

Homework-1

Waheed Algabri, Joe Garcia, Lwin Shwe, Mikhail Broomes

Setup:

First, let's read in the provided dataset.

Data Exploration:

The original training dataset consists of 17 elements and a total of 2276 observations. Among these elements, INDEX is merely an index value utilized for sorting, while TARGET_WINS serves as the response variable for our regression models. The remaining 15 elements represent potential predictor variables for our linear models. Below is a summary table for the dataset.

variables	n	mean	sd	med	min	max	range	skew	kurtosis	se	NAs
TARGET_WINS	2276	81	16	82	0	146	146	-0.40	1.03	0.33	
TEAM_BATTING2B	1470	145	1454	891	2554	1663	1.57	7.28	3.03		
TEAM_BATTING2B1B	241	47	238	69	458	389	0.22	0.01	0.98		
TEAM_BATTING2B1B1R	55	28	47	0	223	223	1.11	1.50	0.59		
TEAM_BATTING2BHR	100	61	102	0	264	264	0.19	-0.96	1.27		
TEAM_BATTING2B3B	502	123	512	0	878	878	-1.03	2.18	2.57		
TEAM_BATTING2B1S0	736	240	750	0	1399	1399	-0.30	-0.32	5.33	102	
TEAM_BASERUN21BB	125	88	101	0	697	697	1.97	5.49	1.90	131	
TEAM_BASERUN150S	53	23	49	0	201	201	1.98	7.62	0.59	772	
TEAM_BATTING19HBP	59	13	58	29	95	66	0.32	-0.11	0.94	2085	
TEAM_PITCHINC21H	1779	1407	1518	1137	30132	28995	10.33	141.84	29.49		
TEAM_PITCHINC21HR	106	61	107	0	343	343	0.29	-0.60	1.28		
TEAM_PITCHINC21B	553	166	536	0	3645	3645	6.74	96.97	3.49		
TEAM_PITCHINC21SO	818	553	813	0	19278	19278	22.17	671.19	11.86	102	
TEAM_FIELDING22E	246	228	159	65	1898	1833	2.99	10.97	4.77		
TEAM_FIELDING19MP	146	26	149	52	228	176	-0.39	0.18	0.59	286	

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

Variable	Definition
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

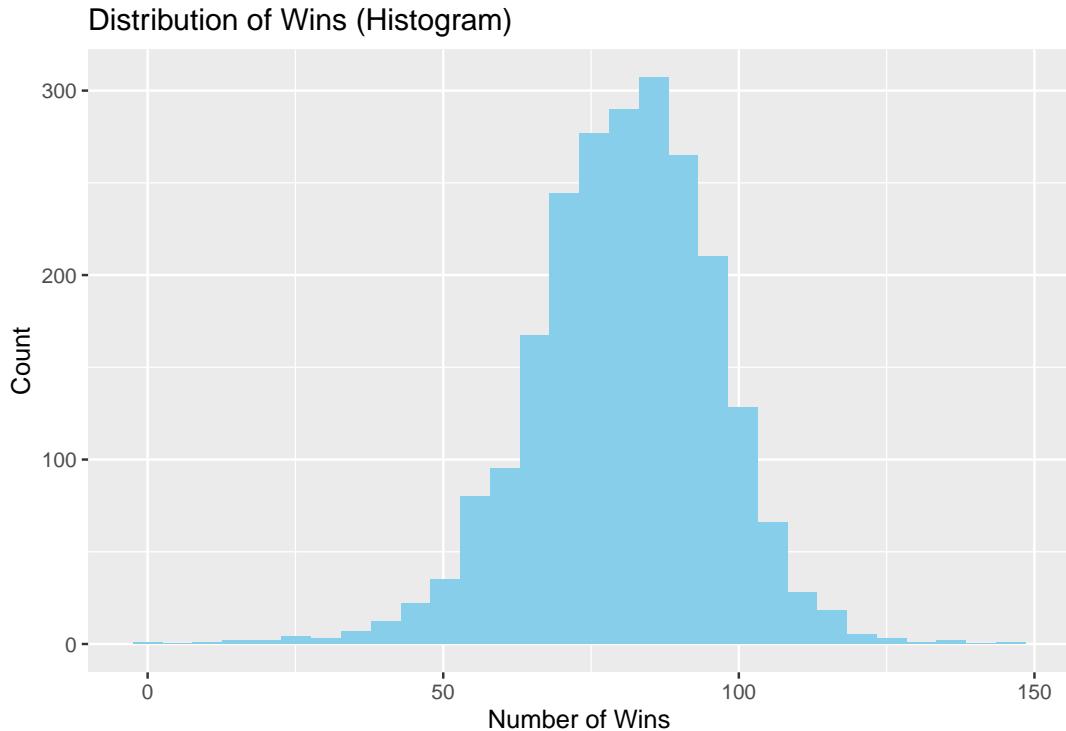
Next, let's print out some summary statistics. We're primarily interested in the TARGET_WINS variable, so we'll look at that first.

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

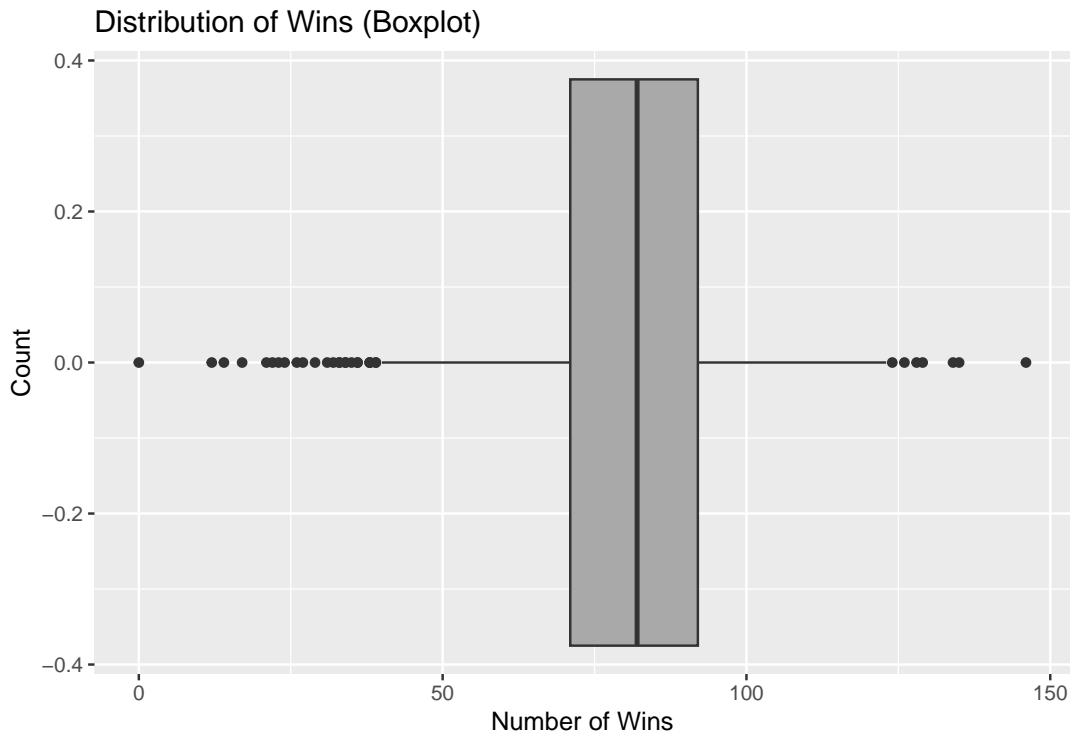
## The median number of wins in a season is 82.

## 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.



We could describe the average team's season using the mean for all variables below:

TARGET_WINS	80.8
TEAM_BATTING_H	1469.3
TEAM_BATTING_2B	241.2
TEAM_BATTING_3B	55.2
TEAM_BATTING_HR	99.6
TEAM_BATTING_BB	501.6
TEAM_BATTING_SO	735.6
TEAM_BASERUN_SB	124.8
TEAM_BASERUN_CS	52.8
TEAM_BATTING_HBP	59.4
TEAM_PITCHING_H	1779.2
TEAM_PITCHING_HR	105.7
TEAM_PITCHING_BB	553.0
TEAM_PITCHING_SO	817.7
TEAM_FIELDING_E	246.5
TEAM_FIELDING_DP	146.4

Data Preparation:

Let's take a closer look at all the summary statistics for these variables and identify any data completeness issues:

```
##   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

```

```

# First, store the summary in an object
summary_data <- summary(train)

# Create a data frame from the summary
summary_df <- as.data.frame(summary_data)

# Print the data frame
summary_df

