Moneyball

Biguzzi, Connin, Greenlee, Moscoe, Sooklall, Telab, and Wright

9/26/2021

Introduction

How can we predict the number of wins for a baseball team in a given season based on information about its performance during the season? In this report, we construct a linear model to answer this question. The `moneyball` dataset may lend some insights into what elements of a baseball team's performance are most important in securing wins. Variables that are strongly associated with a greater number of wins for a team would be worthy of further study.

In this report we will

- explore the data
- transform the data to meet conditions of linear modeling
- compare models
- select an optimal model
- generate predictions for the evaluation data set

Data Exploration

As part of our initial data exploration we want to look at the following key diagnostics:

- missingness:
  - missing values might provide opportunity for dummy variables, or lead to elimination of data
- exploring normality of the variable distributions:
  - diagnostics in distributions that help us detect potential problems or opportunities for modeling such as skewness and outliers
- potential covariance among predictors:
  - covariance may lead us to select or mutate certain variables

Exploring the data is essential to uncovering any surprises or unusual features in the data. As we explore the data, we also bear in mind the conditions for linear modeling:

- The residuals of the model should be nearly normal;

- The variability of the residuals should be nearly constant;

- The residuals are independent;

- Each variable is linearly related to the target variable.

While almost no real-world data set conforms perfectly to these conditions, there may be transformations that we can perform on the data to alleviate some of its violations of these conditions.

The data set contains 2,276 rows each representing the full-season performance of a baseball team between the years 1871 and 2006 inclusive.

Let's look first at a numerical summary of each variable:

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
INDEX	0	1	1268	736	1	631	1270	1916	2535
TARGET_WINS	0	1	81	16	0	71	82	92	146
TEAM_BATTING_H	0	1	1469	145	891	1383	1454	1537	2554
TEAM_BATTING_2B	0	1	241	47	69	208	238	273	458
TEAM_BATTING_3B	0	1	55	28	0	34	47	72	223
TEAM_BATTING_HR	0	1	100	61	0	42	102	147	264
TEAM_BATTING_BB	0	1	502	123	0	451	512	580	878
TEAM_BATTING_SO	102	1	736	249	0	548	750	930	1399
TEAM_BASERUN_SB	131	1	125	88	0	66	101	156	697
TEAM_BASERUN_CS	772	1	53	23	0	38	49	62	201
TEAM_BATTING_HBP	2085	0	59	13	29	50	58	67	95
TEAM_PITCHING_H	0	1	1779	1407	1137	1419	1518	1682	30132
TEAM_PITCHING_HR	0	1	106	61	0	50	107	150	343
TEAM_PITCHING_BB	0	1	553	166	0	476	536	611	3645
TEAM_PITCHING_SO	102	1	818	553	0	615	814	968	19278
TEAM_FIELDING_E	0	1	246	228	65	127	159	249	1898
TEAM_FIELDING_DP	286	1	146	26	52	131	149	164	228

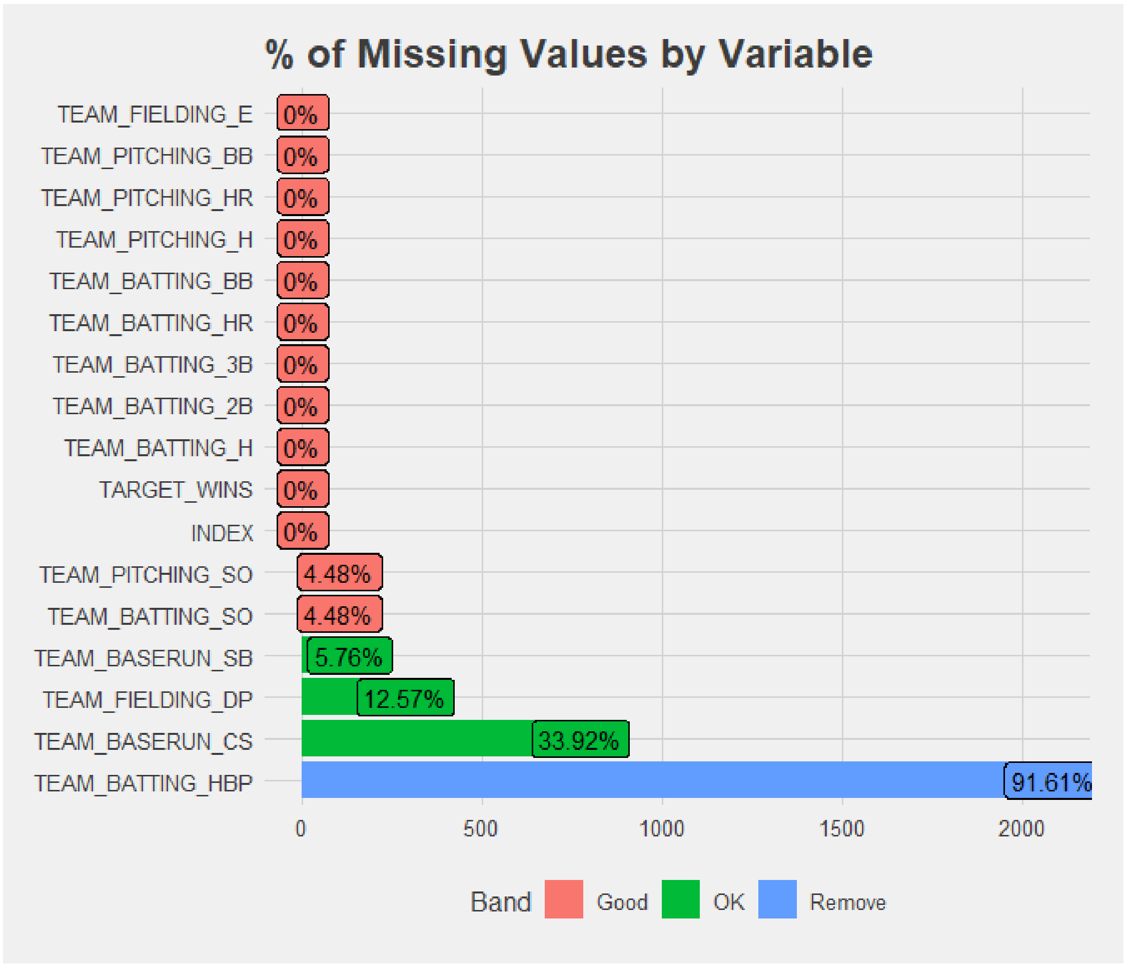
From the summary above, we can see that some variables show a significant number of missing values, especially `TEAM_BATTING_HBP` and `TEAM_BASERUN_CS`.

Some variables show suspiciously large maximum values, such as `TEAM_PITCHING_H`, `TEAM_PITCHING_BB`, `TEAM_PITCHING_SO`, and `TEAM_FIELDING_E`.

We can also see that some variables contain entries of zero that don't make sense in the context of a baseball season. These variables include `TARGET_WINS`, `TEAM_BATTING_3B`, `TEAM_BATTING_HR`, `TEAM_BATTING_BB`, `TEAM_BATTING_SO`, `TEAM_BASERUN_SB`, `TEAM_BASERUN_CS`, `TEAM_PITCHING_HR`, `TEAM_PITCHING_BB`, `TEAM_PITCHING_SO`.

At least some of these entries we know to be erroneous. For example, the all-time minimum batting strikeouts for a team over a complete season was 308, achieved by the 1921 Cincinnati Reds.

How we respond to the missingness in the data will depend on how it is distributed across columns and rows.



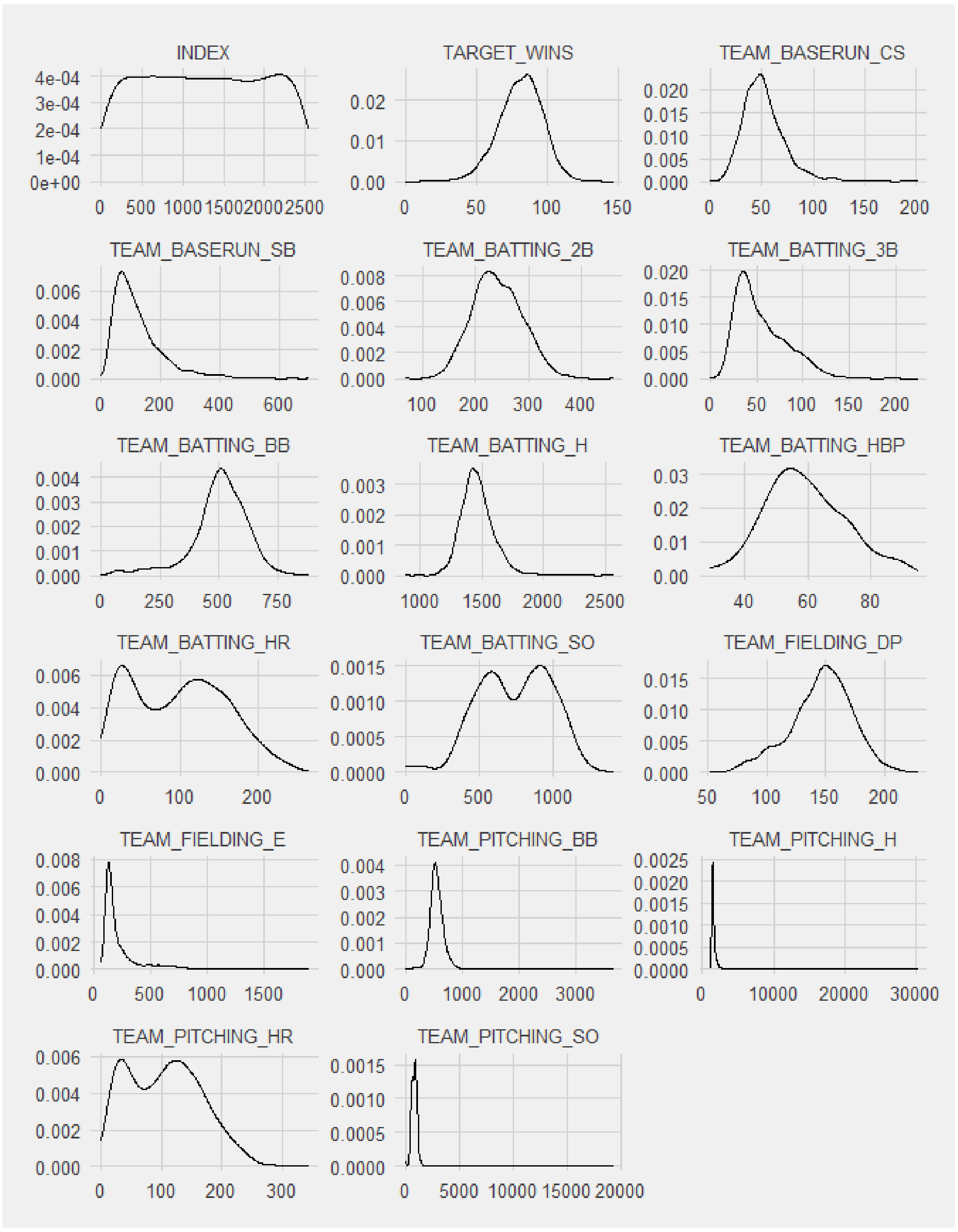
Here we see again that missing entries are not distributed randomly throughout the data set. In particular, `TEAM_BATTING_HBP` is comprised of almost 92% missing values. This variable cannot provide much information to our model and will be dropped.

`TEAM_BASERUN_CS` also displays a very high fraction of missing values: 34%. `TEAM_PITCHING_SO` and `TEAM_BATTING_SO` share the exact same number of missing values. This suggests that the missingness in these variables is not random.

Further investigation reveals that the distribution of missingness in both these variables is identical. One option for responding to the missingness in these variables will be to code it as its own factor variable. Another option may be to drop one or both these variables from our models if there are further reasons to be suspicious about the quality of the data contained in them.

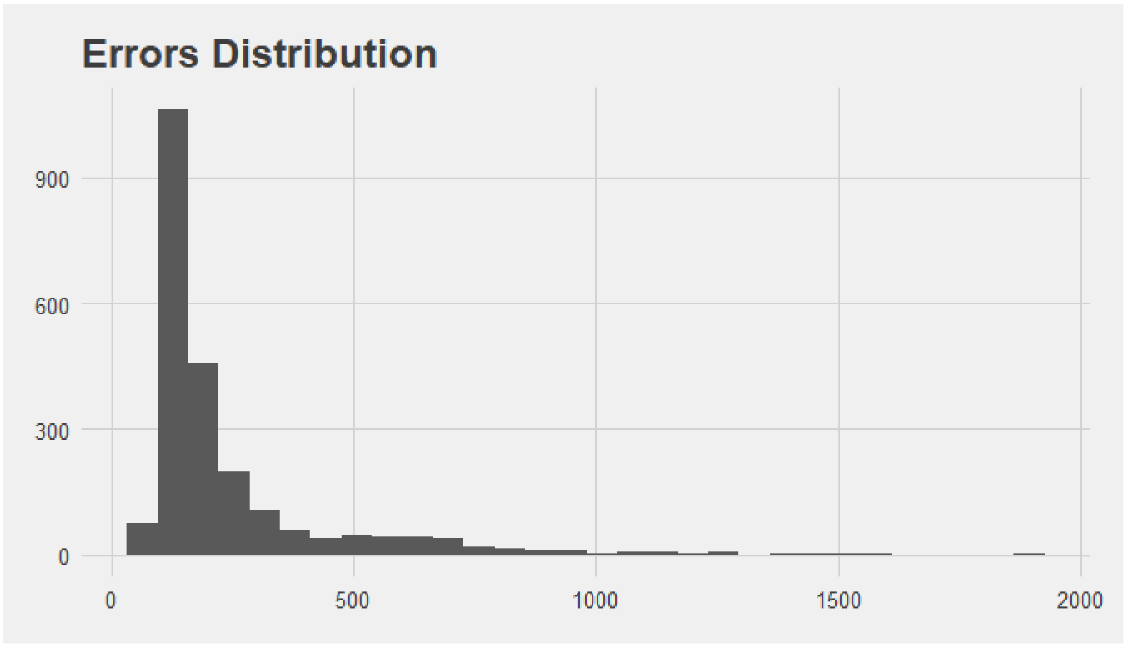
Having looked at missing values, and identifying points of concern that we will return to later in the report, we will now look at some characteristics of the predictor variables.

