HW1: Moneyball Multiple Regression

Critical Thinking Group One

2021-03-06

Contents

AUTHORSHIP

Critical Thinking Group 1:

- Angel Claudio,
- Bonnie Cooper,
- Manolis Manoli,
- Magnus Skonberg,
- Christian Thieme and
- Leo Yi

BACKGROUND

On Sabermetrics

Statistics have played a role in quantifying baseball since Henry Chadwick introduced the box score in 1858. The box score was adopted from cricket scorecards and introduced metrics such as the batting average and earned run average which are still used in analytics to this day. However, the statistics of baseball saw little change until the late 20th century with the advent of sabermetrics.

Sabermetrics introduced a new approach to evaluating baseball and baseball athletes. For instance, the classic approach for weighing a players worth was measured with Batting Average statistics. However, sabermetrics no longer weighed a player only by his direct production but incorporated metrics to describe the players ability to create run opportunities for the team. Sabermetrics was largely marginalized by major league teams which relied primarily on scouts to find talent. This all changed when the Oakland A's began to levy sabermetric principles to fill out their rosters in the 1990s. Ultimately the A's approach is attributed

to a 20 game win streak in the 2002 season. At the time, a streak of this length was unprecedented for modern major league baseball. The Oakland A's achievements were celebrated in the book Moneyball (later made to a movie) and advanced analytics became the mainstay for assessing new talent and quantifying althete and team performance.

Our Approach

In the following analysis, we will use sabermetric principles to explore, analyze and model a data set that describes team performance in a given year. The purpose of the assignment is to build a multiple linear regression model on the training data to predict the number of wins for the teams provided in a given baseball season.

- Each record represents a professional baseball team from the years 1871 to 2006 inclusive.
- Each record has the performance of the team for the given year, with all statistics adjusted to match the performance of a 162 game season.

The following table describes the data set feature variables in detail.

Variable Name	Definition	Theoretical Effect
team_batting_h	Base hits by batters (1b,2b,3b,hr)	Positive impact on Wins
team_batting_2b	Doubles by batters (2b)	Positive impact on Wins
team_batting_3b	Triples by batters (3b)	Positive impact on Wins
team_batting_hr	Homeruns by batters (hr)	Positive impact on Wins
team_batting_bb	Walks by batters	Positive impact on Wins
team_batting_hbp	Batters hit by pitch	Positive impact on Wins
team_batting_so	Strikeouts by batters	Negative impact on Wins
team_baserun_sb	Stolen bases	Positive impact on Wins
team_baserun_cs	Caught stealing	Negative impact on Wins
team_fielding_e	Errors	Negative impact on Wins
team_fielding_dp	Double plays	Positive impact on Wins
team_pitching_bb	Walks allowed	Negative impact on Wins
team_pitching_h	Hits allowed	Negative impact on Wins
team_pitching_hr	Homeruns allowed	Negative impact on Wins
team_pitching_so	Strikeouts by pitchers	Positive impact on Wins

The goal of our modeling approach is to find an optimal multiple linear regression model that utilizes this feature set to predict the number of games won assuming a 162 game season. Working with a training data set, we begin by importing the data and performing a basic EDA and visualization. Following, the data is prepared for further analysis by a series of transformations (e.g. to account for missing data and collinearity). We then take an incremental modeling approach to (1) Begin with all model features included (kitchen sink model), (2) Remove outliers & retrain the model and (3) Remove impertinent features from the model. Finally, model performance is evaluated on a separate test data set.

1. DATA EXPLORATION

We utilize the built-in `glimpse()` method to gain insight into the dimensions, variable characteristics, and value range for our training dataset:

```
## Rows: 2,276
```

```
## Columns: 17
## $ INDEX            <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 15, 16, 17, 18, 1~
## $ TARGET_WINS      <dbl> 39, 70, 86, 70, 82, 75, 80, 85, 86, 76, 78, 68, 72, 7~
## $ TEAM_BATTING_H   <dbl> 1445, 1339, 1377, 1387, 1297, 1279, 1244, 1273, 1391,~
## $ TEAM_BATTING_2B  <dbl> 194, 219, 232, 209, 186, 200, 179, 171, 197, 213, 179~
## $ TEAM_BATTING_3B  <dbl> 39, 22, 35, 38, 27, 36, 54, 37, 40, 18, 27, 31, 41, 2~
## $ TEAM_BATTING_HR  <dbl> 13, 190, 137, 96, 102, 92, 122, 115, 114, 96, 82, 95,~
## $ TEAM_BATTING_BB  <dbl> 143, 685, 602, 451, 472, 443, 525, 456, 447, 441, 374~
## $ TEAM_BATTING_SO  <dbl> 842, 1075, 917, 922, 920, 973, 1062, 1027, 922, 827, ~
## $ TEAM_BASERUN_SB  <dbl> NA, 37, 46, 43, 49, 107, 80, 40, 69, 72, 60, 119, 221~
## $ TEAM_BASERUN_CS  <dbl> NA, 28, 27, 30, 39, 59, 54, 36, 27, 34, 39, 79, 109, ~
## $ TEAM_BATTING_HBP <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ TEAM_PITCHING_H  <dbl> 9364, 1347, 1377, 1396, 1297, 1279, 1244, 1281, 1391,~
## $ TEAM_PITCHING_HR <dbl> 84, 191, 137, 97, 102, 92, 122, 116, 114, 96, 86, 95,~
## $ TEAM_PITCHING_BB <dbl> 927, 689, 602, 454, 472, 443, 525, 459, 447, 441, 391~
## $ TEAM_PITCHING_SO <dbl> 5456, 1082, 917, 928, 920, 973, 1062, 1033, 922, 827,~
## $ TEAM_FIELDING_E  <dbl> 1011, 193, 175, 164, 138, 123, 136, 112, 127, 131, 11~
## $ TEAM_FIELDING_DP <dbl> NA, 155, 153, 156, 168, 149, 186, 136, 169, 159, 141,~
```

We can see above the training data set has:

