)])


# df_status function to show zero's and missing values
train_data_status <- df_status(df_train, print_results=FALSE)
# order by percentage of missing values
train_data_status[order(-train_data_status$p_na),]


# trigger a dummy variable if NA is present
df_train$HBP_missing <- ifelse(is.na(df_train$TEAM_BATTING_HBP), 1, 0)
# imputing NA to mean
df_train$TEAM_BATTING_HBP[is.na(df_train$TEAM_BATTING_HBP)] <-
mean(df_train$TEAM_BATTING_HBP, na.rm=TRUE)


# trigger a dummy variable if NA is present
df_train$CS_missing <- ifelse(is.na(df_train$TEAM_BASERUN_CS), 1, 0)
# imputing NA to mean
df_train$TEAM_BASERUN_CS[is.na(df_train$TEAM_BASERUN_CS)] <-
mean(df_train$TEAM_BASERUN_CS, na.rm=TRUE)


# trigger a dummy variable if NA is present
df_train$DP_missing <- ifelse(is.na(df_train$TEAM_FIELDING_DP), 1, 0)
# imputing NA to mean
df_train$TEAM_FIELDING_DP[is.na(df_train$TEAM_FIELDING_DP)] <-
mean(df_train$TEAM_FIELDING_DP, na.rm=TRUE)


# imputing mean value as a replacement to NA
df_train$TEAM_BASERUN_SB[is.na(df_train$TEAM_BASERUN_SB)] <-
mean(df_train$TEAM_BASERUN_SB, na.rm=TRUE)
df_train$TEAM_BATTING_SO[is.na(df_train$TEAM_BATTING_SO)] <-
mean(df_train$TEAM_BATTING_SO, na.rm=TRUE)
df_train$TEAM_PITCHING_SO[is.na(df_train$TEAM_PITCHING_SO)] <-
mean(df_train$TEAM_PITCHING_SO, na.rm=TRUE)
```

```
# results after imputation, we see all NA's are addressed
df_status(df_train)


# reviewing outliers
boxplot(df_train$TEAM_PITCHING_SO)
# reviewing outliers
boxplot(df_train$TEAM_PITCHING_H)
# reviewing outliers
boxplot(df_train$TEAM_PITCHING_BB)
# determining opportunity for transformation
hist(df_train$TEAM_FIELDING_E)


# assign the TEAM_PITCHING_SO outliers into a vector
outliers_SO <- boxplot(df_train$TEAM_PITCHING_SO, plot=FALSE)$out
# removing TEAM_PITCHING_SO outliers
df_train <- df_train[-which(df_train$TEAM_PITCHING_SO %in% outliers_SO),]
# demonstrating outliers removed, compared to above
boxplot(df_train$TEAM_PITCHING_SO)
# assign the TEAM_PITCHING_H outlier values into a vector
outliers_H <- boxplot(df_train$TEAM_PITCHING_H, plot=FALSE)$out
# removing TEAM_PITCHING_H outliers
df_train <- df_train[-which(df_train$TEAM_PITCHING_H %in% outliers_H),]
# demonstrating outliers removed, compared to above
boxplot(df_train$TEAM_PITCHING_H)
# assign the TEAM_PITCHING_BB outlier values into a vector
outliers_BB <- boxplot(df_train$TEAM_PITCHING_BB, plot=FALSE)$out
# removing TEAM_PITCHING_BB outliers
df_train <- df_train[-which(df_train$TEAM_PITCHING_BB %in% outliers_BB),]
# demonstrating outliers removed, compared to above
boxplot(df_train$TEAM_PITCHING_BB)


# performing log 10 transformation
df_train$TEAM_FIELDING_E = log10(df_train$TEAM_FIELDING_E)
# determining distribution of data
hist(df_train$TEAM_FIELDING_E)


#generating Model lm1 the base model
# all variables except the index
lm1 <- lm(TARGET_WINS ~ . - INDEX, df_train)
(lm1sum <- summary(lm1))
```

```
#generating Model lm2
lm2 <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_BATTING_HBP +
          TEAM_BASERUN_SB + TEAM_BASERUN_CS, df_train)
(lm2sum <- summary(lm2))

#generating Model lm3
lm3 <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B
+TEAM_BATTING_HR +
           TEAM_BATTING_BB + TEAM_BATTING_HBP + TEAM_BATTING_SO, df_train)
(lm3sum <- summary(lm3))


#generating Model lm4
lm4 <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_BATTING_BB +
          TEAM_BATTING_SO + TEAM_BASERUN_SB + TEAM_FIELDING_E +
          TEAM_PITCHING_BB + TEAM_PITCHING_H + TEAM_PITCHING_HR, df_train)
(lm4sum <- summary(lm4))

#generating Model lm5
lm5 <- lm(TARGET_WINS ~
          I(TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_BATTING_HBP) +
          I(TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR) +
          TEAM_BATTING_SO +
          I(TEAM_BASERUN_SB - TEAM_BASERUN_CS) +
          I(TEAM_PITCHING_H + TEAM_PITCHING_BB + TEAM_PITCHING_HR) +
          TEAM_PITCHING_SO +
          TEAM_FIELDING_E + TEAM_FIELDING_DP +
          HBP_missing + CS_missing + DP_missing, df_train)
(lm5sum <- summary(lm5))

