DATA 621: Homework 1 (Baseball Regression)

Shoshana Farber, Josh Forster, Glen Davis, Andrew Bowen, Charles Ugiagbe

October 1, 2023

Setup:

First, let's read in the provided dataset.

Data Exploration:

```
## The dataset consists of 2276 observations of 17 variables.
```

The variables and their definitions can be seen below:

Variable	Definition
INDEX	Identification variable
TARGET_WINS	Number of wins
TEAM_BATTING_H	Base hits by batters (1B, 2B, 3B, HR)
TEAM_BATTING_2B	Doubles by batters (2B)
TEAM_BATTING_3B	Triples by batters (3B)
TEAM_BATTING_HR	Homeruns by batters (4B)
TEAM_BATTING_BB	Walks by batters
TEAM_BATTING_HBP	Batters hit by pitch (get a free base)
TEAM_BATTING_SO	Strikeouts by batters
TEAM_BASERUN_SB	Stolen bases
TEAM_BASRUN_CS	Caught stealing
TEAM_FIELDING_E	Errors
TEAM_FIELDING_DP	Double plays
TEAM_PITCHING_BB	Walks allowed
TEAM_PITCHING_H	Hits allowed
TEAM_PITCHING_HR	Homeruns allowed
TEAM_PITCHING_SO	Strikeouts by pitchers

INDEX is an identifying feature and should not be included in the linear regression model.

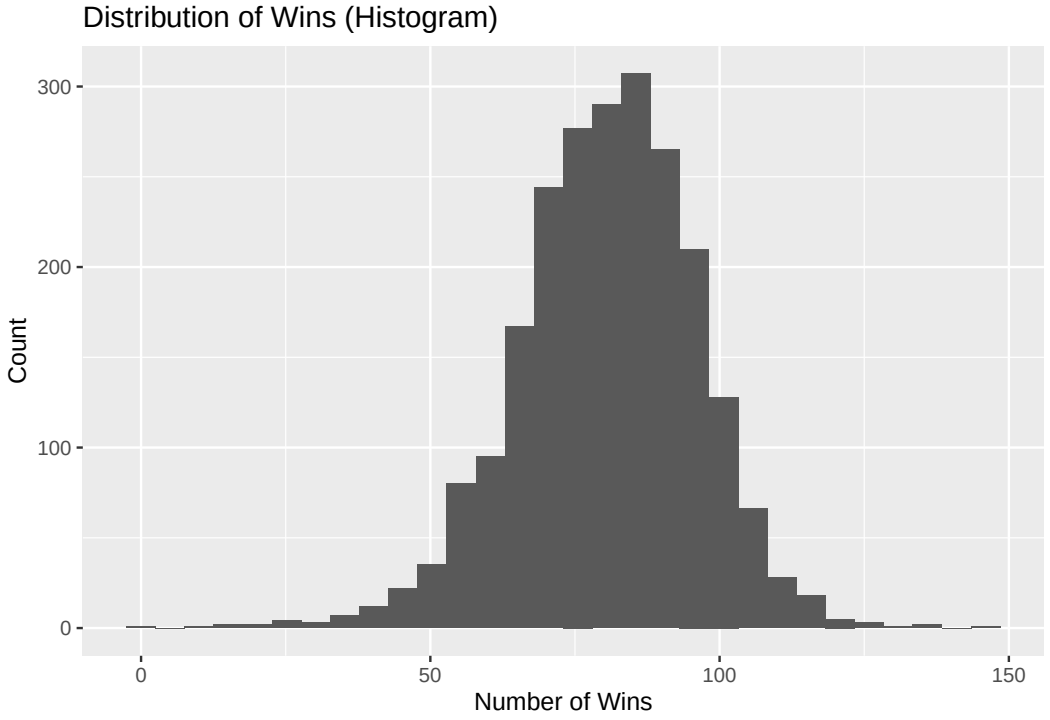
Next, let's print out some summary statistics. We're primarily interested in the TARGET_WINS variable, so we'll look at that first.

```
## The mean number of wins in a season is 80.79.
```

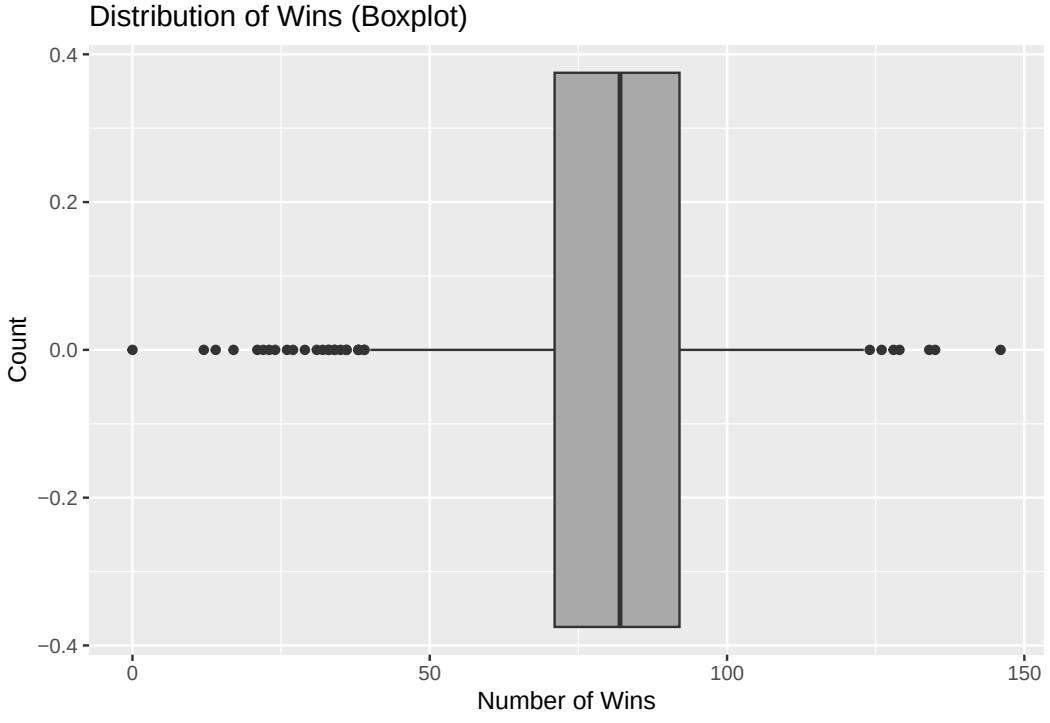
```
## The median number of wins in a season is 82.
```

```
## The standard deviation for number of wins in a season is 15.75.
```

Let's also make a histogram of the `TARGET_WINS` variable. This should give us a sense of the distribution of wins for teams/seasons in our population.



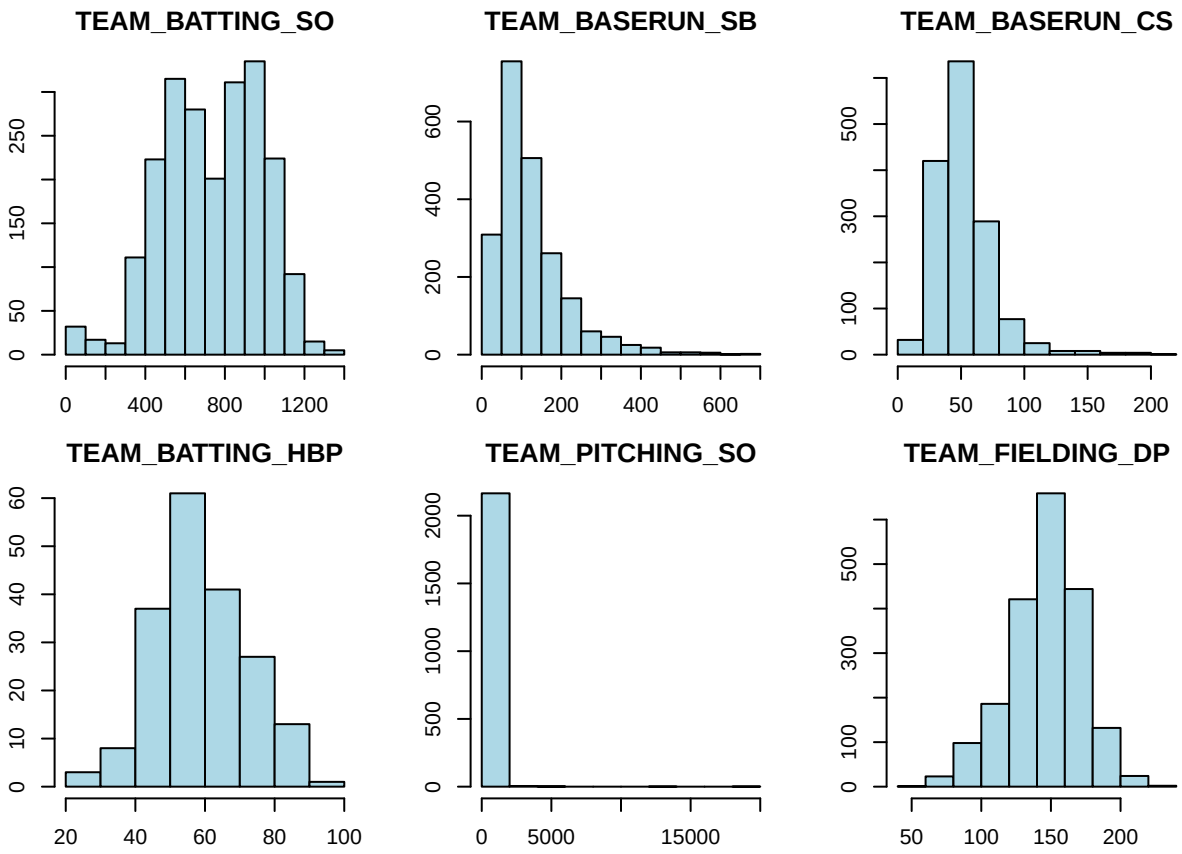
Overall, the number of wins in a season for a given baseball team looks fairly normally distributed. Looking at a boxplot helps to highlight the outliers.