Examining the shape of each variable can help identify any unusual attributes including extreme skewness or low variance.

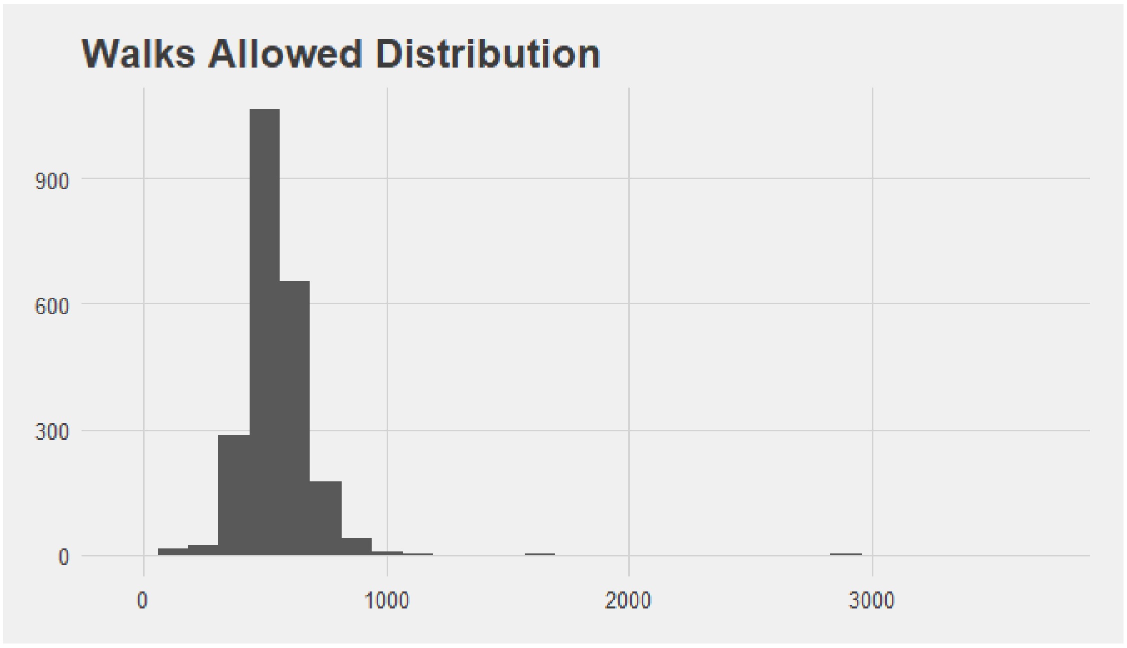


Density plots for each variable reveals the highly skewed shapes of `TEAM_FIELDING_E`, `TEAM_PITCHING_BB`, `TEAM_PITCHING_H`, and `TEAM_PITCHING_SO` as suggested by the numeric summary above. `TEAM_BASERUN_SB` and `TEAM_BATTING_3B` display moderate skewness. Other distributions appear roughly normal or bimodal.

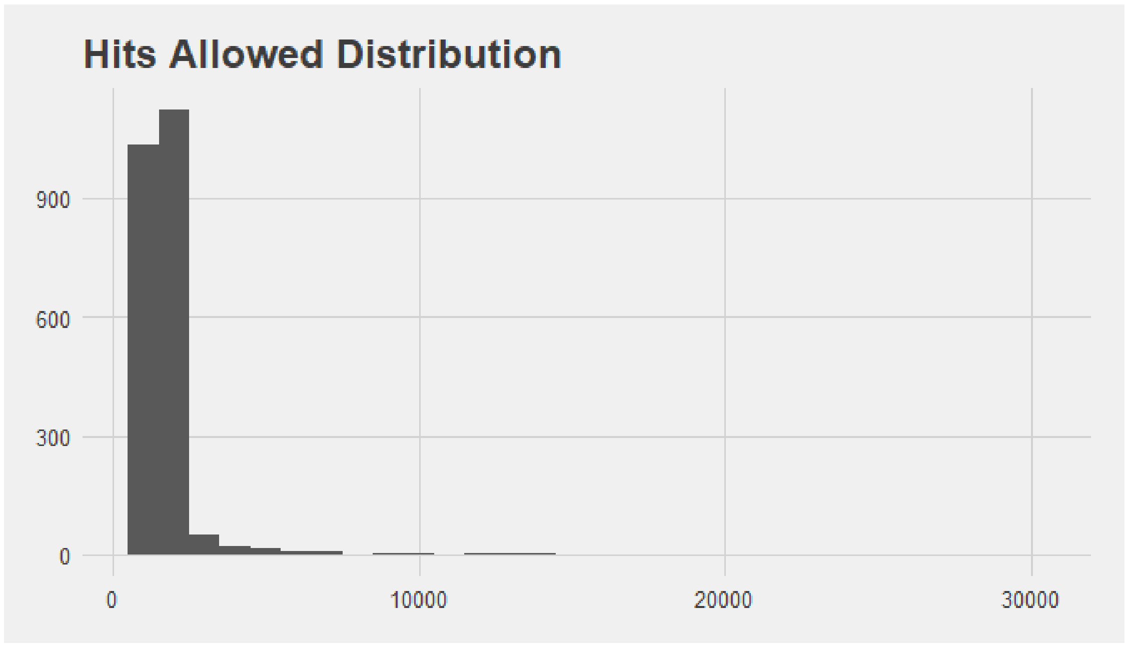
Looking more closely at each of the highly skewed variables may suggest a reason for their non-normality.



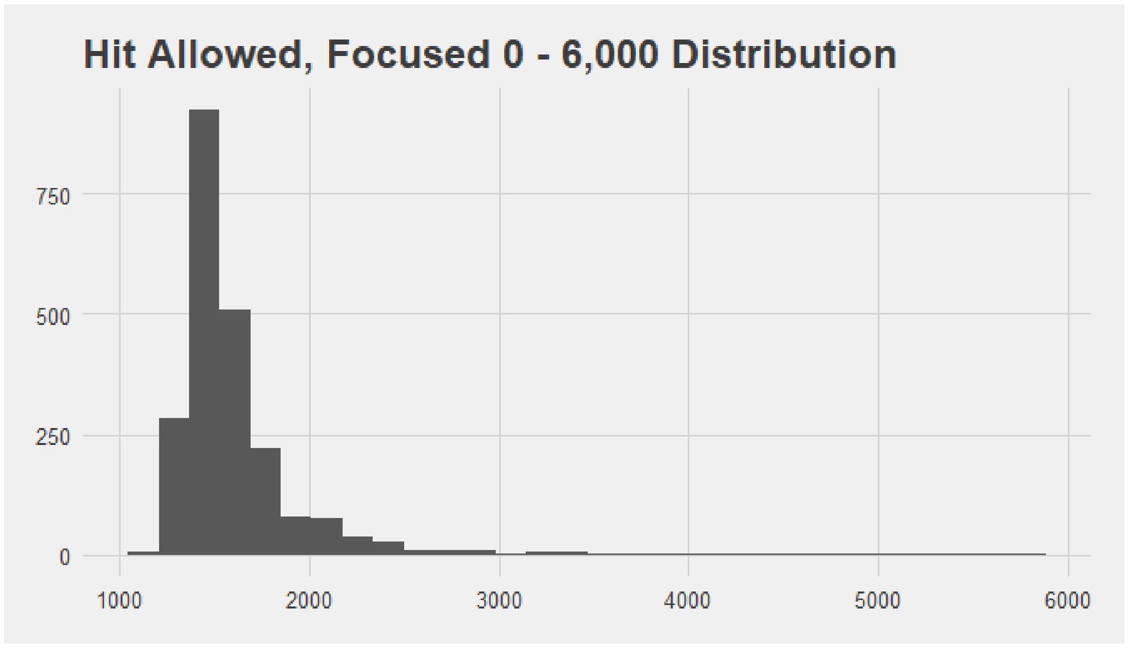
In TEAM_FIELDING_E, a very small number of entries exceed 1000. Excluding these, the distribution is moderately skewed right.



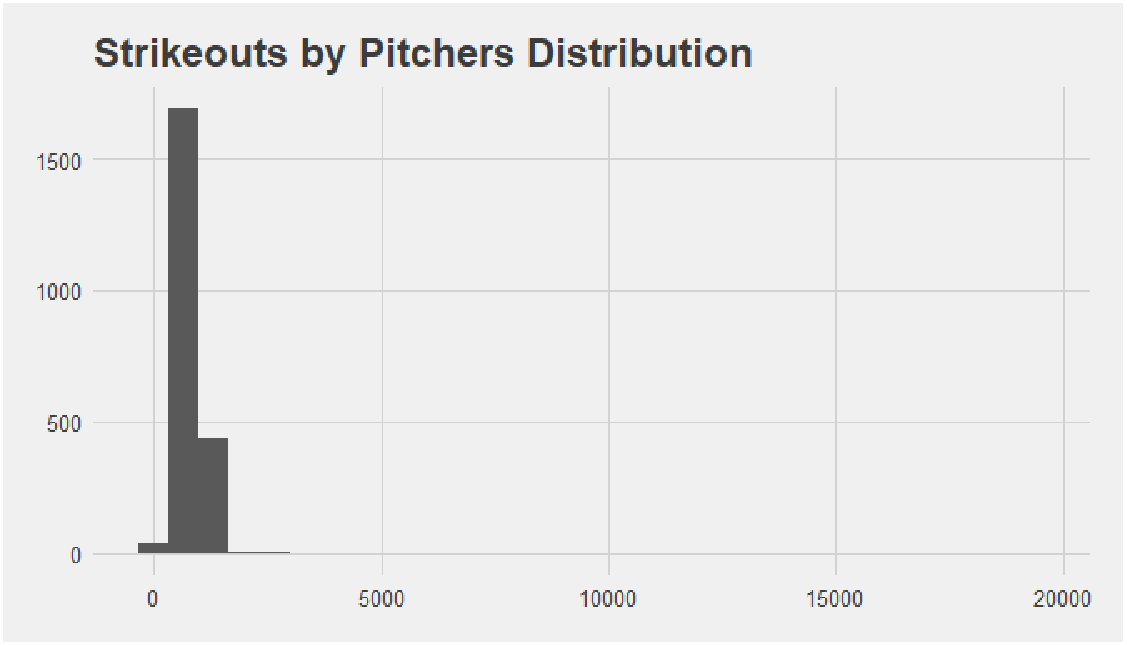
The distribution of TEAM_PITCHING_BB appears normal except for a very small number of entries greater than 1000.



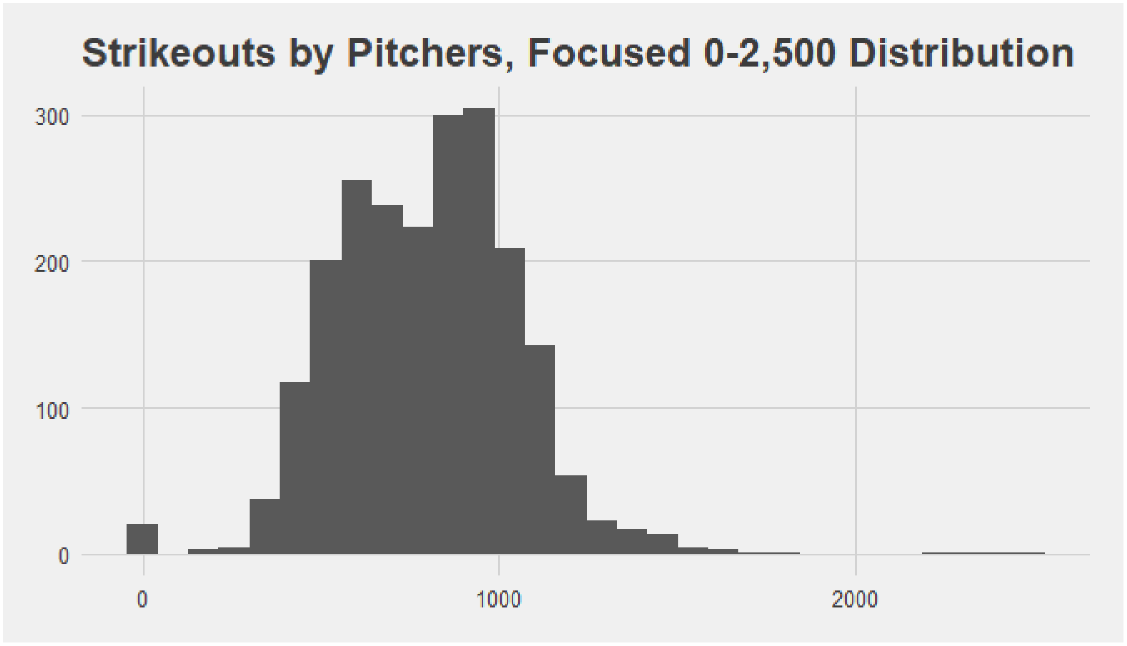
Let's look more closely at the region where most of the values in `TEAM_PITCHING_H` lie, [0,6000]:



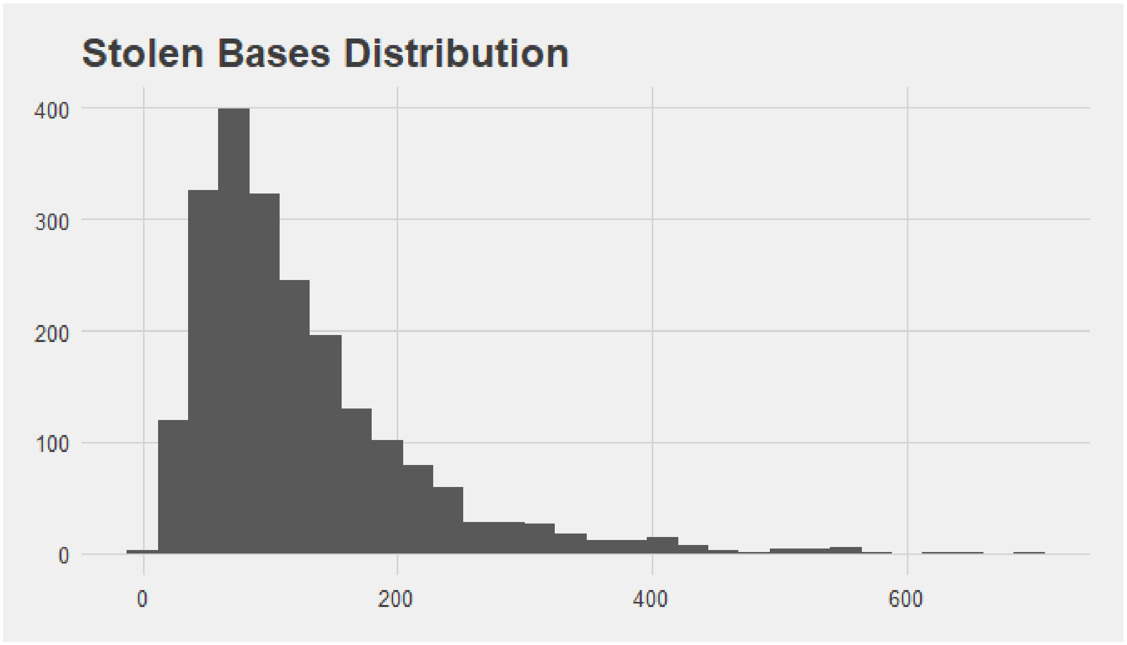
`TEAM_PITCHING_H` is skewed even within this narrower region surrounding the peak. The presence of skew and the wide separation of data points along the right tail may highlight some unique phenomenon, data entry errors, or structural changes, (such as rule changes, or changes in the equipment used) that influenced circumstances on the ground.



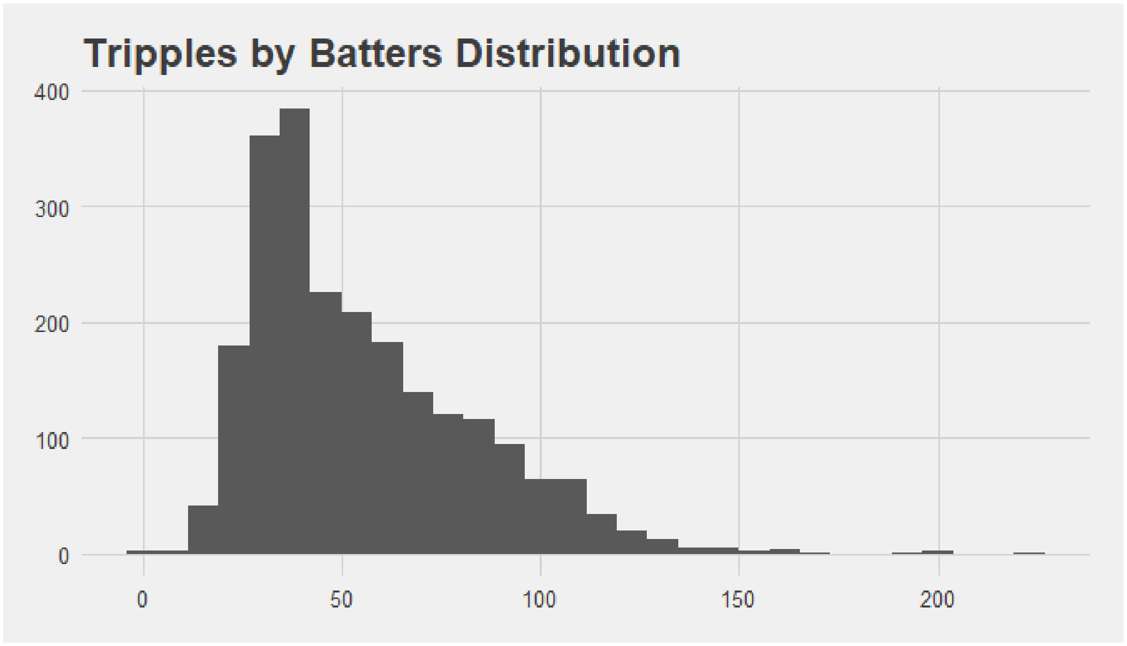
Narrowing in on `TEAM_PITCHING_SO`:



The data appear roughly normal, but there are a significant number of unrealistic values. The record for strikeouts all time was set in 2018 by the Houston Astros at 1,687, and we can see from this distribution there are a significant amount of data above this mark. It is also worth noting the all time record for least strikeouts was set by the 1921 Cincinnati Reds at 308. Not only are there many values below this, but we have a significant amount of zero values.



The shape of TEAM_BASERUN_SB does not suggest any data entry errors, even though there exist a few extreme outliers.



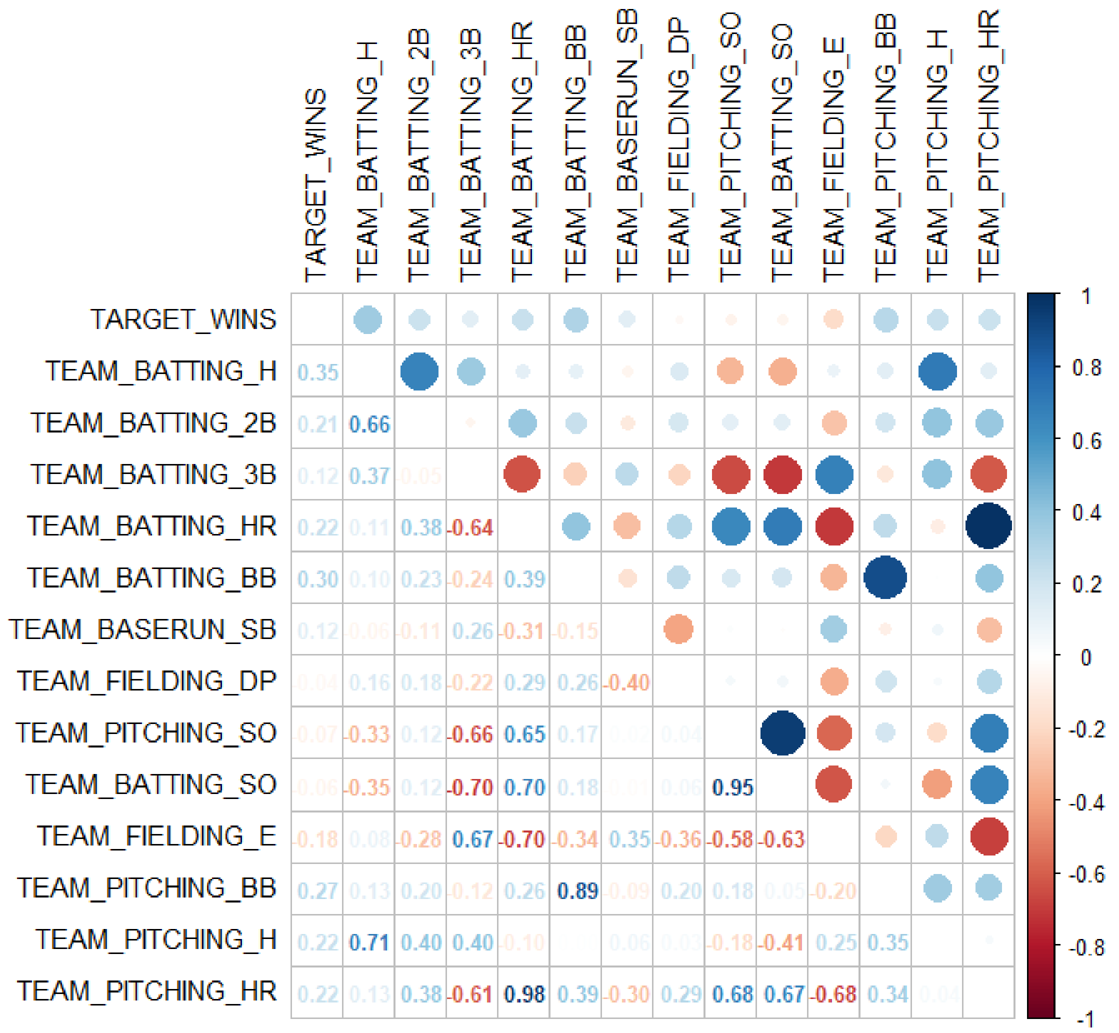
The shape of TEAM_BATTING_3B does not suggest any data entry errors, even though there exist a few extreme outliers.

As demonstrated in our examination of TEAM_PITCHING_SO we are aware of outliers that lie beyond the historical maximum and minimum in these categories. These type of outliers are present in all of the variables that would lead us to conclude at least some of the data is erroneous. The best course of action under these conditions is to pause our investigation and return to the source of the data to discover why our data does not match historic baseball data sets. Because this is not possible, we'll use multiple imputation to replace extreme outliers and missing values.

This decision entails some risk, because replacing data with imputed values could impair the quality of our model. However in this case, it's our judgment that incorporating these extreme, unrealistic values would do greater damage to any models that are trained on them. In order to proceed with caution, we'll train models on two versions of the data–one with extreme values replaced by imputation, and one with extreme values unchanged.

One important condition of linear modeling is that independent variables should be uncorrelated with each other. In real-world data it is rarely possible to fully satisfy this condition. However, we can seek to avoid large pairwise correlations between variables.

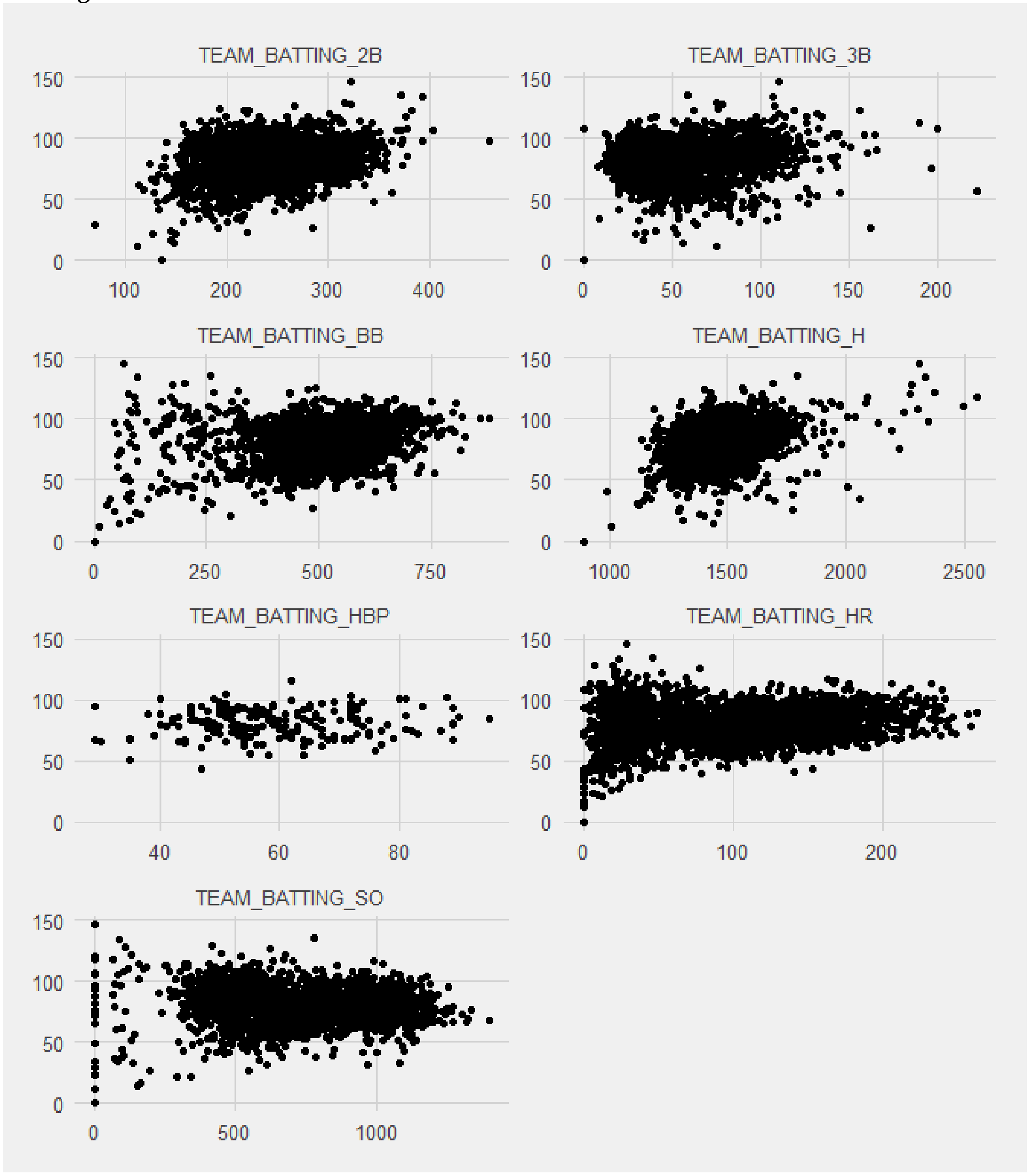
The correlation plot below is arranged so that all variables with a theoretical positive effect on `TARGET_WINS` appear at the top and left, followed by variables with a theoretical negative effect. We would expect positive correlations among variables with positive effects, positive correlations among variables with negative effects, and negative correlations among variables with opposite effects.