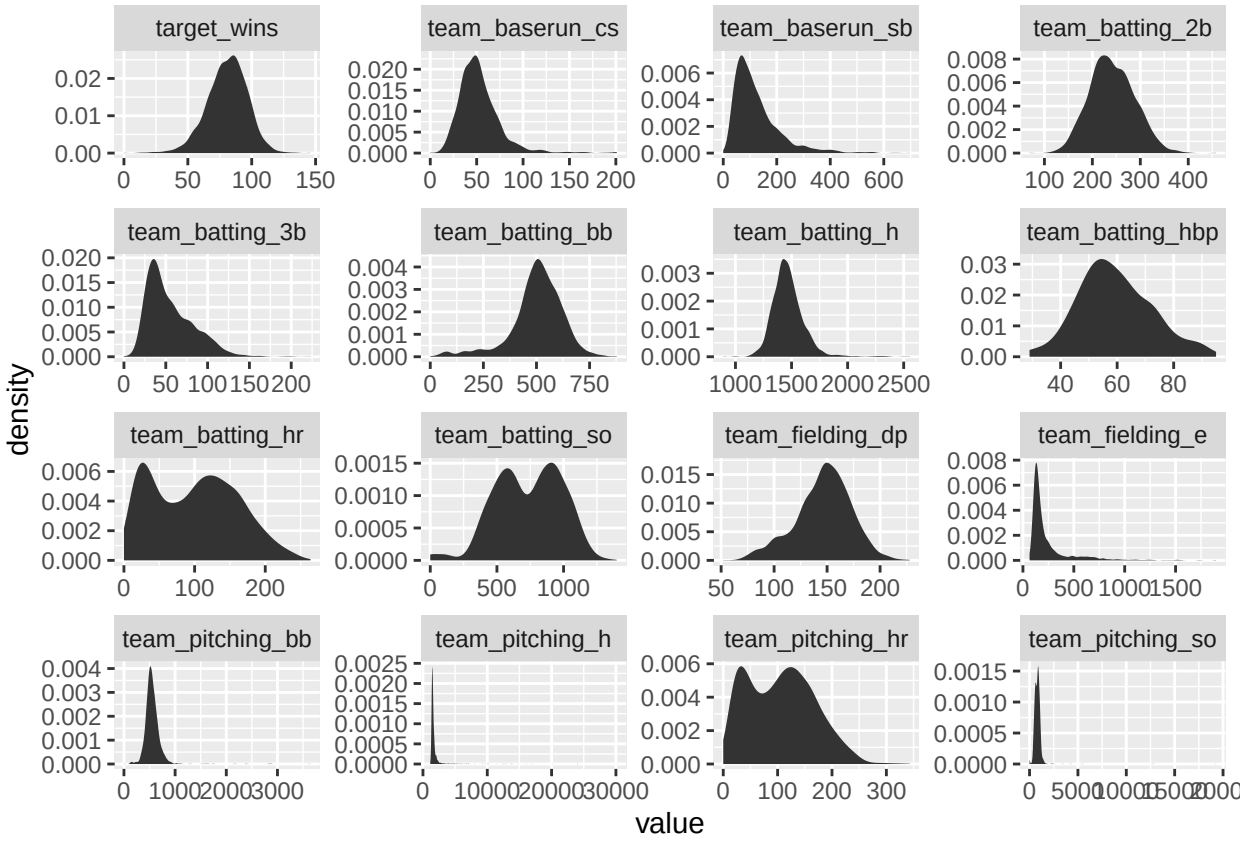
- **17** features, all of which are of data type integer.
- **2276** observations.
- Features with **NA** values that are candidates for imputation or exclusion (ie. team_batting_hbp, team_baserun_cs).
- An ***index*** feature which may also be excluded.

Per the final two notes above, we clean the original data before moving on to EDA:

- Converting columns to ***lower case*** for personal manageability.
- Dropping the ***index*** feature from the training data set since we will not be using it.

Variable Distributions

After taking a high level overview of the data, we can now investigate it in greater depth. We can seek anomalies, patterns, and otherwise visualize and analyze our data so that we might chart a better course when feature engineering and comparing models.



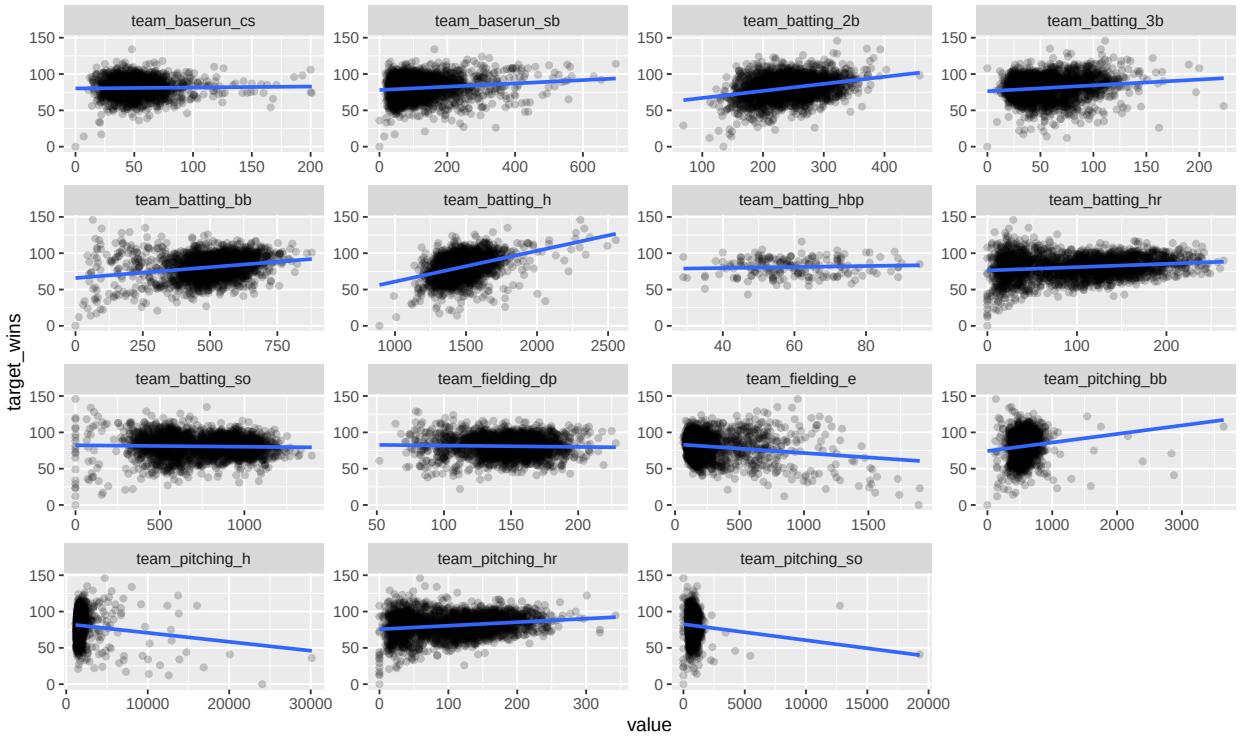
From the plots above, there are a number of points to highlight:

- target_wins, team_batting_h each appear to have a **normal distrution**,
- team_baserun_cs, team_baserun_sb, team_batting_3b each appear to be **right skewed normal**,
- team_batting_bb appears to be **left skewed normal**,
- team_batting_hr, team_batting_so, and team_pitching_hr appear to be **bimodal**,
- team_batting_hbp appears to be arguably **multimodal**,
- and remaining variables do not appear to have normal distributions. We may either deal with outliers / perform transformations to gain greater insight or discount these variables altogether from consideration from our model.

Variable Correlation

With target_wins as our dependent variable and an idea of each variable's distribution, as a next step we can visualize the correlation between our independent variables and our dependent variable :

```
## `geom_smooth()` using formula 'y ~ x'
```



From our regression visualizations above, we observe:

- team_baserun_cs and team_batting_hbp appear to be non-correlated and thus likely candidates for our exclusion from a model.
- a **strong positive correlation** between team_batting_h, team_pitching_bb and our dependent variable target_wins.
- a **positive correlation** between team_baserun_sb, team_batting_2b, team_batting_3b, team_batting_bb, team_batting_hr, and target_wins although some of these correlations aren't very strong and it'd be tough to judge based on these plots whether or not the variables are worth keeping in the model.
- a **strong negative correlation** between team_pitching_h, team_pitching_so and our dependent variable target_wins.
- a **negative correlation** between team_fielding_e and target_wins although it's tough to judge on this plot alone whether the variable is worth keeping in our model.
- **outliers**, that we'll likely need to take care of, affecting the fit line for team_pitching_bb, team_pitching_h, and team_pitching_so.

We've observed our data at a high level, familiarized ourselves with the variable distributions and correlations and are prepared, at this point, to tidy and transform our data.