Let's take a look at the summary statistics for all the variables:

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

We can see quite a few NA values for `TEAM_BATTING_SO`, `TEAM_BASERUN_SB`, `TEAM_BASERUN_CS`, `TEAM_BATTING_HBP`, `TEAM_PITCHING_SO`, and `TEAM_FIELDING_DP`. Let's take a look at the distributions of these variables to see how to impute these missing values.



TEAM_BASERUN_SB, TEAM_PITCHING_SO, and TEAM_BASERUN_CS seem to be skewed to the right so we should probably impute the missing values using the median value for these variables. TEAM_BATTING_HBP and TEAM_FIELDING_DP seem basically normally distributed so we can use the mean here, although TEAM_BATTING_HBP has 2,085 NA values out of 2,276 observations so it may make sense to leave this variable out of our model entirely. TEAM_BATTING_SO is bimodally distributed, so we have decided to use KNN imputation, which does not rely on the shape of the distribution, for this variable.

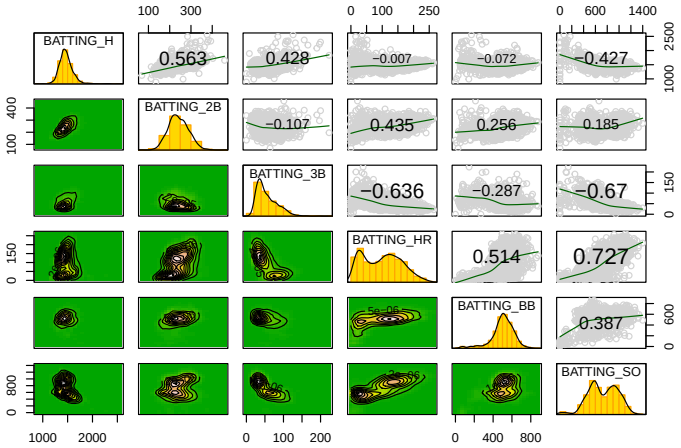
Let's look at raw correlations between our other included variables and a team's win total for a season:

```
##                         [,1]
## TARGET_WINS       1.00000000
## TEAM_BATTING_H    0.38876752
## TEAM_BATTING_2B   0.28910365
## TEAM_BATTING_3B   0.14260841
## TEAM_BATTING_HR   0.17615320
## TEAM_BATTING_BB   0.23255986
## TEAM_BATTING_SO  -0.03606403
## TEAM_BASERUN_SB   0.12361087
## TEAM_BASERUN_CS   0.01595982
## TEAM_PITCHING_H  -0.10993705
## TEAM_PITCHING_HR  0.18901373
## TEAM_PITCHING_BB  0.12417454
## TEAM_PITCHING_SO -0.07579967
## TEAM_FIELDING_E  -0.17648476
## TEAM_FIELDING_DP -0.02884126
```

None of the independent variables seem to have such high correlation with TARGET_WINS. TEAM_BATTING_H is

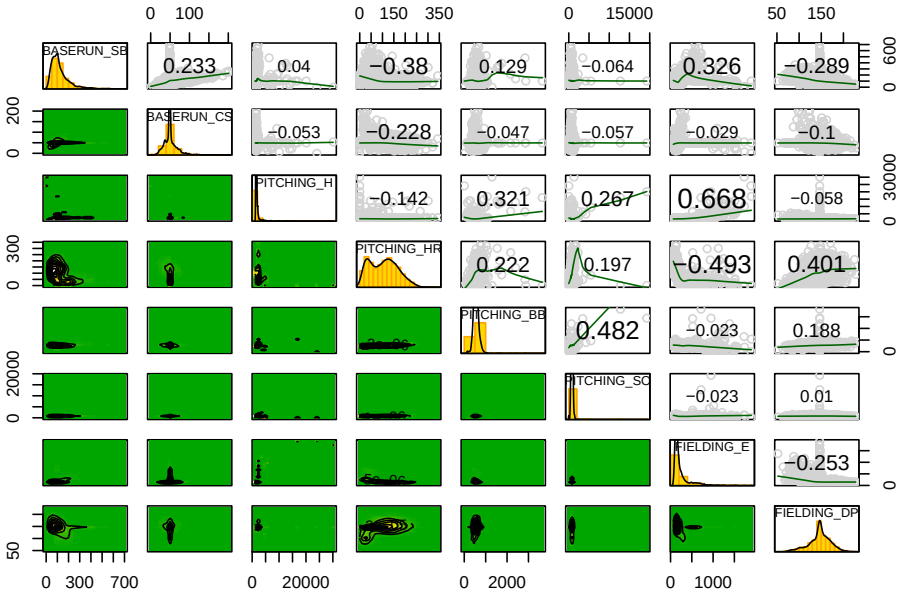
most highly correlated, with a correlation of 0.39. TEAM_BATTING_H, TEAM_BATTING_2B, TEAM_BATTING_3B, TEAM_BATTING_HR, TEAM_BATTING_BB, TEAM_BASERUN_SB, TEAM_BASERUN_CS, TEAM_PITCHING_HR, and TEAM_PITCHING_BB are all positively correlated with TARGET_WINS while TEAM_BATTING_SO, TEAM_PITCHING_H, TEAM_PITCHING_SO, TEAM_FIELDING_E, and TEAM_FIELDING_DP are negatively correlated.

Let's review relationships between batting independent variables.



Most of the batting variables appear to be somewhat approximately normal although there are some cases of right skew. Overall, there aren't any very strong correlations between these statistics at least from a preliminary visual inspection. From the distributions of these variables, we can see some that require transforming to normalize them before we use them in our linear model.

Let's review relationships between other independent variables.



There isn't very strong correlation between the other independent variables similar to the batting statistics although there are more examples of skewed data with these inputs. Once again, we can see that we will need to transform some of these variables.

Modeling

First, let's create a basic model with the untransformed variables:

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = train_imputed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.260  -8.612   0.151   8.425  59.018
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.657e+01  5.234e+00   5.078 4.14e-07 ***
## TEAM_BATTING_H    4.708e-02  3.699e-03  12.729  < 2e-16 ***
## TEAM_BATTING_2B  -1.788e-02  9.206e-03  -1.942 0.052245 .
## TEAM_BATTING_3B   6.137e-02  1.678e-02   3.657 0.000261 ***
## TEAM_BATTING_HR   5.752e-02  2.749e-02   2.093 0.036500 *
## TEAM_BATTING_BB   1.085e-02  5.816e-03   1.865 0.062310 .
## TEAM_BATTING_SO  -1.141e-02  2.579e-03  -4.427 1.00e-05 ***
## TEAM_BASERUN_SB   2.580e-02  4.317e-03   5.976 2.66e-09 ***
## TEAM_BASERUN_CS  -7.159e-03  1.577e-02  -0.454 0.649853
## TEAM_PITCHING_H  -8.980e-04  3.673e-04  -2.445 0.014562 *
## TEAM_PITCHING_HR  1.612e-02  2.431e-02   0.663 0.507243
## TEAM_PITCHING_BB -2.408e-05  4.124e-03  -0.006 0.995341
## TEAM_PITCHING_SO  3.201e-03  9.134e-04   3.505 0.000466 ***
## TEAM_FIELDING_E  -1.961e-02  2.448e-03  -8.011 1.80e-15 ***
## TEAM_FIELDING_DP -1.201e-01  1.293e-02  -9.290  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.05 on 2261 degrees of freedom
## Multiple R-squared:  0.3181, Adjusted R-squared:  0.3139
## F-statistic: 75.34 on 14 and 2261 DF,  p-value: < 2.2e-16
```

We can see that the R^2 value of a model that includes all the variables is not particularly high.

We can remove some variables that are not significant using backward step-wise elimination.

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO +
##     TEAM_BASERUN_SB + TEAM_PITCHING_H + TEAM_PITCHING_SO + TEAM_FIELDING_E +
##     TEAM_FIELDING_DP, data = train_imputed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -50.201  -8.548   0.137   8.404  59.080
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      25.5925419  5.0907238   5.027 5.36e-07 ***
## TEAM_BATTING_H    0.0473024  0.0036728  12.879  < 2e-16 ***
## TEAM_BATTING_2B  -0.0182083  0.0091927  -1.981 0.047742 *
## TEAM_BATTING_3B   0.0633643  0.0165996   3.817 0.000139 ***
## TEAM_BATTING_HR   0.0752404  0.0098361   7.649 2.97e-14 ***
## TEAM_BATTING_BB   0.0109356  0.0033639   3.251 0.001167 **
## TEAM_BATTING_SO  -0.0114146  0.0024962  -4.573 5.07e-06 ***
## TEAM_BASERUN_SB   0.0254110  0.0041873   6.069 1.51e-09 ***
## TEAM_PITCHING_H  -0.0008562  0.0003209  -2.669 0.007672 **
## TEAM_PITCHING_SO  0.0032329  0.0006703   4.823 1.51e-06 ***
## TEAM_FIELDING_E  -0.0192393  0.0023792  -8.086 9.91e-16 ***
## TEAM_FIELDING_DP -0.1201245  0.0129038  -9.309  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.04 on 2264 degrees of freedom
## Multiple R-squared:  0.3179, Adjusted R-squared:  0.3145
## F-statistic: 95.91 on 11 and 2264 DF,  p-value: < 2.2e-16
```

The R^2 for this model is not much imporoved. Let's check for multicollinearity between variables.