But this is not what we see. Instead we see that no variable alone has correlation greater than 0.35 in either direction with the target. We also see that some pairs of variables have correlations that are so strong or misdirected that we have reason to doubt the integrity of the data. Note that `TEAM_PITCHING_HR` and `TEAM_BATTING_HR`, with correlation 0.98, contain essentially the same information. `TEAM_PITCHING_SO` and `TEAM_BATTING_SO`, the variables whose patterns of missingness are identical, also demonstrate an unreasonably high correlation of 0.95.

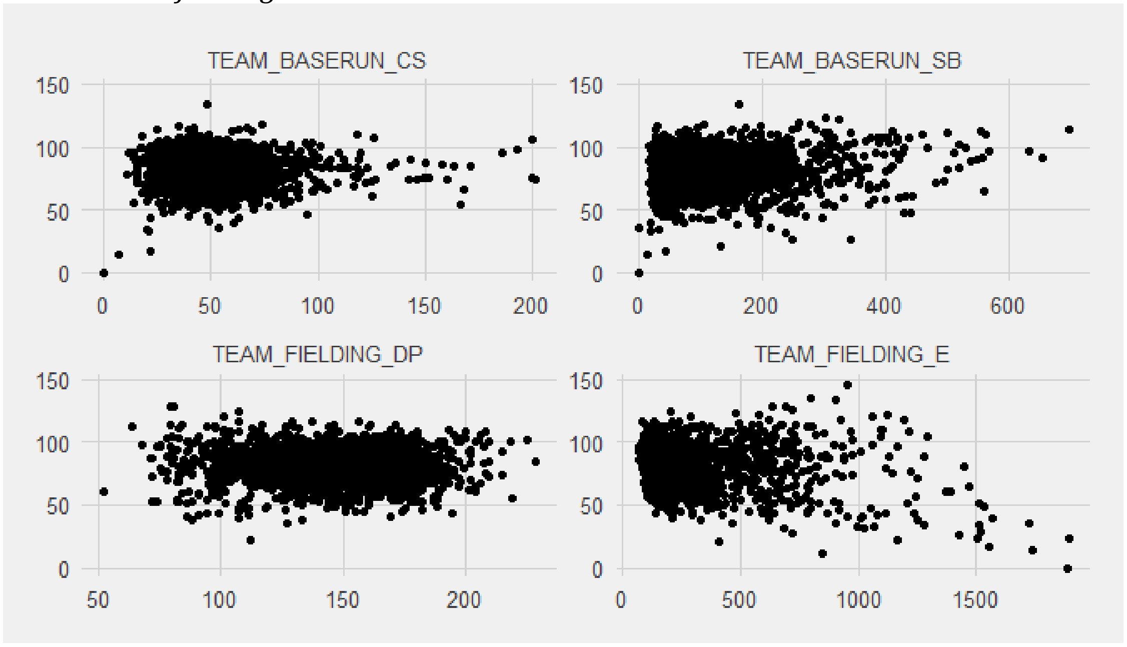
Another condition of linear models is that independent variables be linearly related to the target. We can examine this using scatterplots, one group of variables at a time.

Batting variables



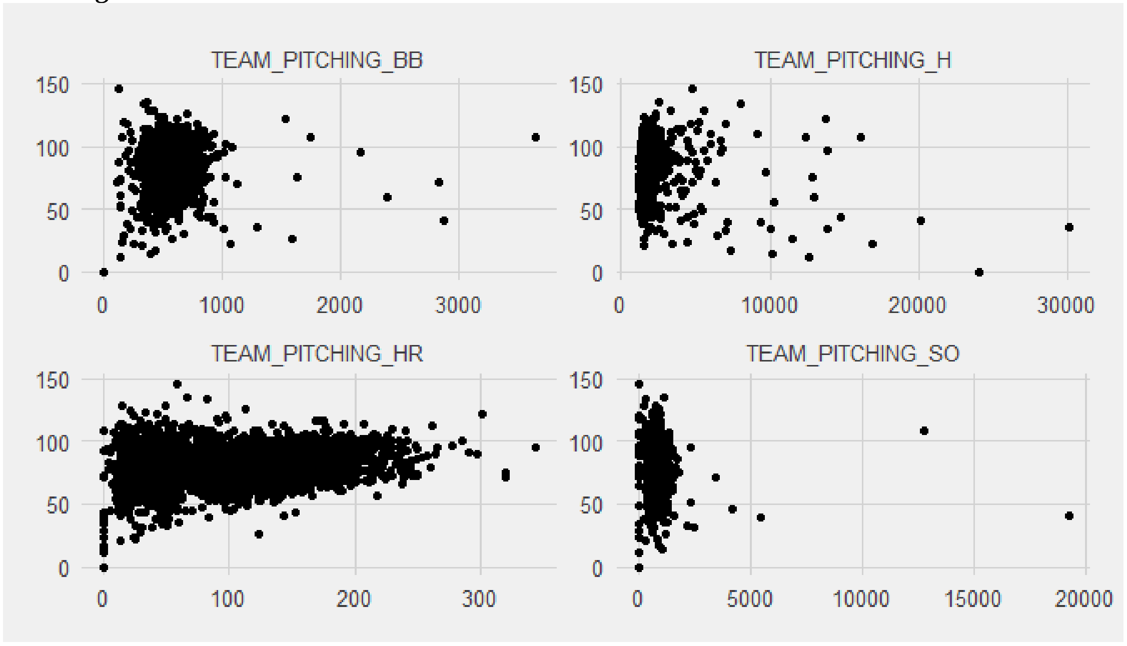
Excluding `TEAM_BATTING_HBP`, the batting variables have potential for positive linear relationships. These will need to be investigated further.

Baserun and fielding variables



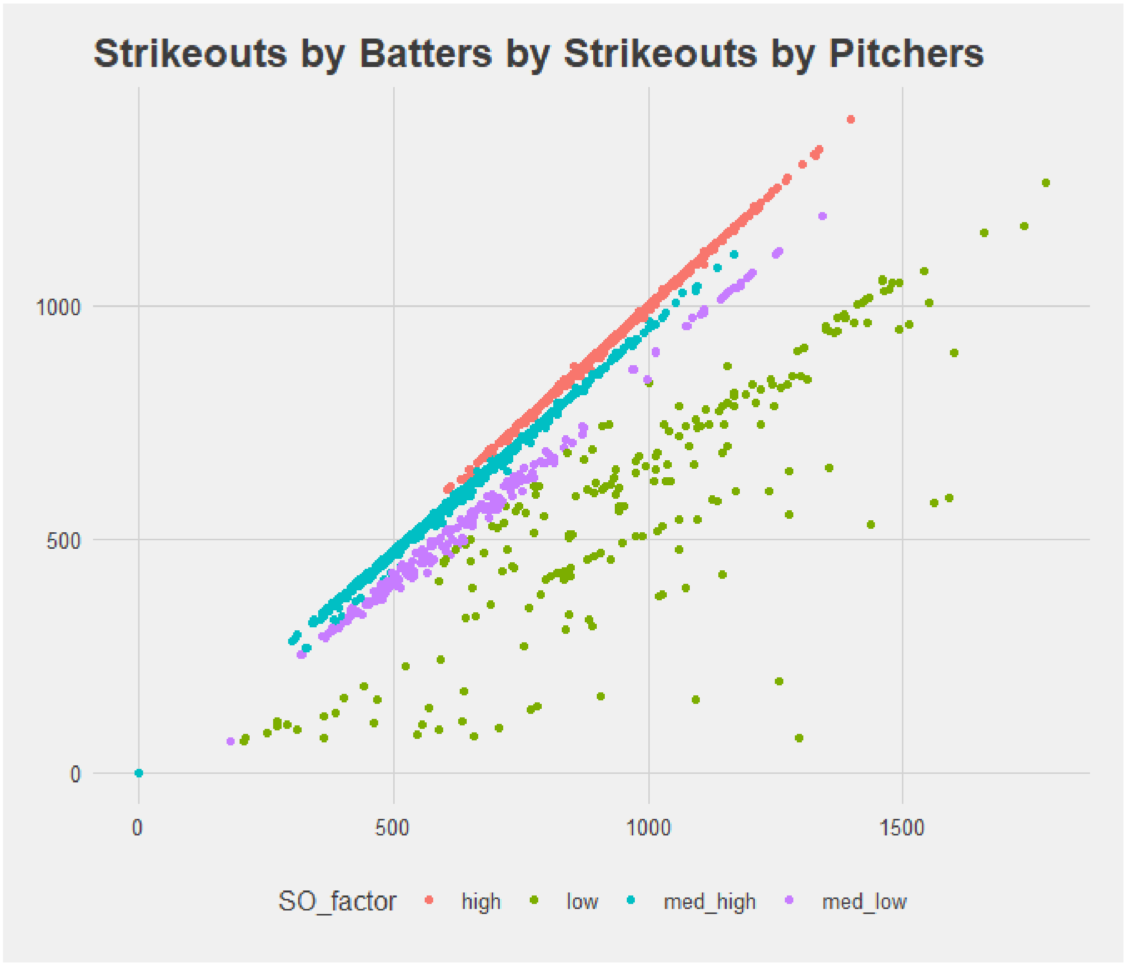
The situation with these variables is similar to that of the batting variables. There is potential for linearity, but we will have to investigate further.

Pitching variables



No linearity is evident in the scatterplots for the pitching variables. What is evident again is the presence of extreme outliers and skewness in three of the four pitching variables.

An unexpected finding in our exploratory data analysis is that a scatterplot of `TEAM_BATTING_SO` and `TEAM_PITCHING_SO` suggests that the data can be divided into 4 distinct groups, three of which contain highly correlated data.



There is no theoretical reason to expect such a grouping, or to expect such high correlations between the number of strikeouts a team incurs while batting, and the number of strikeouts a team achieves while pitching. The fact that there is no “theoretical reason” would imply there is a problem with the data. Since we have already decided to press on with our analysis, these patterns could be treated as natural features of the data that we want to draw out in our model.

Data Preparation

To prepare the data for modeling, we first simplify variable names by removing the prefix `TEAM_`, and we add the grouping variable `SO_FACTOR`.

An understanding of baseball suggests how to combine some variables into other meaningful measures of team performance, such as `_OBP`. This is a rate statistic based on how many times you reach base on a hit or walk, vs how many times you came to the plate. Creating new variables as combinations of existing variables will help alleviate collinearity among some pairs of variables, and it will lead to models that better reflect commonly used measures of team performance.

We create variables to represent net stolen bases (`BASERUN_NET_SB`), offensive on-base percentage (`OFFENSE_OBP`), defensive on-base percentage (`DEFENSE_OBP`), and total at-bats (`TOT_AT_BATS`). Then we drop the variables that were combined to produce these new variables.

After these initial transformations, we split the data into a training set (80% = 1820 rows) and a test set (20% = 456 rows).

During our exploration of the data, we noted several variables with extreme outliers and described our strategy of examining two versions of our models: one trained on data with only missing values imputed, and one trained on data with missing values and outliers imputed. We'll consider all these models before making our final selection.

Replacing missing data within a given variable with it's column mean or median will introduce bias in the data. This results from ignoring the influence of other variables on these values as well as naturally occurring variability in the data. To reduce the potential for such bias, we employed a coupled regression/bootstrapping method (via the MICE package) to impute our missing data. Our diagnostic checks (not included here) indicated that this method produced better results (distributions comparable to the actual data) relative to other available methods (e.g., pmm, random forest, etc.).

A look at the summaries of our test sets, both with and without outliers imputed, shows the raw materials for our models.