#generating Model lm6
lm6 <- lm(TARGET_WINS ~
          I(TEAM_BATTING_H + TEAM_BATTING_BB
            - TEAM_PITCHING_H - TEAM_PITCHING_BB) +
          I(TEAM_BATTING_HR - TEAM_PITCHING_HR) +
          I(TEAM_BATTING_SO - TEAM_PITCHING_SO) +
          I(TEAM_BASERUN_SB - TEAM_BASERUN_CS) +
          TEAM_FIELDING_E + TEAM_FIELDING_DP +
          HBP_missing + CS_missing + DP_missing, df_train)
(lm6sum <- summary(lm6))
```

```
# list of models and model summaries
models <- list(lm1, lm2, lm3, lm4, lm5, lm6)
modsums <- list(lm1sum, lm2sum, lm3sum, lm4sum, lm5sum, lm6sum)
nmod <- length(modsums)

# storage variables
nvar <- integer(nmod)
sigma <- numeric(nmod)
rsq <- numeric(nmod)
adj_rsq <- numeric(nmod)
fstat <- numeric(nmod)
fstat_p <- numeric(nmod)
mse <- numeric(nmod)
rmse <- numeric(nmod)


# loop through model summaries
for (j in 1:nmod) {
  nvar[j] <- modsums[[j]]$df[1]
  sigma[j] <- modsums[[j]]$sigma
  rsq[j] <- modsums[[j]]$r.squared
  adj_rsq[j] <- modsums[[j]]$adj.r.squared
  fstat[j] <- modsums[[j]]$fstatistic[1]
  fstat_p[j] <- 1 - pf(modsums[[j]]$fstatistic[1], modsums[[j]]$fstatistic[2],
                  modsums[[j]]$fstatistic[3])
  mse[j] <- mean(modsums[[j]]$residuals^2)
  rmse[j] <- sqrt(mse[j])
}

modnames <- paste0("lm", c(1:nmod))

# evaluation dataframe
eval <- data.frame(Model = modnames,
              N_Vars = nvar,
              Sigma = sigma,
              R_Sq = rsq,
              Adj_R_Sq = adj_rsq,
              F_Stat = fstat,
              F_P_Val = fstat_p,
              MSE = mse,
              RMSE = rmse)

kable(eval, digits = 3, align = 'c', caption = 'Model Summary Statistics')
```

```
# champion model
champ <- lm1
# plot diagnostics
par(mfrow = c(2, 2))
plot(champ)


# list of variables in champion model
attach(df_train)
var_list <- list(TEAM_BATTING_H, TEAM_BATTING_2B, TEAM_BATTING_3B,
             TEAM_BATTING_HR, TEAM_BATTING_BB, TEAM_BATTING_HBP,
             TEAM_BATTING_SO, TEAM_BASERUN_SB, TEAM_BASERUN_CS,
             TEAM_PITCHING_H, TEAM_PITCHING_HR, TEAM_PITCHING_BB,
             TEAM_PITCHING_SO, TEAM_FIELDING_E, TEAM_FIELDING_DP)
var_names <- c("TEAM_BATTING_H", "TEAM_BATTING_2B", "TEAM_BATTING_3B",
            "TEAM_BATTING_HR", "TEAM_BATTING_BB", "TEAM_BATTING_HBP",
            "TEAM_BATTING_SO", "TEAM_BASERUN_SB", "TEAM_BASERUN_CS",
            "TEAM_PITCHING_H", "TEAM_PITCHING_HR", "TEAM_PITCHING_BB",
            "TEAM_PITCHING_SO", "TEAM_FIELDING_E", "TEAM_FIELDING_DP")
detach(df_train)


# plot standardized residuals vs predictor variables
par(mfrow = c(1, 2))
for (j in 1:length(var_list))
   plot(rstandard(champ) ~ var_list[[j]], ylab = "Standardized Residuals",
       xlab = var_names[j])


# view the test data
glimpse(df_test)
summary(df_test)


# prepare the data (same as done for training dataset):
#   - indicator variables for NA's
#   - mean imputation for NA's
#   - log10 transform for fielding_error variable


# indicator variables for NA's
df_test$HBP_missing <- ifelse(is.na(df_test$TEAM_BATTING_HBP), 1, 0)
df_test$CS_missing <- ifelse(is.na(df_test$TEAM_BASERUN_CS), 1, 0)
df_test$DP_missing <- ifelse(is.na(df_test$TEAM_FIELDING_DP), 1, 0)
```

```
# mean imputation for NA's
colavg <- colMeans(df_test, na.rm = TRUE)
df_test_prep <- df_test
for (j in 2:ncol(df_test))
   df_test_prep[is.na(df_test[ , j]), j] <- colavg[j]

# log10 transform for fielding_error variable
df_test_prep$TEAM_FIELDING_E <- log10(df_test_prep$TEAM_FIELDING_E)

# make predictions
predictions <- predict(champ, newdata = df_test_prep)
df_pred <- cbind(df_test, PREDICT_WINS = predictions)

# review the final test dataset
glimpse(df_pred)
df_pred

# save as csv file
write_csv(df_pred, "moneyball-predictions.csv")
```