Reviewing the variance inflation factors:

```
##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_HR
##         3.772304         2.475869         2.876917         4.744128
##  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB  TEAM_PITCHING_H
##         2.277645         5.005090         1.710653         2.725441
## TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##         1.755773         3.928217         1.339341
```

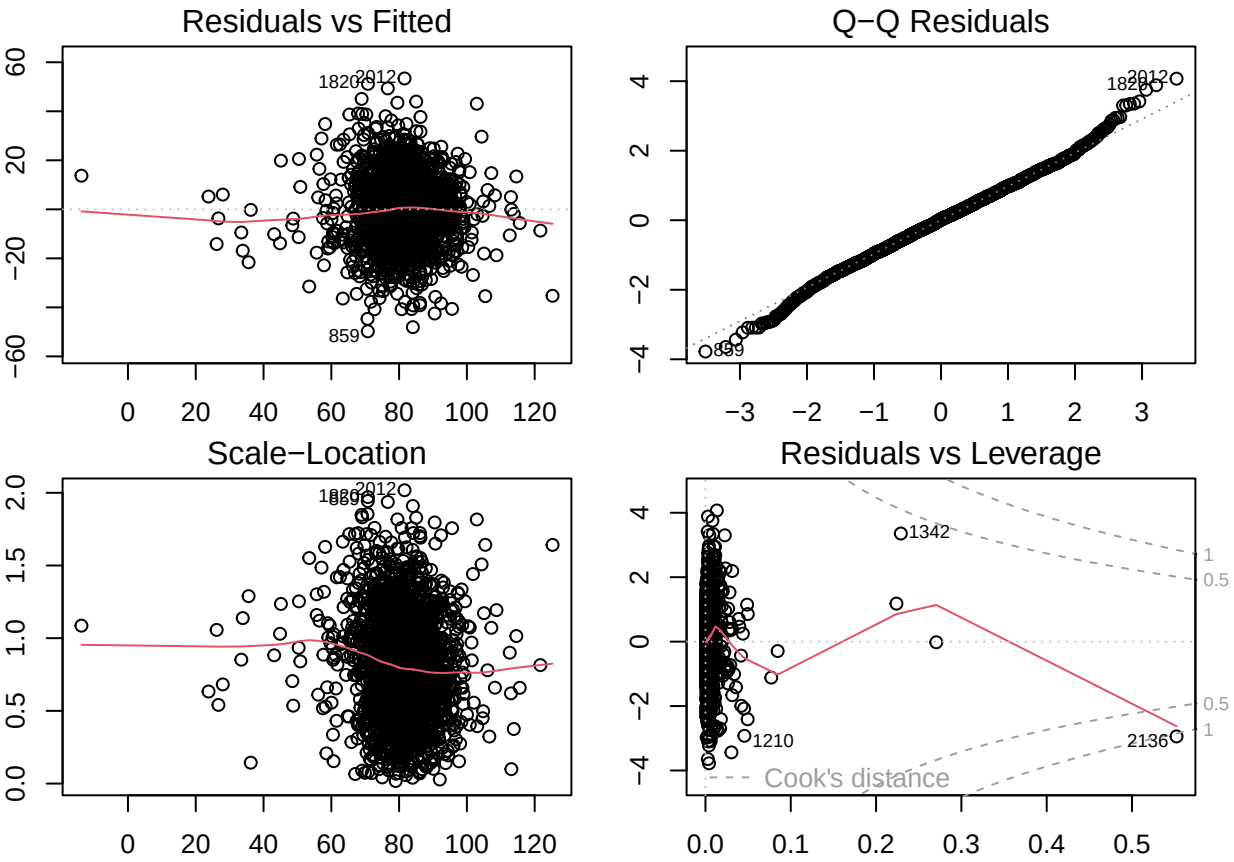
The variance inflation factors for `TEAM_BATTING_HR` and `TEAM_BATTING_SO` are greater than 4. We can remove these.

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_PITCHING_H +
##     TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP, data = train_imputed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.769  -8.557   0.284   8.688  53.397
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.9496247  3.8436725   1.548   0.1218
## TEAM_BATTING_H    0.0588461  0.0032943  17.863  < 2e-16 ***
## TEAM_BATTING_2B  -0.0200145  0.0089864  -2.227   0.0260 *
## TEAM_BATTING_3B   0.0245482  0.0144737   1.696   0.0900 .
## TEAM_BATTING_BB   0.0182069  0.0032668   5.573 2.80e-08 ***
```