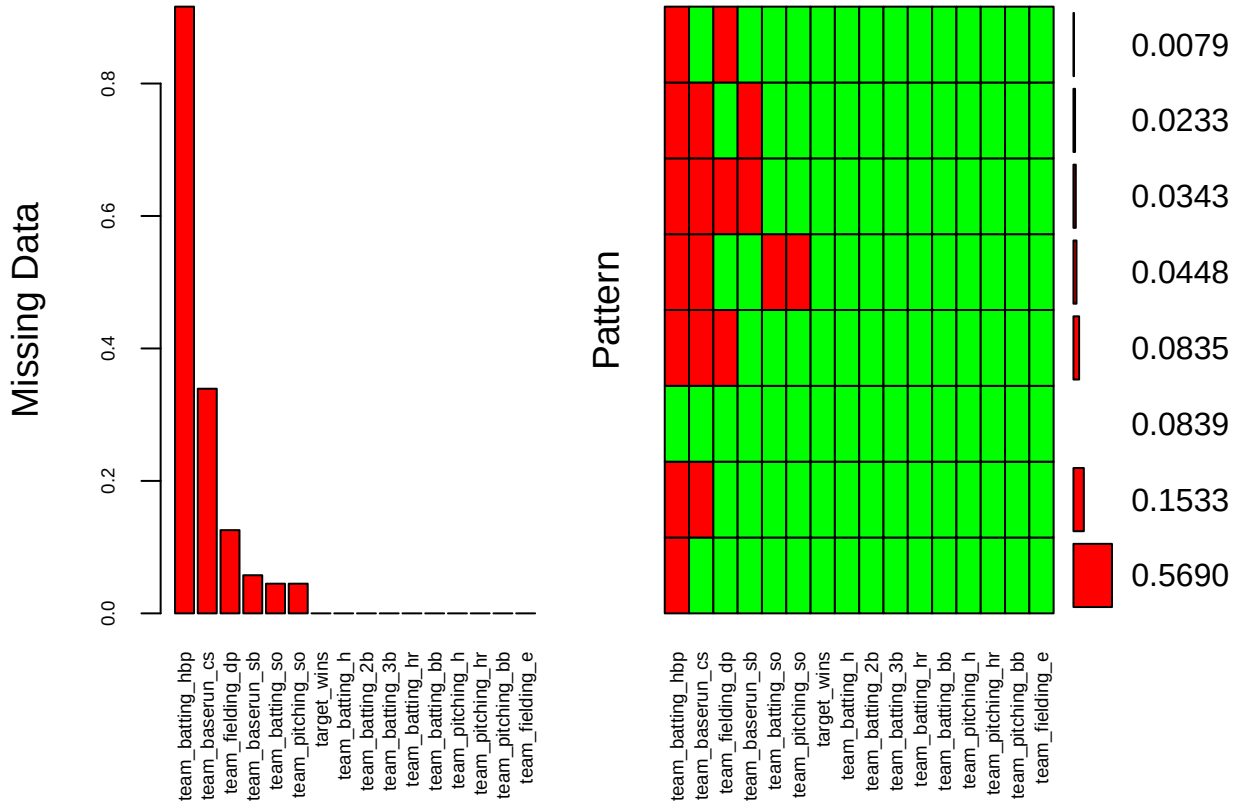
2. DATA PREPARATION

With insights gained via EDA (exploratory data analysis), we can now handle NA values, drop inessential variables, impute missing values, and otherwise transform our dataset into a form more favorable for analysis.

Handling NA Values

Being that we noticed NA values in our preliminary investigation of the dataset, we'll explore further to probe the level of severity and plot a proper response (imputation or exclusion):

```
##      target_wins   team_batting_h  team_batting_2b  team_batting_3b
##                0                0                0                0
##  team_batting_hr  team_batting_bb  team_batting_so  team_baserun_sb
##                0                0              102              131
##  team_baserun_cs team_batting_hbp  team_pitching_h team_pitching_hr
##              772             2085                0                0
## team_pitching_bb team_pitching_so  team_fielding_e team_fielding_dp
##                0              102                0              286
```




```
##
##  Variables sorted by number of missings:
##          Variable      Count
##  team_batting_hbp 0.91608084
##   team_baserun_cs 0.33919156
##  team_fielding_dp 0.12565905
##   team_baserun_sb 0.05755712
##   team_batting_so 0.04481547
##  team_pitching_so 0.04481547
##       target_wins 0.00000000
##    team_batting_h 0.00000000
##   team_batting_2b 0.00000000
##   team_batting_3b 0.00000000
##   team_batting_hr 0.00000000
##   team_batting_bb 0.00000000
##   team_pitching_h 0.00000000
##  team_pitching_hr 0.00000000
```

```
##  team_pitching_bb 0.00000000
##   team_fielding_e 0.00000000
```

team_batting_hbp and team_baserun_cs raise red flags due to their high NA counts of 2085/2276 (~92%) and 772/2276 (~34%) respectively. We'll exclude team_batting_hbp from consideration for the simple fact that the majority of its data is missing and we'll carry team_baserun_cs to see if imputing values proves to be an effective remedy.

Remaining variables do not have nearly the same magnitude of NA values and thus imputation appears to be a better route than exclusion.

Feature Exclusions

We had previously identified that ***index*** and ***team_batting_hbp*** would be excluded. ***index*** because it was a non-valuable variable and ***team_batting_hbp*** because more than 90% of its data was missing. Being that ***index*** already was during data pre-processing, we exclude ***team_batting_hbp*** from our training and test sets below:

Prior to excluding the bi-modal distributions identified earlier (**team_batting_hr**, **team_batting_so**, and **team_pitching_hr**), we verify their predictive power.

We apply a 'kitchen sink' multinomial linear regression model, accounting for all independent variables, and interpret associated p-values to judge predictive capability:

```
##
## Call:
## lm(formula = target_wins ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -30.5627  -6.6932  -0.1328   6.5249  27.8525
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      57.912438   6.642839   8.718  < 2e-16 ***
## team_batting_h    0.015434   0.019626   0.786   0.4318
## team_batting_2b  -0.070472   0.009369  -7.522 9.36e-14 ***
## team_batting_3b   0.161551   0.022192   7.280 5.43e-13 ***
## team_batting_hr   0.073952   0.085392   0.866   0.3866
## team_batting_bb   0.043765   0.046454   0.942   0.3463
## team_batting_so   0.018250   0.023463   0.778   0.4368
## team_baserun_sb   0.035880   0.008687   4.130 3.83e-05 ***
## team_baserun_cs   0.052124   0.018227   2.860   0.0043 **
## team_pitching_h   0.019044   0.018381   1.036   0.3003
## team_pitching_hr  0.022997   0.082092   0.280   0.7794
## team_pitching_bb -0.004180   0.044692  -0.094   0.9255
## team_pitching_so -0.038176   0.022447  -1.701   0.0892 .
## team_fielding_e  -0.155876   0.009946 -15.672  < 2e-16 ***
## team_fielding_dp -0.112885   0.013137  -8.593  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.556 on 1471 degrees of freedom
##   (790 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.4386, Adjusted R-squared:  0.4333
## F-statistic:  82.1 on 14 and 1471 DF,  p-value: < 2.2e-16
```

Based on the high p-values we see above for our bi-modal distributions (**team_batting_hr**, **team_batting_so**, and **team_pitching_hr**), it appears that these features do not play a significant role in predicting team wins and thus there's reason for exclusion. With these variables in mind, we'll move on to imputation and dealing with multicollinearity to confirm the effect eliminating one variable may have on the performance of our model vs. eliminating them all at once.

Imputation

From excluding variables and simplifying the breadth of our models consideration, we move on to accounting for missing values.

From the mice library, we impute using the **pmm** method (predictive mean matching):

Predictive mean matching calculates the predicted value for our target variable, and, for missing values, forms a small set of "candidate donors" from the complete cases that are closest to the predicted value for our missing entry. Donors are then randomly chosen from candidates and imputed where values were once missing. *To apply pmm we assume that the distribution is the same for missing cells as it is for observed data, and thus, the approach may be more limited when the % of missing values is higher.*

Once we've imputed missing values into our training dataset and returned the data in proper form, we verify whether our operation was successful:

```
##      target_wins   team_batting_h  team_batting_2b  team_batting_3b
##                0                0                0                0
##  team_batting_hr  team_batting_bb  team_batting_so  team_baserun_sb
##                0                0                0                0
##  team_baserun_cs  team_pitching_h team_pitching_hr team_pitching_bb
##                0                0                0                0
## team_pitching_so  team_fielding_e team_fielding_dp
##                0                0                0
```

The presence of all 0's above confirms the success of imputation.

