DATA 621—Business Analytics and Data Mining

Fall 2020—Group 2—Homework #1

Avraham Adler, Samantha Deokinanan, Amber Ferger, John Kellogg, Bryan Persaud, Jeff Shamp

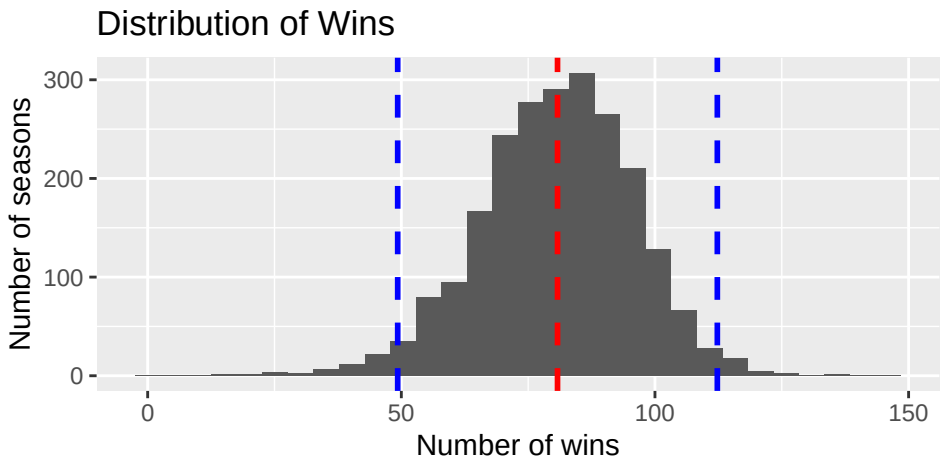
9/27/2020

Contents

DATA EXPLORATION

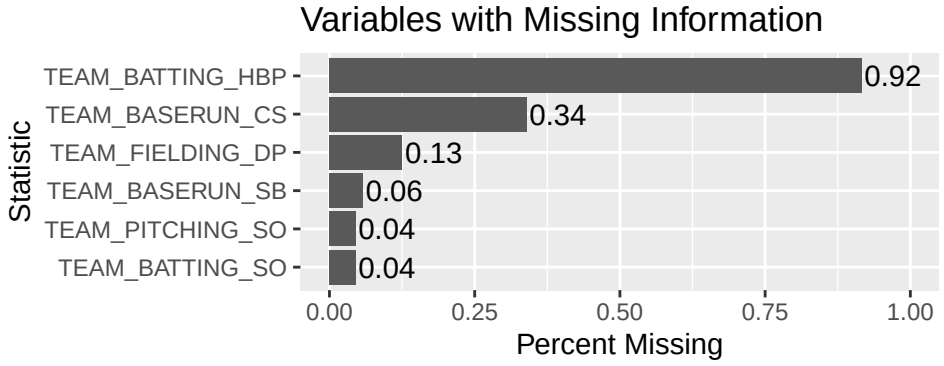
How Often Does The Team Win?

The data set provided comprises 2,276 records of baseball teams seasons. There are 15 features, statistics corresponding to a team's performance in a season, and a target variable of the team's number of wins in that season. A baseball season contains 162 games. On average, about 50% of games played are won—81 games out of 162—with the best individual season having 146 wins and the worst season having 0 wins. The data is near normally distributed and most years have between 49 and 112 wins, represented by the blue lines in the histogram below. The nature of the distribution indicates that there aren't too many extreme seasons where wins are significantly higher or lower than usual. This serves as a good gut-check for our final predictions; if the predicted wins are too high or too low, we know something in our model is probably off.



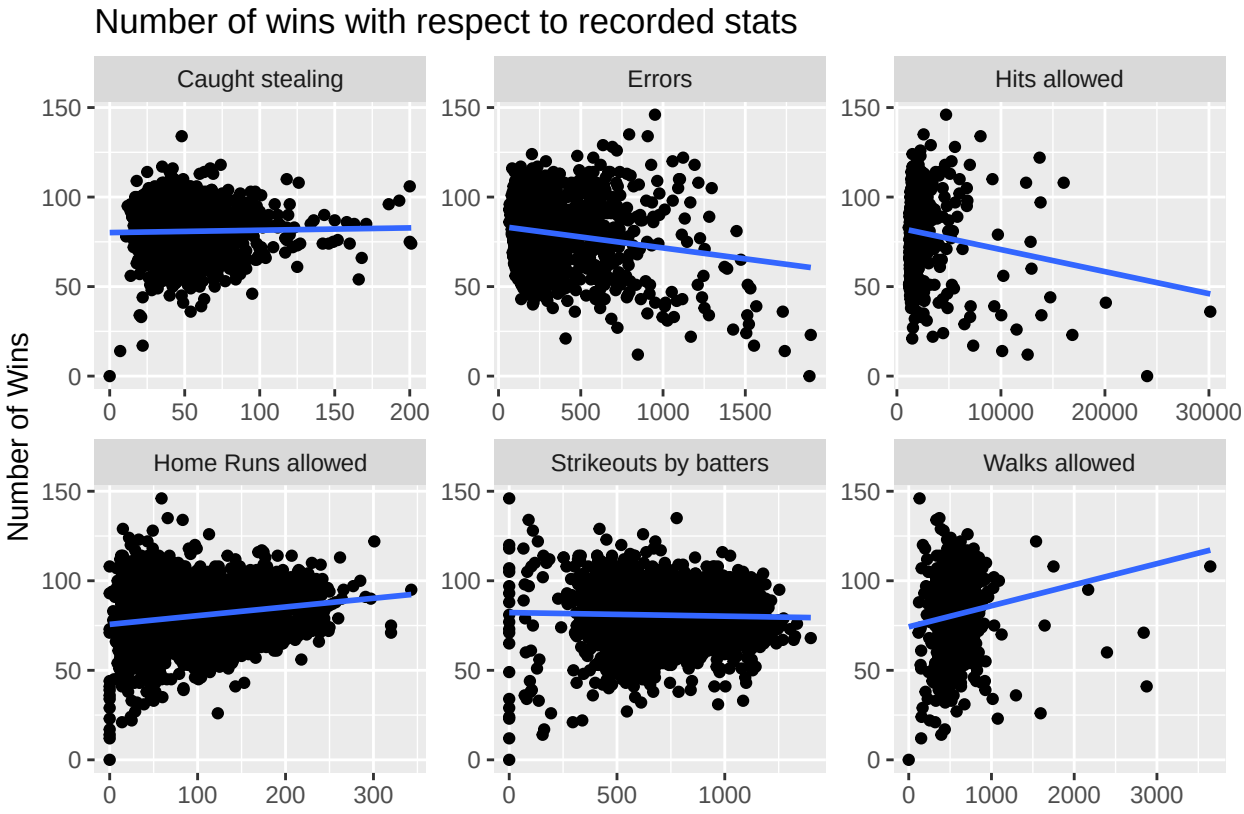
What's Missing?

A first look at the data shows that only about 8% of the records have a full set of information. The good news is that most of the missing values come from statistics that don't happen too often: hit-by-pitch (`TEAM_BATTING_HBP`, 92% missing!), caught stealing (`TEAM_BASERUN_CS`, 34% missing), and double plays (`TEAM_FIELDING_DP`, 13% missing). Since we have so little hit-by-pitch data, we expect that it doesn't contribute much to overall wins and will eliminate it from a few of the models we propose. The other two stats have less than half of the data missing, so we'll need to think of a clever way to fill in these values. The remaining missing information is from a combination of stolen bases and strikeouts by pitchers and batters (`TEAM_BASERUN_SB`, `TEAM_PITCHING_SO`, `TEAM_BATTING_SO`). It seems **completely unreasonable** to have zero strike outs in a season, so this is something we'll most certainly have to impute.



Do the Individual Stats Affect Winning?