```
## TEAM_BASERUN_SB   0.0177654  0.0040714   4.363 1.34e-05 ***
## TEAM_PITCHING_H  -0.0006359  0.0003150  -2.019   0.0436 *
## TEAM_PITCHING_SO  0.0026330  0.0006164   4.272 2.02e-05 ***
## TEAM_FIELDING_E  -0.0204497  0.0023866  -8.569  < 2e-16 ***
## TEAM_FIELDING_DP -0.1055554  0.0129132  -8.174 4.90e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.2 on 2266 degrees of freedom
## Multiple R-squared:  0.3002, Adjusted R-squared:  0.2974
## F-statistic:   108 on 9 and 2266 DF,  p-value: < 2.2e-16
```

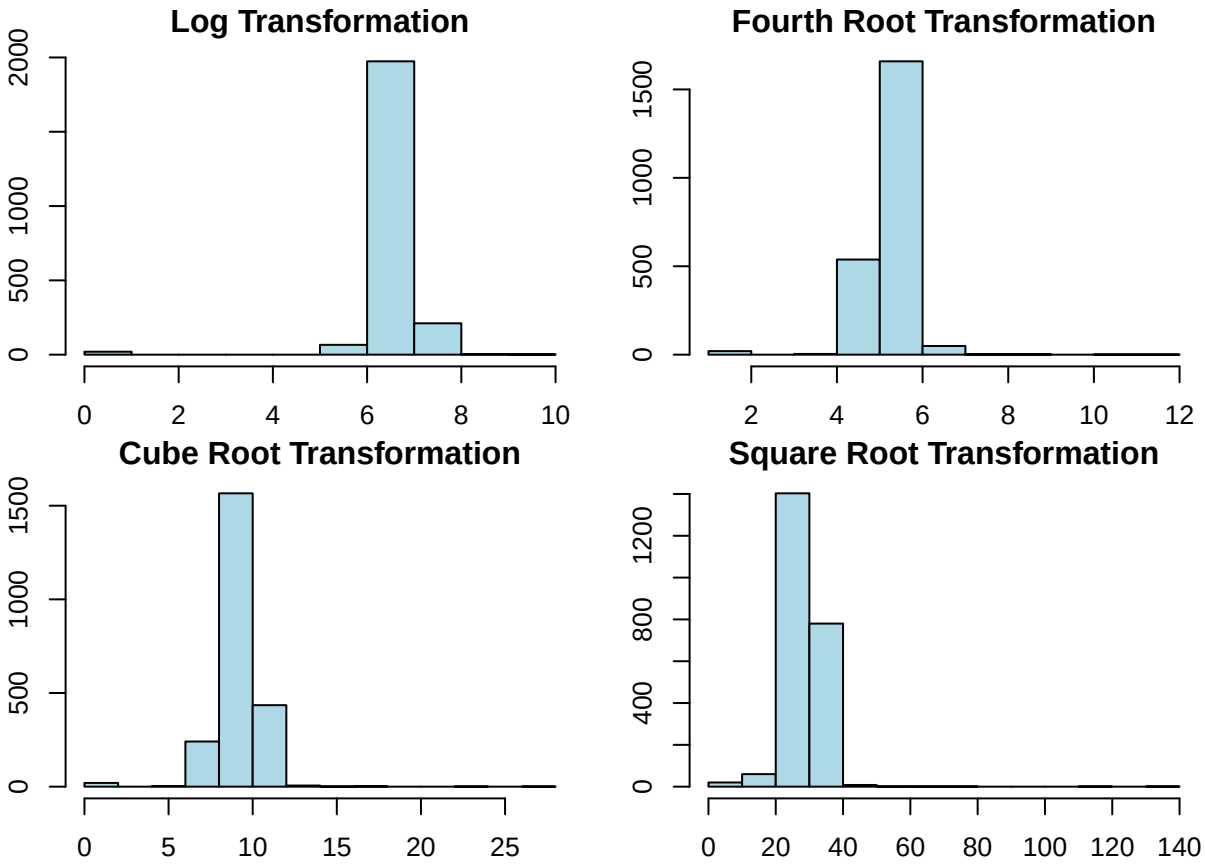
```
##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_BB
##         2.960992         2.308371         2.133943         2.095811
##  TEAM_BASERUN_SB  TEAM_PITCHING_H TEAM_PITCHING_SO  TEAM_FIELDING_E
##         1.577874         2.563471         1.448651         3.856179
## TEAM_FIELDING_DP
##         1.308614
```

We can make some plots to help test our assumptions of our basic model using the `plot` function on our model variable:



We can see from the diagnostic plots that there is definitely something up with this model. Let's try creating a better model by transforming some of the variables.

`TEAM_PITCHING_SO` is a right-skewed variable with very large outliers. Let's compare how four common transformations (log, fourth root, cube root, and square root) would normalize the distribution of this variable (after adding a small constant since the variable includes accurate values of 0).

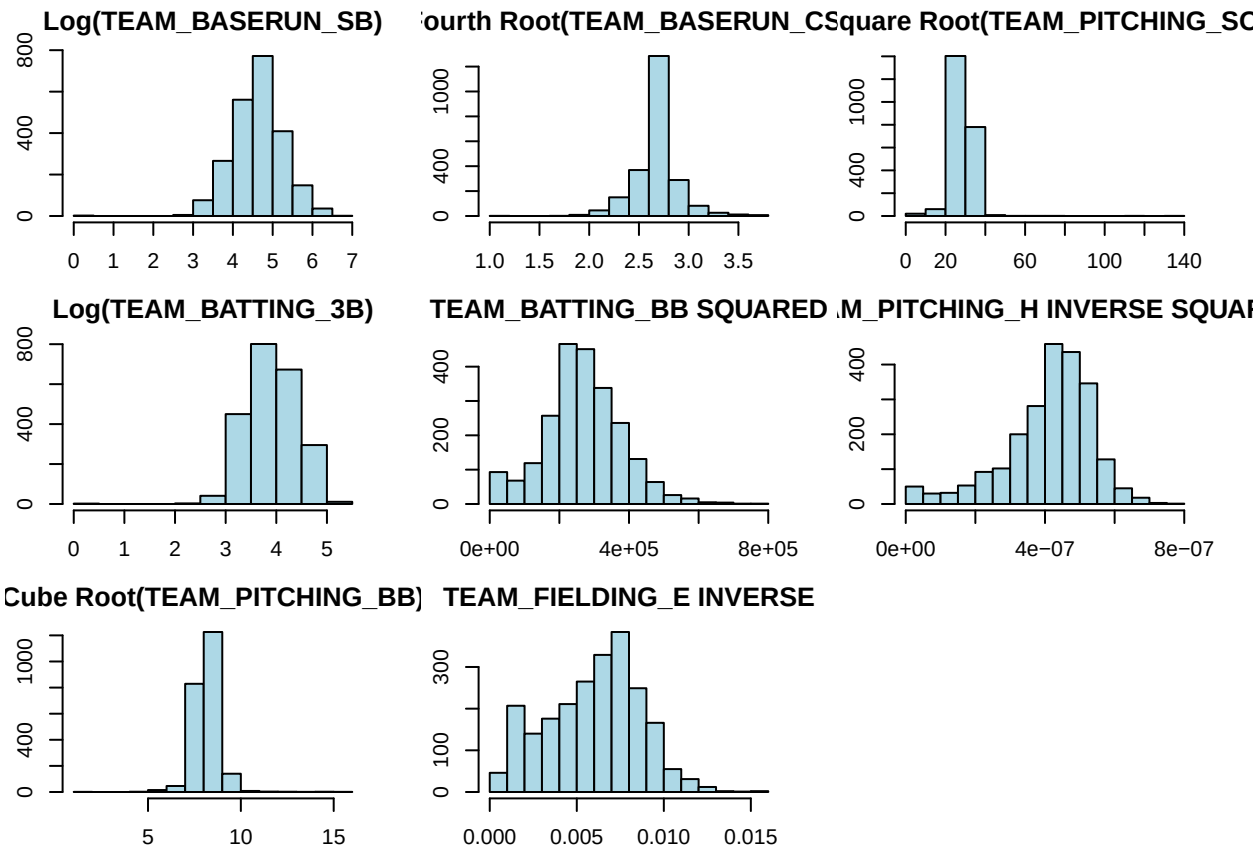


The square root transformation appears to normalize the data best. Let's confirm the ideal lambda proposed by the boxcox function from the MASS library is similar to to the square root transformation lambda (0.5) we assume will work best for this data.

```
## [1] 0.45
```

The proposed lambda of 0.45 is in fact very close to 0.5, so we will go with the easier to understand square root transformation. We will follow a similar process to find reasonable transformations for several other variables in our model without showing the process repeatedly.

variables	lambdas
TEAM_BASERUN_SB	0.1
TEAM_BASERUN_CS	0.2
TEAM_PITCHING_SO	0.45
TEAM_BATTING_3B	0.15
TEAM_BATTING_BB	1.8
TEAM_PITCHING_H	-2
TEAM_PITCHING_BB	0.3
TEAM_FIELDING_E	-0.95



Adjusting the ideal lambdas proposed for several variables to commonly understand transformations, we see mixed results on normalizing the distributions. However, we will attempt to use all these transformed variables in the model for now.

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = train_imputed_transformed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.614  -7.799   0.094   8.254  67.095
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.008e+01  8.329e+00  -1.210   0.2265
## TEAM_BATTING_H    4.628e-02  3.912e-03  11.828  < 2e-16 ***
## TEAM_BATTING_2B  -2.152e-02  9.249e-03  -2.326   0.0201 *
## TEAM_BATTING_3B   6.107e+00  8.813e-01   6.930 5.48e-12 ***
## TEAM_BATTING_HR   4.686e-02  2.755e-02   1.701   0.0891 .
## TEAM_BATTING_BB   4.614e-05  5.283e-06   8.734  < 2e-16 ***
## TEAM_BATTING_SO  -2.694e-02  3.531e-03  -7.629 3.46e-14 ***
## TEAM_BASERUN_SB   4.811e+00  5.861e-01   8.209 3.71e-16 ***
## TEAM_BASERUN_CS  -1.248e+00  1.450e+00  -0.861   0.3893
## TEAM_PITCHING_H   9.291e+06  4.518e+06   2.056   0.0399 *
## TEAM_PITCHING_HR  1.074e-02  2.438e-02   0.441   0.6595
## TEAM_PITCHING_BB -4.446e+00  8.817e-01  -5.042 4.97e-07 ***
```

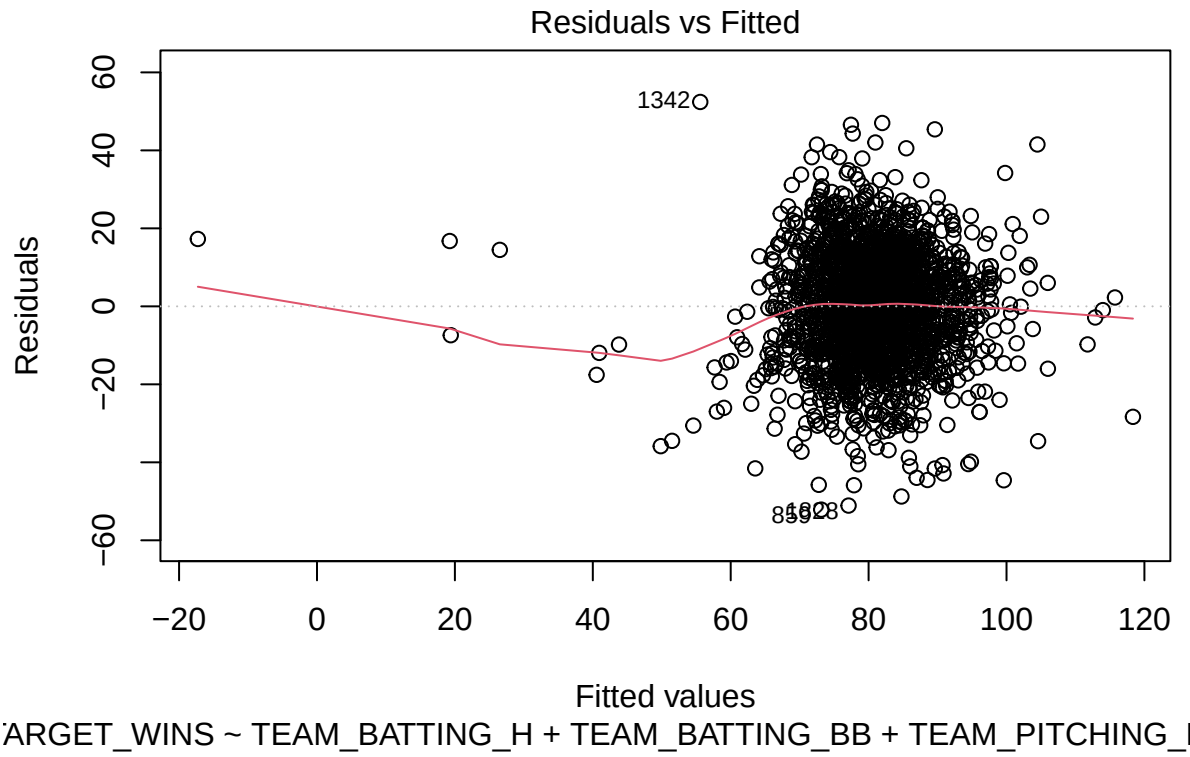
```
## TEAM_PITCHING_SO  7.626e-01  1.079e-01   7.071 2.05e-12 ***
## TEAM_FIELDING_E   2.231e+03  2.147e+02  10.391  < 2e-16 ***
## TEAM_FIELDING_DP -1.113e-01  1.319e-02  -8.439  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.95 on 2261 degrees of freedom
## Multiple R-squared:  0.3283, Adjusted R-squared:  0.3241
## F-statistic: 78.92 on 14 and 2261 DF,  p-value: < 2.2e-16
```