Dealing with Multicollinearity

For a given predictor, multicollinearity can be measured by computing the associated variance inflation factor (VIF). The VIF measures the inflation of our regression coefficient due to multicollinearity.

The smallest possible value of VIF is 1 (no multicollinearity) whereas values in excess of 5 or 10 are indicative that we may have a problem. When faced with multicollinearity, the problematic variables should be removed since they are redundant in the presence of the other variables in our model.

To deal with multicollinearity, we check VIF values:

```
##   team_batting_h  team_batting_2b  team_batting_3b  team_batting_hr
##         3.861840         2.474136         3.050518        36.774802
##  team_batting_bb  team_batting_so  team_baserun_sb  team_baserun_cs
##         6.736903         5.384670         4.107918         4.024636
##  team_pitching_h team_pitching_hr team_pitching_bb team_pitching_so
##         4.127498        29.570742         6.340675         3.324785
##  team_fielding_e team_fielding_dp
##         5.291335         1.975921
```

There's quite a bit of multicollinearity present in our model as indicated by the five variables with a VIF value in excess of 5 and the two variables in far excess of 10.

In order to fix the dependency between variables we remove the variable with the highest VIF score, **team_batting_hr**, and revisit our VIF scores again to verify that multicollinearity has been addressed:

```
##   team_batting_h  team_batting_2b  team_batting_3b  team_batting_bb
##        3.851784        2.473524        2.927524        5.330006
##  team_batting_so  team_baserun_sb  team_baserun_cs  team_pitching_h
##        5.351003        4.105937        4.023803        4.111405
## team_pitching_hr team_pitching_bb team_pitching_so  team_fielding_e
##        3.743155        4.896353        3.076201        5.289992
## team_fielding_dp
##        1.970173
```

Multi-collinearity appears to be present to a much lesser degree after our exclusion of the aforementioned variable. There are now only three variables whose VIF scores exceed 5 and they do so to a very minor degree. Thus, we'll carry all variables forward into the model-building phase.

3. MODEL BUILDING & SELECTION

We started with 16 independent variables, excluded **index** (irrelevant), **team_batting_hbp** (high NA %), and **team_batting_hr** (multicollinearity) to widdle variables still within consideration down to 13.

At this point we've explored and prepared our data and are ready to build and assess different models. We'll take an approach of incremental improvement (or kaizen):

- **Model 1** (l.model): the 'kitchen sink' model that accounts for all 13 variables,
- **Model 2** (l.model2): the removal of apparent outliers, and
- **Model 3** (l.model3): the removal of impertinent features.

Model 1

Let's start by applying regression to all variables still within consideration:

```
##
## Call:
## lm(formula = target_wins ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.478  -8.407   0.217   8.179  55.310
##
## Coefficients:
##                    Estimate Std. Error t value         Pr(>|t|)
## (Intercept)      36.0665258  5.1314727   7.028 0.00000000000275 ***
## team_batting_h    0.0433043  0.0035755  12.111          < 2e-16 ***
## team_batting_2b  -0.0197764  0.0088521  -2.234         0.025574 *
## team_batting_3b   0.0235020  0.0161321   1.457         0.145298
## team_batting_bb   0.0176715  0.0049576   3.565         0.000372 ***
## team_batting_so  -0.0158466  0.0024751  -6.402 0.00000000018546 ***
## team_baserun_sb   0.0527118  0.0053120   9.923          < 2e-16 ***
```

```
## team_baserun_cs  -0.0050001  0.0109418  -0.457         0.647733
## team_pitching_h   0.0015449  0.0003797   4.069 0.00004878476115 ***
## team_pitching_hr  0.0719202  0.0083140   8.650          < 2e-16 ***
## team_pitching_bb -0.0062891  0.0035038  -1.795         0.072795 .
## team_pitching_so  0.0023424  0.0008524   2.748         0.006042 **
## team_fielding_e  -0.0425099  0.0026600 -15.981          < 2e-16 ***
## team_fielding_dp -0.1253568  0.0127340  -9.844          < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.56 on 2262 degrees of freedom
## Multiple R-squared:  0.3674, Adjusted R-squared:  0.3638
## F-statistic: 101.1 on 13 and 2262 DF,  p-value: < 2.2e-16
```

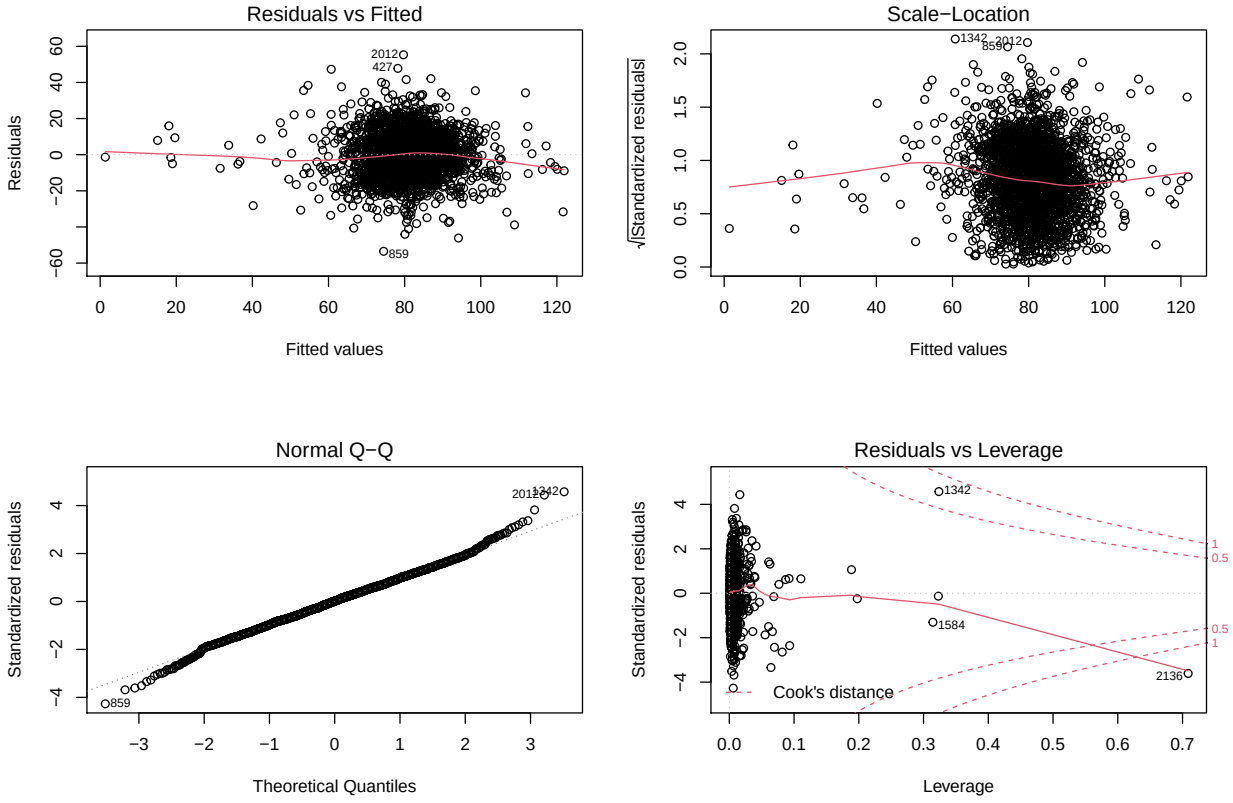
We can interpret the coefficients of our model as follows:

- **positive coefficients**: team_batting_h, team_batting_3b, team_batting_bb, team_baserun_sb, and team_pitching_so make sense in the context of a team having a winning record. We can imagine that hitting the ball, getting walked, stealing a base, and striking out the opposing team would all positively impact a team's chance in winning the game. On the flip-side, the positive coefficients for team_pitching_h (hits against) and team_pitching_hr (home runs against) don't appear to make sense because we wouldn't want the other team to get on base or hit a home run. These coefficients raise red flags.
- **negative coefficients**: negative coefficients for team_batting_so, team_baserun_cs, team_pitching_bb, team_fielding_e make sense. We can imagine being struck out, caught stealing, walking the opposing team, and making fielding errors adversely effecting a team's ability to win. The negative coefficients for team_batting_2b (getting to 2nd base) and team_fielding_dp (getting a double play) don't make sense, since we'd expect getting further on base and making an exceptional play to positively rather than negatively impact a team's ability to win.