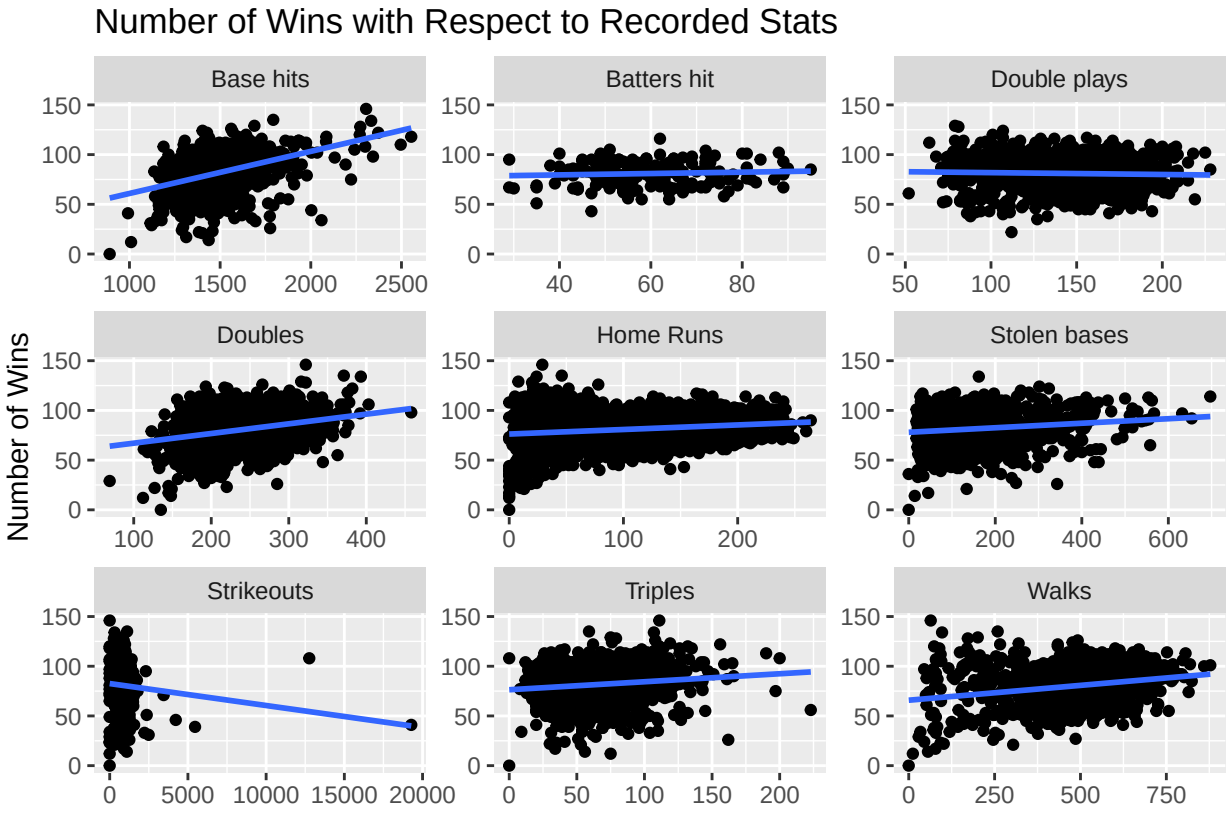
Stats with an expected negative impact: Intuitively, we expect that Caught Stealing, Errors, Hits Allowed, Home Runs Allowed, Strikeouts by Batters, and Walks Allowed would all have a **negative** impact on the total wins. In other words, as these values increase, we expect that the team is less likely to win.



When we take a closer look at the data, these negative relationships aren't obvious. In fact, only **Errors** and **Hits Allowed** seem to have a negative impact on wins. **Caught Stealing** and **Strikeouts by Batters** appear to be random; this means that whether the stat for a particular season is high or low doesn't affect the overall number of wins.

Even more interestingly, **Home Runs Allowed** and **Walks Allowed** have the *opposite* effect—as these stats increase, so do the number of wins!

Stats with an expected positive impact: We can look at the same information for the stats that we expect to have a **positive** effect on wins: Base Hits, Doubles, Triples, Home Runs, Walks, Batters getting hit by pitches, Stolen Bases, Double Plays, and Strikeouts by Pitchers.

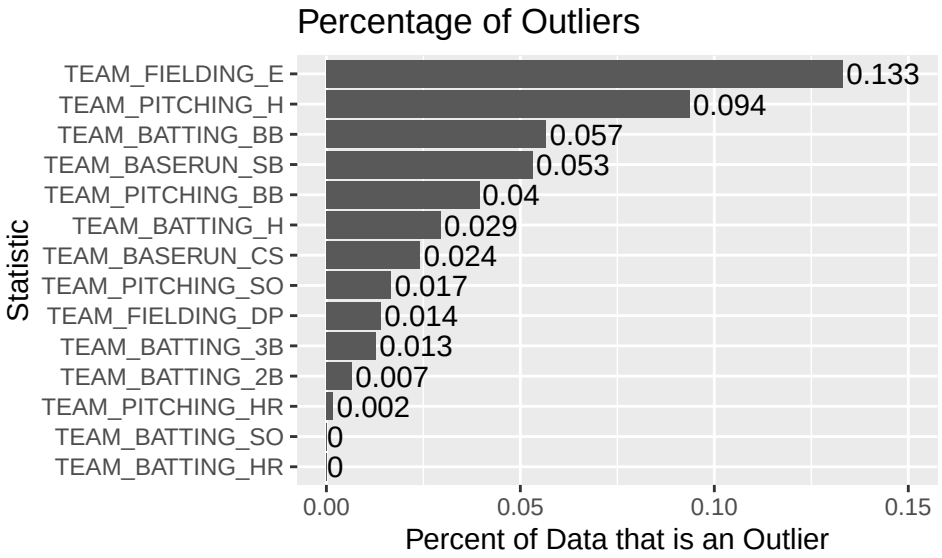


Many of these stats *do* seem to have an effect on the number of wins; most notably, **Base Hits** and **Walks**. We see weaker positive relationships for **Home Runs**, **Doubles**, **Triples**, and **Stolen Bases**. On the surface, this makes sense as these events tend to happen less often in games than base hits and walks, so they don't have as much of an effect on winning. Finally, **Double Plays** and **Batters Hit** don't appear to have any correlation with the number of wins. Once again, this intuitively makes sense because they are less likely to happen in a game.

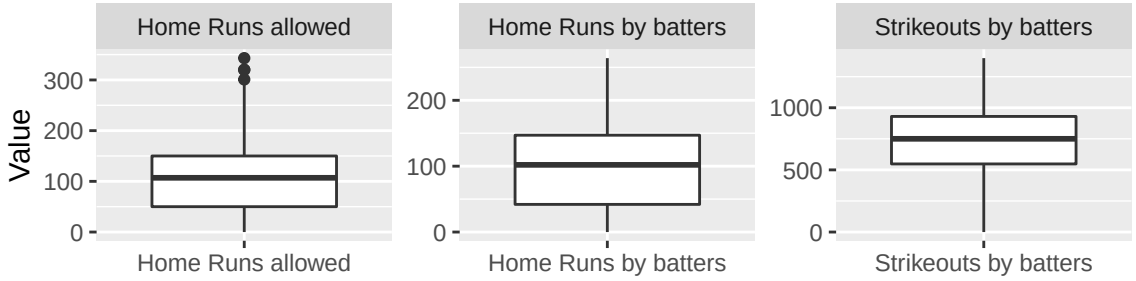
One thing to note is the number of strikeouts compared to the number of wins. We can see that there are a few outliers—abnormally high numbers of strikeouts in a season. This should be taken with caution, as they don't represent a typical season's stats.

Are Some Stats More Skewed Than Others?

Before using any of the statistics in a model, we need to take a closer look at the variation in the data. We call out-of-the-ordinary values, exceptionally high or low values, **outliers**. We need to take these into account in our modeling to prevent them from skewing our predictions.



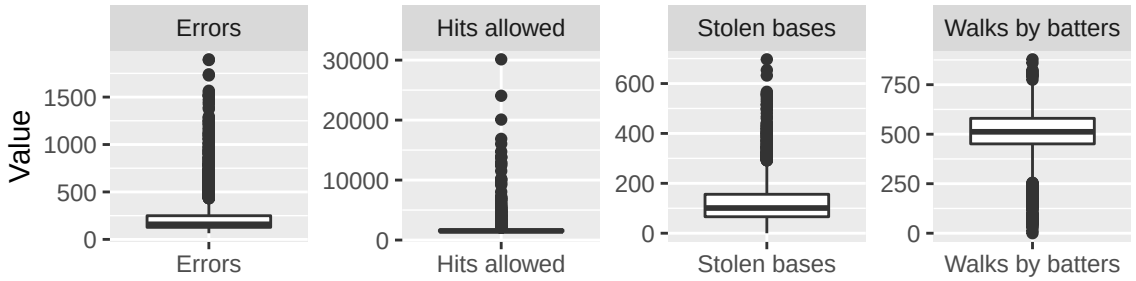
As we can see, some of the provided statistics are well-balanced in the sense that there are very *few*, or no, extreme values. **Home Runs allowed** (`TEAM_PITCHING_HR`), **Strikeouts by batters** (`TEAM_BATTING_SO`), and **Home Runs by batters** (`TEAM_BATTING_HR`) are some examples.



Some things to note about each of these statistics:

- **Home Runs Allowed** (average ~100/year) and **Home Runs by Batters** (average ~106/year) have a very similar mid-range distribution—50% of the data lies between ~50 and 150. The slight difference in average stats means that teams tend to have a higher number of Home Runs than the opposition team.
- The only thing that stands out about **Strikeouts by Batters** (average ~736/year) is how nearly perfectly normal it is. 50% of the data is between about 500 and 1000 and there are absolutely no outliers in the dataset! This means that there were no surprisingly high or low seasons.

