Data 621 Homework 1: Moneyball

Tommy Jenkins, Violeta Stoyanova, Todd Weigel, Peter Kowalchuk, Eleanor R-Secoquian

September, 2019

1 OVERVIEW

In this homework assignment, we will explore, analyze and model a data set containing approximately 2200 records. 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 of the statistics adjusted to match the performance of a 162 game season.

1.1 Objective:

To build a multiple linear regression model on the training data to predict the number of wins for the team. We can only use the variables provided (or variables that we will derive from the variables provided).

2 DATA EXPLORATION

2.1 Data Summary

The dataset consists of two data files: training and evaluation. The training dataset contains 17 columns, while the evaluation dataset contains 16. The evaluation dataset is missing column TARGET_WINS. We will start by exploring the training data set since it will be the one used to generate the regression model.

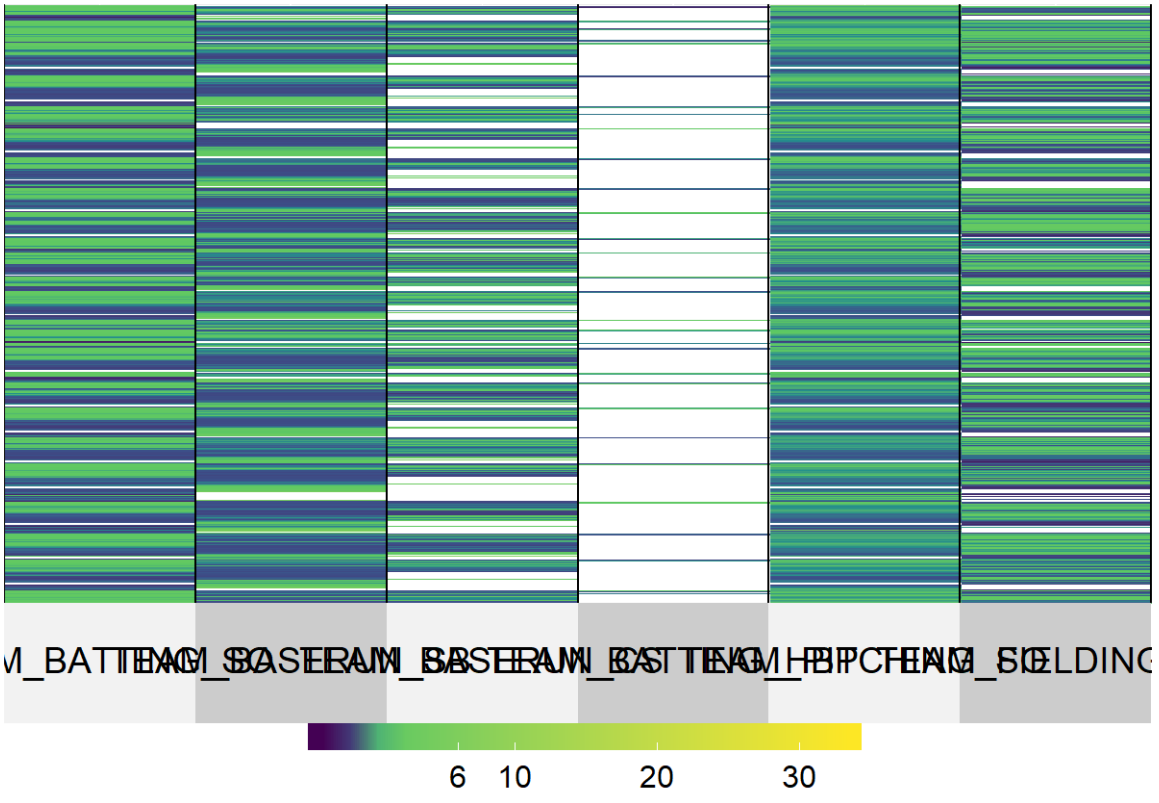
First we see that all data is numeric.