Moving on to the summary statistics relevant to model assessment:

- **p-values**: while the majority of variables under consideration do appear to play a significant role in predicting wins, the relatively high p-values associated with team_baserun_cs, team_batting_3b, and team_pitching_bb are indicative that these variables may be excluded,
- **RSE**: being that a lower value is typically better, our RSE value appears to be OK. The residual standard error provides indication of how far (on average) our residuals are from the fit line and we'll keep an eye on this metric as we tune the model.
- **F-statistic**: the f-statistic provides a measure of variability between groups over variability within each group. Typically, a higher value is stronger evidence in favor of a model's efficacy and our value appears to be promising.
- **Adj. R^2**: typically values near 1 are indicative of a strong model. Our low value is concerning and indicative of a weak model. With this said, each dataset has nuance and the adjusted R squared does not provide any sort of final decision on the validity of a model. *Thus, visualizations and all model statistics (in conjunction) are to be considered when tuning a model.*

Let's turn our eye to the diagnostics for our 1st model:



We can interpret the residual plots as follows:

- **residuals vs. fitted**: our data are not equally dispersed but there doesn't appear to be an unaccounted for non-linear pattern to our data. The line is relatively straight and influential outliers are marked (427, 859, 2012).
- **scale-location**: our data are not equally dispersed and thus there may be reason to hold concern regarding homoscedasticity but the line is relatively horizontal. The slight curvature might improve if we deal with the influential outliers (859, 1342, 2012)
- **normal Q-Q**: with influential outliers marked (859, 1342, 2012) and our data ing a relatively straight line, our normal Q-Q plot is promising.
- **residuals vs. leverage**: this plot helps identify influential cases (1242, 1584, 2136) which may alter the trend of our regression line. Dealing with these cases, and others like them, may improve our fit.

From the summary statistics and residuals plots of model 1, we gather that the next step for improving our model would be to deal with influential outliers.

Model 2

For model 2, we'll see if we removing these leverage points improves our model:

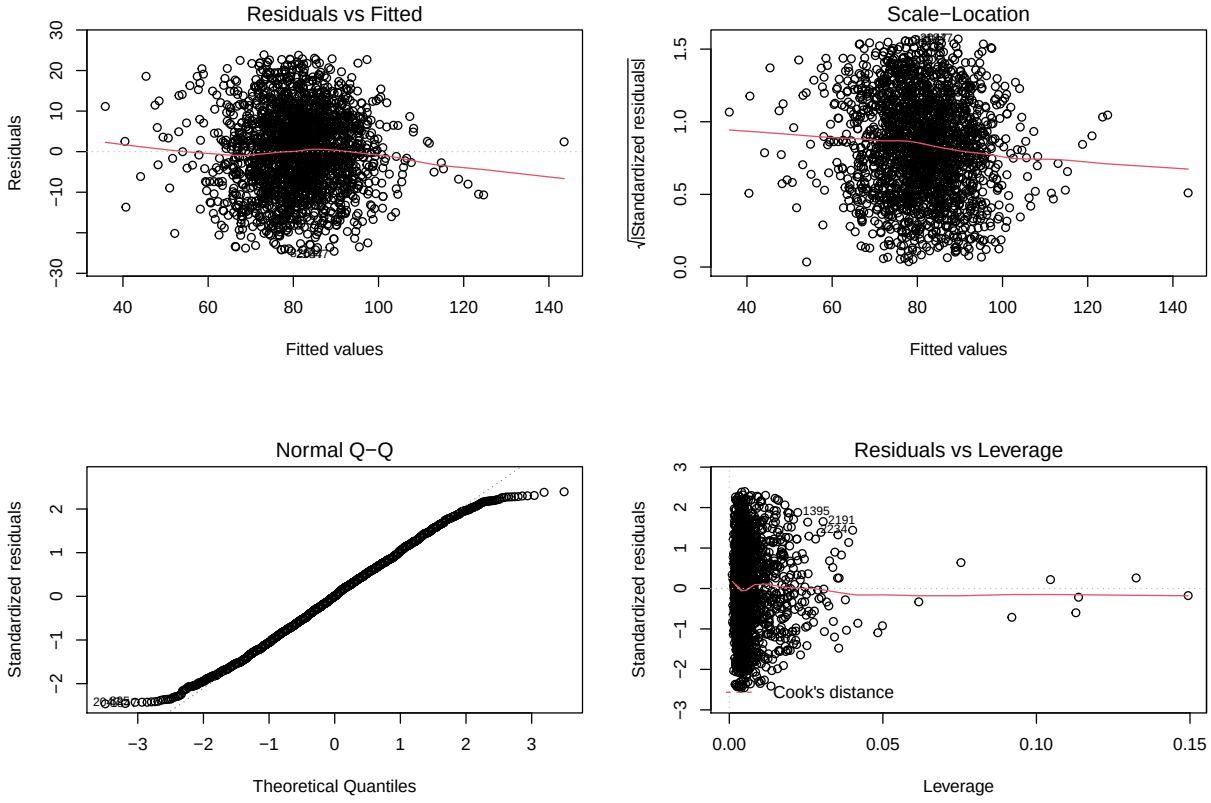
```
## [1] 2108   14
```

```
##
## Call:
## lm(formula = target_wins ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.5720  -7.1206   0.0283   6.9831  23.8858
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      46.643648   5.040204   9.254  < 2e-16 ***
## team_batting_h   -0.007290   0.004540  -1.606    0.108
## team_batting_2b  -0.044492   0.007923  -5.616 2.22e-08 ***
## team_batting_3b   0.147625   0.015780   9.355  < 2e-16 ***
## team_batting_bb   0.248073   0.013513  18.358  < 2e-16 ***
## team_batting_so  -0.068881   0.006749 -10.207  < 2e-16 ***
## team_baserun_sb   0.073737   0.004930  14.958  < 2e-16 ***
## team_baserun_cs  -0.007810   0.009507  -0.822    0.411
## team_pitching_h   0.037541   0.002102  17.858  < 2e-16 ***
## team_pitching_hr  0.090095   0.007685  11.724  < 2e-16 ***
## team_pitching_bb -0.200263   0.011754 -17.038  < 2e-16 ***
## team_pitching_so  0.047830   0.006160   7.764 1.28e-14 ***
## team_fielding_e  -0.081129   0.003429 -23.660  < 2e-16 ***
## team_fielding_dp -0.123805   0.010778 -11.487  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.998 on 2094 degrees of freedom
## Multiple R-squared:  0.4797, Adjusted R-squared:  0.4765
## F-statistic: 148.5 on 13 and 2094 DF,  p-value: < 2.2e-16
```

Being that an in-depth analysis, as well as an explanation of how to interpret the coefficients as well as each statistic, was provided for model 1, here we'll highlight the changes in value from model 1 to model 2:

- **RSE**: improvement from 12.56 to 9.998.
- **F-statistic**: improvement from 101.1 to 148.5.
- **Adj. R^2**: improvement from 0.3638 to 0.4765.