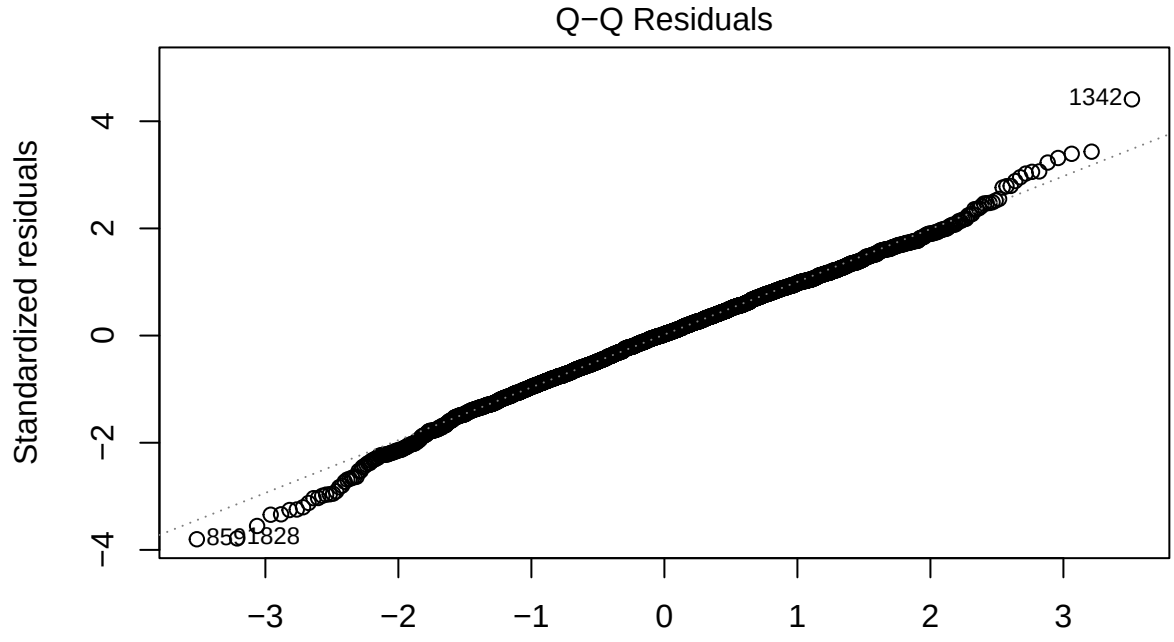
Now we can make a model with inputs that we know from baseball.

- Total hits (`TEAM_BATTING_H`)
- Total walks gained (`TEAM_BATTING_BB`)
- Total hits allowed (`TEAM_PITCHING_H`)
- Total walks allowed (`TEAM_PITCHING_BB`)

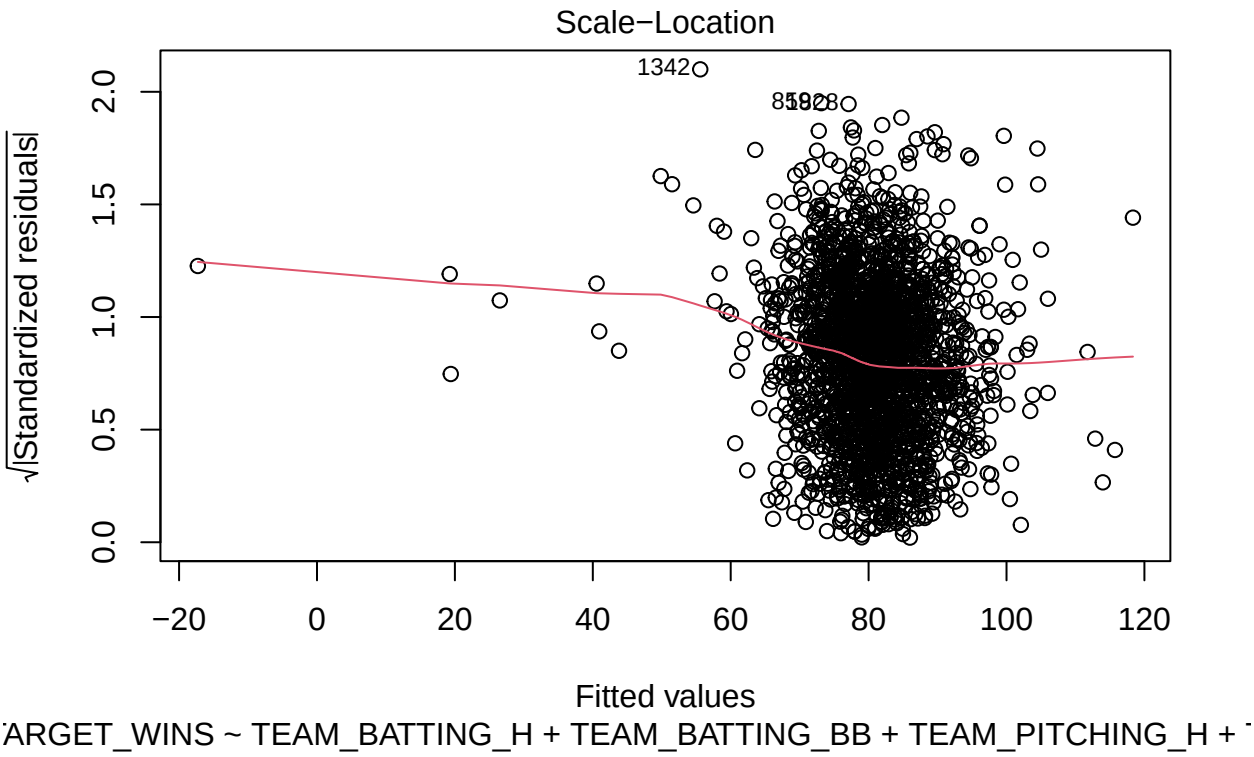
We chose these variables based on our understanding that good teams generally tend to get on base more frequently (positive predictor variables `TEAM_BATTING_HITS` and `TEAM_BATTING_BB`) while allowing *fewer* runners on base (negative predictor variables `TEAM_PITCHING_H` and `TEAM_PITCHING_BB`).

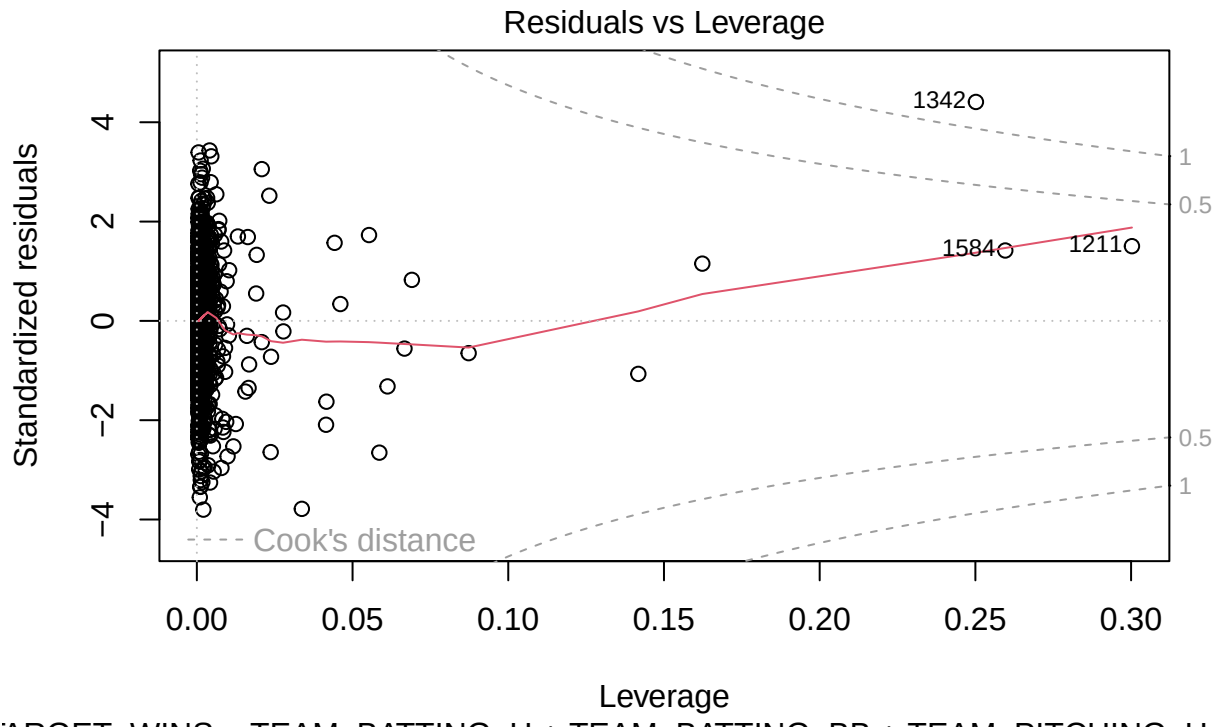
```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB +
##     TEAM_PITCHING_H + TEAM_PITCHING_BB, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.133  -8.860   0.379   9.373  52.416
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -0.3518000  3.2552864  -0.108 0.913949
## TEAM_BATTING_H    0.0497667  0.0021032  23.663  < 2e-16 ***
## TEAM_BATTING_BB   0.0148499  0.0039923   3.720 0.000204 ***
## TEAM_PITCHING_H  -0.0025469  0.0003317  -7.679 2.36e-14 ***
## TEAM_PITCHING_BB  0.0092317  0.0027681   3.335 0.000867 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.73 on 2271 degrees of freedom
## Multiple R-squared:  0.2416, Adjusted R-squared:  0.2403
## F-statistic: 180.9 on 4 and 2271 DF,  p-value: < 2.2e-16
```