With missing values imputed

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
INDEX	0	1	1266	739	1	621	1280	1903	2534
BATTING_2B	0	1	242	47	69	209	238	274	458
BATTING_3B	0	1	55	28	0	34	47	72	223
BATTING_HR	0	1	101	61	0	44	104	148	264
BATTING_SO	0	1	731	253	0	543	748	928	1399
PITCHING_HR	0	1	107	61	0	53	108	152	343
PITCHING_SO	0	1	810	592	-248	603	810	966	19278
BASERUN_NET_SB	0	1	38	40	-140	13	35	60	628
OFFENSE_OBP	0	1	0	0	0	0	0	0	0
DEFENSE_OBP	0	1	0	0	0	0	0	0	4
TOT_AT_BATS	0	1	6325	226	1270	6217	6317	6424	7518
TARGET_WINS	0	1	81	16	0	71	82	92	146

With missing values and outliers imputed

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
INDEX	0	1	1266	739	1	621	1280	1903	2534
BATTING_2B	0	1	241	41	120	211	238	272	330
BATTING_3B	0	1	54	24	3	35	47	70	117
BATTING_HR	0	1	100	57	-35	46	104	147	232
BATTING_SO	0	1	739	233	-788	556	748	925	1246
PITCHING_HR	0	1	106	57	-2	56	109	150	243
PITCHING_SO	0	1	798	220	289	620	810	957	1453
BASERUN_NET_SB	0	1	38	30	-54	16	36	57	129
OFFENSE_OBP	0	1	0	0	0	0	0	0	0
DEFENSE_OBP	0	1	0	0	0	0	0	0	1
TOT_AT_BATS	0	1	6314	131	5898	6219	6309	6401	6723
TARGET_WINS	0	1	81	16	0	71	82	92	146

Build Models

In this section we construct models first for training sets with only missing values imputed, and we evaluate them with a test set with only missing values imputed. Then, we train and test analogous models with outliers imputed as well.

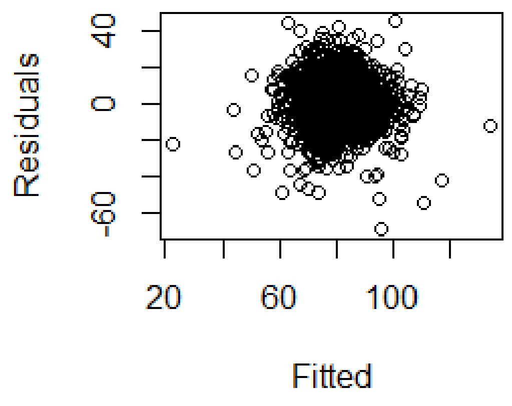
Model 1. Almost all variables

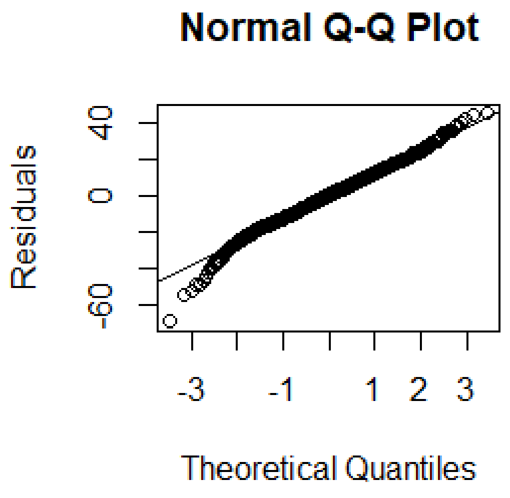
This model contains all variables except `BATTING_HR`, `BATTING_SO`, and `TOT_AT_BATS`, because each of these variables is highly correlated with at least one other variable.

With only missing values imputed

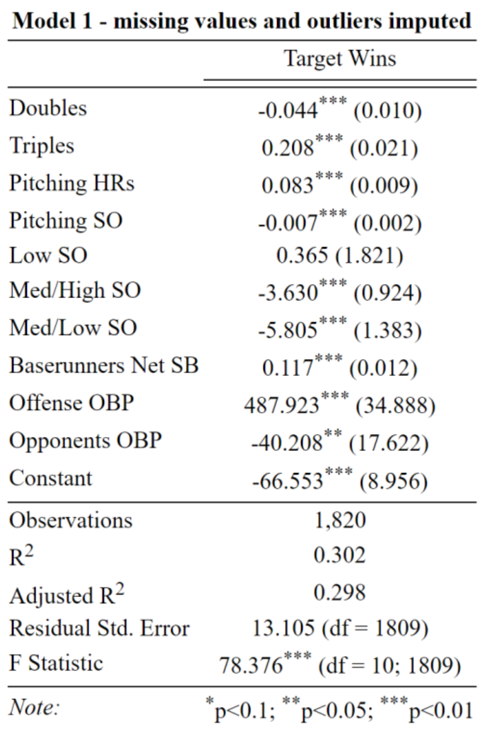
Model 1 - with only missing values imputed

	Target Wins
Doubles	-0.001 (0.008)
Triples	0.176^{***} (0.016)
Pitching HRs	0.082^{***} (0.008)
Pitching SO	-0.005^{**} (0.002)
Low SO	-9.508^{***} (1.341)
Med/High SO	-0.699 (0.887)
Med/Low SO	-5.064^{***} (1.324)
Baserunners Net SB	0.130^{***} (0.010)
Offense OBP	220.226^{***} (23.585)
Opponents OBP	36.197^{***} (8.948)
Constant	-18.956^{***} (6.597)
Observations	1,820
R^2	0.318
Adjusted R^2	0.315
Residual Std. Error	12.953 (df = 1809)
F Statistic	84.478^{***} (df = 10; 1809)
Note:	*p<0.1; **p<0.05; ***p<0.01



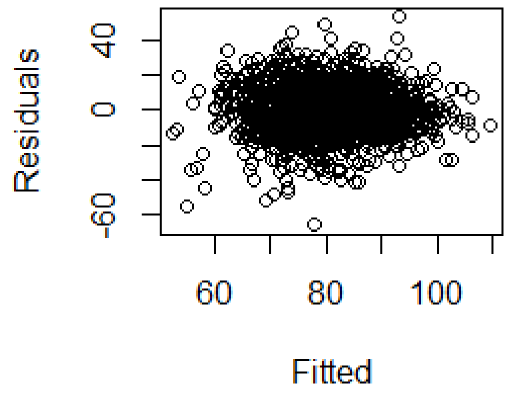


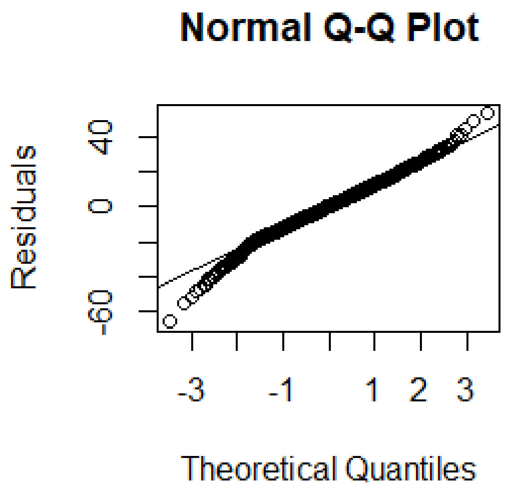
With missing values and outliers imputed



Model 1 - missing values and outliers imputed

	Target Wins
Doubles	-0.044*** (0.010)
Triples	0.208*** (0.021)
Pitching HRs	0.083*** (0.009)
Pitching SO	-0.007*** (0.002)
Low SO	0.365 (1.821)
Med/High SO	-3.630*** (0.924)
Med/Low SO	-5.805*** (1.383)
Baserunners Net SB	0.117*** (0.012)
Offense OBP	487.923*** (34.888)
Opponents OBP	-40.208** (17.622)
Constant	-66.553*** (8.956)
Observations	1,820
R^2	0.302
Adjusted R^2	0.298
Residual Std. Error	13.105 (df = 1809)
F Statistic	78.376*** (df = 10; 1809)
Note:	*p<0.1; **p<0.05; ***p<0.01





Evaluation and discussion

The root mean square error for each model is reported below.

Test set RMSE for model 1 with only missing imputed: 15.57
Test set RMSE for model 1 with missing and outliers imputed: 14.59

Our initial look at the residuals for the model with only missing values imputed shows some heteroskedasticity, violating one of our modeling conditions. It appears the variance of the residuals decreases at extreme values. We also notice from the QQ plot that the residuals depart from a normal distribution at the extremes.

Plots for the model trained on the imputed outliers show improvement on the issues of the previous model. This could be a hint that the extreme outliers that have been removed were affecting our model.

RMSE values for these models on the test set are similar to the RMSE values for the training sets. This means that these models avoid overfitting the data.

Model 2. Piecewise by FACTOR_SO

The scatterplot of `BATTING_SO` vs. `PITCHING_SO` suggested four groups in this data. Here we fit one model to each group. Since we conduct a piecewise analysis on two separate training sets, 8 sets of outputs are generated for models in this section. Rather than display this model output, we summarize our findings.

Evaluation and discussion

The root mean square error for each model is reported below.

RMSE for piecewise models with missing values imputed
RMSE for training set: 12.3 RMSE for test set: 15.43

RMSE for piecewise models with missing values and outliers imputed
RMSE for training set: 12.67 RMSE for test set: 14.82

This piecewise modeling approach returned RMSE values similar to those of our previous model. Residual and QQ plots for component models where only missing values were imputed demonstrate the same heteroskedasticity and deviation from normality that we saw in Model 1. Again, these issues are somewhat mitigated in the model where outliers are also imputed.

This second model was prompted by the results of our `BATTING_SO` vs. `PITCHING_SO` plot. We acknowledge some issues with this approach– one being that the models differ greatly in population size, so a visual comparison may not be trustworthy. Also, we may not have the appropriate context to make the decision to divide the data based on the relationship between these two predictors. While this might have led to overfitting, RMSE values that are similar in both the training and test sets suggest that we avoided this.

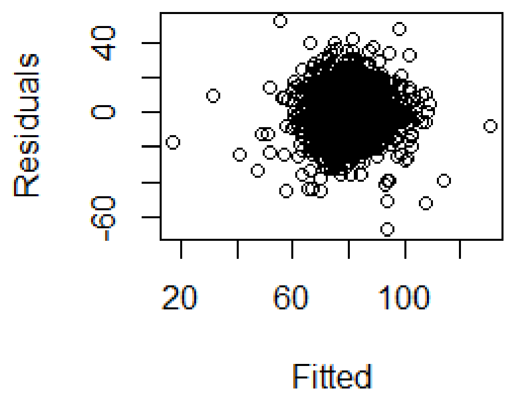
Model 3. Drop DEFENSE_OBP

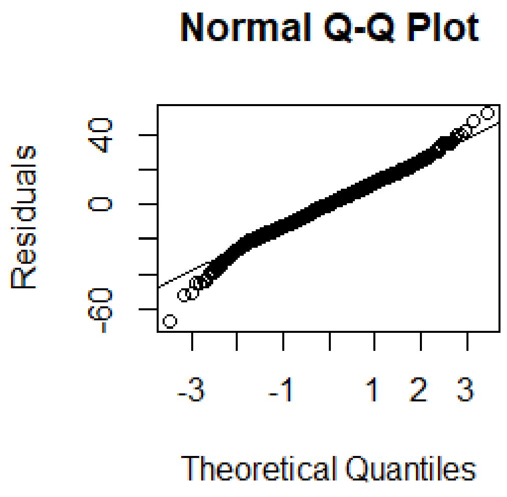
Here we drop `DEFENSE_OBP` from Model 1. Even though this variable is a statistically significant predictor of `TARGET_WINS`, its large coefficient and standard error compared to the other variables might be due to its high correlation with `OFFENSE_OBP`.

With only missing values imputed

Model 3 - only missing values imputed

	Target Wins
Doubles	0.001 (0.008)
Triples	0.192^{***} (0.016)
Pitching HRs	0.065^{***} (0.007)
Pitching SO	0.003^{***} (0.001)
Low SO	-7.935^{***} (1.289)
Med/High SO	0.760 (0.814)
Med/Low SO	-3.925^{***} (1.299)
Baserunners Net SB	0.123^{***} (0.010)
Offense OBP	269.455^{***} (20.288)
Constant	-28.857^{***} (6.152)
Observations	1,820
R^2	0.312
Adjusted R^2	0.309
Residual Std. Error	13.008 (df = 1810)
F Statistic	91.271^{***} (df = 9; 1810)
Note:	*p<0.1; **p<0.05; ***p<0.01

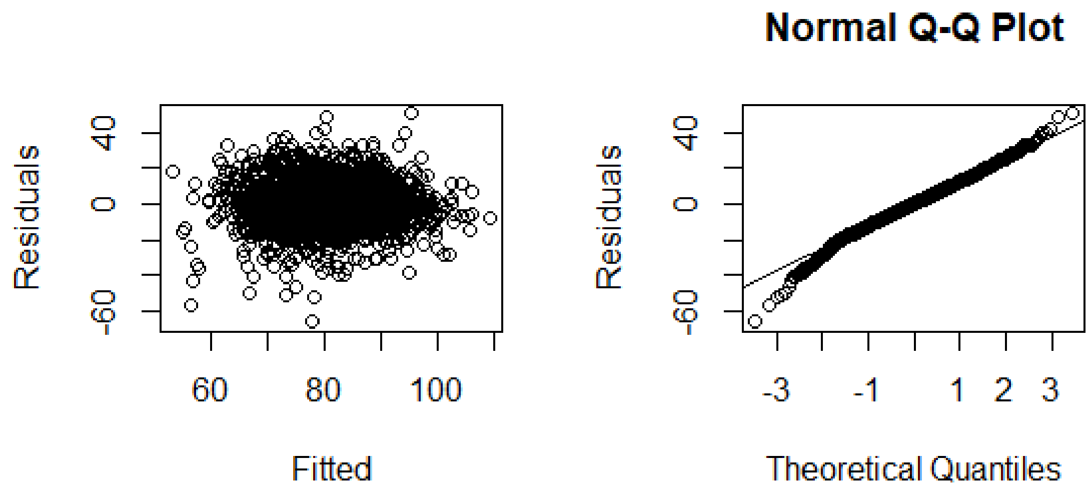




With missing values and outliers imputed

Model 3 - missing values and outliers imputed

	Target Wins
Doubles	-0.043*** (0.010)
Triples	0.196*** (0.020)
Pitching HRs	0.083*** (0.009)
Pitching SO	-0.008*** (0.002)
Low SO	-2.658** (1.251)
Med/High SO	-4.107*** (0.901)
Med/Low SO	-6.805*** (1.313)
Baserunners Net SB	0.120*** (0.012)
Offense OBP	449.444*** (30.576)
Constant	-66.136*** (8.965)
Observations	1,820
R^2	0.300
Adjusted R^2	0.297
Residual Std. Error	13.120 (df = 1810)
F Statistic	86.306*** (df = 9; 1810)
Note:	*p<0.1; **p<0.05; ***p<0.01



Evaluation and discussion

The root mean square error for each model is reported below.