An important aspect of any dataset is to determine how much, if any, data is missing. We look at all the variables to see which if any have missing data. We look at the basic descriptive statistics as well as the missing data and their percentages:

	vars	n	mean	sd	median	trimmed	mad	min	max	range	
TARGET_WINS	2	191	80.92670	12.115013	82	81.11765	13.3434	43	116	73	-0.16
TEAM_BATTING_H	3	191	1478.62827	76.147869	1477	1477.42484	74.1300	1308	1667	359	0.13
TEAM_BATTING_2B	4	191	297.19895	26.329335	296	296.62745	25.2042	201	373	172	0.09
TEAM_BATTING_3B	5	191	30.74346	9.043878	29	30.13072	8.8956	12	61	49	0.70
TEAM_BATTING_HR	6	191	178.05236	32.413243	175	176.81046	35.5824	116	260	144	0.29
TEAM_BATTING_BB	7	191	543.31937	74.842133	535	541.31373	74.1300	365	775	410	0.31
TEAM_BATTING_SO	8	191	1051.02618	104.156382	1050	1046.95425	97.8516	805	1399	594	0.39
TEAM_BASERUN_SB	9	191	90.90576	29.916401	87	89.06536	29.6520	31	177	146	0.55

	vars	n	mean	sd	median	trimmed	mad	min	max	range	
TEAM_BASERUN_CS	10	191	39.94241	11.898334	38	39.49020	11.8608	12	74	62	0.34
TEAM_BATTING_HBP	11	191	59.35602	12.967123	58	58.86275	11.8608	29	95	66	0.31
TEAM_PITCHING_H	12	191	1479.70157	75.788625	1480	1478.50327	72.6474	1312	1667	355	0.12

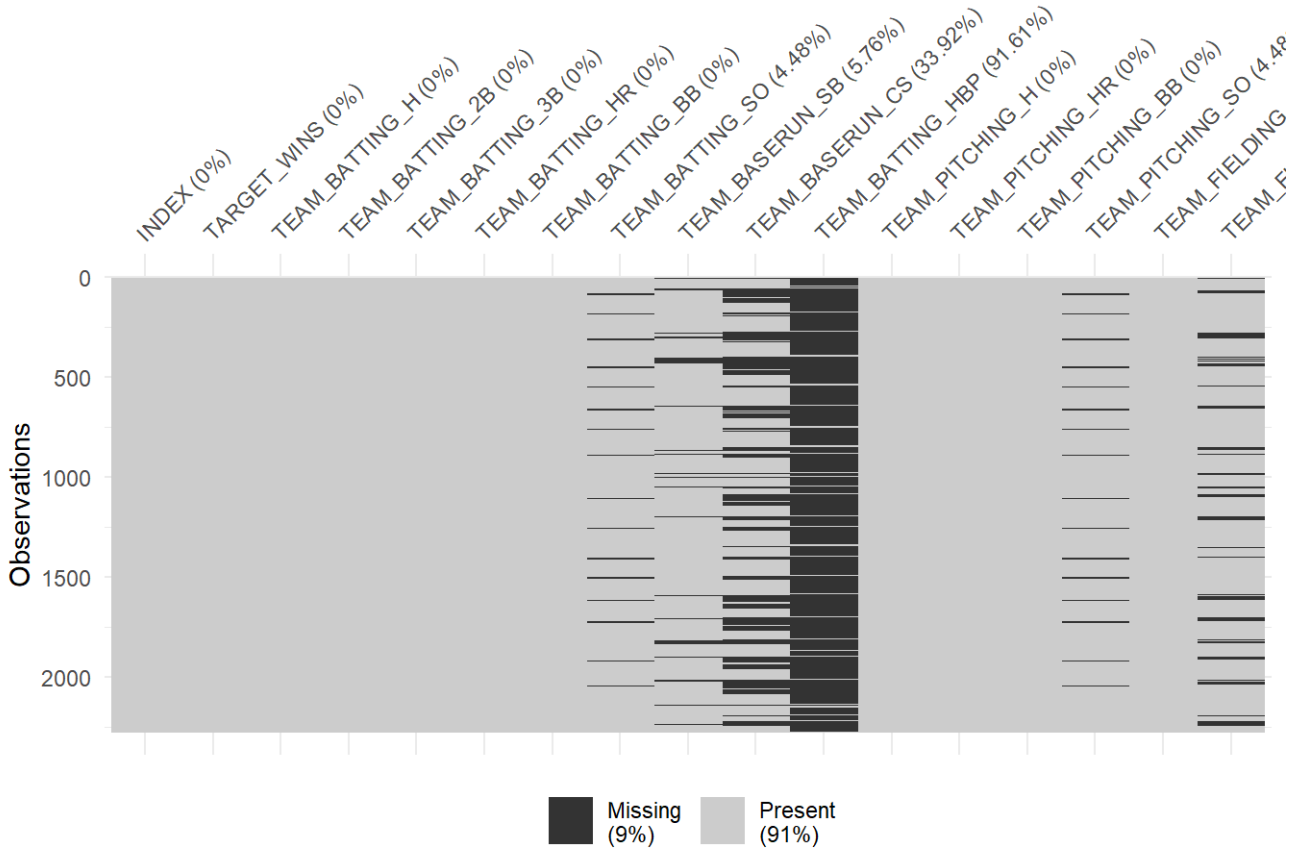
```
##             INDEX      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B
##               0.0              0.0              0.0              0.0
##  TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO
##               0.0              0.0              0.0              4.5
##  TEAM_BASERUN_SB  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H
##               5.8             33.9             91.6              0.0
## TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E
##               0.0              0.0              4.5              0.0
## TEAM_FIELDING_DP
##              12.6
```