Residuals vs Fitted
1342
Residuals
60
40
20
0
−20
−60
−20
0
20
40
60
80
100
120
Fitted values
ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_PITCHING_H +



ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_PITCHING_H +

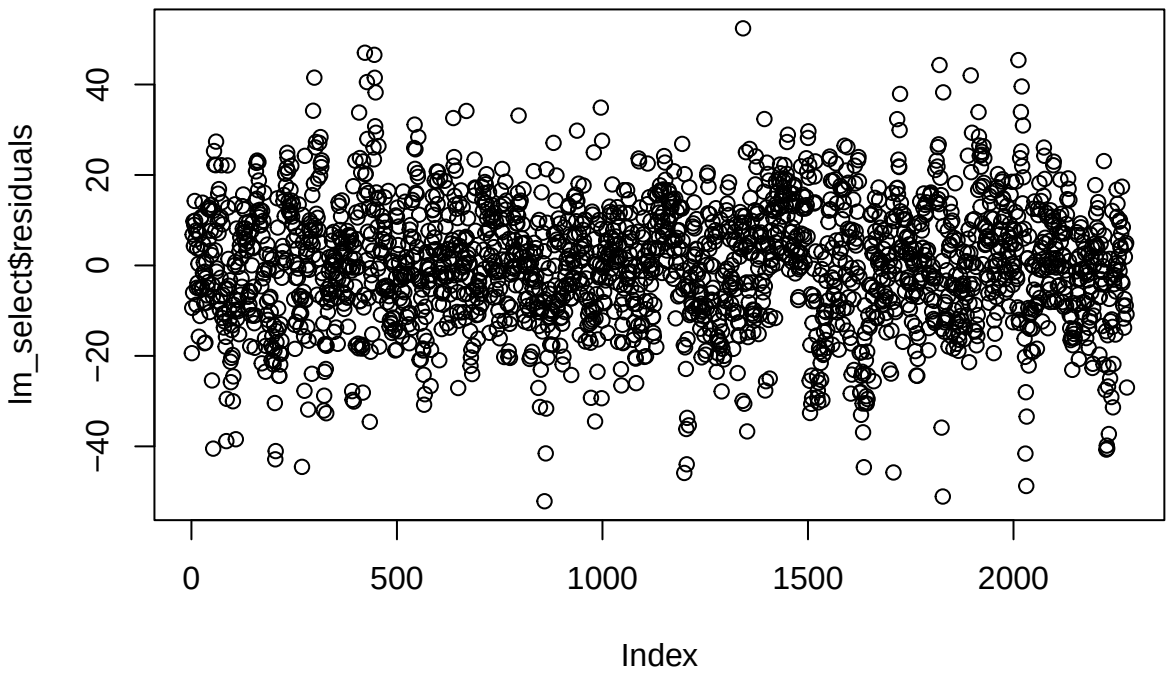

Scale−Location
1342
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
−20
0
20
40
60
80
100
120
Fitted values
ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_PITCHING_H +



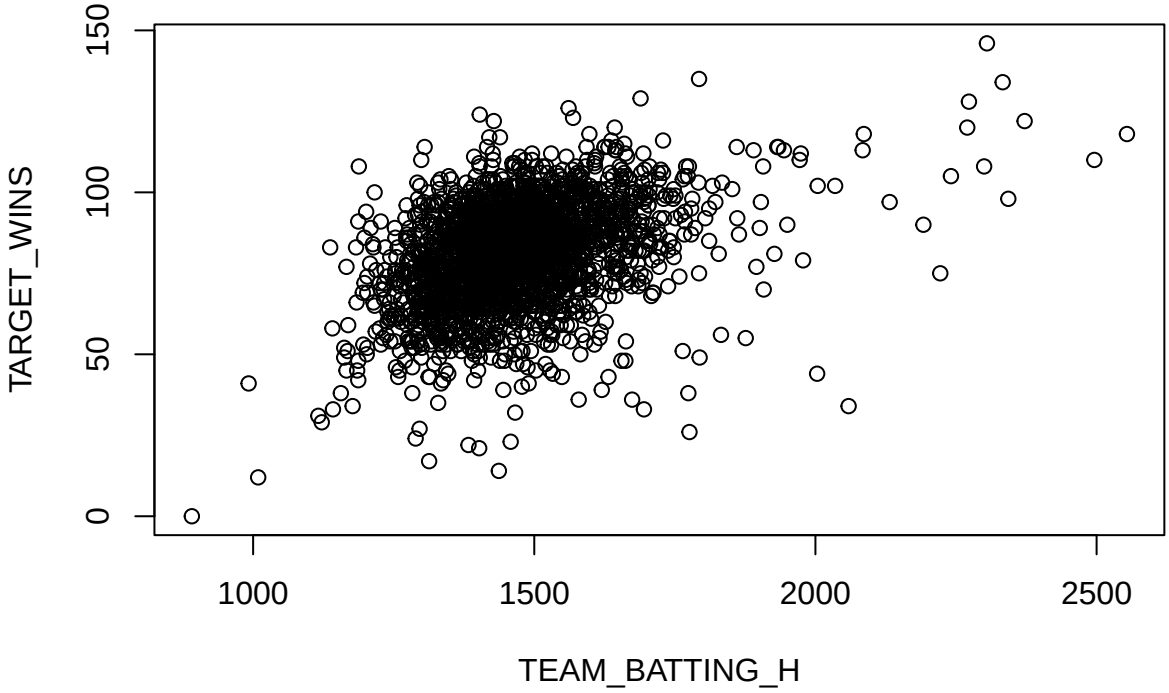
ARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_PITCHING_H +

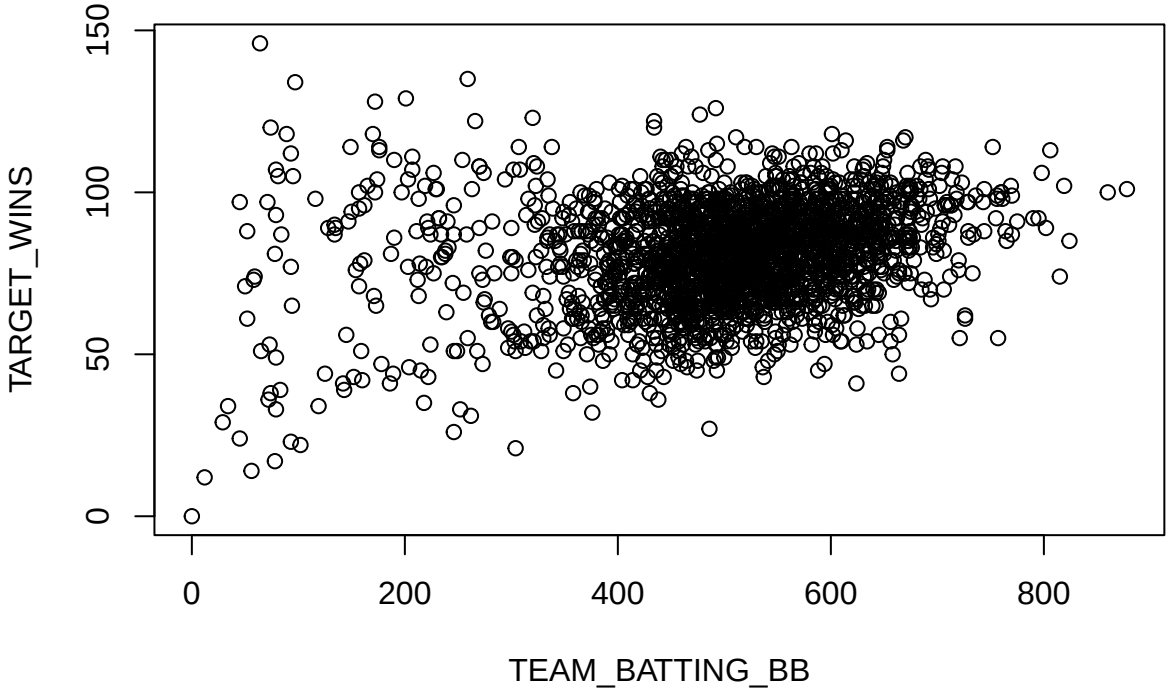
It's interesting to note that with selected variables (walks and hits gained/allowed per team) that our adjusted R^2 actually went *down*, indicating the amount of variability in `TARGET_WINS` explained by our more selective walks/hits model is *less* than the model including all variables.