For further insight, let's visit the diagnostic plots for model 2:



There's a noticeable improvement in curvature across our **residuals vs. fitted**, **scale-location**, and **normal Q-Q** plots. The curve in the red line on these plots is less pronounced, which is an indicator of model improvement. Additionally, when we look at our **residuals vs. leverage** plot, we note that Cook's distance lines (the red dashed lines) have receded toward the edges of our plot. This is also a positive indicator.

Based on our summary statistics and residual plots, removing the aforementioned leverage points improved our model.

Model 3

As a next step, we can deal with the non-pertinent variables we've carried thus far. These variables are distinguishable in model 1 and model 2 by their high p-values.

We'll remove the least significant feature (team_baserun_cs) to see the impact it has on improving our model:

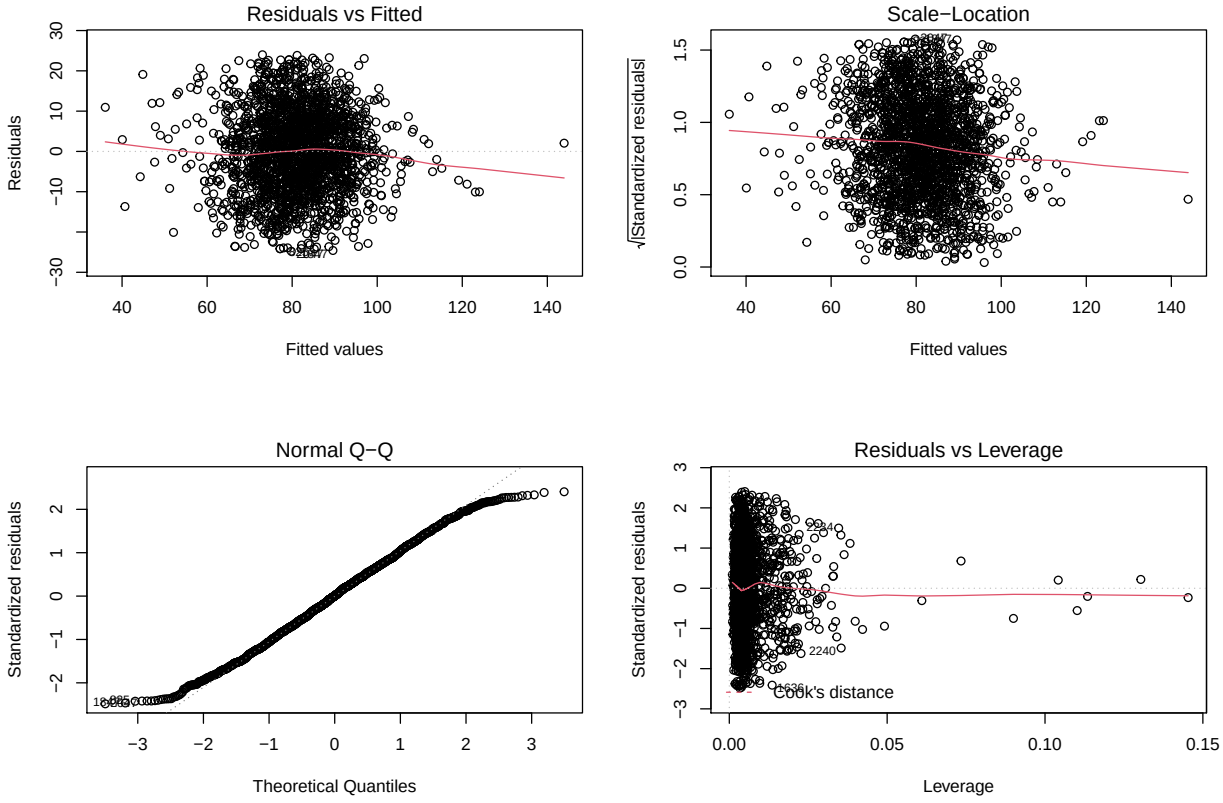
```
##
## Call:
## lm(formula = target_wins ~ team_batting_h + team_batting_2b +
##     team_batting_3b + team_batting_bb + team_batting_so + team_baserun_sb +
##     team_pitching_so + team_pitching_h + team_pitching_hr + team_pitching_bb +
##     team_fielding_e + team_fielding_dp, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.7909  -7.0989   0.0746   6.9930  23.9867
```

```
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      46.106661   4.997256   9.226  < 2e-16 ***
## team_batting_h   -0.007440   0.004536  -1.640    0.101
## team_batting_2b  -0.043877   0.007887  -5.563 2.98e-08 ***
## team_batting_3b   0.145478   0.015561   9.349  < 2e-16 ***
## team_batting_bb   0.248012   0.013512  18.355  < 2e-16 ***
## team_batting_so  -0.068382   0.006721 -10.175  < 2e-16 ***
## team_baserun_sb   0.071448   0.004067  17.570  < 2e-16 ***
## team_pitching_so  0.047436   0.006141   7.724 1.73e-14 ***
## team_pitching_h   0.037667   0.002097  17.966  < 2e-16 ***
## team_pitching_hr  0.090668   0.007652  11.848  < 2e-16 ***
## team_pitching_bb -0.200145   0.011752 -17.030  < 2e-16 ***
## team_fielding_e  -0.081171   0.003428 -23.677  < 2e-16 ***
## team_fielding_dp -0.122987   0.010731 -11.461  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.997 on 2095 degrees of freedom
## Multiple R-squared:  0.4795, Adjusted R-squared:  0.4765
## F-statistic: 160.8 on 12 and 2095 DF,  p-value: < 2.2e-16
```

The changes in value from model 2 to model 3 are noted below:

- **p-values**: there's now only one variable above the 0.05 threshold. *Removing it had worsened our performance, so we kept it.*
- **RSE**: improvement from 9.998 to 9.997.
- **F-statistic**: improvement from 148.5 to 160.8
- **Adj. R^2**: no change

Let's visit the diagnostic plots for model 3 to see if there are any noteworthy changes:



From model 2 to model 3 there's no noticeable change in our residual plots. A net neutral.

With no real change in residual plots and positive indicators in our summary statistics, it appears that removing the aforementioned features from consideration improved our model.

Model Selection

Being that our model building process was one of continually fine-tuning the entry model, with model 2 being better than model 1 and model 3 being better than model 2, our choice in model is rather clear up to this point: **model 3**. Model 3 was our best performing model.

4. MODEL TESTING

Being that explanations of selection have been documented up to this point, we'll assess model 3's performance and predictive capabilities.

We'll first assess its performance on unseen data and compare this performance to that with seen data, we'll then explore which features carry the most influence in predicting team wins, and finally, we'll cast our predictions based on the evaluation dataset.