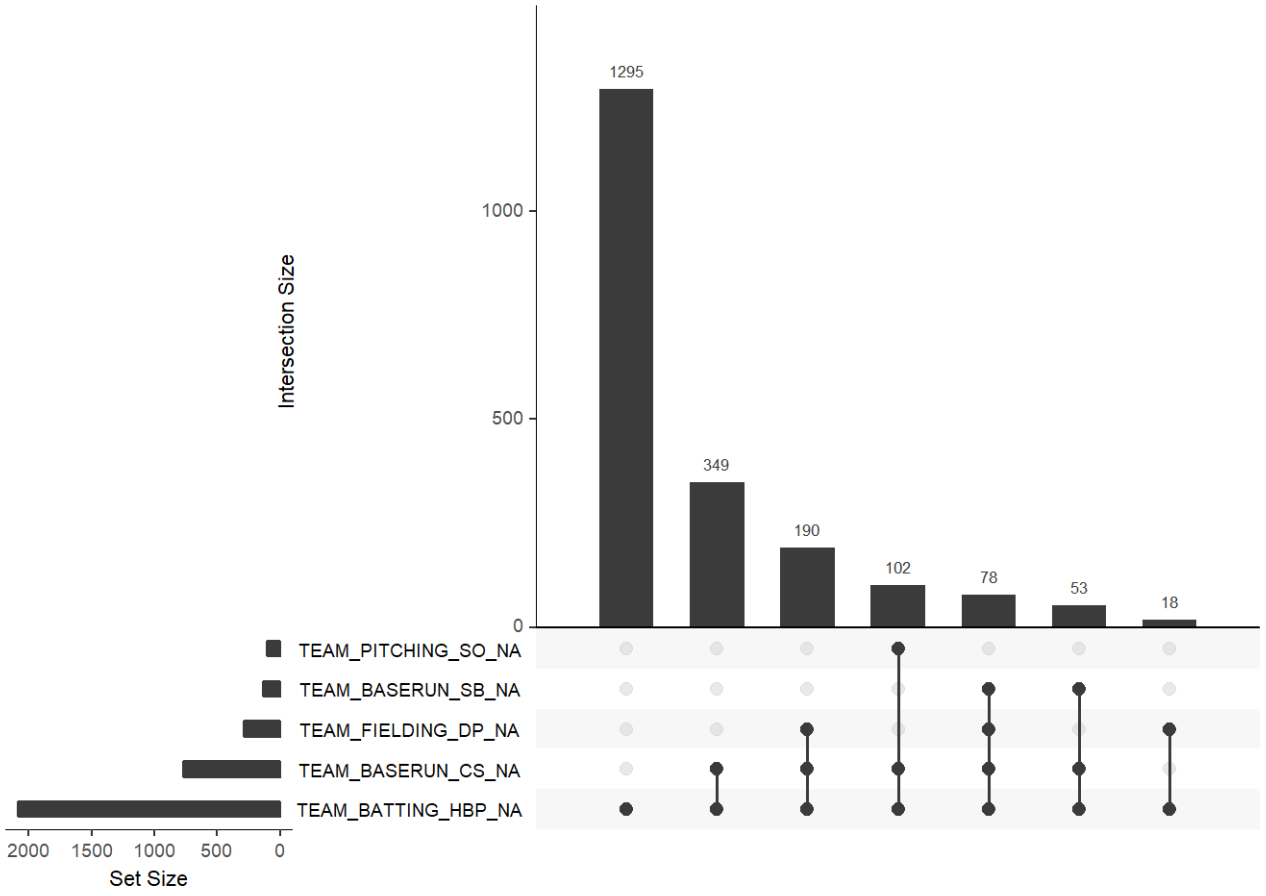

```
#install.packages("naniar")
library(naniar)
```

```
## Warning: package 'naniar' was built under R version 3.5.3
```

```
vis_miss(mbTrain)
```