```

	Var1	Var2	Freq
## 1	TARGET_WINS	Min. : 0.00	
## 2	TARGET_WINS	1st Qu.: 71.00	
## 3	TARGET_WINS	Median : 82.00	
## 4	TARGET_WINS	Mean : 80.79	
## 5	TARGET_WINS	3rd Qu.: 92.00	
## 6	TARGET_WINS	Max. :146.00	
## 7	TARGET_WINS	<NA>	
## 8	TEAM_BATTING_H	Min. : 891	
## 9	TEAM_BATTING_H	1st Qu.:1383	
## 10	TEAM_BATTING_H	Median :1454	
## 11	TEAM_BATTING_H	Mean :1469	
## 12	TEAM_BATTING_H	3rd Qu.:1537	
## 13	TEAM_BATTING_H	Max. :2554	
## 14	TEAM_BATTING_H	<NA>	
## 15	TEAM_BATTING_2B	Min. : 69.0	

```

## 16 TEAM_BATTING_2B 1st Qu.:208.0
## 17 TEAM_BATTING_2B Median :238.0
## 18 TEAM_BATTING_2B Mean   :241.2
## 19 TEAM_BATTING_2B 3rd Qu.:273.0
## 20 TEAM_BATTING_2B Max.   :458.0
## 21 TEAM_BATTING_2B <NA>
## 22 TEAM_BATTING_3B Min.   : 0.00
## 23 TEAM_BATTING_3B 1st Qu.: 34.00
## 24 TEAM_BATTING_3B Median : 47.00
## 25 TEAM_BATTING_3B Mean   : 55.25
## 26 TEAM_BATTING_3B 3rd Qu.: 72.00
## 27 TEAM_BATTING_3B Max.   :223.00
## 28 TEAM_BATTING_3B <NA>
## 29 TEAM_BATTING_HR Min.   : 0.00
## 30 TEAM_BATTING_HR 1st Qu.: 42.00
## 31 TEAM_BATTING_HR Median :102.00
## 32 TEAM_BATTING_HR Mean   : 99.61
## 33 TEAM_BATTING_HR 3rd Qu.:147.00
## 34 TEAM_BATTING_HR Max.   :264.00
## 35 TEAM_BATTING_HR <NA>
## 36 TEAM_BATTING_BB Min.   : 0.0
## 37 TEAM_BATTING_BB 1st Qu.:451.0
## 38 TEAM_BATTING_BB Median :512.0
## 39 TEAM_BATTING_BB Mean   :501.6
## 40 TEAM_BATTING_BB 3rd Qu.:580.0
## 41 TEAM_BATTING_BB Max.   :878.0
## 42 TEAM_BATTING_BB <NA>
## 43 TEAM_BATTING_SO Min.   : 0.0
## 44 TEAM_BATTING_SO 1st Qu.: 548.0
## 45 TEAM_BATTING_SO Median : 750.0
## 46 TEAM_BATTING_SO Mean   : 735.6
## 47 TEAM_BATTING_SO 3rd Qu.: 930.0
## 48 TEAM_BATTING_SO Max.   :1399.0
## 49 TEAM_BATTING_SO NA's   :102
## 50 TEAM_BASERUN_SB Min.   : 0.0
## 51 TEAM_BASERUN_SB 1st Qu.: 66.0
## 52 TEAM_BASERUN_SB Median :101.0
## 53 TEAM_BASERUN_SB Mean   :124.8
## 54 TEAM_BASERUN_SB 3rd Qu.:156.0
## 55 TEAM_BASERUN_SB Max.   :697.0
## 56 TEAM_BASERUN_SB NA's   :131
## 57 TEAM_BASERUN_CS Min.   : 0.0
## 58 TEAM_BASERUN_CS 1st Qu.: 38.0
## 59 TEAM_BASERUN_CS Median : 49.0
## 60 TEAM_BASERUN_CS Mean   : 52.8
## 61 TEAM_BASERUN_CS 3rd Qu.: 62.0
## 62 TEAM_BASERUN_CS Max.   :201.0
## 63 TEAM_BASERUN_CS NA's   :772
## 64 TEAM_BATTING_HBP Min.   :29.00
## 65 TEAM_BATTING_HBP 1st Qu.:50.50
## 66 TEAM_BATTING_HBP Median :58.00
## 67 TEAM_BATTING_HBP Mean   :59.36
## 68 TEAM_BATTING_HBP 3rd Qu.:67.00
## 69 TEAM_BATTING_HBP Max.   :95.00

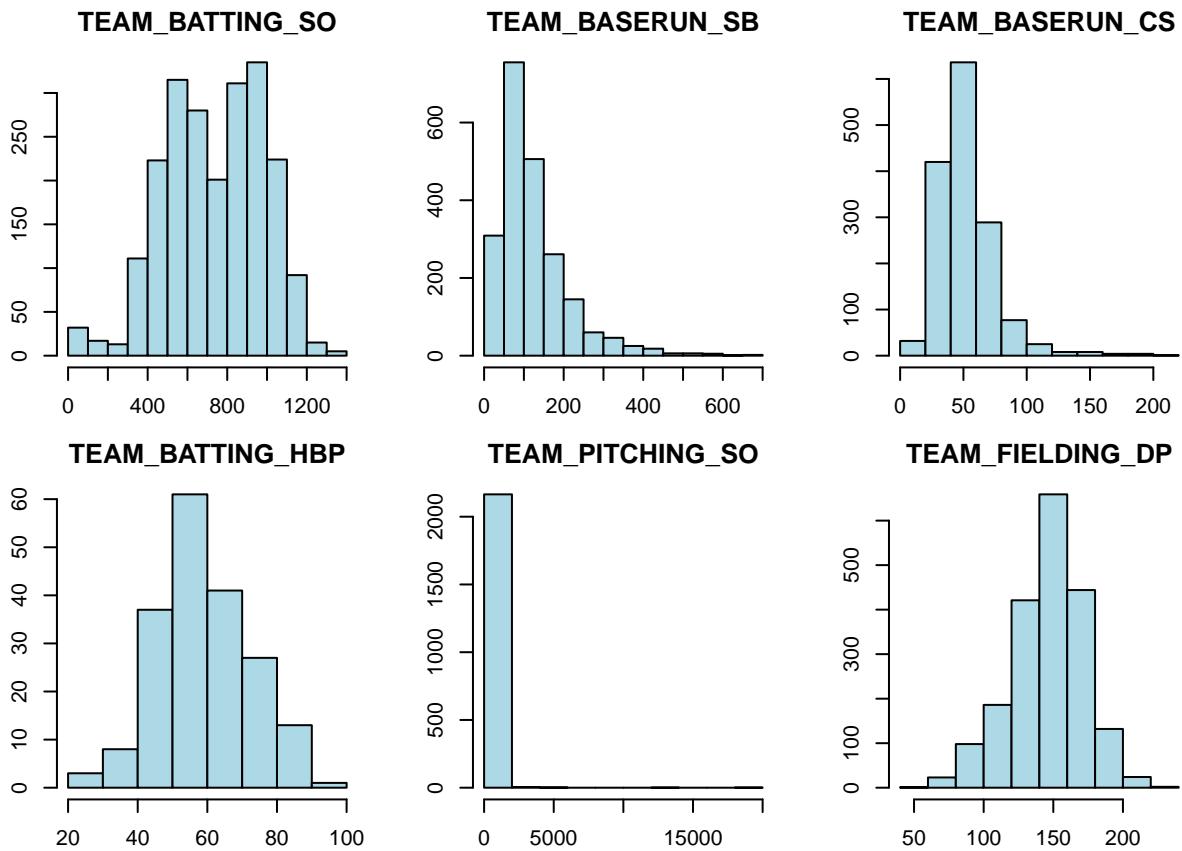
```

```

## 70      TEAM_BATTING_HBP    NA's   :2085
## 71      TEAM_PITCHING_H    Min.   : 1137
## 72      TEAM_PITCHING_H    1st Qu.: 1419
## 73      TEAM_PITCHING_H    Median : 1518
## 74      TEAM_PITCHING_H    Mean   : 1779
## 75      TEAM_PITCHING_H    3rd Qu.: 1682
## 76      TEAM_PITCHING_H    Max.   :30132
## 77      TEAM_PITCHING_H    <NA>
## 78      TEAM_PITCHING_HR   Min.   : 0.0
## 79      TEAM_PITCHING_HR   1st Qu.: 50.0
## 80      TEAM_PITCHING_HR   Median :107.0
## 81      TEAM_PITCHING_HR   Mean   :105.7
## 82      TEAM_PITCHING_HR   3rd Qu.:150.0
## 83      TEAM_PITCHING_HR   Max.   :343.0
## 84      TEAM_PITCHING_HR   <NA>
## 85      TEAM_PITCHING_BB   Min.   : 0.0
## 86      TEAM_PITCHING_BB   1st Qu.: 476.0
## 87      TEAM_PITCHING_BB   Median : 536.5
## 88      TEAM_PITCHING_BB   Mean   : 553.0
## 89      TEAM_PITCHING_BB   3rd Qu.: 611.0
## 90      TEAM_PITCHING_BB   Max.   :3645.0
## 91      TEAM_PITCHING_BB   <NA>
## 92      TEAM_PITCHING_SO   Min.   : 0.0
## 93      TEAM_PITCHING_SO   1st Qu.: 615.0
## 94      TEAM_PITCHING_SO   Median : 813.5
## 95      TEAM_PITCHING_SO   Mean   : 817.7
## 96      TEAM_PITCHING_SO   3rd Qu.: 968.0
## 97      TEAM_PITCHING_SO   Max.   :19278.0
## 98      TEAM_PITCHING_SO   NA's   :102
## 99      TEAM_FIELDING_E    Min.   : 65.0
## 100     TEAM_FIELDING_E   1st Qu.: 127.0
## 101     TEAM_FIELDING_E   Median : 159.0
## 102     TEAM_FIELDING_E   Mean   : 246.5
## 103     TEAM_FIELDING_E   3rd Qu.: 249.2
## 104     TEAM_FIELDING_E   Max.   :1898.0
## 105     TEAM_FIELDING_E   <NA>
## 106     TEAM_FIELDING_DP   Min.   : 52.0
## 107     TEAM_FIELDING_DP   1st Qu.:131.0
## 108     TEAM_FIELDING_DP   Median :149.0
## 109     TEAM_FIELDING_DP   Mean   :146.4
## 110     TEAM_FIELDING_DP   3rd Qu.:164.0
## 111     TEAM_FIELDING_DP   Max.   :228.0
## 112     TEAM_FIELDING_DP   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.

Identify missing data

```
# Identify missing data by Feature and display percent breakout
missing <- colSums(df %>% sapply(is.na))
missing_pct <- round(missing / nrow(df) * 100, 2)
stack(sort(missing_pct, decreasing = TRUE))
```

```
##      values           ind
## 1    91.61 TEAM_BATTING_HBP
## 2    33.92 TEAM_BASERUN_CS
## 3    12.57 TEAM_FIELDING_DP
## 4     5.76 TEAM_BASERUN_SB
## 5     4.48 TEAM_BATTING_SO
## 6     4.48 TEAM_PITCHING_SO
## 7     0.00      INDEX
## 8     0.00 TARGET_WINS
## 9     0.00 TEAM_BATTING_H
## 10    0.00 TEAM_BATTING_2B
## 11    0.00 TEAM_BATTING_3B
## 12    0.00 TEAM_BATTING_HR
```

```

## 13 0.00 TEAM_BATTING_BB
## 14 0.00 TEAM_PITCHING_H
## 15 0.00 TEAM_PITCHING_HR
## 16 0.00 TEAM_PITCHING_BB
## 17 0.00 TEAM_FIELDING_E

```

```

# Drop the BATTING_HBP field
df <- df %>%
  select(-TEAM_BATTING_HBP)

```

Notice that ~91.6% of the rows are missing from the BATTING_HBP field - we will just drop this column from consideration. The columns BASERUN_CS (base run caught stealing) and BASERUN_SB (stolen bases) both have missing values. According to baseball history, stolen bases weren't tracked officially until 1887, so some of the missing data could be from 1871-1886. We will impute those values. There are a high percentage of missing BATTING_SO (batter strike outs) and PITCHING_SO (pitching strike outs) which seem highly unlikely - we will also impute those missing values. We have chosen to impute missing values with the median value of the feature.

```

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)

```

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

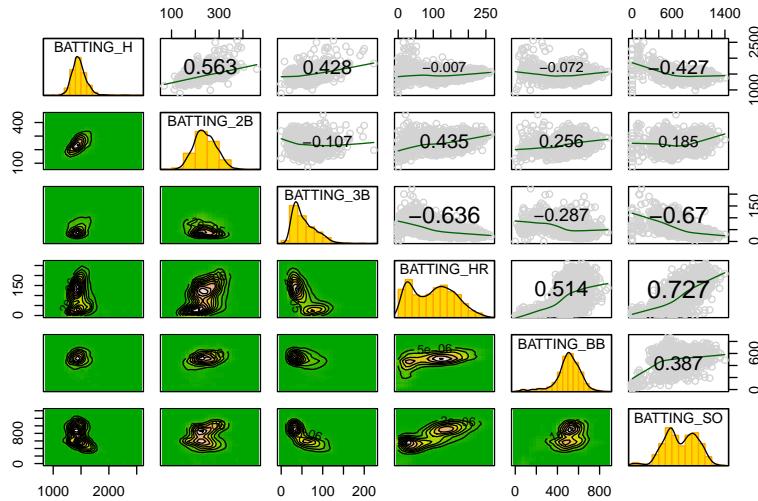
```

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.

Some of these correlations are surprising, as we would have expected TEAM_BASERUN_CS, TEAM_PITCHING_HR, and TEAM_PITCHING_BB to be negatively correlated with TARGET_WINS, and we would have expected TEAM_PITCHING_SO and TEAM_FIELDING_DP to be positively correlated with TARGET_WINS. We won't exclude them from our models based solely on the unexpected relationships themselves.

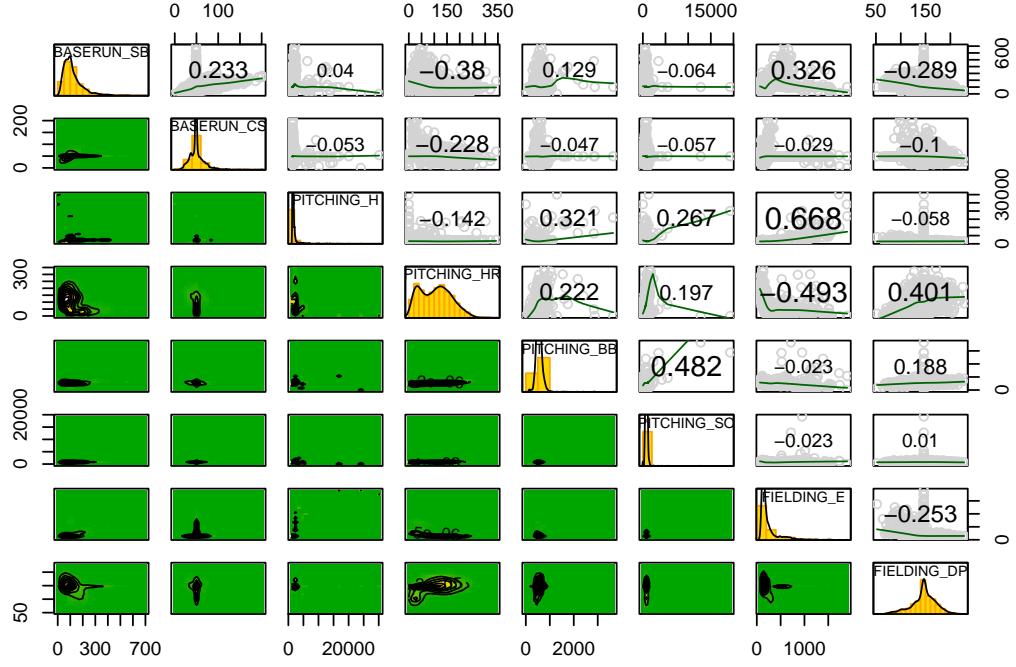
Let's review relationships between batting independent variables.

```
train_cleaned <- train_imputed |> rename_all(~stringr::str_replace(., "TEAM_", ""))
subset_batting <- train_cleaned |> select(contains('batting'))
kdepairs(subset_batting)
```



Let's review relationships between other independent variables.

```
subset_pitching <- train_cleaned |> select(!contains('batting'), -TARGET_WINS)
kdepairs(subset_pitching)
```



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.

Model Development:

First, let's create a basic model with all 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

```

Despite the simplicity of the approach used by including all of the variables provided, there are several variables which indicate multicollinearity thereby impacting the reliability of the variance and coefficients in the model.