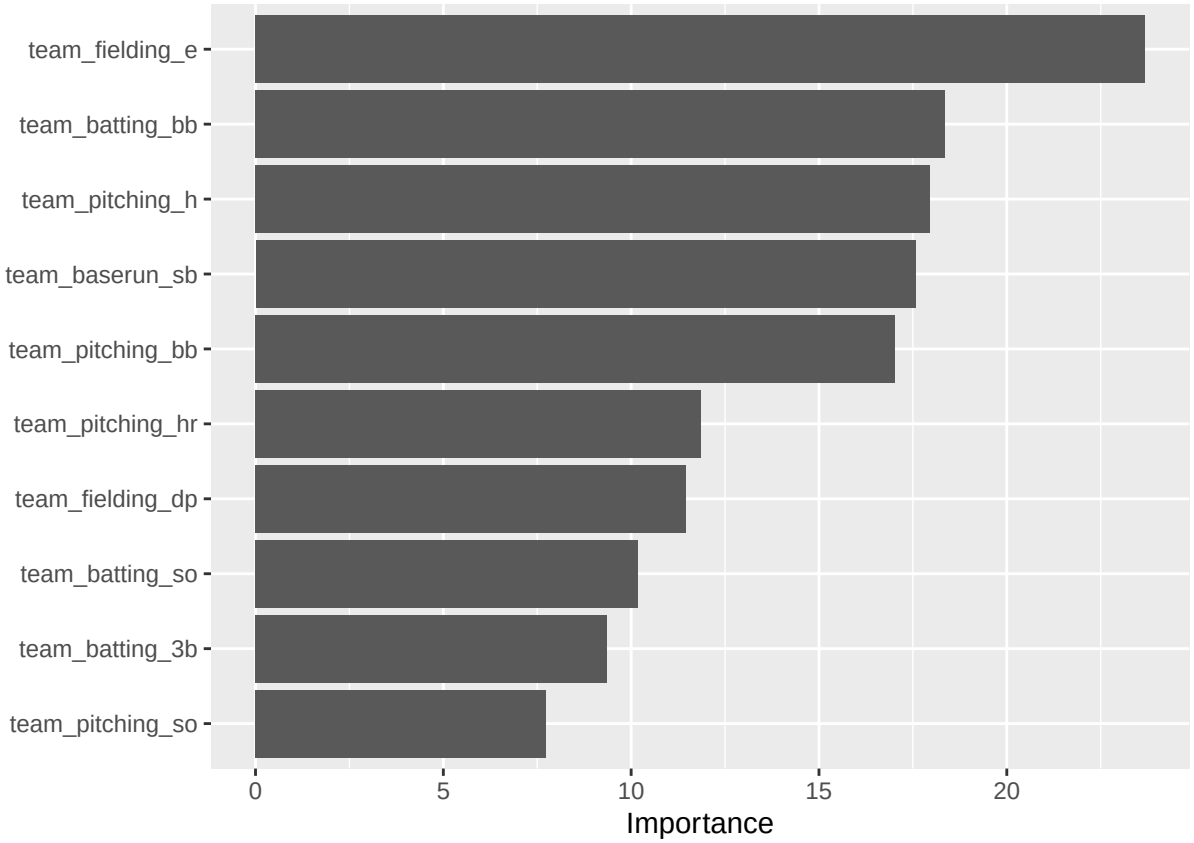
Model Assessment on Unseen Data

We perform a train-test split on the training data, with an 80/20 proportion, apply regression to this unseen data, and then verify the RMSE and R^2 metrics to verify their consistency with values seen earlier:

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard      10.0   Preprocessor1_Model1
## 2 rsq     standard       0.482 Preprocessor1_Model1
```

If you recall the performance metrics for seen data, these values are consistent. The model's R squared was actually slightly better for unseen data, which is curious.

Next, we utilize the vip() function on model 3 to gain insight into the weight each feature carries in predicting whether or not a team will win:



It appears that fielding errors, walks by batters, and hits allowed are most indicative of a team's ability to win a game. These results are somewhat suprising, especially fielding errors.

Predictions

As a final step, we input the provided test data to our model and cast predictions:

```
##     index target_wins
## 1       9          60
## 2      10          67
## 3      14          73
## 4      47          83
## 5      60         125
## 6      63          81
```

```
## 7      74          80
## 8      83          70
## 9      98          73
## 10    120          70
## 11    123          64
## 12    135          82
## 13    138          84
## 14    140          81
## 15    151          86
## 16    153          74
## 17    171          72
## 18    184          76
## 19    193          68
## 20    213          87
## 21    217          84
## 22    226          82
## 23    230          83
## 24    241          70
## 25    291          76
## 26    294          84
## 27    300          99
## 28    348          68
## 29    350          83
## 30    357          66
## 31    367          93
## 32    368          88
## 33    372          85
## 34    382          86
## 35    388          79
## 36    396          87
## 37    398          76
## 38    403          90
## 39    407          80
## 40    410          88
## 41    412          81
## 42    414          93
## 43    436          68
## 44    440         110
## 45    476          97
## 46    479          86
## 47    481          97
## 48    501          75
## 49    503          66
## 50    506          79
## 51    519          72
## 52    522          82
## 53    550          74
## 54    554          72
## 55    566          70
## 56    578          76
## 57    596          91
## 58    599          77
## 59    605          74
## 60    607          88
```

```
## 61    614          88
## 62    644          82
## 63    692          88
## 64    699          82
## 65    700          82
## 66    716         107
## 67    721          66
## 68    722          72
## 69    729          94
## 70    731          96
## 71    746          88
## 72    763          70
## 73    774          80
## 74    776          92
## 75    788          69
## 76    789          78
## 77    792          89
## 78    811          82
## 79    835          72
## 80    837          69
## 81    861          80
## 82    862          84
## 83    863          93
## 84    871          71
## 85    879          88
## 86    887          78
## 87    892          85
## 88    904          86
## 89    909          98
## 90    925          91
## 91    940          78
## 92    951         101
## 93    976          66
## 94    981          69
## 95    983          78
## 96    984          77
## 97    989          99
## 98    995         103
## 99   1000          89
## 100  1001          91
## 101  1007          78
## 102  1016          73
## 103  1027          82
## 104  1033          87
## 105  1070          75
## 106  1081          85
## 107  1084          55
## 108  1098          71
## 109  1150          88
## 110  1160          38
## 111  1169          91
## 112  1172          89
## 113  1174          96
## 114  1176          89
```

```
## 115  1178          77
## 116  1184          76
## 117  1193          84
## 118  1196          82
## 119  1199          66
## 120  1207          70
## 121  1218          98
## 122  1223          62
## 123  1226          63
## 124  1227          50
## 125  1229          70
## 126  1241          83
## 127  1244          86
## 128  1246          75
## 129  1248          92
## 130  1249          88
## 131  1253          82
## 132  1261          81
## 133  1305          76
## 134  1314          86
## 135  1323          89
## 136  1328          64
## 137  1353          76
## 138  1363          74
## 139  1371          93
## 140  1372          82
## 141  1389          57
## 142  1393          79
## 143  1421          92
## 144  1431          69
## 145  1437          75
## 146  1442          72
## 147  1450          76
## 148  1463          82
## 149  1464          78
## 150  1470          82
## 151  1471          81
## 152  1484          78
## 153  1495         747
## 154  1507          60
## 155  1514          78
## 156  1526          68
## 157  1549          90
## 158  1552          65
## 159  1556          86
## 160  1564          70
## 161  1585         108
## 162  1586         115
## 163  1590          98
## 164  1591         108
## 165  1592         103
## 166  1603          96
## 167  1612          85
## 168  1634          84
```

```
## 169  1645          70
## 170  1647          78
## 171  1673          91
## 172  1674          94
## 173  1687          81
## 174  1688          94
## 175  1700          79
## 176  1708          77
## 177  1713          84
## 178  1717          68
## 179  1721          74
## 180  1730          82
## 181  1737         100
## 182  1748          90
## 183  1749          87
## 184  1763          86
## 185  1768         178
## 186  1778          91
## 187  1780          74
## 188  1782          36
## 189  1784          54
## 190  1794         104
## 191  1803          71
## 192  1804          82
## 193  1819          68
## 194  1832          71
## 195  1833          69
## 196  1844          62
## 197  1847          74
## 198  1854          93
## 199  1855          84
## 200  1857          85
## 201  1864          72
## 202  1865          79
## 203  1869          75
## 204  1880          85
## 205  1881          81
## 206  1882          91
## 207  1894          75
## 208  1896          76
## 209  1916          76
## 210  1918          64
## 211  1921         107
## 212  1926          91
## 213  1938          88
## 214  1979          63
## 215  1982          68
## 216  1987          82
## 217  1997          76
## 218  2004          95
## 219  2011          74
## 220  2015          82
## 221  2022          75
## 222  2025          68
```