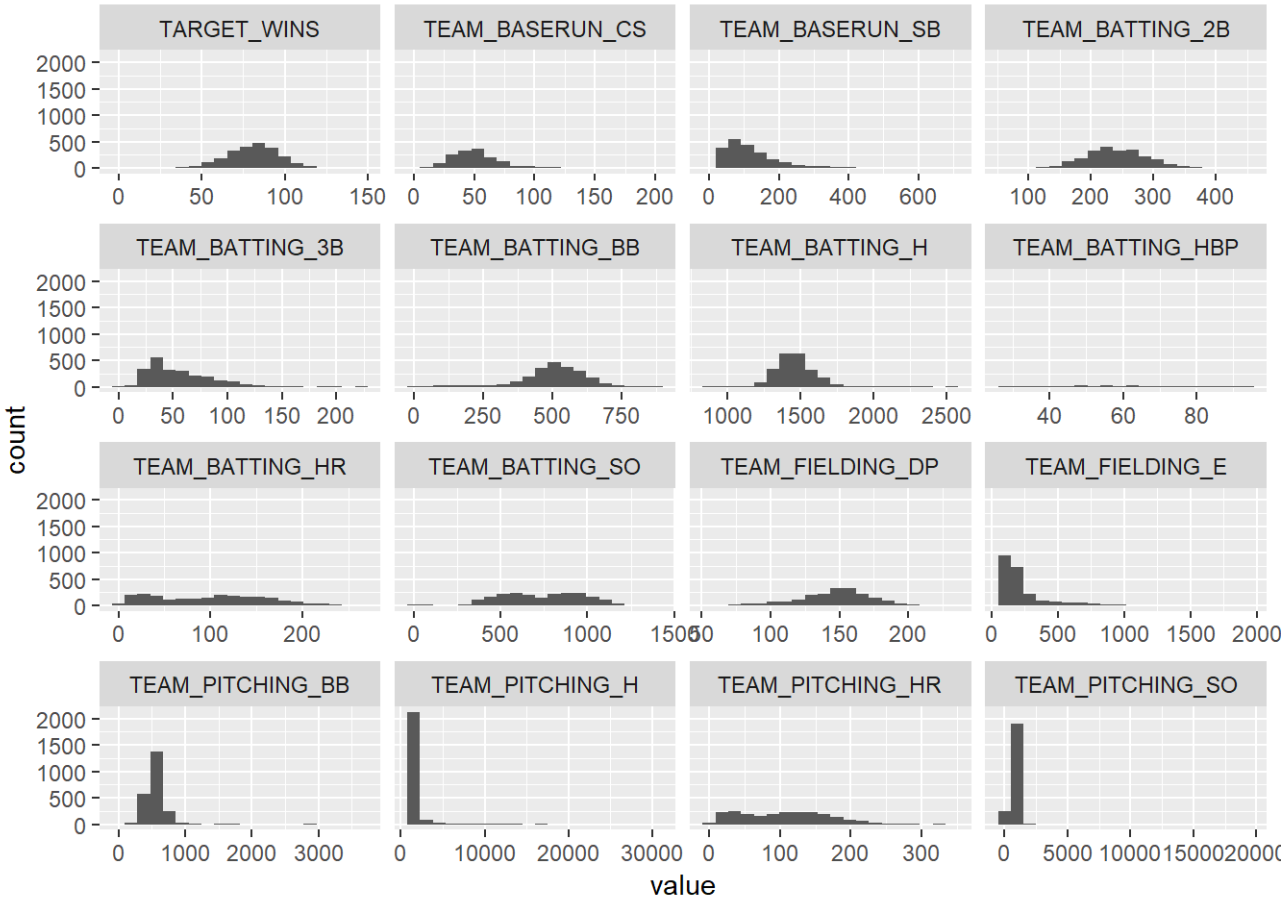

```
gg_miss_upset(mbTrain)
```



2.2 Missing and Invalid Data

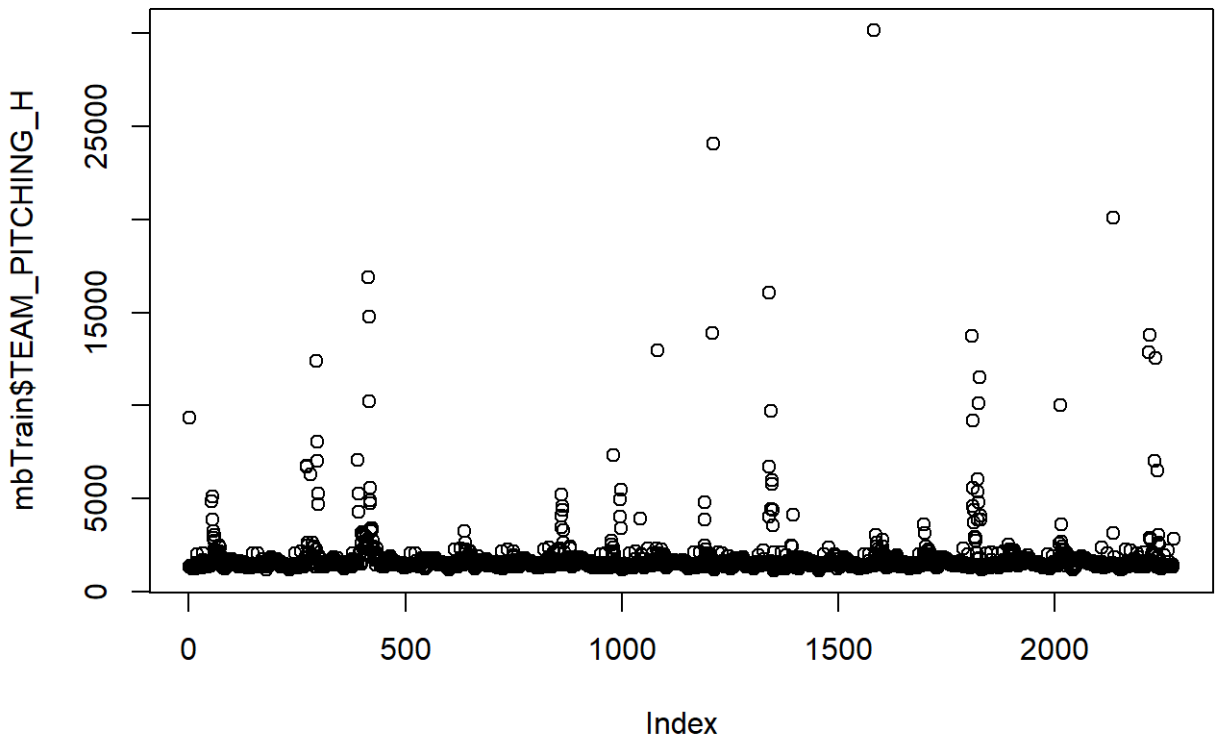
From this result we can see how several variables have a number of missing values. The maximum number of missing values was 2085 in the TEAM_BATTING_HBP variable. This is a significant amount of missing data representing 91.6% of that data.