```
vif(lm_all)
```

```

## TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B TEAM_BATTING_HR
## 3.822624      2.480705      2.936429      37.007740
## TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_BASERUN_CS
## 6.801581      5.336220      1.816473      1.167416
## TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO
## 3.567853      29.669252      6.288097      3.257794
## TEAM_FIELDING_E TEAM_FIELDING_DP
## 4.155586      1.343048

```

TEAM_BATTING_HR and TEAM_PITCHING_HR are the most correlated with other variables which is interesting that the correlation plots did not more clearly emphasize that fact from an initial spot check.

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 improved. Let's check for multicollinearity between variables.

Removing predictors that were not statistically significant slightly increased R^2 for this model. The coefficients for most of the variables around batting are positively associated with target wins which makes some sense as more hits/stolen bases should correspond with more runs and ultimately translate into wins. TEAM_BATTING_2B has a negative coefficient which is not expected at all as what differentiates doubles compared to other hits from the dependent variable. The individual T-test from our sample would also seem to indicate that it is only slightly significant. TEAM_BATTING_SO is expected to have a negative relationship with wins and its coefficient is aligned with the initial expectations. The predictors around pitching do not have very strong coefficients although they are significant to the model and the coefficients align with expectations that allowing hits inversely impacts winning, while striking out opposing players is beneficial as well. Lastly, the fielding variables that remain (TEAM_FIELDING_E and TEAM_FIELDING_DP) appear to be consistent with expectations. Double plays may have one of the strongest impacts given its coefficient although there is more sample variability compared to the other predictors.

Reviewing the variance inflation factors:

```
vif(lm_all_reduced)
```

```

##   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 factor for TEAM_BATTING_SO is greater than 5. We can remove this predictor.

```
lm_all_reduced <- update(lm_all_reduced, .~. - TEAM_BATTING_SO)
summary(lm_all_reduced)
```

```

##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + 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
## -49.354 -8.637  0.046  8.422  56.185
##
```

```

## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)           10.4516005  3.8837691  2.691  0.00717 **
## TEAM_BATTING_H        0.0543248  0.0033510 16.212 < 2e-16 ***
## TEAM_BATTING_2B      -0.0275776  0.0090008 -3.064  0.00221 **
## TEAM_BATTING_3B        0.0742212  0.0165010  4.498 7.21e-06 ***
## TEAM_BATTING_HR       0.0472798  0.0077385  6.110 1.17e-09 ***
## TEAM_BATTING_BB       0.0134412  0.0033335  4.032 5.71e-05 ***
## TEAM_BASERUN_SB        0.0204726  0.0040634  5.038 5.07e-07 ***
## TEAM_PITCHING_H      -0.0005109  0.0003132 -1.631  0.10300
## TEAM_PITCHING_SO       0.0020318  0.0006194  3.281  0.00105 **
## TEAM_FIELDING_E        -0.0186422  0.0023861 -7.813 8.48e-15 ***
## TEAM_FIELDING_DP      -0.1165458  0.0129366 -9.009 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.1 on 2265 degrees of freedom
## Multiple R-squared:  0.3116, Adjusted R-squared:  0.3085
## F-statistic: 102.5 on 10 and 2265 DF, p-value: < 2.2e-16

```

The coefficients were not impacted that substantially once this modification was made. The only caveat to that prior statement is that TEAM_BATTING_2B remains negatively associated with target wins but it's significance to the model has drastically improved with the exclusion of the collinear variable. It is not intuitive why 'TEAM_BATTING_SO' would impact TEAM_BATTING_2B as their correlation was only 0.185. TEAM_PITCHING_SO also became slightly less significant to the model

Let's remove TEAM_PITCHING_H as it is no longer significant.

```

lm_all_reduced <- update(lm_all_reduced, . ~ . - TEAM_PITCHING_H)
summary(lm_all_reduced)

```

```

##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BASERUN_SB +
##     TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP, data = train_imputed)
##
## Residuals:
##      Min      1Q      Median      3Q      Max 
## -51.785   -8.584    0.002    8.446   57.585 
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)           12.1798845  3.7378157  3.259  0.00114 **
## TEAM_BATTING_H        0.0528676  0.0032309 16.363 < 2e-16 ***
## TEAM_BATTING_2B      -0.0272628  0.0090020 -3.029  0.00249 **
## TEAM_BATTING_3B        0.0787841  0.0162681  4.843 1.37e-06 ***
## TEAM_BATTING_HR       0.0481046  0.0077248  6.227 5.64e-10 ***
## TEAM_BATTING_BB       0.0133595  0.0033344  4.007 6.36e-05 ***
## TEAM_BASERUN_SB        0.0216382  0.0040015  5.408 7.06e-08 ***
## TEAM_PITCHING_SO       0.0016132  0.0005639  2.861  0.00426 **
## TEAM_FIELDING_E        -0.0208625  0.0019604 -10.642 < 2e-16 ***
## TEAM_FIELDING_DP      -0.1173637  0.0129316 -9.076 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.1 on 2266 degrees of freedom
## Multiple R-squared:  0.3108, Adjusted R-squared:  0.308
## F-statistic: 113.5 on 9 and 2266 DF,  p-value: < 2.2e-16

```

```
vif(lm_all_reduced)
```

```

##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_HR
##   2.891551      2.351793      2.737074      2.898411
##   TEAM_BATTING_BB  TEAM_BASERUN_SB TEAM_PITCHING_SO  TEAM_FIELDING_E
##   2.216711      1.547476      1.230982      2.641838
##   TEAM_FIELDING_DP
##   1.332410

```

Based on the definitions of TEAM_BATTING_H, TEAM_BATTING_2B, TEAM_BATTING_3B, and TEAM_BATTING_HR, there is probably some multicollinearity going on with these variables. Let's compare a model that uses just the total hits against a model using each individual type of hit.

```
lm_all_reduced_hits <- update(lm_all_reduced, .~. - TEAM_BATTING_2B - TEAM_BATTING_3B - TEAM_BATTING_HR)
summary(lm_all_reduced_hits)
```

```

##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB +
##     TEAM_BASERUN_SB + TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP,
##     data = train_imputed)
##
## Residuals:
##    Min      1Q  Median      3Q      Max
## -48.823 -8.638  0.156  8.473  52.443
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)             9.7032191  3.5315442  2.748  0.00605 **
## TEAM_BATTING_H          0.0542548  0.0021078 25.740 < 2e-16 ***
## TEAM_BATTING_BB         0.0171889  0.0032588  5.275 1.46e-07 ***
## TEAM_BASERUN_SB         0.0237070  0.0037196  6.374 2.23e-10 ***
## TEAM_PITCHING_SO        0.0015116  0.0005316  2.844  0.00450 **
## TEAM_FIELDING_E         -0.0210881 0.0018286 -11.532 < 2e-16 ***
## TEAM_FIELDING_DP        -0.1107454 0.0128421 -8.624 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.24 on 2269 degrees of freedom
## Multiple R-squared:  0.2955, Adjusted R-squared:  0.2936
## F-statistic: 158.6 on 6 and 2269 DF,  p-value: < 2.2e-16

```

```
lm_all_reduced_others <- update(lm_all_reduced, .~. - TEAM_BATTING_H)
summary(lm_all_reduced_others)
```

```
##
```

```

## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B +
##      TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_PITCHING_SO +
##      TEAM_FIELDING_E + TEAM_FIELDING_DP, data = train_imputed)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -61.232 -8.904   0.069   8.910  65.787 
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 55.5049520  2.7892547 19.900 < 2e-16 ***
## TEAM_BATTING_2B 0.0694280  0.0071795  9.670 < 2e-16 ***
## TEAM_BATTING_3B 0.1953241  0.0154629 12.632 < 2e-16 ***
## TEAM_BATTING_HR 0.0748393  0.0079819  9.376 < 2e-16 ***
## TEAM_BATTING_BB 0.0103989  0.0035199  2.954 0.00317 **  
## TEAM_BASERUN_SB 0.0221969  0.0042302  5.247 1.69e-07 ***
## TEAM_PITCHING_SO -0.0012958  0.0005657 -2.291 0.02208 *  
## TEAM_FIELDING_E -0.0110738  0.0019737 -5.611 2.26e-08 ***
## TEAM_FIELDING_DP -0.0947871  0.0135932 -6.973 4.05e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.85 on 2267 degrees of freedom
## Multiple R-squared:  0.2293, Adjusted R-squared:  0.2266 
## F-statistic: 84.31 on 8 and 2267 DF,  p-value: < 2.2e-16

```

Comparing Partial F-Tests/ANOVA of reduced models

```
anova(lm_all_reduced_others,lm_all_reduced)
```

```

## Analysis of Variance Table
##
## Model 1: TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR +
##           TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_PITCHING_SO + TEAM_FIELDING_E +
##           TEAM_FIELDING_DP
## Model 2: TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##           TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_PITCHING_SO +
##           TEAM_FIELDING_E + TEAM_FIELDING_DP
## Res.Df   RSS Df Sum of Sq    F    Pr(>F)    
## 1    2267 435053
## 2    2266 389079  1    45974 267.75 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The results of the partial F-test indicate that there is a statistically significant difference between the full model and the reduced one and it does not seem necessary to exclude the other hit predictors in the model.

```
anova(lm_all_reduced_hits,lm_all_reduced)
```

```

## Analysis of Variance Table
##
```

```

## Model 1: TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_BASERUN_SB +
##           TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP
## Model 2: TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B +
##           TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BASERUN_SB + TEAM_PITCHING_SO +
##           TEAM_FIELDING_E + TEAM_FIELDING_DP
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1    2269 397696
## 2    2266 389079  3     8617.2 16.729 9.486e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Similarly, the partial F-test would indicate that reducing the hits variable is not necessary despite the potential reduced collinearity.

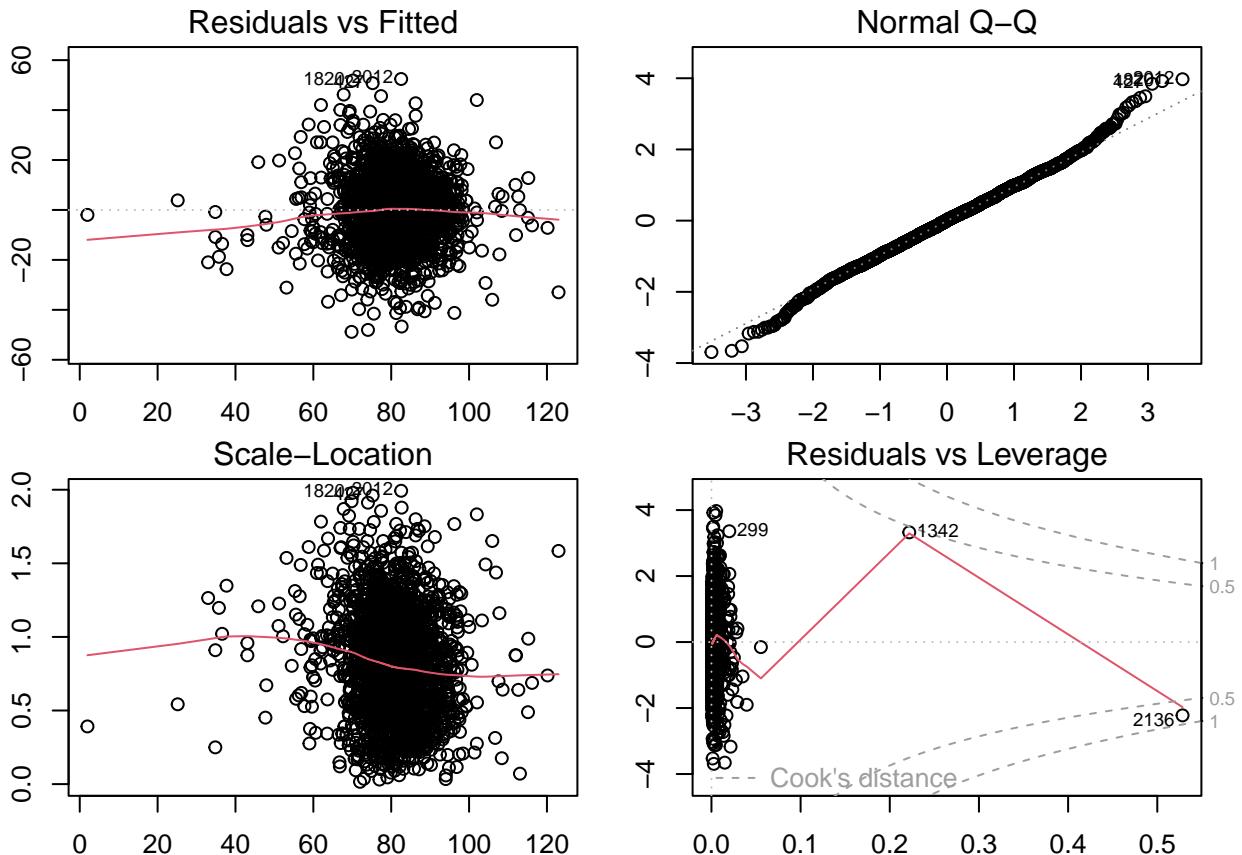
The model using `TEAM_BATTING_HITS` has a higher R^2 so it accounts for more variability. Let's use this variable in our model.