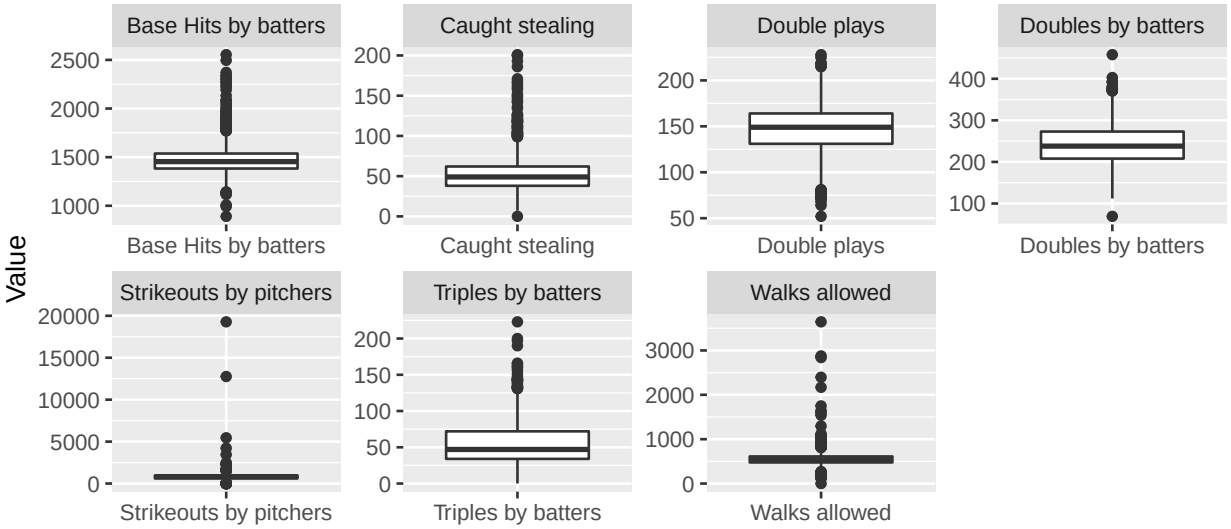
Conversely, some of the stats have a very *high* number of outliers, indicating that there are some seasons with some abnormally high or low values. **Errors** (`TEAM_FIELDING_E`), **Hits allowed** (`TEAM_PITCHING_H`), **Walks by batters** (`TEAM_BATTING_BB`), and **Stolen bases** (`TEAM_BASERUN_SB`) are some examples.



Some things to note about each of these statistics:

- All of the outliers for **Errors**, **Hits Allowed**, and **Stolen Bases** are above the upper tail of the data set. This is further illustrated by the mean and median values for these stats; in all instances, the means per year (Errors ~246/year, Hits Allowed ~1779/year, Stolen Bases ~125/year) are higher than the medians per year (Errors ~159/year, Hits Allowed ~1518/year, Stolen Bases ~101/year). This means that some seasons with exceptionally high values skew the dataset.
- There are a few *very* extreme outliers for **Hits Allowed**. The maximum value is 30,132, which is over 16 times the average number of hits allowed per season!
- There are outliers both above *and* below the tails of the data for the **Walks by Batters** stat. This means that we have exceptionally low (min = 0!) and exceptionally high (max = 878) seasons.

The remaining stats, **Walks Allowed** (`TEAM_PITCHING_BB`), **Base Hits by Batters** (`TEAM_BATTING_H`), **Caught Stealing** (`TEAM_BASERUN_CS`), **Strikeouts by Pitchers** (`TEAM_PITCHING_SO`), **Double Plays** (`TEAM_FIELDING_DP`), **Triples by Batters** (`TEAM_BATTING_3B`), and **Doubles by Batters** (`TEAM_BATTING_2B`) have between 29 and 99 outliers.



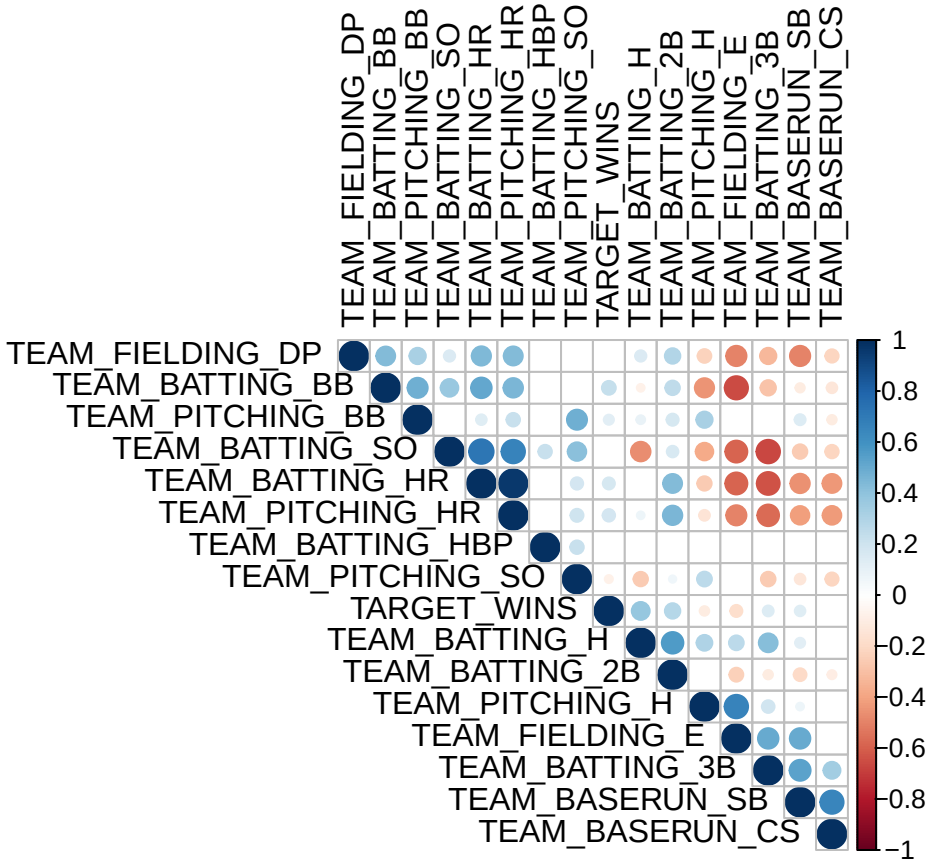
Some things to note:

- All variables are very narrowly distributed, meaning that most of the data falls within a small range.
- **Strikeouts by Pitchers** and **Walks Allowed** have a few very extreme outliers; these represent seasons that have abnormally high values for the statistics.

- The average number of pure **Base Hits** (1469/season) is greater than the average number of Doubles (~241/season) and Triples (~55/season). This isn't at all surprising, but serves as a good gut check on the validity of the data.

Are Stats Correlated?

We would expect that a few things in the dataset might be correlated. Perhaps number of errors and hits/homeruns allowed or the number of base hits by batters and doubles/triples/homeruns. We can visualize the correlations between the statistics to determine if there is a significant relationship between the them. In the graph below, blue dots represent positively correlated variables and red dots represent negatively correlated variables.



Some noteworthy relationships (coincidental or not):

- **Errors** are highly, negatively correlated with walks by batters, strikeouts by batters, and homeruns (both by batters and allowed).
- **Triples by Batters** are highly, negatively correlated with strikeouts by batters and homeruns (both by batters and allowed).
- **Strikeouts by Batters** are highly, positively correlated with homeruns (both by batters and allowed).
- **Homeruns by Batters** and **Homeruns Allowed** are both positively correlated with walks by batters.
- As expected, **Base Hits** are positively correlated with doubles and triples.

We can keep these correlations in mind when developing our models: if we have correlated statistics, there could be in-built redundancy in the features, and we may be able to create a simpler, more accurate model by eliminating some.

DATA PREPARATION

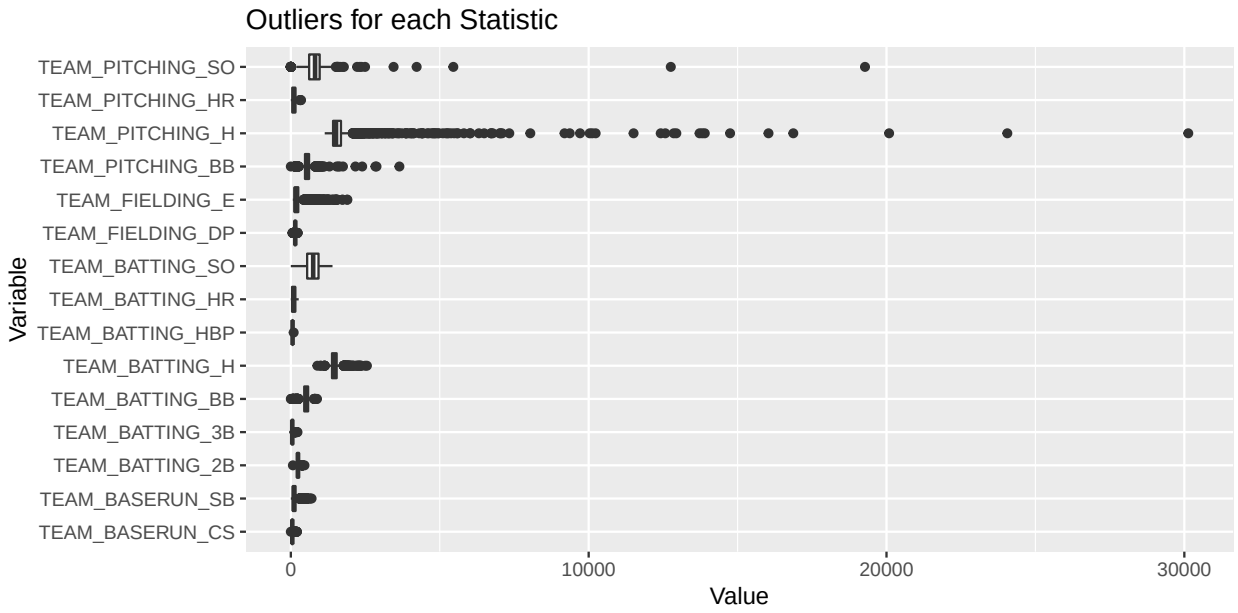
Now we have a good idea of the data we are looking at we can take the next steps to prepare it for building a solid model.