RMSE for piecewise models with missing values imputed
RMSE on training set: 12.3
RMSE on test set: 15.43

RMSE for piecewise models with missing values and outliers imputed
RMSE on training set: 12.67
RMSE on test set: 14.82

In this model, we see residuals behaving similarly to the first model. We can call attention again to the improvement in the residuals by the training of the data with the imputed outliers. Not only is the shape of the residuals more homoskedastic and normal, but the statistical significance of the coefficients is much greater.

Conclusion and model selection

Before we select the best performing model, let's review the above analysis.

Data Exploration. We recognize a high probability of flawed data. Unrelated variables were found to have near perfect correlation. Outliers represent impossible measurements. These occurrences call for additional scrutiny toward the less egregious features of the data, such as bimodal distributions. In a practical setting, this would need to be addressed before moving forward.

Data Preparation. Pushing forward with the suspicious data, we proceeded cautiously by creating a factor that represented particular cohorts within the data. We created new variables based on our knowledge of baseball.

Modeling. In the first model, we incorporated all the remaining features of the data. In the second model, we trained separate models for each level of `SO_FACTOR`, and combined these models into an overall piecewise linear model. In the third model, we improved on the first model by dropping one of a pair of highly correlated variables.

Below, we restate the RMSE for each model constructed.

Models with only missing values imputed
RMSE for model 1, missing values imputed: 15.57
RMSE for model 2, missing values imputed: 15.43
RMSE for model 3, missing values imputed: 15.41

Models with missing vlues and outliers imputed
RMSE for model 1, missing values and outliers imputed: 14.59
RMSE for model 2, missing values and outliers imputed: 14.82
RMSE for model 3, missing values and outliers imputed: 14.59

The best model is Model 3 with missing values and outliers imputed. It uses the full training set in a single model, unlike Model 2. Its predictors and the intercept are highly significant, which is not true for every variable in Model 1. We believe the factors added to our model serve as an effective attempt to counteract the flaws in this data set, and the engineered features made the model more efficient and improved performance. The strategy of imputing outliers both decreased RMSE and improved our residual plots.

Appendix

This appendix shows the R code behind the conclusions and visualizations in the body of the report. The text of the report has been left in to help the reader easily associate a code chunk with a particular claim or visualization. This code also resides in the document "Group1_HW1.Rmd".

```
#Setup

library(mice)
library(tidyverse)
library(GGally)
library(psych)
library(stats)
library(corrplot)
library(ggthemes)

library(cowplot)
library(magrittr)
library(skimr)
library(DataExplorer)
library(caret)
library(MASS)
library(regclass)
library(moderndive)
library(memisc)
library(pander)
library(stargazer)

set.seed(210904)

raw <-
read.csv("https://raw.githubusercontent.com/dmoscoe/DATA621/main/data/moneyba
ll-training-data.csv")
eval <-
read.csv("https://raw.githubusercontent.com/dmoscoe/DATA621/main/data/moneyba
ll-evaluation-data.csv")
```

Introduction

How can we predict the number of wins for a baseball team in a given season based on information about its performance during the season? In this report, we construct a linear model to answer this question. The `moneyball` dataset may lend some insights into what elements of a baseball team's performance are most important in securing wins. Variables that are strongly associated with a greater number of wins for a team would be worthy of further study.

In this report we will

- explore the data
- transform the data to meet conditions of linear modeling
- compare models
- select an optimal model
- generate predictions for the evaluation data set

Data Exploration

As part of our initial data exploration we want to look at the following key diagnostics:

- missingness:
  - missing values might provide opportunity for dummy variables, or lead to elimination of data
- exploring normality of the variable distributions:
  - diagnostics in distributions that help us detect potential problems or opportunities for modeling such as skewness and outliers
- potential covariance among predictors:
  - covariance may lead us to select or mutate certain variables

Exploring the data is essential to uncovering any surprises or unusual features in the data. As we explore the data, we also bear in mind the conditions for linear modeling:

- The residuals of the model should be nearly normal;

- The variability of the residuals should be nearly constant;

- The residuals are independent;

- Each variable is linearly related to the target variable.

While almost no real-world data set conforms perfectly to these conditions, there may be transformations that we can perform on the data to alleviate some of its violations of these conditions.

The data set contains 2,276 rows each representing the full-season performance of a baseball team between the years 1871 and 2006 inclusive.

Let’s look first at a numerical summary of each variable:

```r
skim_without_charts(raw)
```

From the summary above, we can see that some variables show a significant number of missing values, especially `TEAM_BATTING_HBP` and `TEAM_BASERUN_CS`.

Some variables show suspiciously large maximum values, such as `TEAM_PITCHING_H`, `TEAM_PITCHING_BB`, `TEAM_PITCHING_SO`, and `TEAM_FIELDING_E`.

We can also see that some variables contain entries of zero that don't make sense in the context of a baseball season. These variables include TARGET_WINS, TEAM_BATTING_3B, TEAM_BATTING_HR, TEAM_BATTING_BB, TEAM_BATTING_SO, TEAM_BASERUN_SB, TEAM_BASERUN_CS, TEAM_PITCHING_HR, TEAM_PITCHING_BB, TEAM_PITCHING_SO.

At least some of these entries we know to be erroneous. For example, the all-time minimum batting strikeouts for a team over a complete season was 308, achieved by the 1921 Cincinnati Reds.

How we respond to the missingness in the data will depend on how it is distributed across columns and rows.

```
plot_missing(raw, title = "% of Missing Values by Variable", ggtheme =
theme_fivethirtyeight())
```

Here we see again that missing entries are not distributed randomly throughout the data set. In particular, TEAM_BATTING_HBP is comprised of almost 92% missing values. This variable cannot provide much information to our model and will be dropped.

TEAM_BASERUN_CS also displays a very high fraction of missing values: 34%. TEAM_PITCHING_SO and TEAM_BATTING_SO share the exact same number of missing values. This suggests that the missingness in these variables is not random.

```
both_missing <- is.na(raw$TEAM_PITCHING_SO) == is.na(raw$TEAM_BATTING_SO)
(sum(as.numeric(both_missing))/length(both_missing))
```

Further investigation reveals that the distribution of missingness in both these variables is identical. One option for responding to the missingness in these variables will be to code it as its own factor variable. Another option may be to drop one or both these variables from our models if there are further reasons to be suspicious about the quality of the data contained in them.

Having looked at missing values, and identifying points of concern that we will return to later in the report, we will now look at some characteristics of the predictor variables.

Examining the shape of each variable can help identify any unusual attributes including extreme skewness or low variance.

```
raw %>%
  gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free", ncol = 3) +
  geom_density() +
  ggthemes::theme_fivethirtyeight()
```

Density plots for each variable reveals the highly skewed shapes of TEAM_FIELDING_E, TEAM_PITCHING_BB, TEAM_PITCHING_H, and TEAM_PITCHING_SO as suggested by the numeric summary above. TEAM_BASERUN_SB and TEAM_BATTING_3B display moderate skewness. Other distributions appear roughly normal or bimodal.

Looking more closely at each of the highly skewed variables may suggest a reason for their non-normality.

```r
ggplot(data = raw, aes(x = TEAM_FIELDING_E)) +
  geom_histogram() +
  ggtitle("Errors Distribution") +
  theme_fivethirtyeight()
```

In TEAM_FIELDING_E, a very small number of entries exceed 1000. Excluding these, the distribution is moderately skewed right.

```r
ggplot(data = raw, aes(x = TEAM_PITCHING_BB)) +
  geom_histogram() +
  ggtitle("Walks Allowed Distribution") +
  theme_fivethirtyeight()
```

The distribution of TEAM_PITCHING_BB appears normal except for a very small number of entries greater than 1000.

```r
ggplot(data = raw, aes(x = TEAM_PITCHING_H)) +
  geom_histogram() +
  ggtitle("Hits Allowed Distribution") +
  theme_fivethirtyeight()
```

Let's look more closely at the region where most of the values in TEAM_PITCHING_H lie, [0,6000]:

```r
raw %>%
  filter(TEAM_PITCHING_H < 6000) %>%
  ggplot(aes(x = TEAM_PITCHING_H)) +
  geom_histogram() +
  ggtitle("Hit Allowed, Focused 0 - 6,000 Distribution") +
  theme_fivethirtyeight()
```

TEAM_PITCHING_H is skewed even within this narrower region surrounding the peak. The presence of skew and the wide separation of data points along the right tail may highlight some unique phenomenon, data entry errors, or structural changes, (such as rule changes, or changes in the equipment used) that influenced circumstances on the ground.

```r
ggplot(data = raw, aes(x = TEAM_PITCHING_SO)) +
  geom_histogram() +
  ggtitle("Strikeouts by Pitchers Distribution") +
  theme_fivethirtyeight()
```

Narrowing in on TEAM_PITCHING_SO:

```r
raw %>%
  filter(TEAM_PITCHING_SO < 2500) %>%
  ggplot(aes(x = TEAM_PITCHING_SO)) +
  geom_histogram() +
  ggtitle("Strikeouts by Pitchers, Focused 0-2,500 Distribution") +
  theme_fivethirtyeight()
```

The data appear roughly normal, but there are a significant number of unrealistic values. The record for strikeouts all time was set in 2018 by the Houston Astros at 1,687, and we can see from this distribution there are a significant amount of data above this mark. It is also worth noting the all time record for least strikeouts was set by the 1921 Cincinnati Reds at 308. Not only are there many values below this, but we have a significant amount of zero values.

```r
ggplot(data = raw, aes(x = TEAM_BASERUN_SB)) +
  geom_histogram() +
  ggtitle("Stolen Bases Distribution") +
  theme_fivethirtyeight()
```

The shape of `TEAM_BASERUN_SB` does not suggest any data entry errors, even though there exist a few extreme outliers.

```r
ggplot(data = raw, aes(x = TEAM_BATTING_3B)) +
  geom_histogram() +
  ggtitle("Tripples by Batters Distribution") +
  theme_fivethirtyeight()
```

The shape of `TEAM_BATTING_3B` does not suggest any data entry errors, even though there exist a few extreme outliers.

As demonstrated in our examination of `TEAM_PITCHING_SO` we are aware of outliers that lie beyond the historical maximum and minimum in these categories. These type of outliers are present in all of the variables that would lead us to conclude at least some of the data is erroneous. The best course of action under these conditions is to pause our investigation and return to the source of the data to discover why our data does not match historic baseball data sets. Because this is not possible, we'll use multiple imputation to replace extreme outliers and missing values.

This decision entails some risk, because replacing data with imputed values could impair the quality of our model. However in this case, it's our judgment that incorporating these extreme, unrealistic values would do greater damage to any models that are trained on them. In order to proceed with caution, we'll train models on two versions of the data–one with extreme values replaced by imputation, and one with extreme values unchanged.

One important condition of linear modeling is that independent variables should be uncorrelated with each other. In real-world data it is rarely possible to fully satisfy this condition. However, we can seek to avoid large pairwise correlations between variables.

The correlation plot below is arranged so that all variables with a theoretical positive effect on `TARGET_WINS` appear at the top and left, followed by variables with a theoretical negative effect. We would expect positive correlations among variables with positive effects, positive correlations among variables with negative effects, and negative correlations among variables with opposite effects.

```r
#reorder based on variable list on assignment sheet. Drop TEAM_BATTING_HBP
and TEAM_BASERUN_CS due to missingness.
tmp <- raw[,c(2:7,9,17,15,8,16,14,12,13)]
```

```r
correlation <- cor(tmp, use = "complete.obs")
corrplot.mixed(correlation, tl.col = 'black', tl.pos = 'lt', number.cex=
11/ncol(tmp))
```

But this is not what we see. Instead we see that no variable alone has correlation greater than 0.35 in either direction with the target. We also see that some pairs of variables have correlations that are so strong or misdirected that we have reason to doubt the integrity of the data. Note that `TEAM_PITCHING_HR` and `TEAM_BATTING_HR`, with correlation 0.98, contain essentially the same information. `TEAM_PITCHING_SO` and `TEAM_BATTING_SO`, the variables whose patterns of missingness are identical, also demonstrate an unreasonably high correlation of 0.95.

Another condition of linear models is that independent variables be linearly related to the target. We can examine this using scatterplots, one group of variables at a time.