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

```

par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
plot(lm_all_reduced_hits)

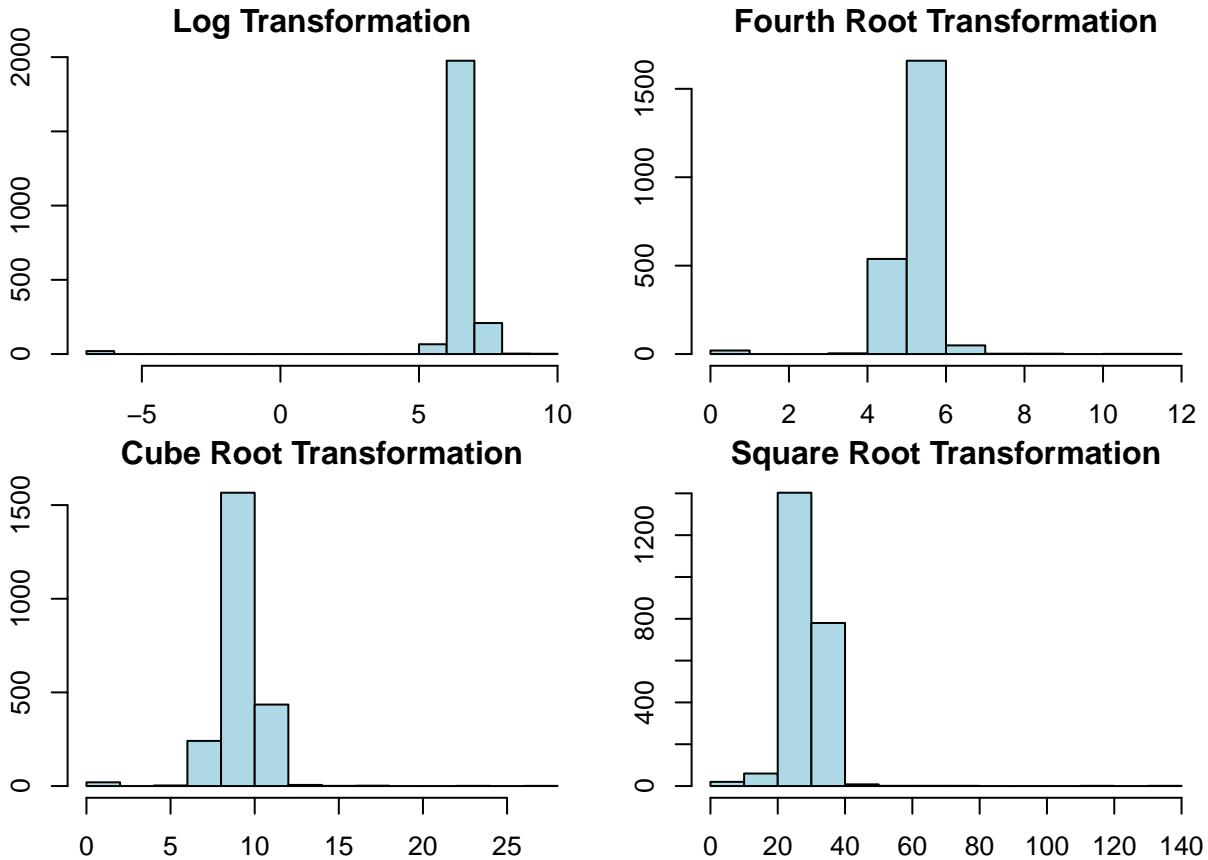
```



The Q-Q plot shows that the residuals of this model are fairly normally distributed. The residuals vs. fitted plot shows a cluster of residuals and a seeming outlying point. There is no general pattern seen here and the cluster of points seems to indicate that homoscedasticity is satisfied for this model.

Let's try transforming some of our variables to come up with a more accurate model.

`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).

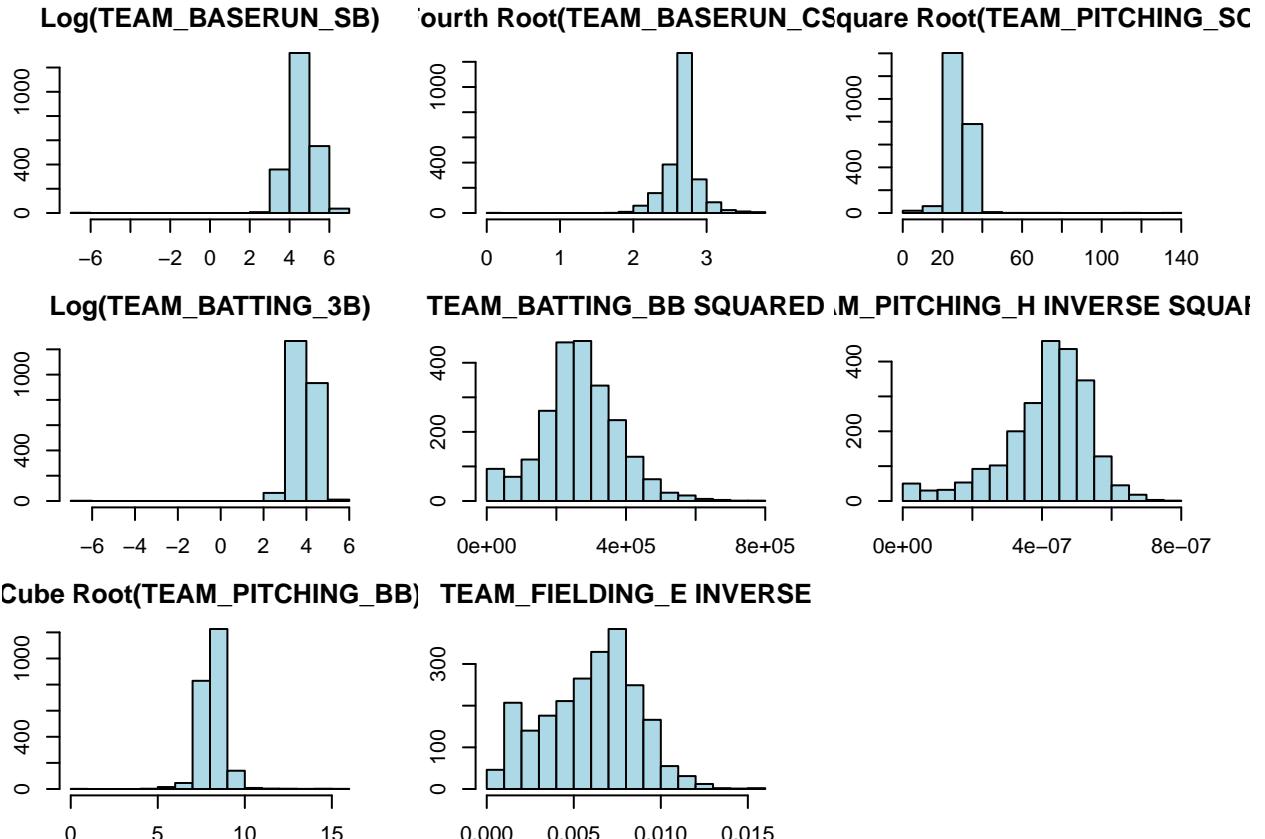


The square root transformation appears to normalize the data best. Let's confirm the ideal lambda proposed by the `boxcox` function from the `MASS` library is similar to the square root transformation lambda (0.5) we assume will work best for this data.

```
## [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	adj
<code>TEAM_BASERUN_SB</code>	0.2	log
<code>TEAM_BASERUN_CS</code>	0.3	fourth root
<code>TEAM_PITCHING_SO</code>	0.45	square root
<code>TEAM_BATTING_3B</code>	0.3	log
<code>TEAM_BATTING_BB</code>	1.75	square
<code>TEAM_PITCHING_H</code>	-2	square inverse
<code>TEAM_PITCHING_BB</code>	0.35	cube root
<code>TEAM_FIELDING_E</code>	-0.95	inverse



Adjusting the ideal lambdas proposed for several variables to commonly understood transformations, we see mixed results on normalizing the distributions. Let's use the same variables from our final untransformed model above to see if we can improve the model using transformations.

```
lm_trans <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + I(TEAM_BATTING_BB**2) + log(TEAM_BASERUN_SB + 1e-04) + I(TEAM_PITCHING_SO^0.5) + I(TEAM_FIELDING_E^-1) + TEAM_FIELDING_DP, data = train_imputed)
```

```
##  
## Call:  
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB +  
##       I(TEAM_BATTING_BB^2) + log(TEAM_BASERUN_SB + 1e-04) + I(TEAM_PITCHING_SO^0.5) +  
##       I(TEAM_FIELDING_E^-1) + TEAM_FIELDING_DP, data = train_imputed)  
##  
## Residuals:  
##      Min      1Q Median      3Q     Max  
## -59.340 -8.557  0.113  8.603 55.118  
##  
## Coefficients:  
##  
## (Intercept)          Estimate Std. Error t value Pr(>|t|)  
## TEAM_BATTING_H        -3.404e+00  4.958e+00 -0.686  0.4925  
## TEAM_BATTING_BB        4.868e-02  2.173e-03 22.405 < 2e-16 ***  
## I(TEAM_BATTING_BB^2)    5.538e-03  1.068e-02  0.518  0.6042  
## log(TEAM_BASERUN_SB + 1e-04) 2.482e-05  1.131e-05  2.195  0.0283 *  
## I(TEAM_PITCHING_SO^0.5) -1.991e-02  5.392e-02 -0.369  0.7119  
## I(TEAM_FIELDING_E^-1)   1.411e+03  1.427e+02  9.885 < 2e-16 ***  
## TEAM_FIELDING_DP      -1.270e-01  1.313e-02 -9.672 < 2e-16 ***
```

```

## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.29 on 2268 degrees of freedom
## Multiple R-squared: 0.2905, Adjusted R-squared: 0.2883
## F-statistic: 132.7 on 7 and 2268 DF, p-value: < 2.2e-16

```

Note: There are instances of TEAM_BASERUN_SB where the value is zero. Because of this, a log transformation creates an error. To account for this we increment by a small number (0.0001) so that the log transformation can be applied.

The transformed TEAM_PITCHING_SO is no longer significant, let's remove it.

```
lm_trans_reduced <- update(lm_trans, .~. - I(TEAM_PITCHING_SO**.5), train_imputed)
summary(lm_trans_reduced)
```

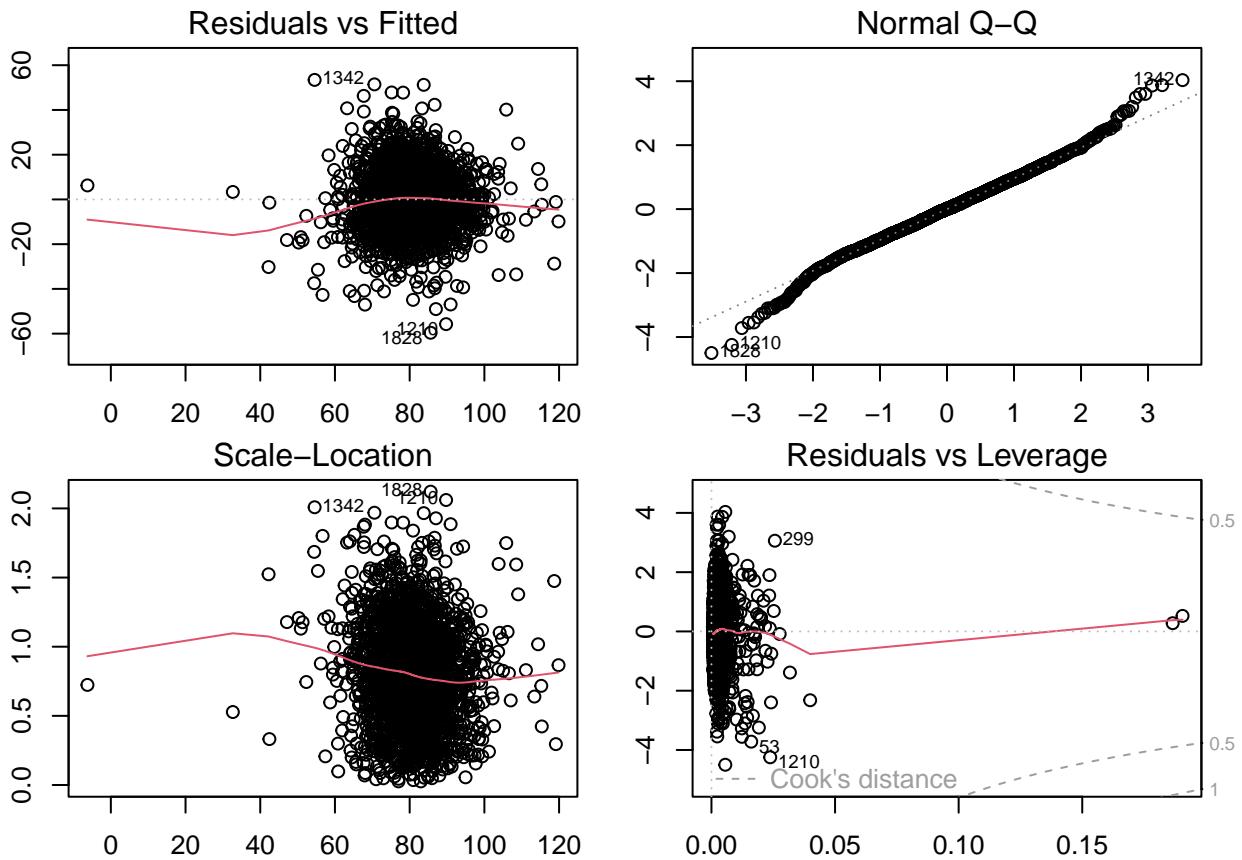
```

##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB +
##      I(TEAM_BATTING_BB^2) + log(TEAM_BASERUN_SB + 1e-04) + I(TEAM_FIELDING_E^-1) +
##      TEAM_FIELDING_DP, data = train_imputed)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -59.611 -8.620   0.107  8.597  53.441 
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                 -4.223e+00  4.433e+00 -0.953   0.3409    
## TEAM_BATTING_H                4.893e-02  2.059e-03 23.771  < 2e-16 ***
## TEAM_BATTING_BB               5.693e-03  1.067e-02  0.533   0.5938    
## I(TEAM_BATTING_BB^2)           2.473e-05  1.131e-05  2.188   0.0288 *  
## log(TEAM_BASERUN_SB + 1e-04)  3.019e+00  4.233e-01  7.132  1.33e-12 ***
## I(TEAM_FIELDING_E^-1)          1.394e+03  1.354e+02 10.297  < 2e-16 ***
## TEAM_FIELDING_DP              -1.269e-01  1.312e-02 -9.668  < 2e-16 *** 
## ---                        
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.29 on 2269 degrees of freedom
## Multiple R-squared: 0.2905, Adjusted R-squared: 0.2886
## F-statistic: 154.8 on 6 and 2269 DF, p-value: < 2.2e-16

```

The adjusted R^2 is slightly less for this model than for the untransformed one. The coefficients/relationships for the predictor variables become a little less intuitive due to the transformations. It makes sense that the inverse fielding errors would create a positive relationship with target wins as if its some type of rate rather than count data. The other variables that were transformed do not appear to be that much different. Let's take a look at the diagnostic plots for this transformed model.