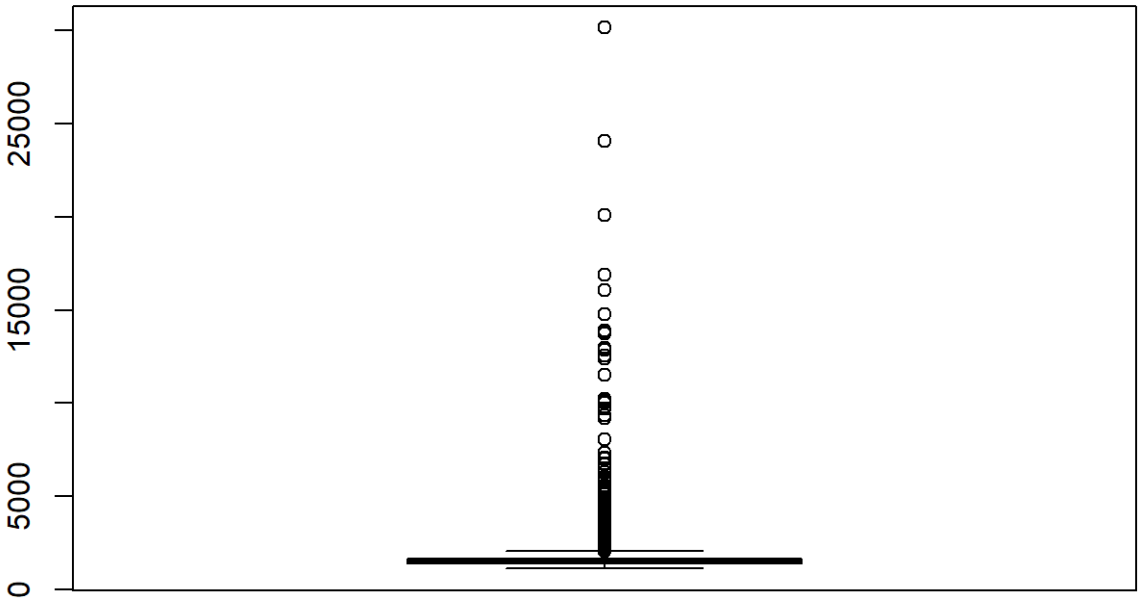
With missing data assessed, we can look into descriptive statistics in more detail. Interestingly we find that the difference between means and medians is fairly small for all data columns. The maximum difference is in fact only 4.77%. This means that we are to expect the distributions of this data to be fairly uniform. To visualize this we plot histograms for each data.