Batting variables

```r
raw %>%
  gather(starts_with("TEAM_BAT"), key = "var", value = "value") %>%
  ggplot(aes(x = value, y = TARGET_WINS)) +
  geom_point() +
  facet_wrap(~ var, scales = "free", ncol = 2) +
  ggthemes::theme_fivethirtyeight()
```

Excluding `TEAM_BATTING_HBP`, the batting variables have potential for positive linear relationships. These will need to be investigated further.

Baserun and fielding variables

```r
raw %>%
  gather(c(starts_with("TEAM_BASERUN"), starts_with("TEAM_FIELD")), key =
"var", value = "value") %>%
  ggplot(aes(x = value, y = TARGET_WINS)) +
  geom_point() +
  facet_wrap(~ var, scales = "free") +
  ggthemes::theme_fivethirtyeight()
```

The situation with these variables is similar to that of the batting variables. There is potential for linearity, but we will have to investigate further.

Pitching variables

```r
raw %>%
  gather(starts_with("TEAM_PITCH"), key = "var", value = "value") %>%
  ggplot(aes(x = value, y = TARGET_WINS)) +
  geom_point() +
  facet_wrap(~ var, scales = "free") +
  ggthemes::theme_fivethirtyeight()
```

No linearity is evident in the scatterplots for the pitching variables. What is evident again is the presence of extreme outliers and skewness in three of the four pitching variables.

An unexpected finding in our exploratory data analysis is that a scatterplot of TEAM_BATTING_SO and TEAM_PITCHING_SO suggests that the data can be divided into 4 distinct groups, three of which contain highly correlated data.

```
raw %>%
  mutate(SO_factor = case_when(TEAM_BATTING_SO >= TEAM_PITCHING_SO*.96+10~
'high',
                               (TEAM_BATTING_SO<TEAM_PITCHING_SO*.96+10
                                & TEAM_BATTING_SO>TEAM_PITCHING_SO*.96-50)
~'med_high',
                               (TEAM_BATTING_SO<TEAM_PITCHING_SO*.96-50
                                & TEAM_BATTING_SO>TEAM_PITCHING_SO*.96-120)
~'med_low',
                               TEAM_BATTING_SO<TEAM_PITCHING_SO*.96-120
~'low')) %>%
  filter(TEAM_PITCHING_SO < 2000) %>%
  ggplot(aes(x = TEAM_PITCHING_SO, y = TEAM_BATTING_SO, colour = SO_factor))
+
  geom_point() +
  ggtitle("Strikeouts by Batters by Strikeouts by Pitchers") +
  theme_fivethirtyeight()
```

There is no theoretical reason to expect such a grouping, or to expect such high correlations between the number of strikeouts a team incurs while batting, and the number of strikeouts a team achieves while pitching. The fact that there is no “theoretical reason” would imply there is a problem with the data. Since we have already decided to press on with our analysis, these patterns could be treated as natural features of the data that we want to draw out in our model.

Data Preparation

To prepare the data for modeling, we first simplify variable names by removing the prefix TEAM_, and we add the grouping variable SO_FACTOR.

```
#Simplify column names
names(raw) <- gsub('TEAM_', '', x = names(raw))

#Add group variable, SO_FACTOR
raw <- raw %>% mutate(
  SO_FACTOR = case_when(
    BATTING_SO >= PITCHING_SO*.96+10 ~ 'high',
    (BATTING_SO<PITCHING_SO*.96+10 & BATTING_SO>PITCHING_SO*.96-50) ~
'med_high',
    (BATTING_SO<PITCHING_SO*.96-50 & BATTING_SO>PITCHING_SO*.96-120) ~
'med_low',
    BATTING_SO<PITCHING_SO*.96-120 ~ 'low'))
```

An understanding of baseball suggests how to combine some variables into other meaningful measures of team performance, such as _OBP. This is a rate statistic based on how many times you reach base on a hit or walk, vs how many times you came to the plate. Creating new variables as combinations of existing variables will help alleviate collinearity among some pairs of variables, and it will lead to models that better reflect commonly used measures of team performance.

We create variables to represent net stolen bases (`BASERUN_NET_SB`), offensive on-base percentage (`OFFENSE_OBP`), defensive on-base percentage (`DEFENSE_OBP`), and total at-bats (`TOT_AT_BATS`). Then we drop the variables that were combined to produce these new variables.

```
raw <- raw %>%
  mutate("BASERUN_NET_SB" = BASERUN_SB - BASERUN_CS) %>%
  mutate("OFFENSE_OBP" = (BATTING_H + BATTING_BB)/(BATTING_H + BATTING_BB -
BASERUN_CS + (162*27))) %>%
  mutate("DEFENSE_OBP" = (PITCHING_H + FIELDING_E + PITCHING_BB -
FIELDING_DP)/(PITCHING_H + FIELDING_E + PITCHING_BB - FIELDING_DP +
(162*27))) %>%
  mutate("TOT_AT_BATS" = BATTING_H + BATTING_BB - BASERUN_CS + (162*27))
raw <- raw[,c(1,4:6,8,13,15,18:22,2)]
```

After these initial transformations, we split the data into a training set (80% = 1820 rows) and a test set (20% = 456 rows).

```
train_rows <- sample(nrow(raw), 0.80 * nrow(raw), replace = FALSE)
train <- raw[train_rows,]
test <- raw[-train_rows,]
```

During our exploration of the data, we noted several variables with extreme outliers and described our strategy of examining two versions of our models: one trained on data with only missing values imputed, and one trained on data with missing values and outliers imputed. We'll consider all these models before making our final selection.

Replacing missing data within a given variable with it's column mean or median will introduce bias in the data. This results from ignoring the influence of other variables on these values as well as naturally occurring variability in the data. To reduce the potential for such bias, we employed a coupled regression/bootstrapping method (via the MICE package) to impute our missing data. Our diagnostic checks (not included here) indicated that this method produced better results (distributions comparable to the actual data) relative to other available methods (e.g., pmm, random forest, etc.).

```
#training set which only missing values will be imputed
train_imp_M <- train

#training set in which missing values and outliers will be imputed
train_imp_OM <- train

#A function to set the upper and lowerbounds of a dataframe vector
bounds <- function(vector,upper_pct,lwr_pct){
  ub <- quantile(vector, upper_pct,na.rm = T)
  lb <- quantile(vector, lwr_pct, na.rm = T)
  return(c(ub,lb))
}

#Setting the upper and lower percentile bounds:
up_l <- 0.97
lo_l <- 0.03

#Impute missing data
imp <- mice(train_imp_M, method = "norm.boot", m = 5) #Alternative: norm.nob,
m = 1
train_imp_M <- complete(imp)

train_imp_M$SO_FACTOR[is.na(train_imp_M$SO_FACTOR)] <- "high"

#Impute missing and outliers
colnames <- c('BATTING_2B', 'BATTING_3B', 'BATTING_HR', 'BATTING_SO',
              'PITCHING_HR', 'PITCHING_SO', 'BASERUN_NET_SB', 'OFFENSE_OBP',
              'DEFENSE_OBP', 'TOT_AT_BATS')

for(col in colnames){
  upper = bounds(train_imp_OM[,col],up_l,lo_l)[1]
  lower = bounds(train_imp_OM[,col],up_l,lo_l)[2]

  train_imp_OM[,col] = ifelse(
    train_imp_OM[,col] < lower | train_imp_OM[,col] > upper,
    NA, train_imp_OM[,col])
}

#Impute to replace all previously missing values along with removed outliers
imp <- mice(train_imp_OM, method = "norm.boot", m = 5) #Alternative:
norm.nob, m = 1
train_imp_OM <- complete(imp)
train_imp_OM$SO_FACTOR[is.na(train_imp_OM$SO_FACTOR)] <- "high"

#Test set in which only missing values will be imputed
test_imp_M <- test
imp <- mice(test_imp_M, method = "norm.boot", m = 5)
test_imp_M <- complete(imp)
test_imp_M$SO_FACTOR[is.na(test_imp_M$SO_FACTOR)] <- "high"

#Test set in which missing values and outliers will be imputed
test_imp_OM <- test

for(col in colnames){
  upper = bounds(test_imp_OM[,col],up_l,lo_l)[1]
  lower = bounds(test_imp_OM[,col],up_l,lo_l)[2]

  test_imp_OM[,col] = ifelse(
    test_imp_OM[,col] < lower | test_imp_OM[,col] > upper,
    NA, test_imp_OM[,col])
}

# Impute to replace all previously missing values along with removed outliers
imp <- mice(test_imp_OM, method = "norm.boot", m = 5)
test_imp_OM <- complete(imp)
test_imp_OM$SO_FACTOR[is.na(test_imp_OM$SO_FACTOR)] <- "high"
```

A look at the summaries of our test sets, both with and without outliers imputed, shows the raw materials for our models.

With missing values imputed

```
train_imp_M %>%
  skim_without_charts() %>%
  yank("numeric")
```

With missing values and outliers imputed

```
train_imp_OM %>%
  skim_without_charts() %>%
  yank("numeric")
```

Build Models

In this section we construct models first for training sets with only missing values imputed, and we evaluate them with a test set with only missing values imputed. Then, we train and test analogous models with outliers imputed as well.

```
#A function to compute RMSE for test sets
rmse <- function(lm, test) {
  preds <- predict(lm, test[,c(2:12)])
  errors <- test$TARGET_WINS - preds
  return(sqrt(sum(errors^2)/(nrow(test) - length(lm$coefficients) - 1)))
}

#A function for generating residuals plots
res_plot <- function(lm){
  plot(fitted(lm),
       residuals(lm),
       xlab = "Fitted",
       ylab = "Residuals")
}
```

Model 1. Almost all variables

This model contains all variables except `BATTING_HR`, `BATTING_SO`, and `TOT_AT_BATS`, because each of these variables is highly correlated with at least one other variable.

With only missing values imputed

```
#Missing imputed
lm1_M <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR + PITCHING_SO
+ SO_FACTOR + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data = train_imp_M)

stargazer(lm1_M,type='html',out='lm1_M.html',title='Model 1 - with only
missing values imputed',
          covariate.labels = c('Doubles','Triples','Pitching HRs','Pitching
SO','Low SO',
                               'Med/High SO','Med/Low SO','Baserunners Net
SB','Offense OBP',
                               'Opponents OBP'),
          dep.var.caption = 'Target Wins',dep.var.labels.include=F,
          single.row=T, no.space = T, header=F,
          digits = 3, model.numbers = F)

res_plot(lm1_M)
qqnorm(residuals(lm1_M), ylab = "Residuals")
qqline(residuals(lm1_M))
```

With missing values and outliers imputed

```
#Missing and outliers imputed
lm1_OM <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + SO_FACTOR + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data =
train_imp_OM)

stargazer(lm1_OM,type='html',out='lm1_OM.html',
          title='Model 1 - missing values and outliers imputed',
          covariate.labels = c('Doubles','Triples','Pitching HRs','Pitching
SO','Low SO',
                               'Med/High SO','Med/Low SO','Baserunners Net
SB','Offense OBP',
                               'Opponents OBP'),
          dep.var.caption = 'Target Wins',dep.var.labels.include=F,
          single.row=T, no.space = T, header=F,
          digits = 3, model.numbers = F)

# summary(lm1_OM)
res_plot(lm1_OM)
qqnorm(residuals(lm1_OM), ylab = "Residuals")
qqline(residuals(lm1_OM))
```

Evaluation and discussion

The root mean square error for each model is reported below.

```
lm1_M_test_rmse <- rmse(lm1_M,test_imp_M)
lm1_OM_test_rmse <- rmse(lm1_OM,test_imp_OM)
```

Test set RMSE for model 1 with only missing imputed: r round(lm1_M_test_rmse,2)
Test set RMSE for model 1 with missing and outliers imputed: r round(lm1_OM_test_rmse,2)

Our initial look at the residuals for the model with only missing values imputed shows some heteroskedasticity, violating one of our modeling conditions. It appears the variance of the residuals decreases at extreme values. We also notice from the QQ plot that the residuals depart from a normal distribution at the extremes.

Plots for the model trained on the imputed outliers show improvement on the issues of the previous model. This could be a hint that the extreme outliers that have been removed were affecting our model.

RMSE values for these models on the test set are similar to the RMSE values for the training sets. This means that these models avoid overfitting the data.