Outlier Removal

As we saw in the data analysis, there are some outlier concerns for some of the variables, so we will need to account for this in our modeling. When performing the processes of outlier removal, a cautious approach is always best. Each outlier is evaluated to ensure:

- It is clearly from incorrectly or mis-centered data.
- Its removal does not affect later assumptions.
- It creates a significant association/relation that is eliminated with its removal.

We can take another look at the outliers for each variable:

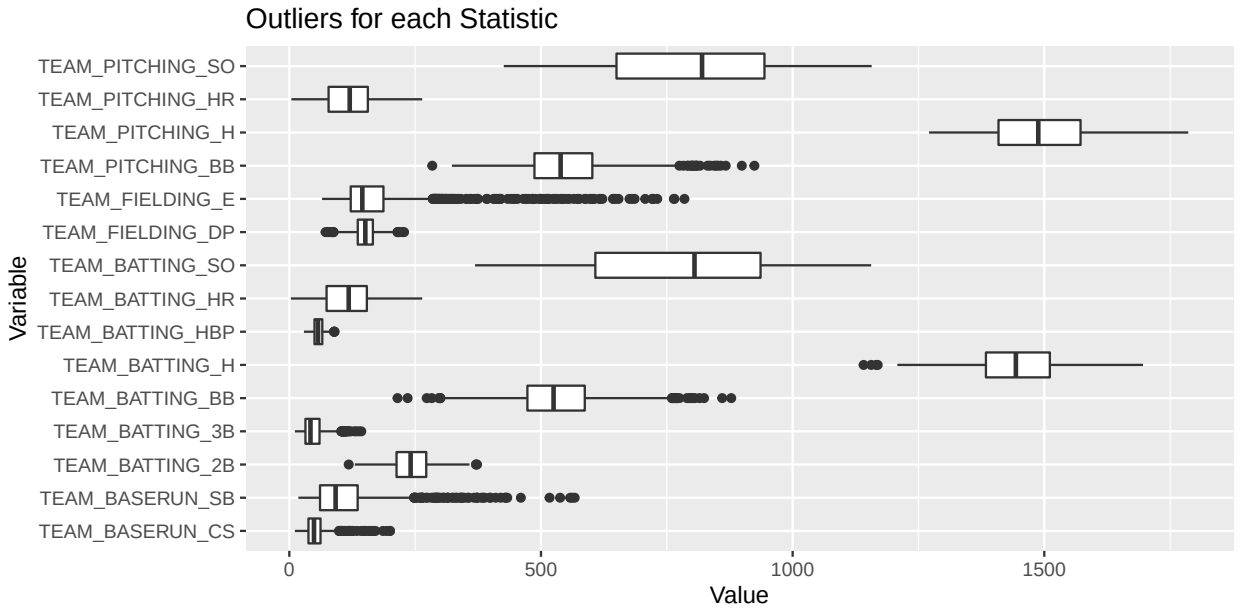


Off the bat (pun intended), there are some clear issues with two of the variables: `TEAM_PITCHING_SO` and `TEAM_PITCHING_H`.

- **TEAM_PITCHING_SO**: The largest outlier is close to 20,000, which averages ~123 strikeouts per game. This is only possible if every one of the 162 games went approximately 40 innings, and each out was a strikeout—which clearly has never happened.
- **TEAM_PITCHING_H**: The largest outlier for this variable is over 30,000 which is similarly highly unlikely.

Since both appear with a heavy right skew, we will use median and IQR to remove the outliers.

We can take a look at the distributions of these variables after removing the outliers:



This has eliminated what appears to be the most extreme outliers. There are still large outlier sets for Errors and Stolen Bases, but none that seem to dwarf the other variables. According to the box plot, observations appear to be on a logical scale given the data.

Imputation

Our early exploration of the data showed that the majority of the data is complete with only a few variables with missing values. We will use the **MICE** (Multivariate Imputation by Chained Equations) method. MICE is a principled method for dealing with missing data. It creates multiple imputations for the mean, as opposed to single imputations, and accounts for the statistical uncertainty in these imputations.

Feature Engineering

Single base hits

With outliers removed and missing values imputed via MICE, we can create new variables. The first will be single base hits, and it will be derived from some of the variables we do have. We know that Team Batting Hits (`TEAM_BATTING_H`) is a combination of *all* hits for the season, so we can create Single Base Hits as follows:

$$Singles = Total\ Hits - (Doubles + Triples + Homeruns)$$

Slugging Percentage

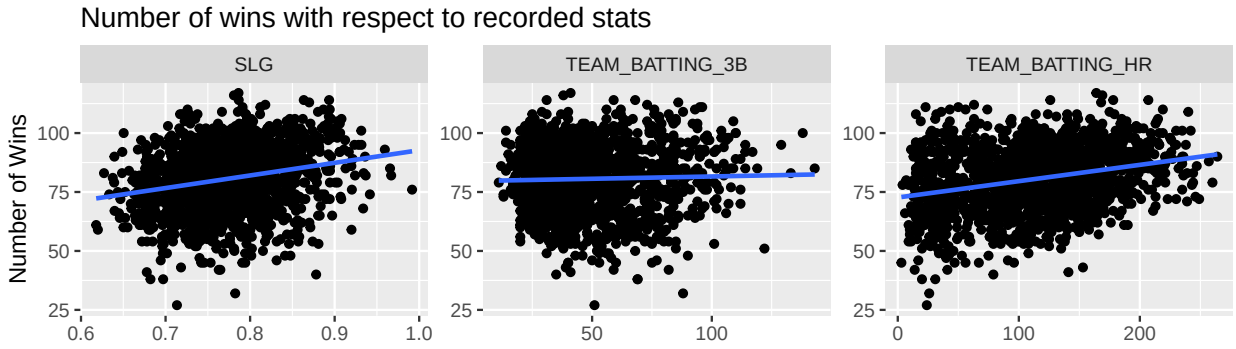
The second variable we will create is **Slugging Percentage**. Slugging is an offensive statistic that is a good predictor of winning and it tends to have better variance and correlation behavior than most other variables. It is composed of singles, doubles, triples, home runs and at-bats:

$$\text{SLG} = \frac{1B + 2 \times 2B + 3 \times 3B + 4 \times HR}{AtBats}$$

Because we don't have a statistic for at-bats, we will approximate it as:

$$\widehat{\text{SLG}} = \frac{1B + 2 \times 2B + 3 \times 3B + 4 \times HR}{SO_{batting} + H + BB}$$

As a sanity check, we can take a look at Slugging in comparison to Triples (`TEAM_BATTING_3B`) and Home Runs (`TEAM_BATTING_HR`). All of these variables should have a positive correlation with the total number of wins.



BUILD MODELS

Model 1

Base Model

We will start with a simple linear model to serve as a baseline. This includes all variables in the dataset.

Table 1: Base Model Regression Output

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	33.730	55.450	0.608	0.543
TEAM_BATTING_H	-0.005	0.028	-0.190	0.849
TEAM_BATTING_2B	-0.055	0.028	-1.949	0.051
TEAM_BATTING_3B	0.163	0.057	2.838	0.005
TEAM_BATTING_HR	0.488	0.155	3.139	0.002
TEAM_BATTING_BB	0.177	0.058	3.064	0.002
TEAM_BATTING_SO	-0.113	0.041	-2.775	0.006
TEAM_BASERUN_SB	0.073	0.007	10.310	0.000
TEAM_BASERUN_CS	0.013	0.015	0.846	0.398
TEAM_BATTING_HBP	0.002	0.030	0.073	0.942
TEAM_PITCHING_H	0.034	0.029	1.163	0.245
TEAM_PITCHING_HR	-0.378	0.166	-2.279	0.023
TEAM_PITCHING_BB	-0.132	0.060	-2.211	0.027
TEAM_PITCHING_SO	0.095	0.045	2.116	0.034
TEAM_FIELDING_E	-0.092	0.006	-15.771	0.000
TEAM_FIELDING_DP	-0.092	0.013	-6.877	0.000
SLG	12.061	71.920	0.168	0.867

We can immediately see that a few variables *exceed* the 0.05 p-value threshold for significance. Additionally, we see that both `TEAM_BATTING_1B` and `AB` have no coefficients at all! This is likely due to the fact that they are colinear with other variables in the model.

Enhanced Model

We used a combination of domain knowledge and data analysis to justify removing some features. First, we are keeping the coefficient for `TEAM_BATTING_HR` since its p-value is marginally above the general threshold and knowledge of the game suggests it is important. We also chose to remove the `TEAM_BATTING_HBP` variable because most of the values are missing from the dataset.

The remaining variables were chosen as a result of **backwards selection** based on null hypothesis testing for non-zero slope. This is done by removing one variable at a time to determine the minimum number of significant variables needed for prediction.

Altogether, we end up **removing** 7 total variables: `TEAM_BATTING_1B`, `AB`, `TEAM_BATTING_3B`, `TEAM_BATTING_H`, `TEAM_PITCHING_BB`, `TEAM_PITCHING_SO`, and `TEAM_BATTING_1B`.

Further data transformation

We used Cook's distance to remove additional outliers that are influencing the fit of the model above. We used a cutoff of $\frac{4}{N}$.