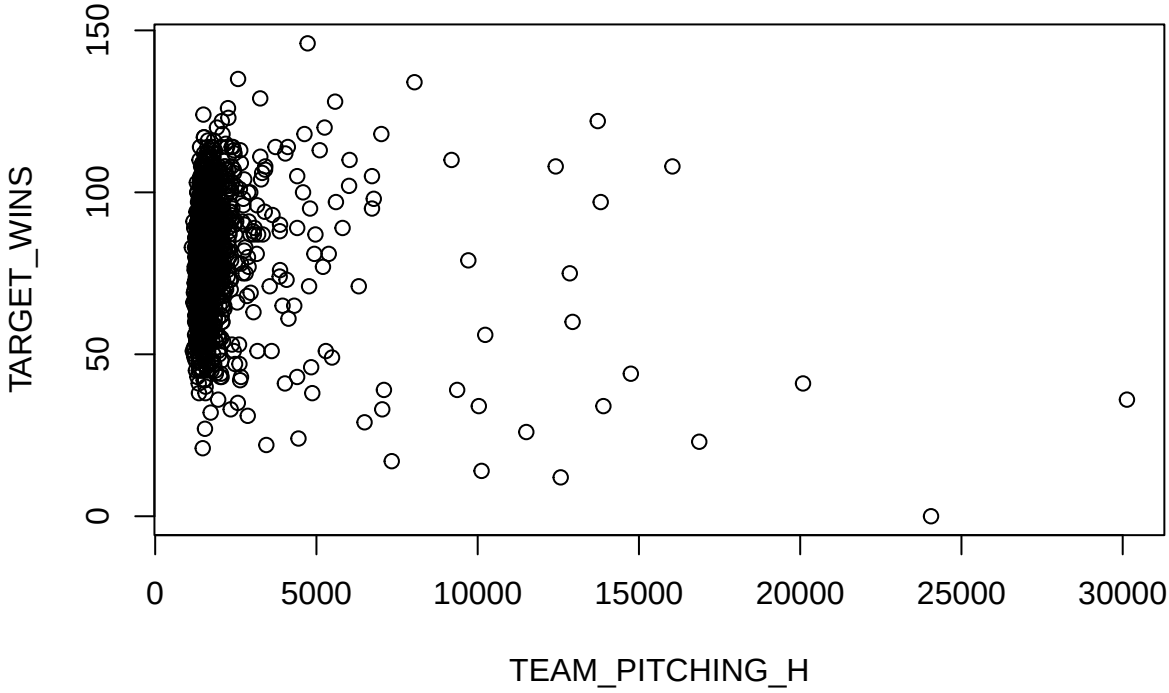
Looking at our residual plot above, there seems to be a clustering of residuals along the x-axis at $X \approx 80$. This shows a pattern in our residuals.

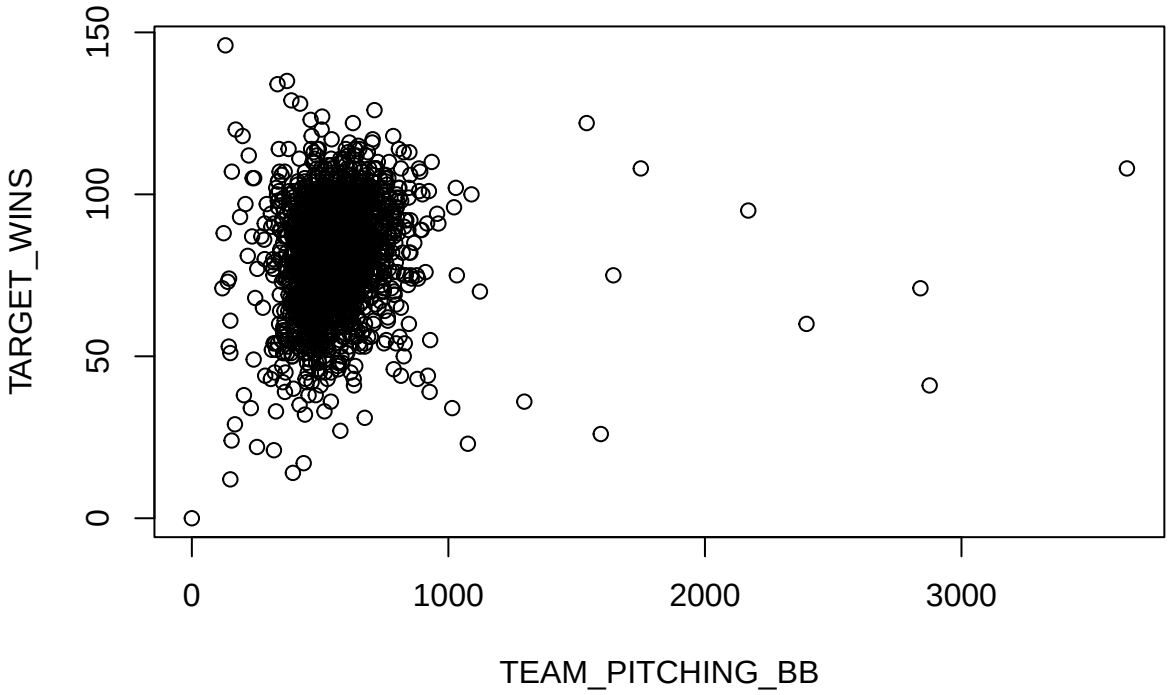


Let's plot our response variable (*Total Wins*) versus each of our predictor variables to get a sense of linear relationships.


TARGET_WINS
0
50
100
150
1000
1500
2000
2500
TEAM_BATTING_H


TARGET_WINS
0
50
100
150
0
200
400
600
800
TEAM_BATTING_BB


TARGET_WINS
0
50
100
150
0
5000
10000
15000
20000
25000
30000
TEAM_PITCHING_H



Model Evaluation:

We'll need to read in our evaluation data, which is hosted on GitHub for reproducability.

```
##        1        2        3        4        5        6        7        8
##  63.23752  65.25091  74.91662  85.85268       NA       NA       NA       NA
##        9       10       11       12       13       14       15       16
##       NA       NA       NA  83.24856  82.85230  83.18648  85.00740  77.70077
##       17       18       19       20       21       22       23       24
##  74.46689  78.45584       NA  91.80399  81.76776  84.12966  81.70425  73.00287
##       25       26       27       28       29       30       31       32
##  81.84364  86.66883       NA  75.85208  84.33511  76.05227  91.10315  85.74013
##       33       34       35       36       37       38       39       40
##  82.70689  85.25490  80.82577  87.26432  76.24566  90.68866  86.60458  92.95775
##       41       42       43       44       45       46       47       48
##  83.19709  90.60743       NA       NA       NA       NA       NA  76.97706
##       49       50       51       52       53       54       55       56
##       NA  79.50105       NA       NA  77.76447  74.16662  75.52183  78.52313
##       57       58       59       60       61       62       63       64
##       NA       NA       NA       NA       NA  74.03299  87.76094  85.45375
##       65       66       67       68       69       70       71       72
##  82.91149       NA       NA       NA       NA       NA       NA       NA
##       73       74       75       76       77       78       79       80
##  77.94184  90.08145  81.62238  85.63397  81.15230  83.00731       NA       NA
##       81       82       83       84       85       86       87       88
```

```
##  85.39582  89.07435  98.03313  75.17161  86.09314  80.53227  82.37750  83.10384
##        89        90        91        92        93        94        95        96
##  87.29584  89.35919        NA        NA        NA        NA        NA        NA
##        97        98        99       100       101       102       103       104
##  86.96781 103.77726  86.94561  86.77105  80.19620  75.20286  84.47147  84.74370
##       105       106       107       108       109       110       111       112
##  79.42791        NA        NA  77.08007  86.58275        NA  83.35088  83.55877
##       113       114       115       116       117       118       119       120
##  92.68745  90.95543  80.63567  77.76307  85.16347  80.04586  74.92314        NA
##       121       122       123       124       125       126       127       128
##        NA        NA        NA        NA        NA  88.66237  92.72478        NA
##       129       130       131       132       133       134       135       136
##        NA        NA        NA        NA  79.34647  85.33517  86.61573        NA
##       137       138       139       140       141       142       143       144
##  73.47446  77.51966  83.74624  79.78469        NA        NA  90.79628  74.41664
##       145       146       147       148       149       150       151       152
##  71.91864  72.68757  77.90232  78.52287  78.75137  82.96559  82.14958  79.51213
##       153       154       155       156       157       158       159       160
##        NA  71.06276  76.61382  69.96117  88.75587        NA        NA        NA
##       161       162       163       164       165       166       167       168
## 105.21553 107.03802  94.58154 104.81534  98.92161  90.15857  81.96876  81.09071
##       169       170       171       172       173       174       175       176
##  72.66974  80.29973        NA        NA  81.31855  94.23222  84.74389  73.87780
##       177       178       179       180       181       182       183       184
##  77.71069  71.54540  75.10454  79.10380  83.64464  88.15654  84.42018  85.42968
##       185       186       187       188       189       190       191       192
##        NA        NA        NA        NA        NA        NA        NA        NA
##       193       194       195       196       197       198       199       200
##  77.28965        NA        NA        NA        NA  84.89327  80.43320  84.98327
##       201       202       203       204       205       206       207       208
##  76.72715  79.90256  73.82346  88.74263  79.88244  82.85087  77.86710  77.23749
##       209       210       211       212       213       214       215       216
##        NA        NA        NA        NA        NA  66.02584  69.64708  84.40647
##       217       218       219       220       221       222       223       224
##  79.54017  90.95238  77.09353  78.09751  78.27320  73.38682  81.35624  73.27862
##       225       226       227       228       229       230       231       232
##        NA  74.71051  81.28468  79.14501  81.36641        NA        NA  92.30581
##       233       234       235       236       237       238       239       240
##        NA  89.06323  80.49576  75.17030  82.51871  77.07366        NA        NA
##       241       242       243       244       245       246       247       248
##        NA        NA        NA  80.63502  60.67621  86.87559  80.93490  84.82017
##       249       250       251       252       253       254       255       256
##  73.10639  82.51300  81.29380        NA        NA        NA  69.32771  76.62011
##       257       258       259
##  81.03232  81.93184  77.65088
```