```
par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
plot(lm_trans_reduced)
```



Once again, the Q-Q plot shows that the residuals are fairly normally distributed. From the plot of Cook's distance, it seems there are fewer possible leverage points. The residuals vs. fitted plot also seems to indicate that homoscedasticity is satisfied.

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).

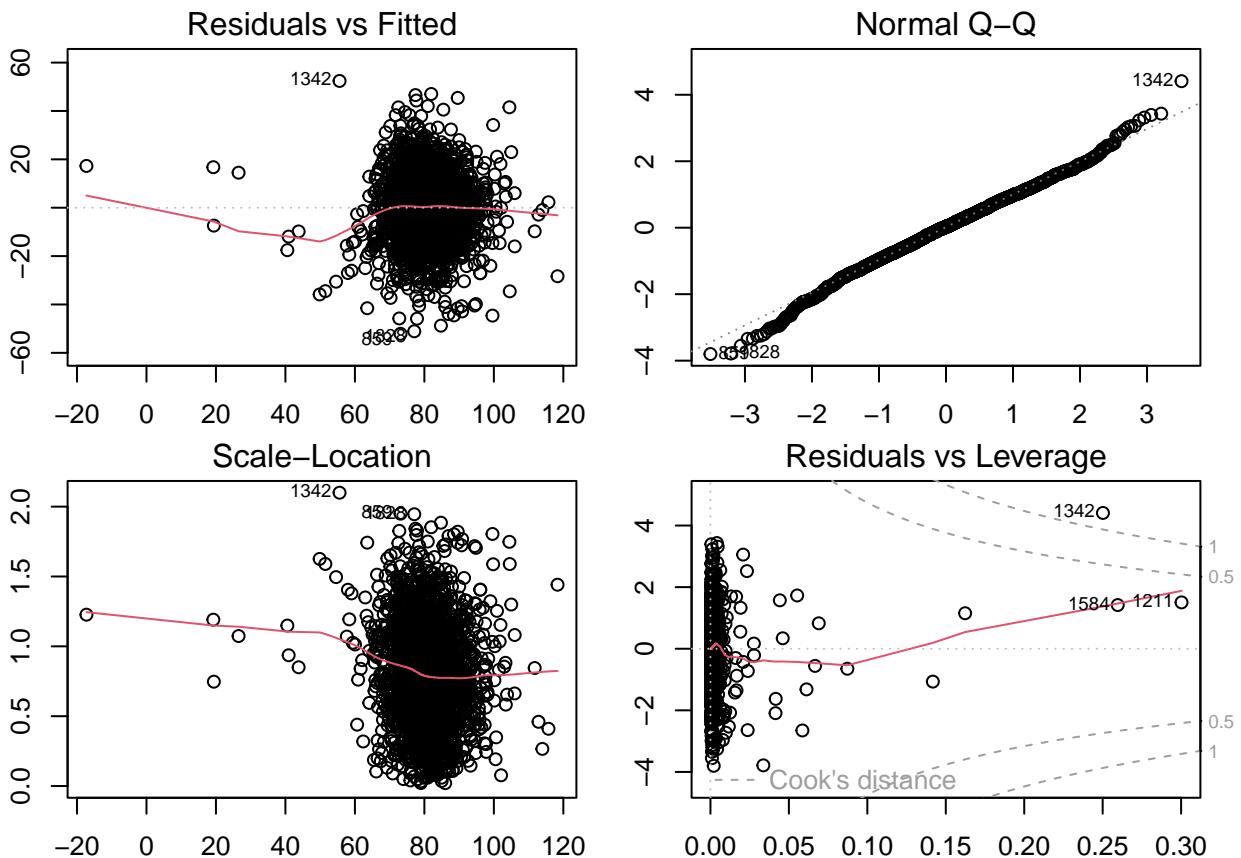
```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB +
##     TEAM_PITCHING_H + TEAM_PITCHING_BB, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -52.133  -8.860   0.379   9.373  52.416 
## 
## Coefficients:
##             Estimate Std. Error 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