Table 2: Backwards Model Regression Output

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-84.046	18.528	-4.536	0.000
TEAM_BATTING_2B	-0.119	0.012	-9.525	0.000
TEAM_BATTING_HR	0.047	0.111	0.426	0.671
TEAM_BATTING_BB	0.085	0.007	11.861	0.000
TEAM_BATTING_SO	0.025	0.007	3.560	0.000
TEAM_BASERUN_SB	0.068	0.007	9.335	0.000
TEAM_BASERUN_CS	0.038	0.015	2.596	0.010
TEAM_PITCHING_HR	-0.108	0.104	-1.035	0.301
TEAM_PITCHING_H	0.023	0.005	4.898	0.000
TEAM_FIELDING_E	-0.099	0.006	-17.479	0.000
TEAM_FIELDING_DP	-0.097	0.012	-7.845	0.000
SLG	159.653	22.559	7.077	0.000

Coefficient Discussion

There are some counter intuitive results, which is expected given that baseball is a messy, imprecise game that has evolved over time. To that end, we should expect some seemingly strange results from algorithmic regression.

First, we see that the intercept itself is negative. This means that, given no information about any of the statistics, we would predict that a team would lose all of its games, and then some!

Next, fielding double plays and batting doubles both appear to have negative impacts on wins even though they *should* be positive impacts. Turning double plays, while a good for the defensive team, may suggest a larger, negative issue. Namely, weak pitching that leads to runners on base. Similarly, batting doubles, leaves runners open to double plays. Allowed hits by pitching, caught stealing, and batting strike-outs are all counter intuitive results as well, but they seem to be small contributors and these are events that happen regularly in every game.

Slugging as an approximation is the major predictor in this regression, by far. The other predictors other than the intercept are orders of magnitude less in predictive value. It should be noted that slugging alone is not a good predictor for wins overall.

Model 2

This is a model allowing pairwise interactions within the data types of baserunning, batting, pitching, and fielding. It is fit using a forward and backwards stepwise regression based on AIC.

Further data transformation

All the features will be centered and scaled. The target variable will be left in normal space. This will make prediction easier.

Stepwise Model

A fully specified model with all the interactions will be the starting point for the stepwise regression, which uses AIC as its target metric. For interest, this model will be compared with both the null model and the fully-specified model from where the stepwise process begins

Discussion

Models

Table 3: Null Model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	80.575	0.332	242.811	0

Table 4: Full Model with All Features & Pairwise Class Interactions

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	80.212	1.503	53.375	0.000
TEAM_BASERUN_CS	0.529	0.620	0.853	0.394
TEAM_BASERUN_SB	3.863	0.557	6.933	0.000
TEAM_BATTING_2B	-1.666	0.433	-3.849	0.000
TEAM_BATTING_3B	3.156	0.516	6.120	0.000
TEAM_BATTING_BB	10.235	36.231	0.282	0.778
TEAM_BATTING_H	-3.167	7.206	-0.440	0.660
TEAM_BATTING_HR	64.905	49.673	1.307	0.192
TEAM_BATTING_SO	-21.545	48.962	-0.440	0.660
TEAM_FIELDING_DP	-2.543	0.316	-8.052	0.000
TEAM_FIELDING_E	-11.439	0.722	-15.851	0.000
TEAM_PITCHING_BB	-7.478	36.134	-0.207	0.836
TEAM_PITCHING_H	7.617	8.683	0.877	0.380
TEAM_PITCHING_HR	-57.448	47.824	-1.201	0.230
TEAM_PITCHING_SO	16.158	44.230	0.365	0.715
TEAM_BASERUN_CS:TEAM_BASERUN_SB	1.066	0.255	4.179	0.000
TEAM_BATTING_2B:TEAM_BATTING_3B	-0.154	0.574	-0.268	0.789
TEAM_BATTING_2B:TEAM_BATTING_BB	0.954	0.463	2.061	0.040
TEAM_BATTING_2B:TEAM_BATTING_H	-0.392	0.338	-1.159	0.247
TEAM_BATTING_2B:TEAM_BATTING_HR	-0.817	0.702	-1.163	0.245
TEAM_BATTING_2B:TEAM_BATTING_SO	-1.094	0.607	-1.803	0.072
TEAM_BATTING_3B:TEAM_BATTING_BB	-0.383	0.440	-0.871	0.384
TEAM_BATTING_3B:TEAM_BATTING_H	-0.023	0.545	-0.042	0.966
TEAM_BATTING_3B:TEAM_BATTING_HR	0.390	0.638	0.611	0.541
TEAM_BATTING_3B:TEAM_BATTING_SO	-1.310	0.670	-1.954	0.051
TEAM_BATTING_BB:TEAM_BATTING_H	-1.190	2.135	-0.557	0.577

	Estimate	Std. Error	t value	Pr(>|t|)
TEAM_BATTING_BB:TEAM_BATTING_HR	5.418	3.695	1.466	0.143
TEAM_BATTING_BB:TEAM_BATTING_SO	-0.225	4.000	-0.056	0.955
TEAM_BATTING_H:TEAM_BATTING_HR	-3.678	3.360	-1.094	0.274
TEAM_BATTING_H:TEAM_BATTING_SO	-0.209	3.387	-0.062	0.951
TEAM_BATTING_HR:TEAM_BATTING_SO	4.585	4.821	0.951	0.342
TEAM_FIELDING_DP:TEAM_FIELDING_E	-1.387	0.422	-3.287	0.001
TEAM_PITCHING_BB:TEAM_PITCHING_H	0.439	2.425	0.181	0.856
TEAM_PITCHING_BB:TEAM_PITCHING_HR	-4.273	3.434	-1.244	0.214
TEAM_PITCHING_BB:TEAM_PITCHING_SO	-0.984	3.533	-0.278	0.781
TEAM_PITCHING_H:TEAM_PITCHING_HR	5.322	3.780	1.408	0.159
TEAM_PITCHING_H:TEAM_PITCHING_SO	0.786	3.639	0.216	0.829
TEAM_PITCHING_HR:TEAM_PITCHING_SO	-5.180	4.270	-1.213	0.225

Table 5: Stepwise Model with Pairwise Class Interactions

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	80.422	0.947	84.943	0.000
TEAM_BASERUN_CS	0.422	0.605	0.697	0.486
TEAM_BASERUN_SB	3.868	0.545	7.102	0.000
TEAM_BATTING_2B	-1.600	0.423	-3.780	0.000
TEAM_BATTING_3B	3.201	0.487	6.567	0.000
TEAM_BATTING_BB	7.167	6.166	1.162	0.245
TEAM_BATTING_H	-2.403	4.910	-0.489	0.625
TEAM_BATTING_HR	82.220	38.739	2.122	0.034
TEAM_BATTING_SO	-28.092	9.488	-2.961	0.003
TEAM_FIELDING_DP	-2.516	0.312	-8.067	0.000
TEAM_FIELDING_E	-11.334	0.695	-16.306	0.000
TEAM_PITCHING_BB	-4.412	6.097	-0.724	0.469
TEAM_PITCHING_H	6.681	5.951	1.123	0.262
TEAM_PITCHING_HR	-74.165	37.312	-1.988	0.047
TEAM_PITCHING_SO	22.182	8.543	2.597	0.010
TEAM_BASERUN_CS:TEAM_BASERUN_SB	1.102	0.248	4.447	0.000
TEAM_BATTING_2B:TEAM_BATTING_BB	1.066	0.439	2.426	0.015
TEAM_BATTING_2B:TEAM_BATTING_H	-0.447	0.315	-1.421	0.156
TEAM_BATTING_2B:TEAM_BATTING_HR	-0.949	0.576	-1.646	0.100
TEAM_BATTING_2B:TEAM_BATTING_SO	-0.780	0.470	-1.658	0.097
TEAM_BATTING_3B:TEAM_BATTING_SO	-1.007	0.445	-2.264	0.024
TEAM_BATTING_BB:TEAM_BATTING_H	-1.030	0.456	-2.261	0.024
TEAM_BATTING_BB:TEAM_BATTING_HR	6.272	3.099	2.024	0.043
TEAM_BATTING_H:TEAM_BATTING_HR	-3.762	2.464	-1.527	0.127
TEAM_FIELDING_DP:TEAM_FIELDING_E	-1.390	0.417	-3.336	0.001
TEAM_PITCHING_BB:TEAM_PITCHING_HR	-4.916	2.913	-1.688	0.092
TEAM_PITCHING_BB:TEAM_PITCHING_SO	-1.253	0.439	-2.852	0.004
TEAM_PITCHING_H:TEAM_PITCHING_HR	5.742	2.940	1.953	0.051
TEAM_PITCHING_HR:TEAM_PITCHING_SO	-1.147	0.406	-2.825	0.005

The null model expects teams to lose slightly more than half their games. The full model has a lower intercept, which indicates that the impact of all the features explains why teams *win* slightly more than just the grand mean does. The stepwise model is similar, having a lower intercept than the null but a higher intercept that the full model.

Also, the full model has the higher R^2 and the stepwise model has the higher *adjusted* R^2. This is expected, as there is no penalty for adding more parameters in regular R^2 measures.

The **stepwise** model will be the one discussed below.

Coefficients Most of the coefficients are reasonable within the context of the game of baseball. These two statements are axiomatic:

- The only way to win is to have the higher score at the end of the game.
- The only way to score is for a baserunner to cross home plate.

With those in mind, we can make the following observations about the linear non-interactive coefficients.