```
## 223  2027          77
## 224  2031          67
## 225  2036         155
## 226  2066          80
## 227  2073          78
## 228  2087          79
## 229  2092          84
## 230  2125          60
## 231  2148          79
## 232  2162          96
## 233  2191          84
## 234  2203          87
## 235  2218          81
## 236  2221          77
## 237  2225          79
## 238  2232          79
## 239  2267          88
## 240  2291          71
## 241  2299          88
## 242  2317          88
## 243  2318          82
## 244  2353          78
## 245  2403          62
## 246  2411          83
## 247  2415          76
## 248  2424          82
## 249  2441          72
## 250  2464          82
## 251  2465          77
## 252  2472          77
## 253  2481          89
## 254  2487         345
## 255  2500          70
## 256  2501          83
## 257  2520          83
## 258  2521          87
## 259  2525          68
```

Our model's predictions are noted above, with the provided team's index in one column and our model's predicted number of wins in the second column.

CONCLUSIONS

Summary

We have presented here an incremental approach to arrive at a multiple linear regression model of annual team wins using the provided feature variables. Our initial data exploration and preparation excluded two feature variables:

- Batter hit by pitch (`team_batting_hbp`) was excluded due to the majority (>90%) of the feature records having missing values.
- Homeruns by batter (`team_batting_hr`) was excluded due to multicolinearity (high VIF score)

After iteratively accounting for leverage points (Model 2) and feature impertinence (Model 3), one more feature variable was removed from model consideration:

- Caught stealing (`team_baserun_cs`)

As a result of this process, we arrive at a streamlined model that utilizes contributing feature variables to predict team wins. Furthermore, we have tested our model on novel data. For complete `R` code detailing our methods, please refer to the Appendix: `R` Statistical Code section.

Future Directions

A weakness of our model is it's construction using a relatively limited set of explanatory feature variables. Future endeavors to more accurately predict team wins would be aided by the addition of key variables used in other analyses. For instance, the 'Pythagorean formula for Baseball' introduced by renowned sabermetrician Bill James, predicts a teams winning percentage: $Win\ \% = RS\hat{}2\ /\ (RS\hat{}2 + RA\hat{}2)$

where RS = the runs scored by a team and RA = the runs allowed for a team.

The equation was adapted as a simple 'Linear Formula for Baseball' by Stanley Rothman using the following form: $Win\% = m\ x\ (RS\text{-}RA) + b$ where m and b are the coefficients of a linear regression.

Unfortunately, our data set does not include features that directly describe a given team's Runs Scored or Runs Allowed (`RS`, `RA` respectively). Furthermore, Rothman notes in his analysis that records that predate 1998 need to be separated from the analysis owing to the fact that it is only after 1998 that the league structure changed to a 30 team system with 162 annual games. In this analysis, we do not have access to the record dates and assume a 162 annual game structure when this is factually incorrect. In conclusion, our model's success could be vastly improved with the incorporation of additional information such as, but not limited to the record date, Runs Scored and Runs Allowed.

Appendix: `R` Statistical Code

DEPENDANCIES

```
library(dplyr)
library(ggplot2)
library(tidyr)
library(car)
library(VIM)
library(corrplot)
library(mice)
library(leaps)
library(tidyverse)
library(tidymodels)
library(vip)
```

IMPORTING DATA

```
test <- readr::read_csv('https://raw.githubusercontent.com/dataconsumer101/data621/main/hw1/moneyball-e
train <- readr::read_csv('https://raw.githubusercontent.com/dataconsumer101/data621/main/hw1/moneyball-
```

DATA EXPLORATION

```
#overview of data set
glimpse( train )

# remove row index field
train <- dplyr::select(train, -INDEX)

# work with lowercase field names
names(train) <- lapply(names(train), tolower)
names(test) <- lapply(names(test), tolower)

#a sample view of the training data after the clean up:
head(train)

#visualize feature distributions
ggplot(gather(train, variable, value), aes(x=value)) + stat_density() +
  facet_wrap(~variable, scales = "free")

# scatter plot all explanatory variables against response
tall <- gather(train, metric, value, -target_wins)

ggplot(tall, aes(x = value, y = target_wins)) +
  geom_point(alpha = .2) +
  geom_smooth(method = 'lm', se = FALSE) +
  facet_wrap(~metric, scales = 'free')
```

DATA PREPARATION

```
#handling NA values
#summarize missing data totals by feature
colSums(is.na(train))

#visualize missing data by feature as well as patterns of missing data across features
mice_plot <- VIM::aggr(train, col=c('green','red'),
                    numbers=TRUE, sortVars=TRUE, only.miss = TRUE,
                     cex.axis=.55,
                    gap=3, ylab=c("Missing Data","Pattern"))

#feature exclusions
#exclude team_batting_hbp from test & train
train <- dplyr::select(train, -c(team_batting_hbp))
test <-  dplyr::select(test, -c(team_batting_hbp))

#multiple lm w/all feature variables
l.model <- lm(data = train, target_wins ~ .)
summary(l.model)

#Imputation
#apply predictive mean matching to train
train <- mice(data = train, m = 1, method = "pmm", seed = 500)
```

```
train <- mice::complete(train, 1)
#apply predictive mean matching to test
test <- mice(data = test, m = 1,  method = "pmm", seed = 500)
test <- mice::complete(test, 1)

#verify absence of NA values in the dataset
colSums(is.na(train))

#Multicollinearity
#display VIF values for each feature variables
l.model <- lm(data = train, target_wins ~ .)
car::vif(l.model)

#remove the feature 'team_batting_hr' from train & test
train <- dplyr::select(train, -c(team_batting_hr))
test <-  dplyr::select(test, -c(team_batting_hr))

#display VIF values for feature variables
l.model <- lm(data = train, target_wins ~ .)
car::vif(l.model)
```

MODEL BUILDING & SELECTION

```
#display summary for Model 1: all inclusive (kitchen sink) model
summary(l.model)

#display Model 1 diagnostic plots
layout(matrix(c(1,2,3,4),2,2))
plot(l.model)

#remove leverage points from train data set
train <- train[c(-2136,-1342,-1584,-859,-2012,-1242,-1211,-1,-2233,-1210,-1083,-1826,-282,-420,-417,-13
#confirm that train has 6 fewer records
dim(train)

#refit multiple linear regression model on train
l.model2 <- lm(data = train, target_wins ~ .)
#display summary for Model 2: excluding high leverage points
summary(l.model2)

#display Model 2 diagnostic plots
plot(l.model2)

#refit model excluding impertinent feature variables
l.model3 <- lm(data = train, target_wins ~ team_batting_h + team_batting_2b + team_batting_3b + team_ba
#display summary for Model 3: excluding impertinent features
summary(l.model3)

#display Model 3 diagnostic plots
plot(l.model3)
```

Model Testing

```
#use tidymodel framework to preprocess the regression model
reg_recipe <-
    recipe(target_wins ~ ., data = train)

set.seed(123)

train_split <- initial_split(train, prop = 0.80)

reg_train <- training(train_split)

lm_model <- linear_reg() %>%
    set_engine('lm') %>%
    set_mode("regression")

reg_workflow <- workflow() %>%
  add_model(lm_model) %>%
  add_recipe(reg_recipe)

reg_fit <- reg_workflow %>%
  last_fit(split = train_split)

reg_fit %>% collect_metrics()

#vip() illustrate the relative importance of feature variables to Model 3
vip(l.model3)

#use the predict function to return predicted values on test data set
l.predict <-  round(predict(l.model3, newdata = test), 0)
#cast result as a dataframe with corresponding index
lm_predicted <- as.data.frame(cbind(test$index, l.predict))
#label with appropriate names
colnames(lm_predicted) <- c('index','target_wins')
#display the model prediction result for the test data
lm_predicted
```