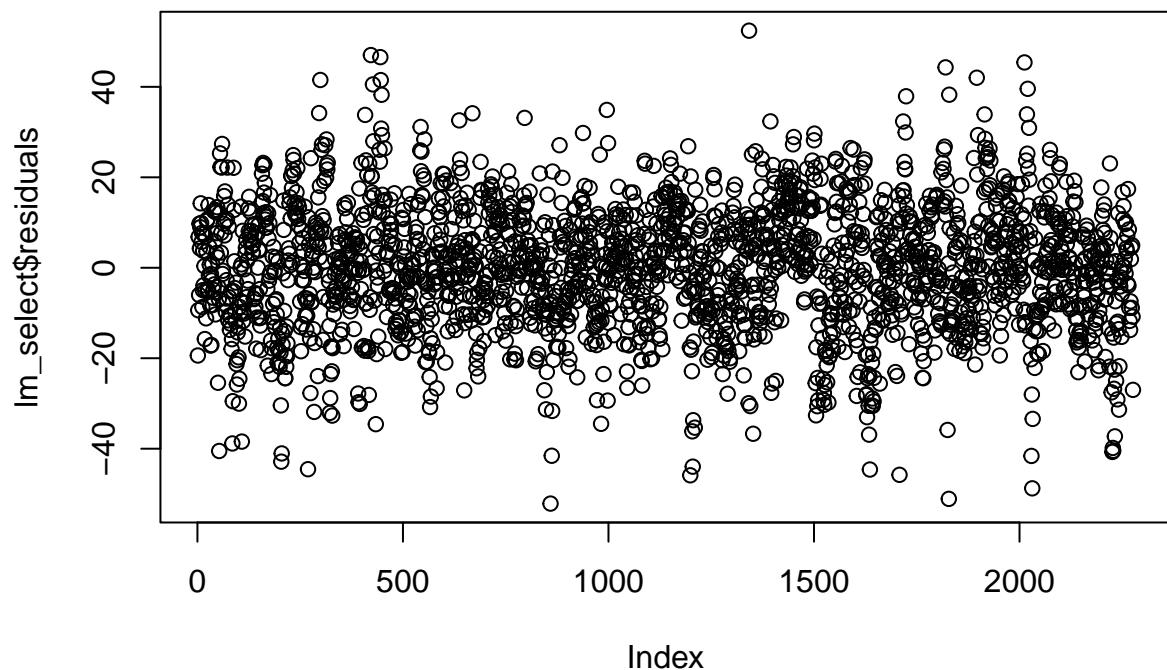
```

When reviewing the model output it appears that all of these factors are statistically significant and some of the intuition around successful baseball performance translates from the statistics. The coefficients are all positively associated with target wins except for TEAM_PITCHING_H which makes sense given that more hits given up typically would indicate the other team may be winning the game. What is slightly surprising is that TEAM_PITCHING_BB has a positive coefficient and is highly significant as in theory more opposing base runners, even via walks, should help the other team win.

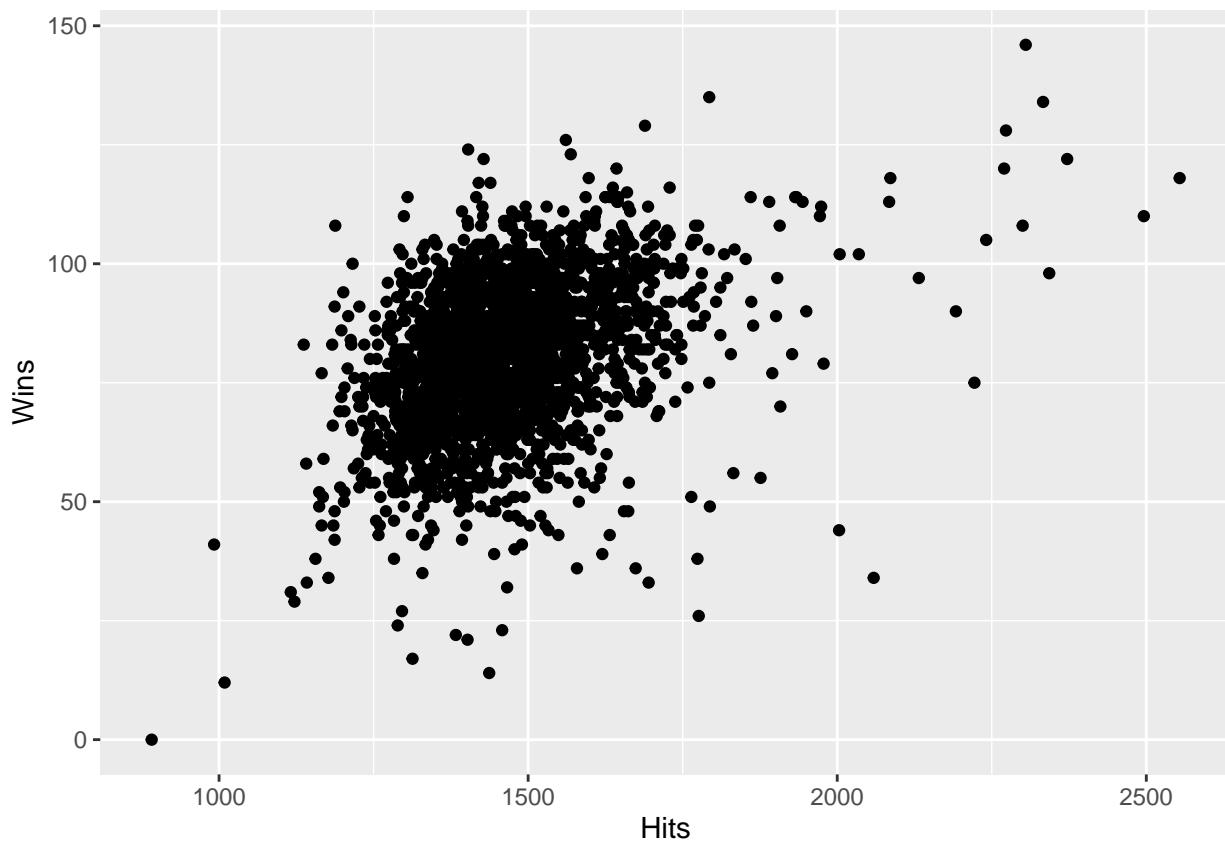


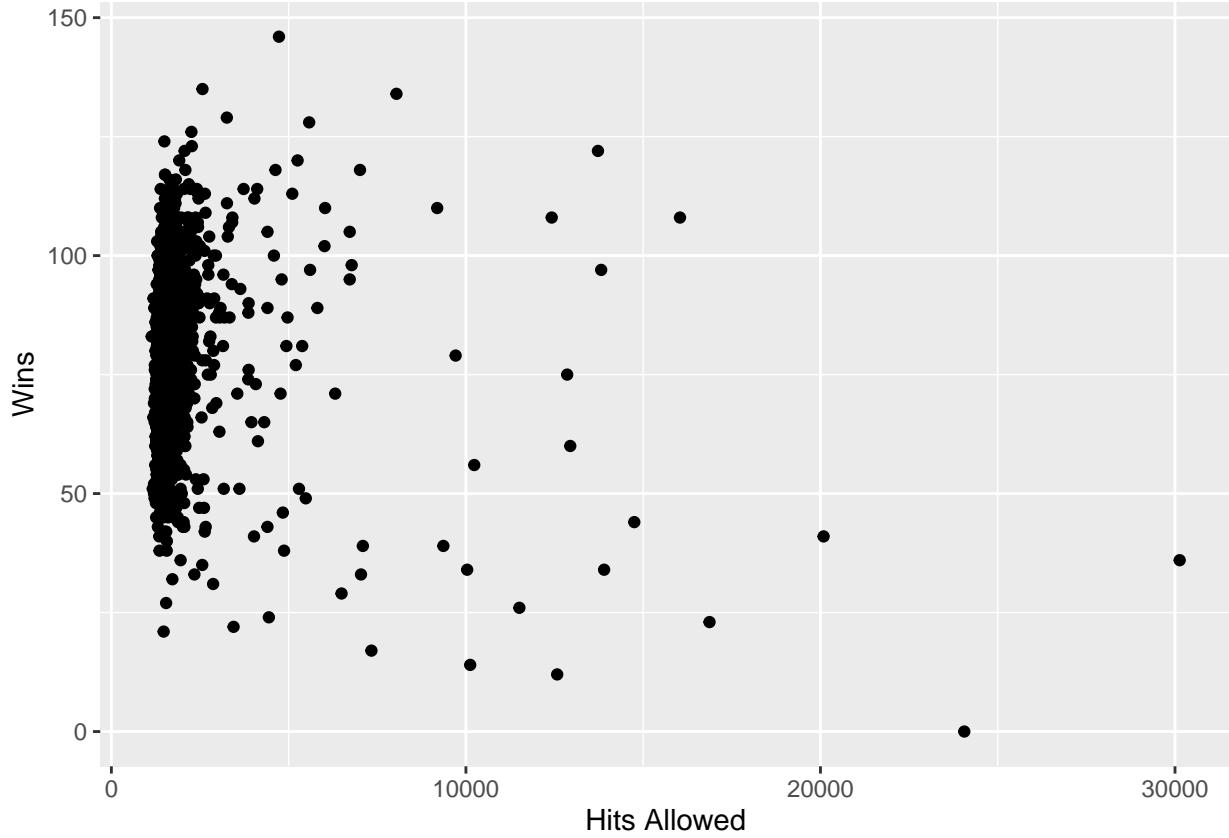
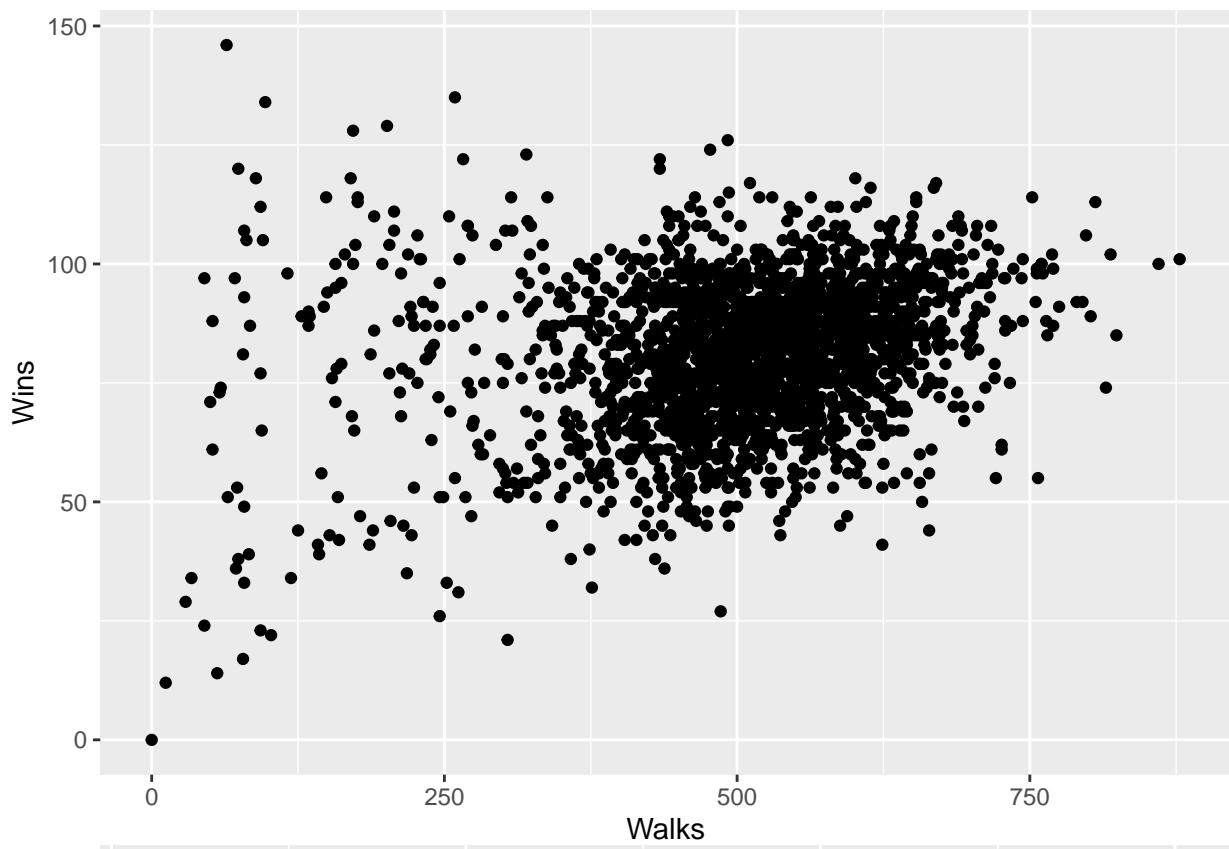
It's interesting to note that with selected variables (walks and hits gained/allowed per team) that our adjusted R^2 actually went *down*, indicating the amount of variability in TARGET_WINS explained by our more selective walks/hits model is *less* than the model including all variables.

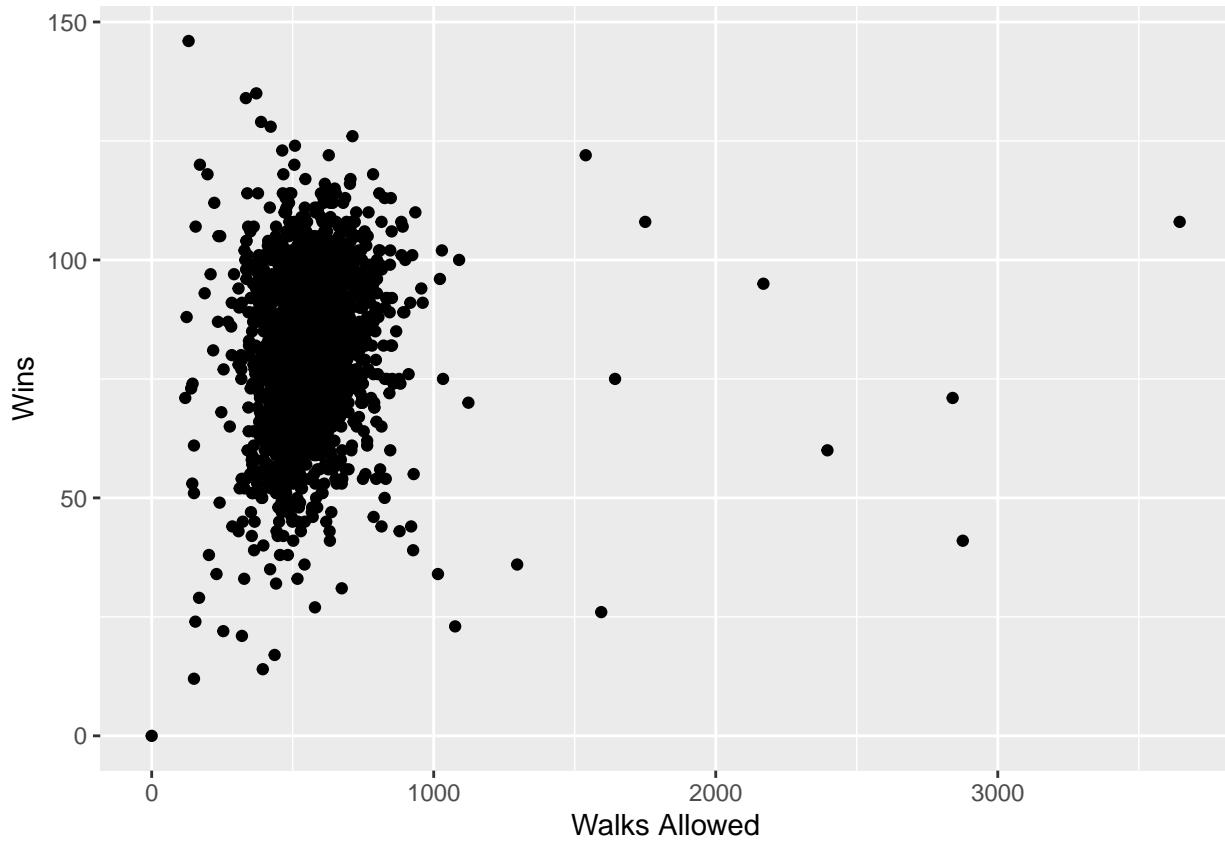
Looking at our residual plot above, there seems to be a clustering of residuals along the x-axis at $X \approx 80$. This shows a pattern in our residuals.



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







Overall, we're seeing some loosely linear relationships between our input variables and wins. For example, offensive hits has a plausibly linear relationship to wins, whereas hits allowed (TEAM_PITCHING_H) does not have as clear of a linear relationship.

One alternative theory that uses some of the same logic as before is to maximize run producing offense (TEAM_BATTING_H, TEAM_BATTING_3B+TEAM_BATTING_2B+TEAM_BATTING_HR) as these hits imply that the batters are getting more bases which should result in runners on base being more likely to score. Another way to supercharge the offense is by advancing the runners when they get on base (TEAM_BASERUN_SB), which in theory should put us in a position to earn more runs. In terms of defense, the key area we will focus on is limiting mistakes (TEAM_FIELDING_E) and when batters get on, cleaning up our mess in an efficient way (TEAM_FIELDING_DP). On paper the lean towards more offense with crisp error free defense is a recipe for success.

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BASERUN_SB + TEAM_FIELDING_DP +
##     TEAM_FIELDING_E, data = train_imputed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -51.143  -8.756  -0.050   8.494  65.205 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  10.0000    1.0000  10.000 0.0000000 ***
## TEAM_BATTING_H  0.0000    0.0000   0.000  0.9999999    
## TEAM_BATTING_2B  0.0000    0.0000   0.000  0.9999999    
## TEAM_BATTING_3B  0.0000    0.0000   0.000  0.9999999    
## TEAM_BATTING_HR  0.0000    0.0000   0.000  0.9999999    
## TEAM_BASERUN_SB  0.0000    0.0000   0.000  0.9999999    
## TEAM_FIELDING_DP  0.0000    0.0000   0.000  0.9999999    
## TEAM_FIELDING_E -0.0000    0.0000   0.000  0.9999999    
##
```

```

## (Intercept) 20.391917 3.301174 6.177 7.71e-10 ***
## TEAM_BATTING_H 0.049586 0.003078 16.108 < 2e-16 ***
## TEAM_BATTING_2B -0.019814 0.008771 -2.259 0.024 *
## TEAM_BATTING_3B 0.082551 0.016245 5.082 4.04e-07 ***
## TEAM_BATTING_HR 0.057850 0.007480 7.734 1.56e-14 ***
## TEAM_BASERUN_SB 0.027281 0.003804 7.171 1.00e-12 ***
## TEAM_FIELDING_DP -0.105804 0.012612 -8.389 < 2e-16 ***
## TEAM_FIELDING_E -0.023849 0.001633 -14.602 < 2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.16 on 2268 degrees of freedom
## Multiple R-squared: 0.3039, Adjusted R-squared: 0.3017
## F-statistic: 141.4 on 7 and 2268 DF, p-value: < 2.2e-16

```

One modification that was made due to the number of null values in fielding predictors is that we used the imputed data set to run the regression analysis to limit the records that would be excluded from the model. Overall we can see a competitive one and appears to be similar enough to the automated backward selection despite have less independent variables. `TEAM_BATTING_2B` has a negative coefficient consistent with other models which doesn't make so much sense, but perhaps the distribution of these values and the higher p-value could mean this could be excluded; however, we will keep it in the model given that it would seem to have predictive value on paper and meets the statistical baselines.

Let's confirm there isn't any collinearity issues with this model variant:

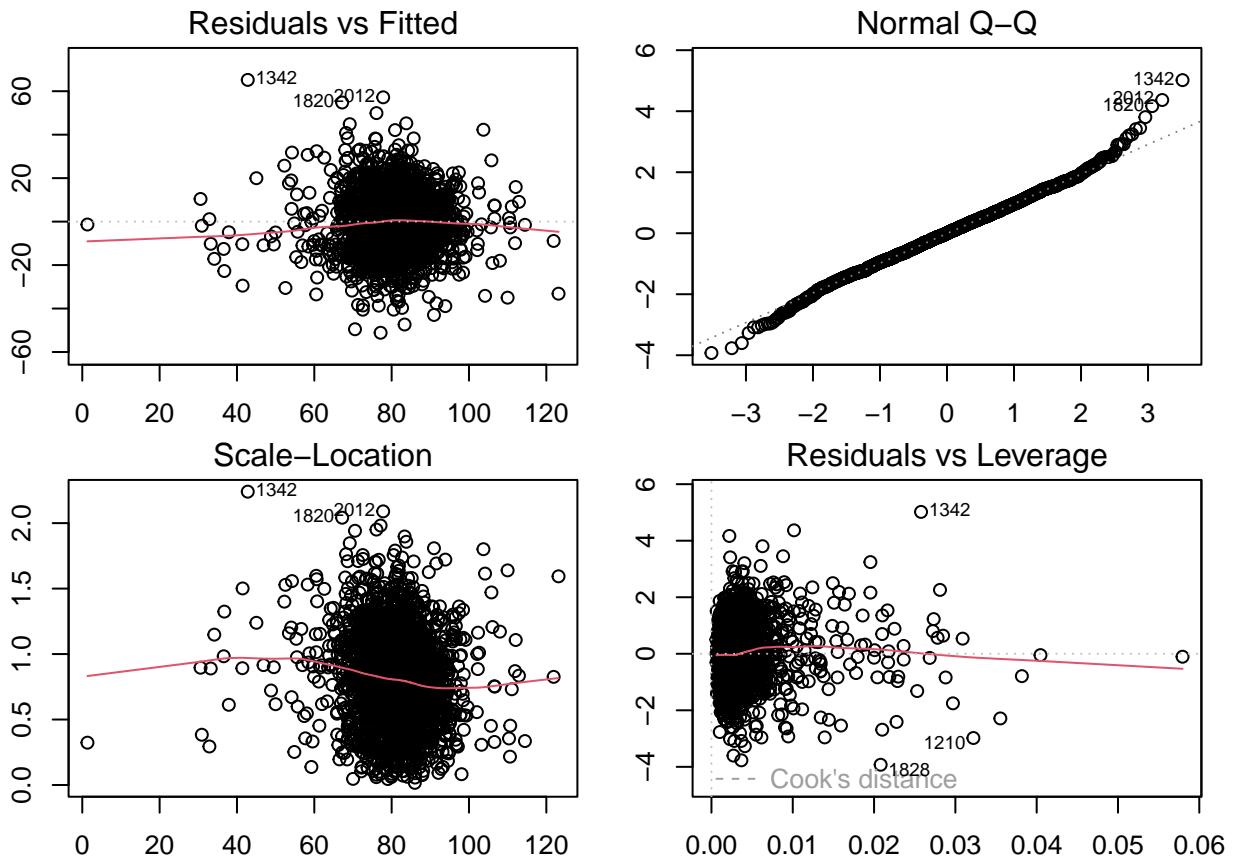
```

##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_HR
##   2.601543      2.212394      2.704599      2.693294
##   TEAM_BASERUN_SB TEAM_FIELDING_DP  TEAM_FIELDING_E
##   1.386145      1.255963      1.817196

```

There is some collinearity which is to be expected with the data that is available to us, but nothing that would be problematic that requires any major changes.