- Reasonable
  - Caught stealing removes baserunners but it isn't significant and can be ignored on its own.
  - Stolen bases get a runner closer to home plate; positive coefficient makes sense.
  - Triples get a runner very close to home plate; positive coefficient makes sense.
  - Walks are free baserunners; positive coefficient makes sense but it isn't significant and can be ignored on its own.
  - Hitting home runs directly increase the score; positive coefficient makes sense.
  - Striking out reduces the number of baserunners; negative coefficient makes sense.
  - Errors allow the other team free baserunners; negative coefficient makes sense.
  - Giving up walks allows the other team free baserunners; negative coefficient makes sense but it isn't significant and can be ignored on its own.
  - Giving up home runs gives the opponent scores; negative coefficient makes sense.
  - Getting strikeouts reduces the opponents baserunners; positive coefficient makes sense.
- Curious
  - Hits increase the number of baserunners; why negative and insignificant?
  - Hitting doubles get runners on base; why negative?
  - Turning double plays reduce baserunners; why negative?
  - Giving up hits allows the other team baserunners; why positive although insignificant?

First, an absolutely fascinating observation from an earlier state of the model. In the very first draft of this exercise, prior to some of the data adjustments performed here, `TEAM_BATTING_HR` was removed due to its high correlation with `TEAM_PITCHING_HR`. In that first draft, pitching giving up home runs was given a *positive* coefficient. This made no sense. Allowing batting home runs to re-enter the model, In hindsight this becomes obvious. The pitching *was being used as an indicator for* ***batting!!*** They are almost 100% correlated! This correlation allowed the use of pitching HRs as an indicator for the hidden batting HRs which has a larger magnitude! Once both variables were restored to the model, the logical coefficients surfaced. This is yet another reason why models should not be trusted out of the box, but all model results should be reviewed for sanity and sense!!

The curiosities above can *possibly* be resolved by looking at the interaction terms. The pitching and batting home run and strikeout coefficients are an order of magnitude greater than all the others. It is likely that the negative coefficients in some of the batting stats and positive coefficients in some of the pitching stats are being used to "temper" the effect of SOs and HRs.

A possible explanation for the negative coefficient for getting double plays, is that double plays require at least two people on base. That means that the opponent has a lot of base runners, which is very highly correlated with scoring.

The behavior of doubles remains confusing. Perhaps its presence in a number of the interaction terms means that the singleton variable needs to act as a balance. Another theory may be that it's hiding something like a team's propensity to strand runners on base. It would be interesting to see a breakdown between the American and National leagues, as the latter tends to be somewhat better at "small ball" and moving the runners along.

Model 3

The final model is a Stepwise Regression with Repeated k-fold Cross-Validation, and higher order polynomials variables were introduced into the full model.

A stepwise variable selection model is conducted to determine what are the variables that can help predict the number of wins for the team. The method allows variables to be added one at a time to the model, as long as the F-statistic is below the specified α, in this case $\alpha = 0.05$. However, variables already in the model do not necessarily stay in. The steps evaluate all of the variables already included in the model and remove any variable that has an insignificant F-statistic. Only after this test ends, is the best model found, that is when none of the variables can be excluded and every variable included in the model is significant.

Here, the dependent variable is the continuous variable, `TARGET_WINS`, and the independent variables are the full model to identify the most contributing predictors. In addition, a robust method for estimating the accuracy of a model, the k-fold cross-validation method, was performed evaluate the model performance on different subset of the training data and then calculate the average prediction error rate.

Further data transformation

Once again, Cook's distance is use to decide if an individual entity is an extreme value or not.

Coefficient Discussion

Table 6: K-fold Step Model with Higher Order Polynomials Output

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	129.200	39.642	3.259	0.001
TEAM_BATTING_H	0.177	0.022	7.890	0.000
TEAM_BATTING_2B	-0.225	0.022	-10.227	0.000
TEAM_BASERUN_SB	0.056	0.008	7.267	0.000
TEAM_BASERUN_CS	-0.095	0.033	-2.869	0.004
TEAM_PITCHING_H	-0.151	0.052	-2.894	0.004
TEAM_PITCHING_HR	-0.075	0.019	-3.874	0.000
TEAM_FIELDING_E	-0.195	0.016	-12.222	0.000
TEAM_FIELDING_DP	-0.401	0.106	-3.766	0.000
TEAM_BATTING_BB_1	0.000	0.000	3.183	0.001
TEAM_BATTING_SO_1	0.000	0.000	-8.989	0.000
TEAM_BASERUN_CS_1	0.001	0.000	5.045	0.000
TEAM_PITCHING_H_1	0.000	0.000	3.674	0.000
TEAM_PITCHING_BB_1	0.000	0.000	-2.182	0.029
TEAM_FIELDING_E_1	0.000	0.000	5.792	0.000
TEAM_FIELDING_DP_1	0.001	0.000	2.850	0.004
TEAM_BATTING_1B_1	0.000	0.000	-8.461	0.000

This discussion begins with the variables that follow the logic of winning and are self-evident. Firstly, the model suggests that winning is in favor of the team which when batting has more total bases, (1B, 2B, 3B, HR). It further reveals that an increase in singles would lead to better chances of winning. A hit is a hit, and more hits than the opponent will inevitably lead to a win. Likewise, an increase in walks by the batting team is awarded first base, further increasing base counts and, thus the chances of winning. In terms of the base run, an increase in the number of stolen bases, and a decrease in caught steals would lead to a win for the batting team. These variables have a positive effect on the number of wins, and it is understandable as to why.

When interpreting the model based on the pitching team's plays, it is perceptible that walks allowed should be reduced so that the batting team hits does not increase. Moreover, preventing errors and increasing double

plays in which two players are put out will also increase the chances of winning.

On the other hand, it is noteworthy that the model suggests some counter-intuitive play strategies. Firstly, when all variables are null, the intercept predicts that a team would win, with a target win of 129. Moreover, the model suggests that a decrease in double hits and an increase in strikeouts by the batting team will improve winning. These variables were kept because when a batter steps to the plate, the player is more likely to strike out than to get a hit. Trying to hit the ball out of the park will come with strikeouts but it will also increase the chances of hitting home runs (even 1B, 2B, 3B), and that is a pretty good exchange that most teams are willing to perform.

SELECT MODELS

Criteria

In order to select the best model to make predictions, we looked at the following goodness of fit metrics:

1. R^2, which represents the proportion of the variance explained by the model
2. Adjusted R^2, which is a modified version of R^2 taking parsimony into consideration
3. *Root Mean Squared Error* (RMSE), which is the square root of the mean squared difference between the observation and the fitted value
4. *Akaike Information Criterion* (AIC), which is an information-criteria based estimate of the Kullback-Leibler divergence from the current model to the "true" model. One of the nice results of the theory of information-criteria based measures is that there is included recognition of parsimony.

Performance

Table 7: Performance Statistics & Error Measure of Models

Models	R.squared	adj.R.squared	RSME	AIC
Model #1: Backwards	0.436	0.432	9.509	11516.86
Model #2: Pairwise	0.427	0.417	10.193	12389.22
Model #3: K-fold	0.456	0.450	9.531	11636.83

It was interesting to compare the three models given that their performance statistics, error measurements, and residual distributions were not drastically different from each other. **Model #2: Pairwise** was the worst performing model for all of our metrics, and was eliminated immediately. **Model #3: K-fold** accounts for nearly 45% of the variation in the dependent variable with the independent variables because its R^2 is 0.456 and its adjusted R^2 is 0.450, both of which are acceptable for a good model. However, **Model #1: Backwards** has the smallest RSME of all the models. RMSE is a measure of how far off predictions are from the actual, giving extra weight to outliers through summing the squared differences. This demonstrates a greater reduction in the randomness by **Model #1: Backwards**. But its difference from **Model #3: K-fold** is only 0.022. It is hard to judge if that is a significant difference. We thus also considered AIC. With AIC, it is not the magnitude of the value which is important, but the difference between the various model AIC's. The magnitude is a function of the number of observations. Accepted rules of thumb in the statistical literature are that if the difference is greater than 10, there is effectively no support for the model with the larger AIC (Section 2.6, Burnham & Anderson 2002). As a result, **Model #1: Backwards** is our selected model.

Discussion

This is an interesting model in that given no other information, a team would be expected to win negative games, which is clearly impossible. That being said, it is an indication of the predictive power of the engineered **slugging** feature that allows small increments in slugging to result in significant shifts in target

wins. This engineered feature contains within it numerous interactions. It combines scaled versions of some features and normalizes them for the combination of other features. Perhaps that is why it is so significant.