Appendix: Report Code

Below is the code for this report to generate the models and charts above.

```
knitr::opts_chunk$set(echo = FALSE, warning = FALSE, message = FALSE)
library(glue)
```

```
library(tidyverse)
library(car)
library(ResourceSelection)
library(VIM)
library(pracma)
library(MASS)
select <- dplyr::select
library(knitr)

df <- read.csv("https://raw.githubusercontent.com/andrewbowen19/businessAnalyticsDataMiningDATA621/main/
df <- data.frame(df)
dim = dim(df)

print(glue("The dataset consists of {dim[1]} observations of {dim[2]} variables."))
train <- subset(df, select=-c(INDEX))
mean_wins <- mean(train$TARGET_WINS)
median_wins <- median(train$TARGET_WINS)
sd_wins <- sd(train$TARGET_WINS)

# Print summary stats
print(glue("The mean number of wins in a season is {round(mean_wins,2)}."))
print(glue("The median number of wins in a season is {median_wins}."))
print(glue("The standard deviation for number of wins in a season is {round(sd_wins,2)}."))
ggplot(train, aes(x=TARGET_WINS)) +
  geom_histogram() +
  labs(title = "Distribution of Wins (Histogram)", x = "Number of Wins", y = "Count")
ggplot(train, aes(x=TARGET_WINS)) +
  geom_boxplot(fill="darkgrey") +
  labs(title = "Distribution of Wins (Boxplot)", x = "Number of Wins", y = "Count")
summary(train)
par(mfrow=c(2,3))
par(mai=c(.3,.3,.3,.3))

variables <- c("TEAM_BATTING_SO", "TEAM_BASERUN_SB", "TEAM_BASERUN_CS", "TEAM_BATTING_HBP", "TEAM_PITCH

for (i in 1:(length(variables))) {
    hist(train[[variables[i]]], main = variables[i], col = "lightblue")
}
train_imputed <- train |>
    mutate(TEAM_BASERUN_SB = replace(TEAM_BASERUN_SB, is.na(TEAM_BASERUN_SB),
                                     median(TEAM_BASERUN_SB, na.rm=T)),
           TEAM_BASERUN_CS = replace(TEAM_BASERUN_CS, is.na(TEAM_BASERUN_CS),
                                     median(TEAM_BASERUN_CS, na.rm=T)),
           TEAM_PITCHING_SO = replace(TEAM_PITCHING_SO, is.na(TEAM_PITCHING_SO),
                                      median(TEAM_PITCHING_SO, na.rm=T)),
           TEAM_FIELDING_DP = replace(TEAM_FIELDING_DP, is.na(TEAM_FIELDING_DP),
                                      mean(TEAM_FIELDING_DP, na.rm=T))) |>
    select(-TEAM_BATTING_HBP)


train_imputed <- train_imputed |>
    VIM::kNN(variable = "TEAM_BATTING_SO", k = 15, numFun = weighted.mean,
             weightDist = TRUE) |>
```

```
    select(-TEAM_BATTING_SO_imp)

cor(train_imputed, df$TARGET_WINS)
train_cleaned <- train_imputed |> rename_all(~stringr::str_replace(.,"^TEAM_",""))
subset_batting <- train_cleaned |> select(contains('batting'))
kdepairs(subset_batting)
subset_pitching <- train_cleaned |> select(!contains('batting'), -TARGET_WINS)
kdepairs(subset_pitching)
lm_all <- lm(TARGET_WINS~., train_imputed)
summary(lm_all)
lm_all_reduced <- step(lm_all, direction="backward", trace = 0)
summary(lm_all_reduced)
vif(lm_all_reduced)
lm_all_reduced <- update(lm_all_reduced, .~. - TEAM_BATTING_HR - TEAM_BATTING_SO)
summary(lm_all_reduced)
vif(lm_all_reduced)
par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
plot(lm_all_reduced)
train_imputed_transformed <- train_imputed
#Add a small constant to TEAM_PITCHING_SO so there are no 0 values.
train_imputed_transformed$TEAM_PITCHING_SO <- train_imputed_transformed$TEAM_PITCHING_SO + 1
par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
#Compare how easy to understand transformations alter the distribution
hist(log(train_imputed_transformed$TEAM_PITCHING_SO),
     main = "Log Transformation", col="lightblue")
hist(nthroot(train_imputed_transformed$TEAM_PITCHING_SO, 4),
     main = "Fourth Root Transformation", col="lightblue")
hist(nthroot(train_imputed_transformed$TEAM_PITCHING_SO, 3),
     main = "Cube Root Transformation", col="lightblue")
hist(sqrt(train_imputed_transformed$TEAM_PITCHING_SO),
     main = "Square Root Transformation", col="lightblue")

bc <- boxcox(lm(train_imputed_transformed$TEAM_PITCHING_SO ~ 1),
             lambda = seq(-2, 2, length.out = 81),
             plotit = FALSE)
lambda <- bc$x[which.max(bc$y)]
lambda

variables <- c("TEAM_BASERUN_SB", "TEAM_BASERUN_CS", "TEAM_PITCHING_SO",
               "TEAM_BATTING_3B", "TEAM_BATTING_BB", "TEAM_PITCHING_H",
               "TEAM_PITCHING_BB", "TEAM_FIELDING_E")
for (i in 1:(length(variables))){
    #Add a small constant to columns with any 0 values
    if (sum(train_imputed_transformed[[variables[i]]] == 0) > 0){
        train_imputed_transformed[[variables[i]]] <-
            train_imputed_transformed[[variables[i]]] + 1
    }
}
for (i in 1:(length(variables))){
    if (i == 1){
        lambdas <- c()
```

```
    }
    bc <- boxcox(lm(train_imputed_transformed[[variables[i]]] ~ 1),
                 lambda = seq(-2, 2, length.out = 81),
                 plotit = FALSE)
    lambda <- bc$x[which.max(bc$y)]
    lambdas <- append(lambdas, lambda)
}
lambdas <- as.data.frame(cbind(variables, lambdas))
kable(lambdas, format = "simple")
par(mfrow=c(3, 3))
par(mai=c(.3,.3,.3,.3))
#Compare how easy to understand transformations alter the distribution
hist(log(train_imputed_transformed$TEAM_BASERUN_SB),
     main = "Log(TEAM_BASERUN_SB)", col="lightblue")
hist(nthroot(train_imputed_transformed$TEAM_BASERUN_CS, 4),
     main = "Fourth Root(TEAM_BASERUN_CS)", col="lightblue")
hist(sqrt(train_imputed_transformed$TEAM_PITCHING_SO),
     main = "Square Root(TEAM_PITCHING_SO)", col="lightblue")
hist(log(train_imputed_transformed$TEAM_BATTING_3B),
     main = "Log(TEAM_BATTING_3B)", col="lightblue")
hist(train_imputed_transformed$TEAM_BATTING_BB^2,
     main = "TEAM_BATTING_BB SQUARED", col="lightblue")
hist(train_imputed_transformed$TEAM_PITCHING_H^-2,
     main = "TEAM_PITCHING_H INVERSE SQUARED", col="lightblue")
hist(nthroot(train_imputed_transformed$TEAM_PITCHING_BB, 3),
     main = "Cube Root(TEAM_PITCHING_BB)", col="lightblue")
hist(train_imputed_transformed$TEAM_FIELDING_E^-1,
     main = "TEAM_FIELDING_E INVERSE", col="lightblue")

adj <- c(NA, 0.25, 0.5, NA, 2, -2, 0.33, -1)
lambdas <- cbind(lambdas, adj)
for (i in 1:(length(variables))){
    if (is.na(lambdas[[i, 3]])){
        train_imputed_transformed[[variables[i]]] <-
            log(train_imputed_transformed[[variables[i]]])
    }else{
        train_imputed_transformed[[variables[i]]] <-
            train_imputed_transformed[[variables[i]]]^lambdas[[i, 3]]
    }
}

lm_all_transformed <- lm(TARGET_WINS ~ ., train_imputed_transformed)
summary(lm_all_transformed)

# Create model with select inputs (walks and hits allowed/gained)
lm_select <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_PITCHING_H + TEAM_PITCHING_BB, tra

summary(lm_select)
plot(lm_select)

# Plot selective model residuals
plot(lm_select$residuals)
```

```r
# Plot our response variable for each predictor variable to get a sense of
plot(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_PITCHING_H + TEAM_PITCHING_BB, data=train)

eval_data_url <- "https://raw.githubusercontent.com/andrewbowen19/businessAnalyticsDataMiningDATA621/ma

test <- read.csv(eval_data_url)
predict(lm_all, test)
```