Let's confirm that the assumptions of OLS are not violated:



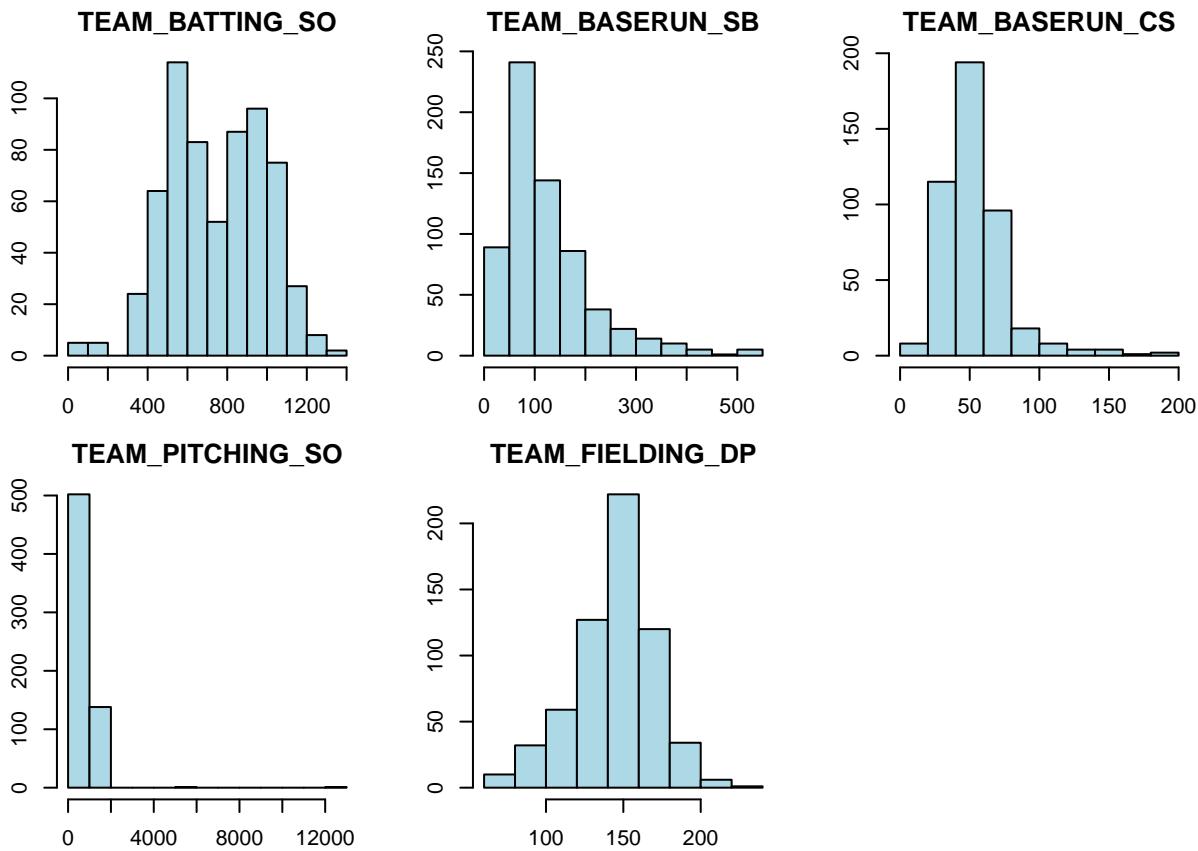
The diagnostic plots don't seem to indicate a violation of the assumptions as the variance across the x-values while mostly concentrated at 80 does not show much heteroskedasticity in the residuals or the transformed standardized ones. There are some leverage points (1342 in particular) in the data set and it can definitely skew the data although some teams could have very strong seasons that aren't invalid or anomalous to be worthy of exclusion.

Model Evaluation/Selection:

First we read in our evaluation data.

Now we can make some predictions on the test holdout data and compare results from our models before we select the best model to use on the evaluation data. First we compare the distributions of the test data to confirm we can use the same imputation methods we used to fill missing values for variables in the train data.

```
par(mfrow=c(2,3))
par(mai=c(.3,.3,.3,.3))
variables <- c("TEAM_BATTING_SO", "TEAM_BASERUN_SB", "TEAM_BASERUN_CS",
              "TEAM_PITCHING_SO", "TEAM_FIELDING_DP")
for (i in 1:(length(variables))) {
  hist(test[[variables[i]]], main = variables[i], col = "lightblue")
}
```



The test data distributions are similar to the distributions observed in the train data for these variables, so the same imputation methods can be used for each of them.

```

test <- test |>
  select(-TEAM_BATTING_HBP) |>
  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)))

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

# Predict using the model using all input variables
predict_all <- predict(lm_all, test)
predict_reduced <- predict(lm_all_reduced, test)

predict_transformed_reduced <- predict(lm_trans_reduced, test)

predict_select <- predict(lm_select, test)
predict_off <- predict(lm_off,test)

```

```
predict_trans_reduced <- predict(lm_trans_reduced, test)
```

We can use Root-mean Squared Error (RMSE) to analyze our models from above. This is one way to measure the performance of a model. In simple terms, a smaller RMSE value indicates better model performance when predicting on new data.

```
rmse <- function(c1, c2){  
  sqrt(mean((c1 - c2)^2))  
}  
#Calculate RMSE and print to screen  
rmse_all <- rmse(predict_all, test$TARGET_WINS)  
rmse_reduced <- rmse(predict_reduced, test$TARGET_WINS)  
rmse_transformed_reduced <- rmse(predict_transformed_reduced, test$TARGET_WINS)  
  
print(rmse_all)
```

```
## [1] 12.29232
```

```
print(rmse_reduced)
```

```
## [1] 12.35444
```

```
print(rmse_transformed_reduced)
```

```
## [1] 12.67368
```

The model with the lowest RMSE is our all variable model without transformations, so we will make predictions on the evaluation data using this model. Since we don't have TARGET_WINS in our evaluation data, we won't be able to evaluate the model performance against actual win totals.

We can use the Root-mean Squared Error (RMSE) (from the `modelr` package) to analyze our models from above. This is one way to measure the performance of a model. In simple terms, a smaller RMSE value indicates better model performance when predicting on new data.

```
# Calculate RMSE and print to screen  
rmse_all <- modelr::rmse(lm_all, test)  
rmse_reduced <- modelr::rmse(lm_all_reduced, test)  
rmse_select <- modelr::rmse(lm_select, test)  
rmse_off <- modelr::rmse(lm_off, test)  
rmse_trans_reduced <- modelr::rmse(lm_trans_reduced, test)  
  
print(glue('RMSE lm_all: {rmse_all}'))
```

```
## RMSE lm_all: 12.2923197085304
```

```
print(glue('RMSE lm_reduced: {rmse_reduced}'))
```

```
## RMSE lm_reduced: 12.3544360642616
```

```

print(glue('RMSE lm_select: {rmse_select}'))

## RMSE lm_select: 13.2136000264059

print(glue('RMSE lm_off: {rmse_off}'))

## RMSE lm_off: 12.5057843276575

print(glue('RMSE lm_transreduced {rmse_trans_reduced}'))

## RMSE lm_transreduced 12.6736755963075

```

Lastly, we can predict on our `eval` data based on the best RMSE value coming from our *reduced* model. Since we don't have `TARGET_WINS` in our evaluation data, we won't be able to evaluate the model performance against actual win totals.

The model with the lowest RMSE is our all variable model without transformations, but given the collinearity concerns it probably isn't fair to use this version as a means of prediction, so we will also use offensive and mistake free defensive one for evaluation as well. Since we don't have `TARGET_WINS` in our evaluation data, we won't be able to identify the model performance against actual win totals.

However, we can look at the distribution of predicted wins to make sure our model predicts reasonable values. Knowing what we know about baseball, average teams tend to win 80 games in a season (out of 162 total regular season games).

```

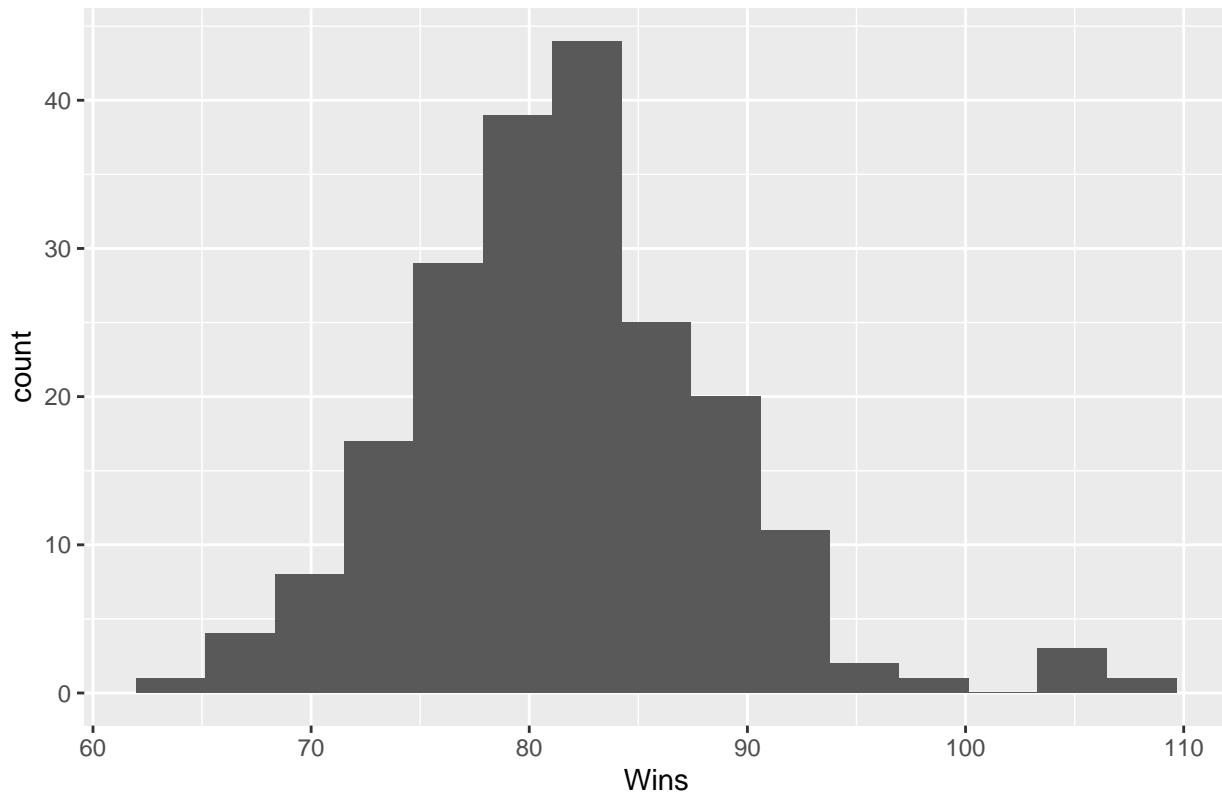
# Predict and plot on evaluation data (no wins listed)
predict_all_eval <- as.data.frame(predict(lm_all, eval))

prediction_reduced <- predict(lm_all_reduced, eval)
predict_reduced_eval <- as.data.frame(prediction_reduced)

# Plot reduced model evaluation
ggplot(predict_reduced_eval,
       aes(x=prediction_reduced)) +
  geom_histogram(bins=15) +
  labs(x="Wins",
       title="Predicted Wins (Reduced Model): Evaluation Data.")

```

Predicted Wins (Reduced Model): Evaluation Data.