Predictions

Model #1: Backwards was used to predict the target wins base on the evaluation set, once processed similarly to the training data. As the number of games won must be integral, the values were rounded to the closest integer.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   49.00   74.00   81.00   79.89   86.00   97.00
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##  72  84  75  68  70  64  83  85  83  87  77  78  83  84  83  71  78  85  68  63
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##  92  90  86  86  78  86  76  90  87  80  91  72  62  74  83  74  69  77  73  83
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##  88  85  86  63  93  71  82  94  70  80  91  83  71  89  81  86  86  96  91  97
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##  87  76  72  82  88  87  89  88  94  88  77  75  84  81  66  64  73  80  76  74
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##  93  83  49  67  73  73  75  80  78  81  83  79  60  78  89  97  88  86  79  93
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##  81  78  76  84  69  73  83  91  89  86  69  71  62  73  92  83  85  79  81  89
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##  75  61  86  67  82  77  96  74  81  68  77  69  79  83  86  89  80  80  80  73
## 141 142 143 144 145 146 147 148 149 150 151 152
##  87  88  83  78  84  77  83  72  83  81  85  68
```

REFERENCES

- Burnham, Kenneth P., and David R. Anderson. 2002. Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach. Second. New York: Springer Science+Business Media, Inc.

CODE APPENDIX

The code chunks below represent the R code called in order during the analysis. They are reproduced in the appendix for review and comment.

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

```
# Libraries and variables
library(tidyverse)
library(ggplot2)
library(corrplot)
library(tidymodels)
library(mice)
library(psych)
library(Hmisc)
library(MASS)
library(dplyr)
library(caret)

urlRemote  = "https://raw.githubusercontent.com/"
pathGithub = "aadler/DT621_Fall2020_Group2/master/HW1/data/"
```

```
fileTrain   = "moneyball-training-data.csv"
train_set = read.csv(paste0(urlRemote, pathGithub, fileTrain))

fileEval   = "moneyball-evaluation-data.csv"
eval_set = paste0(urlRemote, pathGithub, fileEval) %>% read.csv()

set.seed(9450)
```

```
# DATA EXPLORATION
avgWins <- mean(train_set$TARGET_WINS)
sdev <- sd(train_set$TARGET_WINS)
l1 <- mean(train_set$TARGET_WINS) - (2 * sdev)
l2 <- mean(train_set$TARGET_WINS) + (2 * sdev)
# histogram of wins
p <- ggplot(train_set, aes(x=TARGET_WINS)) +
  geom_histogram() +
  geom_vline(aes(xintercept = avgWins),
             color = "red", linetype = "dashed", size = 1)+
  geom_vline(aes(xintercept = l1),
             color="blue", linetype = "dashed", size = 1) +
    geom_vline(aes(xintercept = l2),
             color = "blue", linetype = "dashed", size = 1) +
  labs(title = "Distribution of Wins", x = "Number of wins",
       y = "Number of seasons")
p
```

```
# percent null for each variable
pct_null <- data.frame(do.call("rbind", map(train_set %>% dplyr::select(-INDEX),
                                            ~ mean(is.na(.)))))
colnames(pct_null) <- c('PCT_NULL')
totalNulls <- pct_null %>%
  mutate(VARIABLE = rownames(.)) %>%
  arrange(desc(PCT_NULL)) %>%
  filter(PCT_NULL > 0) %>%
  dplyr::select(VARIABLE, PCT_NULL)
ggplot(totalNulls, aes(x = reorder(VARIABLE, PCT_NULL), y = PCT_NULL,
                       label = round(PCT_NULL, 2))) +
  geom_text(vjust = 0.5, hjust = -0.05)+
  geom_bar(stat = "identity") +
  ggtitle("Variables with Missing Information") +
  xlab("Statistic") + ylab("Percent Missing") +
  coord_flip() + expand_limits(y = 1)
```

```
# Variables expected to have a negative impact on wins
negSet <- train_set %>%
  dplyr::select(TARGET_WINS,
         'Errors' = TEAM_FIELDING_E,
         'Hits allowed' = TEAM_PITCHING_H,
         'Strikeouts by batters' = TEAM_BATTING_SO,
         'Caught stealing' = TEAM_BASERUN_CS,
         'Walks allowed' = TEAM_PITCHING_BB,
         'Home Runs allowed' = TEAM_PITCHING_HR)
ggplot(data = negSet %>%
  gather(-TARGET_WINS, key = "STAT", value = "VALUE"),
```

```
  aes(x = VALUE, y = TARGET_WINS)) +
  geom_point() + geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Number of wins with respect to recorded stats",
       x = "", y = "Number of Wins") + facet_wrap(~ STAT, scales = "free")
```

```
# Variables expected to have a positive impact on wins
posSet <- train_set %>%
  dplyr::select(TARGET_WINS,
         'Base hits' = TEAM_BATTING_H,
         'Doubles' = TEAM_BATTING_2B,
         'Triples' = TEAM_BATTING_3B,
         'Home Runs' = TEAM_BATTING_HR,
         'Walks' = TEAM_BATTING_BB,
         'Batters hit' = TEAM_BATTING_HBP,
         'Stolen bases' = TEAM_BASERUN_SB,
         'Double plays' = TEAM_FIELDING_DP,
         'Strikeouts' = TEAM_PITCHING_SO)
ggplot(data = posSet %>%
  gather(-TARGET_WINS, key = "STAT", value = "VALUE"),
  aes(x = VALUE, y = TARGET_WINS)) + geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Number of Wins with Respect to Recorded Stats", x = "",
       y = "Number of Wins") + facet_wrap(~ STAT, scales = "free")
```

```
# Understanding outliers by variable
totalOutliers <- data.frame(
  sapply(train_set %>% dplyr::select(-INDEX, -TARGET_WINS, - TEAM_BATTING_HBP),
       function(x){length(boxplot.stats(x)$out)/nrow(train_set)}))
totalOutliers$VARIABLE_NM <- rownames(totalOutliers)
colnames(totalOutliers) <- c('PCT_OUTLIERS', 'VARIABLE_NM')
ggplot(totalOutliers,
       aes(x = reorder(VARIABLE_NM, PCT_OUTLIERS), y=PCT_OUTLIERS,
           label = round(PCT_OUTLIERS, 3))) +
  geom_text(vjust = 0.5, hjust = -0.05)+ geom_bar(stat = "identity") +
  ggtitle("Percentage of Outliers") + xlab("Statistic") +
  ylab("Percent of Data that is an Outlier") + coord_flip() +
  expand_limits(y = 0.15)
```

```
# Distributions of variables with few outliers
set1 <- train_set %>%
  dplyr::select('Home Runs by batters' = TEAM_BATTING_HR,
         'Strikeouts by batters' = TEAM_BATTING_SO,
         'Home Runs allowed' = TEAM_PITCHING_HR) %>%
  gather("stat", "value") %>%
  filter(complete.cases(.) == TRUE)
vals <- ggplot(set1 , aes(x = stat, y = value)) +
    geom_boxplot() + labs(title = "", x = "", y = "Value") +
  facet_wrap( ~ stat, scales = "free")
vals
```

```
# Distributions of variables with many outliers
set2 <-  train_set %>%
  dplyr::select('Walks by batters' = TEAM_BATTING_BB,
         'Stolen bases' = TEAM_BASERUN_SB,
         'Hits allowed' = TEAM_PITCHING_H,
```

```
        'Errors' = TEAM_FIELDING_E) %>%
  gather("stat", "value") %>%
  filter(complete.cases(.) == TRUE)
vals2 <- ggplot(set2 , aes(x=stat, y=value)) +
    geom_boxplot() +
  labs(title = "",x = "", y = "Value") +
  facet_wrap(~ stat, scales = 'free', ncol = 4)
vals2
```

```
# Distribution of all other variables
set3 <-  train_set %>%
  dplyr::select('Base Hits by batters' = TEAM_BATTING_H,
        'Doubles by batters' = TEAM_BATTING_2B,
        'Triples by batters' = TEAM_BATTING_3B,
        'Caught stealing' = TEAM_BASERUN_CS,
        'Walks allowed' = TEAM_PITCHING_BB,
        'Strikeouts by pitchers' = TEAM_PITCHING_SO,
        'Double plays' = TEAM_FIELDING_DP) %>%
  gather("stat", "value") %>%
  filter(complete.cases(.) == TRUE)
vals3 <- ggplot(set3, aes(x=stat, y=value)) +
    geom_boxplot() +
  labs(title = "", x = "",y = "Value") +
  facet_wrap(~ stat, scales = 'free', ncol = 4)
vals3
```

```
# Correlation matrix with significance levels (p-value)
res2 <- rcorr(as.matrix(train_set %>% dplyr::select(-INDEX)))
# Insignificant correlation are crossed
corrplot(res2$r, type="upper", order="hclust",
         tl.col = "black", p.mat = res2$P, sig.level = 0.01, insig = "blank")
```

```
# Visualizing variables prior to outlier removal
train_set %>%
  dplyr::select(-TARGET_WINS, -INDEX) %>%
  pivot_longer(everything(),
               names_to = "Variable",
               values_to = "Value") %>%
  ggplot(aes(x = Variable, y = Value)) +
  geom_boxplot(na.rm = TRUE) +
  labs(title="Outliers for each Statistic",x="Variable", y = "Value") +
  coord_flip()
```

```
# Remove outliers
train_set<- train_set %>%
  na.omit() %>%
  summarise(iqr = IQR(TEAM_PITCHING_SO)) %>%
  bind_cols(
            train_set
            )  %>%
  filter(
    TEAM_PITCHING_SO > quantile(TEAM_PITCHING_SO,
                                probs = c(0.25),
                                na.rm = TRUE) - 1.5 * iqr,
    TEAM_PITCHING_SO < quantile(TEAM_PITCHING_SO,
```

```
                                probs = c(0.75),
                                na.rm = TRUE) + 1.5 * iqr
        ) %>%
  dplyr::select(-iqr)
```

```
train_set<- train_set %>%
  na.omit() %>%
  summarise(iqr = IQR(TEAM_PITCHING_H)) %>%
  bind_cols(
            train_set
            )  %>%
  filter(
    TEAM_PITCHING_H > quantile(TEAM_PITCHING_H,
                                probs = c(0.25),
                                na.rm = TRUE) - 1.5 * iqr,
    TEAM_PITCHING_H < quantile(TEAM_PITCHING_H,
                                probs = c(0.75),
                                na.rm = TRUE) + 1.5 * iqr
        ) %>%
  dplyr::select(-iqr)
```