Model 2. Piecewise by FACTOR_SO

The scatterplot of `BATTING_SO` vs. `PITCHING_SO` suggested four groups in this data. Here we fit one model to each group. Since we conduct a piecewise analysis on two separate training sets, 8 sets of outputs are generated for models in this section. Rather than display this model output, we summarize our findings.

```r
trainlow_M <- train_imp_M %>%
  filter(SO_FACTOR == "low") %>%
  dplyr::select(-SO_FACTOR)

trainmed_low_M <- train_imp_M %>%
  filter(SO_FACTOR == "med_low") %>%
  dplyr::select(-SO_FACTOR)

trainmed_high_M <- train_imp_M %>%
  filter(SO_FACTOR == "med_high") %>%
  dplyr::select(-SO_FACTOR)

trainhigh_M <- train_imp_M %>%
  filter(SO_FACTOR == "high") %>%
  dplyr::select(-SO_FACTOR)

trainlow_OM <- train_imp_OM %>%
  filter(SO_FACTOR == "low") %>%
  dplyr::select(-SO_FACTOR)

trainmed_low_OM <- train_imp_OM %>%
  filter(SO_FACTOR == "med_low") %>%
  dplyr::select(-SO_FACTOR)

trainmed_high_OM <- train_imp_OM %>%
  filter(SO_FACTOR == "med_high") %>%
  dplyr::select(-SO_FACTOR)

trainhigh_OM <- train_imp_OM %>%
  filter(SO_FACTOR == "high") %>%
  dplyr::select(-SO_FACTOR)

testlow_M <- test_imp_M %>%
  filter(SO_FACTOR == "low") %>%
  dplyr::select(-SO_FACTOR)

testmed_low_M <- test_imp_M %>%
  filter(SO_FACTOR == "med_low") %>%
  dplyr::select(-SO_FACTOR)

testmed_high_M <- test_imp_M %>%
  filter(SO_FACTOR == "med_high") %>%
  dplyr::select(-SO_FACTOR)

testhigh_M <- test_imp_M %>%
  filter(SO_FACTOR == "high") %>%
  dplyr::select(-SO_FACTOR)

testlow_OM <- test_imp_OM %>%
  filter(SO_FACTOR == "low") %>%
  dplyr::select(-SO_FACTOR)

testmed_low_OM <- test_imp_OM %>%
  filter(SO_FACTOR == "med_low") %>%
  dplyr::select(-SO_FACTOR)

testmed_high_OM <- test_imp_OM %>%
  filter(SO_FACTOR == "med_high") %>%
  dplyr::select(-SO_FACTOR)

testhigh_OM <- test_imp_OM %>%
  filter(SO_FACTOR == "high") %>%
  dplyr::select(-SO_FACTOR)

#IMPUTING MISSING VALUES ONLY

lm_low_M <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data = trainlow_M)
lm_med_low_M <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data =
trainmed_low_M)
lm_med_high_M <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data =
trainmed_high_M)
lm_high_M <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data = trainhigh_M)

summary(lm_low_M)
summary(lm_med_low_M)
```

```r
summary(lm_med_high_M)
summary(lm_high_M)

plot(lm_low_M)
plot(lm_med_low_M)
plot(lm_med_high_M)
plot(lm_high_M)

#IMPUTING MISSING VALUES AND OUTLIERS
lm_low_OM <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data = trainlow_OM)
lm_med_low_OM <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data =
trainmed_low_OM)
lm_med_high_OM <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data =
trainmed_high_OM)
lm_high_OM <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + BASERUN_NET_SB + OFFENSE_OBP + DEFENSE_OBP, data =
trainhigh_OM)

summary(lm_low_OM)
summary(lm_med_low_OM)
summary(lm_med_high_OM)
summary(lm_high_OM)

plot(lm_low_OM)
plot(lm_med_low_OM)
plot(lm_med_high_OM)
plot(lm_high_OM)
```

Evaluation and discussion

The root mean square error for each model is reported below.

```r
#RMSE on training set for piecewise model with missing values imputed:
trn_rmse_low_M <- rmse(lm_low_M,trainlow_M)
trn_rmse_med_low_M <- rmse(lm_med_low_M,trainmed_low_M)
trn_rmse_med_high_M <- rmse(lm_med_high_M,trainmed_high_M)
trn_rmse_high_M <- rmse(lm_high_M,trainhigh_M)

trn_rmse_total_M <- (trn_rmse_low_M * nrow(trainlow_M) + trn_rmse_med_low_M *
nrow(trainmed_low_M) + trn_rmse_med_high_M * nrow(trainmed_high_M) +
trn_rmse_high_M * nrow(trainhigh_M)) / nrow(train_imp_M)

#RMSE on test set for piecewise model with missing values imputed:
test_rmse_low_M <- rmse(lm_low_M,testlow_M)
test_rmse_med_low_M <- rmse(lm_med_low_M,testmed_low_M)
test_rmse_med_high_M <- rmse(lm_med_high_M,testmed_high_M)
test_rmse_high_M <- rmse(lm_high_M,testhigh_M)
```

```
tst_rmse_total_M <- (test_rmse_low_M * nrow(testlow_M) + test_rmse_med_low_M
* nrow(testmed_low_M) + test_rmse_med_high_M * nrow(testmed_high_M) +
test_rmse_high_M * nrow(testhigh_M)) / nrow(test_imp_M)

#RMSE on training set for piecewise model with missing values and outliers
imputed:
trn_rmse_low_OM <- rmse(lm_low_OM,trainlow_OM)
trn_rmse_med_low_OM <- rmse(lm_med_low_OM,trainmed_low_OM)
trn_rmse_med_high_OM <- rmse(lm_med_high_OM,trainmed_high_OM)
trn_rmse_high_OM <- rmse(lm_high_OM,trainhigh_OM)

trn_rmse_total_OM <- (trn_rmse_low_OM * nrow(trainlow_OM) +
trn_rmse_med_low_OM * nrow(trainmed_low_OM) + trn_rmse_med_high_OM *
nrow(trainmed_high_OM) + trn_rmse_high_OM * nrow(trainhigh_OM)) /
nrow(train_imp_OM)

#RMSE on test set for piecewise model with missing values and outliers
imputed:
rmse_low_OM <- rmse(lm_low_OM,testlow_OM)
rmse_med_low_OM <- rmse(lm_med_low_OM,testmed_low_OM)
rmse_med_high_OM <- rmse(lm_med_high_OM,testmed_high_OM)
rmse_high_OM <- rmse(lm_high_OM,testhigh_OM)

tst_rmse_total_OM <- (rmse_low_OM * nrow(testlow_OM) + rmse_med_low_OM *
nrow(testmed_low_OM) + rmse_med_high_OM * nrow(testmed_high_OM) +
rmse_high_OM * nrow(testhigh_OM)) / nrow(test_imp_OM)
```

RMSE for piecewise models with missing values imputed
RMSE for training set: r round(trn_rmse_total_M, 2) RMSE for test set: r round(tst_rmse_total_M, 2)

RMSE for piecewise models with missing values and outliers imputed
RMSE for training set: r round(trn_rmse_total_OM, 2) RMSE for test set: r round(tst_rmse_total_OM, 2)

This piecewise modeling approach returned RMSE values similar to those of our previous model. Residual and QQ plots for component models where only missing values were imputed demonstrate the same heteroskedasticity and deviation from normality that we saw in Model 1. Again, these issues are somewhat mitigated in the model where outliers are also imputed.

This second model was prompted by the results of our `BATTING_SO` vs. `PITCHING_SO` plot. We acknowledge some issues with this approach– one being that the models differ greatly in population size, so a visual comparison may not be trustworthy. Also, we may not have the appropriate context to make the decision to divide the data based on the relationship between these two predictors. While this might have led to overfitting, RMSE values that are similar in both the training and test sets suggest that we avoided this.

Model 3. Drop DEFENSE_OBP

Here we drop DEFENSE_OBP from Model 1. Even though this variable is a statistically significant predictor of TARGET_WINS, its large coefficient and standard error compared to the other variables might be due to its high correlation with OFFENSE_OBP.

With only missing values imputed

```r
lm3_M <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR + PITCHING_SO
+ SO_FACTOR + BASERUN_NET_SB + OFFENSE_OBP, data = train_imp_M)
summary(lm3_M)

stargazer(lm3_M,type='html',out='lm3_M.html',title='Model 3 - only missing
values imputed',
          covariate.labels = c('Doubles','Triples','Pitching HRs','Pitching
SO','Low SO',
                               'Med/High SO','Med/Low SO','Baserunners Net
SB','Offense OBP'),
          dep.var.caption = 'Target Wins',dep.var.labels.include=F,
          single.row=T, no.space = T, header=F,
          digits = 3, model.numbers = F)

res_plot(lm3_M)
qqnorm(residuals(lm3_M), ylab = "Residuals")
qqline(residuals(lm3_M))
```

With missing values and outliers imputed

```r
lm3_OM <- lm(TARGET_WINS ~ BATTING_2B + BATTING_3B + PITCHING_HR +
PITCHING_SO + SO_FACTOR + BASERUN_NET_SB + OFFENSE_OBP, data = train_imp_OM)
summary(lm3_OM)

stargazer(lm3_OM,type='html',out='lm3_OM.html',
          title='Model 3 - missing values and outliers imputed',
          covariate.labels = c('Doubles','Triples','Pitching HRs','Pitching
SO','Low SO',
                               'Med/High SO','Med/Low SO','Baserunners Net
SB','Offense OBP'),
          dep.var.caption = 'Target Wins',dep.var.labels.include=F,
          single.row=T, no.space = T, header=F,
          digits = 3, model.numbers = F)

res_plot(lm3_OM)
qqnorm(residuals(lm3_OM), ylab = "Residuals")
qqline(residuals(lm3_OM))
```

Evaluation and discussion

The root mean square error for each model is reported below.

```r
lm3_M_test_rmse <- rmse(lm3_M,test_imp_M)
lm3_OM_test_rmse <- rmse(lm3_OM,test_imp_OM)
```

RMSE for piecewise models with missing values imputed
RMSE on training set: r round(trn_rmse_total_M,2)
RMSE on test set: r round(tst_rmse_total_M,2)

RMSE for piecewise models with missing values and outliers imputed
RMSE on training set: r round(trn_rmse_total_OM,2)
RMSE on test set: r round(tst_rmse_total_OM,2)

In this model, we see residuals behaving similarly to the first model. We can call attention again to the improvement in the residuals by the training of the data with the imputed outliers. Not only is the shape of the residuals more homoskedastic and normal, but the statistical significance of the coefficients is much greater.

Conclusion and model selection

Before we select the best performing model, let's review the above analysis.

Data Exploration. We recognize a high probability of flawed data. Unrelated variables were found to have near perfect correlation. Outliers represent impossible measurements. These occurrences call for additional scrutiny toward the less egregious features of the data, such as bimodal distributions. In a practical setting, this would need to be addressed before moving forward.

Data Preparation. Pushing forward with the suspicious data, we proceeded cautiously by creating a factor that represented particular cohorts within the data. We created new variables based on our knowledge of baseball.

Modeling. In the first model, we incorporated all the remaining features of the data. In the second model, we trained separate models for each level of `SO_FACTOR`, and combined these models into an overall piecewise linear model. In the third model, we improved on the first model by dropping one of a pair of highly correlated variables.

Below, we restate the RMSE for each model constructed.

Models with only missing values imputed
RMSE for model 1, missing values imputed: r round(lm1_M_test_rmse,2)
RMSE for model 2, missing values imputed: r round(tst_rmse_total_M,2)
RMSE for model 3, missing values imputed: r round(lm3_M_test_rmse,2)

Models with missing vlues and outliers imputed
RMSE for model 1, missing values and outliers imputed: r round(lm1_OM_test_rmse,2)
RMSE for model 2, missing values and outliers imputed: r round(tst_rmse_total_OM,2)
RMSE for model 3, missing values and outliers imputed: r round(lm3_OM_test_rmse,2)

The best model is Model 3 with missing values and outliers imputed. It uses the full training set in a single model, unlike Model 2. Its predictors and the intercept are highly significant, which is not true for every variable in Model 1. We believe the factors added to our model serve as an effective attempt to counteract the flaws in this data set, and the engineered features made the model more efficient and improved performance. The strategy of imputing outliers both decreased RMSE and improved our residual plots.

```
#Simplify column names
names(eval) <- gsub('TEAM_', '', x = names(eval))

#Add group variable, SO_FACTOR
eval <- eval %>% mutate(
  SO_FACTOR = case_when(
    BATTING_SO >= PITCHING_SO*.96+10 ~ 'high',
    (BATTING_SO<PITCHING_SO*.96+10 & BATTING_SO>PITCHING_SO*.96-50) ~
'med_high',
    (BATTING_SO<PITCHING_SO*.96-50 & BATTING_SO>PITCHING_SO*.96-120) ~
'med_low',
    BATTING_SO<PITCHING_SO*.96-120 ~ 'low'))

eval <- eval %>%
  mutate("BASERUN_NET_SB" = BASERUN_SB - BASERUN_CS) %>%
  mutate("OFFENSE_OBP" = (BATTING_H + BATTING_BB)/(BATTING_H + BATTING_BB -
BASERUN_CS + (162*27))) %>%
  mutate("DEFENSE_OBP" = (PITCHING_H + FIELDING_E + PITCHING_BB -
FIELDING_DP)/(PITCHING_H + FIELDING_E + PITCHING_BB - FIELDING_DP +
(162*27))) %>%
  mutate("TOT_AT_BATS" = BATTING_H + BATTING_BB - BASERUN_CS + (162*27))

eval <- eval[,c(1,3:5,7,12,14,17:21)]

for(col in colnames){
  upper = bounds(eval[,col],up_l,lo_l)[1]
  lower = bounds(eval[,col],up_l,lo_l)[2]

  eval[,col] = ifelse(
    eval[,col] < lower | eval[,col] > upper,
    NA, eval[,col])
}

#Impute to replace all previously missing values along with removed outliers
imp <- mice(eval, method = "norm.boot", m = 5)
eval <- complete(imp)
eval$SO_FACTOR[is.na(eval$SO_FACTOR)] <- "high"

pred_TARGET_WINS <- predict(lm3_OM, eval)
eval_final <-
read.csv("https://raw.githubusercontent.com/dmoscoe/DATA621/main/data/moneyba
ll-evaluation-data.csv")
eval_final$pred_TARGET_WINS <- pred_TARGET_WINS
write.csv(eval_final, "C:/Users/dmosc/OneDrive/Documents/academic/CUNY
SPS/DATA621/HW1/group1preds_final.csv", row.names = FALSE)
```