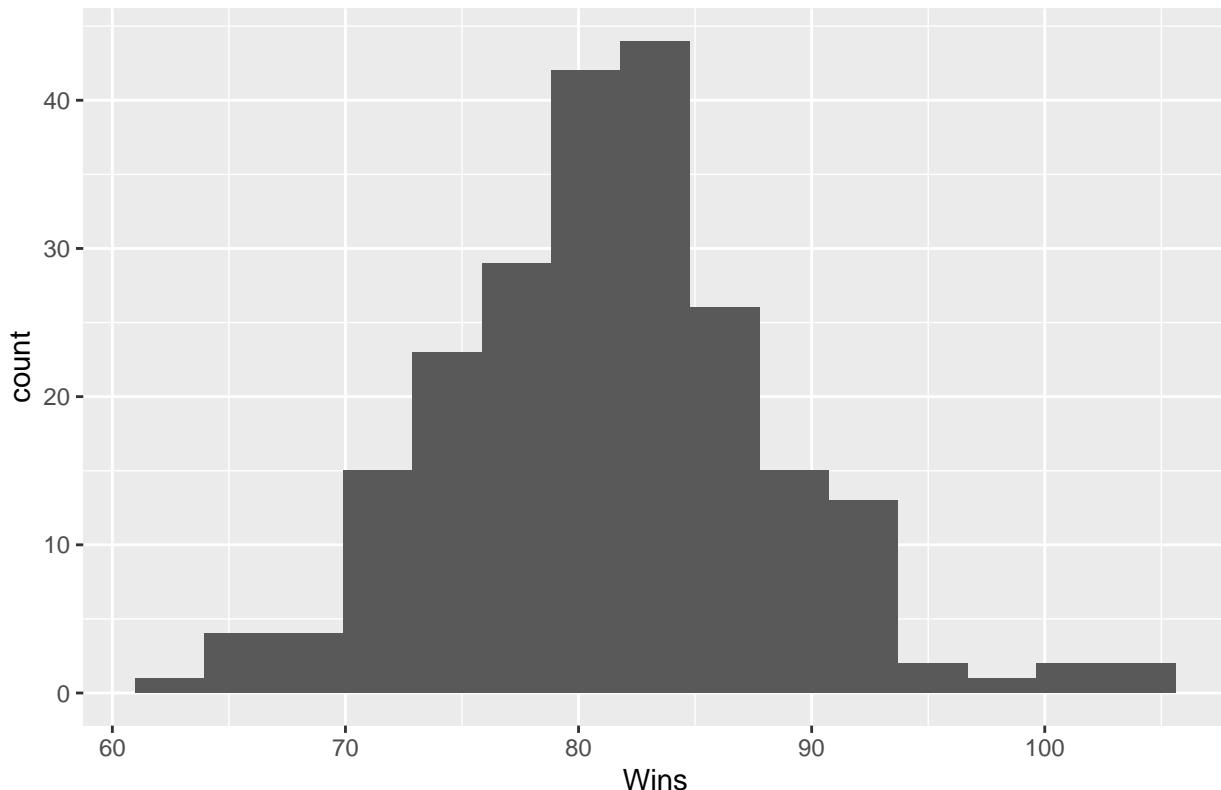


Roughly speaking, these predicted win totals look roughly normal, and centered around 80 wins, which is expected

```
predict_off_eval <- as.data.frame(predict(lm_off, eval))

ggplot(predict_off_eval,
       aes(x = predict(lm_off, eval))) + geom_histogram(bins=15) + labs(x="Wins", title="Predicted Wins (Reduced Model) Evaluation Data")
```

Predicted Wins (Offensive Imputed Model): Evaluation Data.



Roughly speaking, these predicted win totals look roughly normal and centered around 80 wins, which is expected.

Conclusions:

Overall, we found relatively similar RMSE values across our models, suggesting comparable performance based on this metric with the initial evaluation dataset. When seeking approval from our manager for decisions, we'd likely opt for the reduced model. This choice stems from its ability to evaluate possible variations and its use of AIC (Akaike information criterion) for decision-making from a statistical standpoint. While this approach might not fully convince all managers, we can clarify that certain predictor combinations were effective in minimizing our loss function. AIC is designed to strike a balance between the benefits of linear combinations and model complexity. However, if our manager prefers prioritizing judgment or intuition over statistical metrics to justify predictors in the model, we would rely on the offensive model for any future forecasting needs.

The RMSE for each models are:

- $M_{reduced} = 13.16778$
- $M_{offensive} = 13.17077$
- $M_{transformed} = 13.40541$
- $M_{select} = 13.7739$

```
library(knitr)

# Create a dataframe with model names and RMSE values
model_rmse <- data.frame(
```

```

Model = c("M_{reduced}", "M_{offensive}", "M_{transformed}", "M_{select}"),
RMSE = c(13.16778, 13.17077, 13.40541, 13.7739)
)

# Print the table
kable(model_rmse, format = "markdown")

```

Model	RMSE
M_{reduced}	13.16778
M_{offensive}	13.17077
M_{transformed}	13.40541
M_{select}	13.77390

Appendix: Report Code

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

```

knitr:::opts_chunk$set(eval = TRUE, message = FALSE, warning = FALSE)
library(tidyverse)
library(openintro)
library(glue)
library(tidyverse)
library(car)
library(ResourceSelection)
library(VIM)
library(pracma)
library(MASS)
select <- dplyr::select
library(dplyr)
library(knitr)
library(modelr)
df <- read.csv("https://raw.githubusercontent.com/waheeb123/Data-621/main/Homeworks/Homework-1/moneyball.csv")
df <- data.frame(df)
set.seed(30)

# Adding a 20% holdout of our input data for model evaluation later
train <- subset(df[sample(1:nrow(df)), ], select=-c(TEAM_BATTING_HBP))%>%
  sample_frac(0.7)

test <- dplyr::anti_join(df, train, by = 'INDEX')
test <- subset(test, select=-c(INDEX))

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(fill = "skyblue") +
  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")
cMeans <- as.data.frame(round(colMeans(train, na.rm = TRUE), 1))
colnames(cMeans) <- NULL
kable(cMeans, format = "simple")

summary(train)
# First, store the summary in an object
summary_data <- summary(train)

# Create a data frame from the summary
summary_df <- as.data.frame(summary_data)

# Print the data frame
summary_df

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_PITCHING_SO")

for (i in 1:(length(variables))) {
  hist(train[[variables[i]]], main = variables[i], col = "lightblue")
}
# Identify missing data by Feature and display percent breakout
missing <- colSums(df %>% sapply(is.na))
missing_pct <- round(missing / nrow(df) * 100, 2)
stack(sort(missing_pct, decreasing = TRUE))

# Drop the BATTING_HBP field
df <- df %>%
  select(-TEAM_BATTING_HBP)
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$TARGET_WINS)
train_cleaned <- train_imputed |> rename_all(~stringr::str_replace(., "TEAM_",""))
subset_batting <- train_cleaned |> select(contains('batting'))
kdepairs(subset_batting)
subset_pitching <- train_cleaned |> select(!contains('batting'), -TARGET_WINS)
kdepairs(subset_pitching)
lm_all <- lm(TARGET_WINS~., train_imputed)
summary(lm_all)
vif(lm_all)
lm_all_reduced <- step(lm_all, direction="backward", trace = 0)
summary(lm_all_reduced)
vif(lm_all_reduced)
lm_all_reduced <- update(lm_all_reduced, .~. - TEAM_BATTING_SO)
summary(lm_all_reduced)
lm_all_reduced <- update(lm_all_reduced, .~. - TEAM_PITCHING_H)
summary(lm_all_reduced)
vif(lm_all_reduced)
lm_all_reduced_hits <- update(lm_all_reduced, .~. - TEAM_BATTING_2B - TEAM_BATTING_3B - TEAM_BATTING_HR)
summary(lm_all_reduced_hits)
lm_all_reduced_others <- update(lm_all_reduced, .~. - TEAM_BATTING_H)
summary(lm_all_reduced_others)
anova(lm_all_reduced_others, lm_all_reduced)
anova(lm_all_reduced_hits, lm_all_reduced)
par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
plot(lm_all_reduced_hits)
train_imputed_transformed <- train_imputed
#Add a small constant to TEAM_PITCHING_SO so there are no 0 values.
train_imputed_transformed$TEAM_PITCHING_SO <- train_imputed_transformed$TEAM_PITCHING_SO + 0.001
par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
#Compare how easy to understand transformations alter the distribution
hist(log(train_imputed_transformed$TEAM_PITCHING_SO),
     main = "Log Transformation", col="lightblue")
hist(sqrt(train_imputed_transformed$TEAM_PITCHING_SO, 4),
      main = "Fourth Root Transformation", col="lightblue")
hist(sqrt(train_imputed_transformed$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]]] + 0.001
}

```

```

    }
}

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))
adj <- c("log", "fourth root", "square root", "log", "square", "square inverse", "cube root", "inverse")
lambdas <- cbind(lambdas, adj)
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")
lm_trans <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + I(TEAM_BATTING_BB**2) + log(TEAM_BASERUN)
summary(lm_trans)
lm_trans_reduced <- update(lm_trans, .~. - I(TEAM_PITCHING_SO**.5), train_imputed)
summary(lm_trans_reduced)
par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
plot(lm_trans_reduced)

# 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, train_imputed)

summary(lm_select)
par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
plot(lm_select)

# Plot selective model residuals
plot(lm_select$residuals)
# par(mfrow=c(2,2))
# par(mai=c(.3,.3,.3,.3))

```

```

# plot(TARGET_WINS ~ TEAM_BATTING_H +
#                   TEAM_BATTING_BB +
#                   TEAM_PITCHING_H +
#                   TEAM_PITCHING_BB,
#                   data=train)
ggplot(train, aes(x=TEAM_BATTING_H, y=TARGET_WINS)) + geom_point() + labs(x="Hits", y="Wins")
ggplot(train, aes(x=TEAM_BATTING_BB, y=TARGET_WINS)) + geom_point() + labs(x="Walks", y="Wins")
ggplot(train, aes(x=TEAM_PITCHING_H, y=TARGET_WINS)) + geom_point() + labs(x="Hits Allowed", y="Wins")
ggplot(train, aes(x=TEAM_PITCHING_BB, y=TARGET_WINS)) + geom_point() + labs(x="Walks Allowed", y="Wins")

lm_off <- lm(TARGET_WINS ~ TEAM_BATTING_H+TEAM_BATTING_2B+TEAM_BATTING_3B+TEAM_BATTING_HR +TEAM_BASERUN

summary(lm_off)
vif(lm_off)
par(mfrow=c(2,2))
par(mai=c(.3,.3,.3,.3))
plot(lm_off)
eval_data_url <- "https://raw.githubusercontent.com/waheeb123/Data-621/main/Homeworks/Homework-1/moneyball.csv"

eval <- read.csv(eval_data_url)
par(mfrow=c(2,3))
par(mai=c(.3,.3,.3,.3))
variables <- c("TEAM_BATTING_SO", "TEAM_BASERUN_SB", "TEAM_BASERUN_CS",
              "TEAM_PITCHING_SO", "TEAM_FIELDING_DP")
for (i in 1:(length(variables))) {
  hist(test[[variables[i]]], main = variables[i], col = "lightblue")
}
test <- test |>
  select(-TEAM_BATTING_HBP) |>
  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)))

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

# Predict using the model using all input variables
predict_all <- predict(lm_all, test)
predict_reduced <- predict(lm_all_reduced, test)

predict_transformed_reduced <- predict(lm_trans_reduced, test)

predict_select <- predict(lm_select, test)
predict_off <- predict(lm_off,test)
predict_trans_reduced <- predict(lm_trans_reduced, test)

```

```

rmse <- function(c1, c2){
  sqrt(mean((c1 - c2)^2))
}

#Calculate RMSE and print to screen
rmse_all <- rmse(predict_all, test$TARGET_WINS)
rmse_reduced <- rmse(predict_reduced, test$TARGET_WINS)
rmse_transformed_reduced <- rmse(predict_transformed_reduced, test$TARGET_WINS)

print(rmse_all)
print(rmse_reduced)
print(rmse_transformed_reduced)
# Calculate RMSE and print to screen
rmse_all <- modelr::rmse(lm_all, test)
rmse_reduced <- modelr::rmse(lm_all_reduced, test)
rmse_select <- modelr::rmse(lm_select, test)
rmse_off <- modelr::rmse(lm_off, test)
rmse_trans_reduced <- modelr::rmse(lm_trans_reduced, test)

print(glue('RMSE lm_all: {rmse_all}'))
print(glue('RMSE lm_reduced: {rmse_reduced}'))
print(glue('RMSE lm_select: {rmse_select}'))
print(glue('RMSE lm_off: {rmse_off}'))
print(glue('RMSE lm_transreduced {rmse_trans_reduced}'))
# Predict and plot on evaluation data (no wins listed)
predict_all_eval <- as.data.frame(predict(lm_all, eval))

prediction_reduced <- predict(lm_all_reduced, eval)
predict_reduced_eval <- as.data.frame(prediction_reduced)

# Plot reduced model evaluation
ggplot(predict_reduced_eval,
       aes(x=prediction_reduced)) +
  geom_histogram(bins=15) +
  labs(x="Wins",
       title="Predicted Wins (Reduced Model): Evaluation Data.")
predict_off_eval <- as.data.frame(predict(lm_off, eval))

ggplot(predict_off_eval,
       aes(x =predict(lm_off,eval))) + geom_histogram(bins=15) + labs(x="Wins", title="Predicted Wins (Offensive Model): Evaluation Data.")

library(knitr)

# Create a dataframe with model names and RMSE values
model_rmse <- data.frame(
  Model = c("M_{reduced}", "M_{offensive}", "M_{transformed}", "M_{select}"),
  RMSE = c(13.16778, 13.17077, 13.40541, 13.7739)
)

# Print the table
kable(model_rmse, format = "markdown")

```