```
# visualizing variables post outlier removal
train_set %>%
  dplyr::select(-TARGET_WINS, -INDEX) %>%
  pivot_longer(everything(),
               names_to ="Variable",
               values_to="Value") %>%
  ggplot(aes(x=Variable, y=Value)) +
  geom_boxplot(na.rm = TRUE) +
  labs(title="Outliers for each Statistic",x="Variable", y = "Value") +
  coord_flip()
```

```
# Imputing values
train_set <- complete(mice(data = train_set,
                           method = "pmm",
                           seed = 9450,
                           print=FALSE), 3)
```

```
# Defining singles variable
train_set<- train_set %>%
  dplyr::select(-INDEX) %>%
  mutate_if(is.integer, as.numeric) %>%
  mutate(TEAM_BATTING_1B =
           TEAM_BATTING_H -
           (TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR))
```

```
# Defining at bats and slugging variables
train_set<- train_set %>%
  mutate(AB = TEAM_BATTING_SO + TEAM_BATTING_BB + TEAM_BATTING_H,
         SLG = (TEAM_BATTING_1B + 2 * TEAM_BATTING_2B + 3 * TEAM_BATTING_3B +
                  4 * TEAM_BATTING_HR) / AB)
```

```
# Visualizing slugging compared to wins
modified_set <- train_set %>%
  dplyr::select(TARGET_WINS, SLG, TEAM_BATTING_3B, TEAM_BATTING_HR)
```

```
ggplot(data = modified_set %>%
  gather(-TARGET_WINS, key = "STAT", value = "VALUE"),
  aes(x = VALUE, y = TARGET_WINS)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Number of wins with respect to recorded stats", x = "",
       y = "Number of Wins") +
  facet_wrap(~ STAT, scales = "free")
```

```
# BUILD MODELS
# Model 1: Backwards Selection
lm_reg = lm(data=train_set, TARGET_WINS ~ .)

knitr::kable(summary(lm_reg)$coefficients, digits = 3L,
             caption = 'Base Model Regression Output')
```

```
# Removing outliers via cooks distance
lm_1<- lm(TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_HR +
          TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB +
            TEAM_BASERUN_CS + TEAM_PITCHING_HR + TEAM_PITCHING_H +
            TEAM_FIELDING_E + TEAM_FIELDING_DP + SLG, data = train_set)
cooks_dis<- cooks.distance(lm_1)
influential<- as.numeric(names(cooks_dis)[(cooks_dis > (4 / nrow(train_set)))])
train_df2<- train_set[-influential, ]
lm_1<- lm(TARGET_WINS ~ TEAM_BATTING_2B + TEAM_BATTING_HR +
            TEAM_BATTING_BB + TEAM_BATTING_SO + TEAM_BASERUN_SB +
            TEAM_BASERUN_CS + TEAM_PITCHING_HR + TEAM_PITCHING_H +
            TEAM_FIELDING_E + TEAM_FIELDING_DP + SLG, data = train_df2)
mod_sum1 = summary(lm_1)
```

```
# Model 2: Pairwise Model
# Cleaning data for model 2
ts2 <- train_set[, -(17:19)]
ts2 <- data.frame(apply(ts2, 2, scale))
ts2$TARGET_WINS <- train_set$TARGET_WINS
```

```
nullModel <- lm(TARGET_WINS ~ 1, data = ts2)
fullModel <- lm(TARGET_WINS ~
  (TEAM_BASERUN_CS + TEAM_BASERUN_SB) ^ 2 + (TEAM_BATTING_2B + TEAM_BATTING_3B +
  TEAM_BATTING_BB + TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_BATTING_SO) ^ 2 +
  (TEAM_FIELDING_DP + TEAM_FIELDING_E) ^ 2 + (TEAM_PITCHING_BB +
  TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_PITCHING_SO) ^ 2, data = ts2)
stepModel <- stepAIC(fullModel, scope = list(upper = ~
  (TEAM_BASERUN_CS + TEAM_BASERUN_SB) ^ 2 + (TEAM_BATTING_2B + TEAM_BATTING_3B +
  TEAM_BATTING_BB + TEAM_BATTING_H + TEAM_BATTING_HR + TEAM_BATTING_SO) ^ 2 +
  (TEAM_FIELDING_DP + TEAM_FIELDING_E) ^ 2 + (TEAM_PITCHING_BB +
  TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_PITCHING_SO) ^ 2, lower = ~ 1),
  direction = 'both', trace = 0)
snm <- summary(nullModel)
sfm <- summary(fullModel)
sstm <- summary(stepModel)
stmr2 <- sstm$r.squared
stmar2 <- sstm$adj.r.squared
stmAIC <- AIC(stepModel)
stmRMSE <- sqrt(mean(stepModel$residuals ^ 2))
```

```r
# Model 3: K-fold Model
stepdata <- subset(train_set, select = -c(AB, SLG, TEAM_BATTING_HBP))
stepdata <- cbind(stepdata,
                  sapply(stepdata[2:length(stepdata)], function(x) x^2))
names(stepdata) <- make.unique(names(stepdata), sep = "_")
model <- lm(TARGET_WINS ~ ., data = stepdata)
cooksd <- cooks.distance(model)
influential <- as.numeric(
  names(cooksd)[(cooksd > 4 * mean(cooksd, na.rm = TRUE))])
stepdata.2 <- stepdata[-c(influential), ]

# Set up repeated k-fold cross-validation
set.seed(525)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
step.model <- train(TARGET_WINS ~ ., data = stepdata.2, method = "lmStepAIC",
                    trControl = train.control, trace = FALSE)
# Final model
step.model <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B +
                   TEAM_BASERUN_SB + TEAM_BASERUN_CS + TEAM_PITCHING_H +
                   TEAM_PITCHING_HR + TEAM_FIELDING_E +  TEAM_FIELDING_DP +
                   TEAM_BATTING_BB_1 + TEAM_BATTING_SO_1 +  TEAM_BASERUN_CS_1 +
                   TEAM_PITCHING_H_1 + TEAM_PITCHING_BB_1 +  TEAM_FIELDING_E_1 +
                   TEAM_FIELDING_DP_1 + TEAM_BATTING_1B_1, data = stepdata.2)

# Summary of the model
mod_sum3 <- summary(step.model)
```

```r
# SELECT MODELS
# Model accuracy
results <- data.frame(
  Models = c("Model #1: Backwards" , "Model #2: Pairwise", "Model #3: K-fold"),
  R.squared = c(mod_sum1[["r.squared"]], stmr2, mod_sum3[["r.squared"]]),
  adj.R.squared = c(mod_sum1[["adj.r.squared"]], stmar2,
                    mod_sum3[["adj.r.squared"]]),
  RSME = c(sqrt(mean(lm_1$residuals ^ 2)), stmRMSE,
           sqrt(mean(step.model$residuals ^ 2))),
  AIC = c(AIC(lm_1), AIC(stepModel), AIC(step.model)))
results %>% knitr::kable(digits = 3L,
                         caption = 'Performance Statistics & Error Measure of Models')
```

```r
# Predict target wins
# outlier removal - TEAM_PITCHING_SO
eval_set <- eval_set %>% na.omit() %>%
  summarise(iqr = IQR(TEAM_PITCHING_SO)) %>%
  bind_cols(eval_set) %>%
  filter(TEAM_PITCHING_SO > quantile(TEAM_PITCHING_SO, probs = c(0.25), na.rm = TRUE)
         - 1.5 * iqr,
         TEAM_PITCHING_SO < quantile(TEAM_PITCHING_SO, probs = c(0.75), na.rm = TRUE)
         + 1.5 * iqr) %>%
  dplyr::select(-iqr)

# outlier removal - TEAM_PITCHING_H
```

```r
eval_set <- eval_set %>% na.omit() %>%
  summarise(iqr = IQR(TEAM_PITCHING_H)) %>%
  bind_cols(eval_set) %>%
  filter(TEAM_PITCHING_H > quantile(TEAM_PITCHING_H, probs = c(0.25), na.rm = TRUE)
         - 1.5 * iqr,
         TEAM_PITCHING_H < quantile(TEAM_PITCHING_H, probs = c(0.75), na.rm = TRUE)
         + 1.5 * iqr) %>%
  dplyr::select(-iqr)

# mice imputation
eval_set <- complete(mice(data = eval_set, method = "pmm", seed = 9450, print=FALSE), 3)

# feature engineering
eval_set <- eval_set %>% dplyr::select(-INDEX) %>%
  mutate_if(is.integer, as.numeric) %>%
  mutate(TEAM_BATTING_1B = TEAM_BATTING_H - (TEAM_BATTING_2B + TEAM_BATTING_3B +
                                               TEAM_BATTING_HR))
eval_set <- eval_set %>%
  mutate(AB = TEAM_BATTING_SO + TEAM_BATTING_BB + TEAM_BATTING_H,
         SLG = (TEAM_BATTING_1B + 2 * TEAM_BATTING_2B + 3 * TEAM_BATTING_3B +
                  4 * TEAM_BATTING_HR) / AB)

# final predictions
predictions <- round(predict(lm_1, newdata = eval_set))
summary(predictions)
predictions
```