The plot of distributions does show fairly uniform data, but it also show the potential presence of outliers in at least two of the predictors. This is not the best way to vizualise ouliers. Instead we identify the predictors which seem to have outliers by looking at the scattered and box plots. Two variables with outliers appear to be TEAM_PITCHING_H, TEAM_PITCHING_SO, TEAM_PITCHING_BB and TEAM_FIELDING_E. We highlight these variables from the desity plots since we can see most of the data concentrated at the lower end of the scales which show tailing off to high values.

TEAM_PITCHING_H


mbTrain$TEAM_PITCHING_H
0
5000
15000
25000
0
500
1000
1500
2000
Index


0
5000
15000
25000

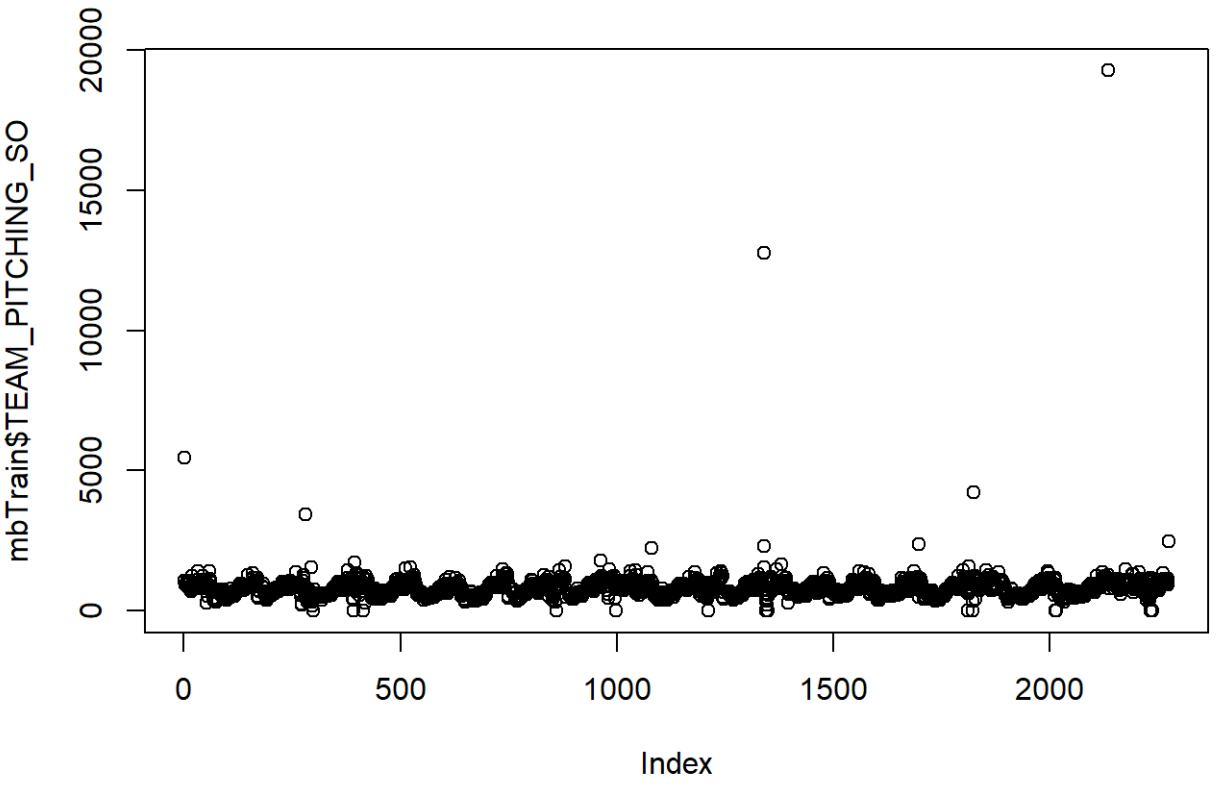
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1137    1419    1518    1779    1682   30132
```

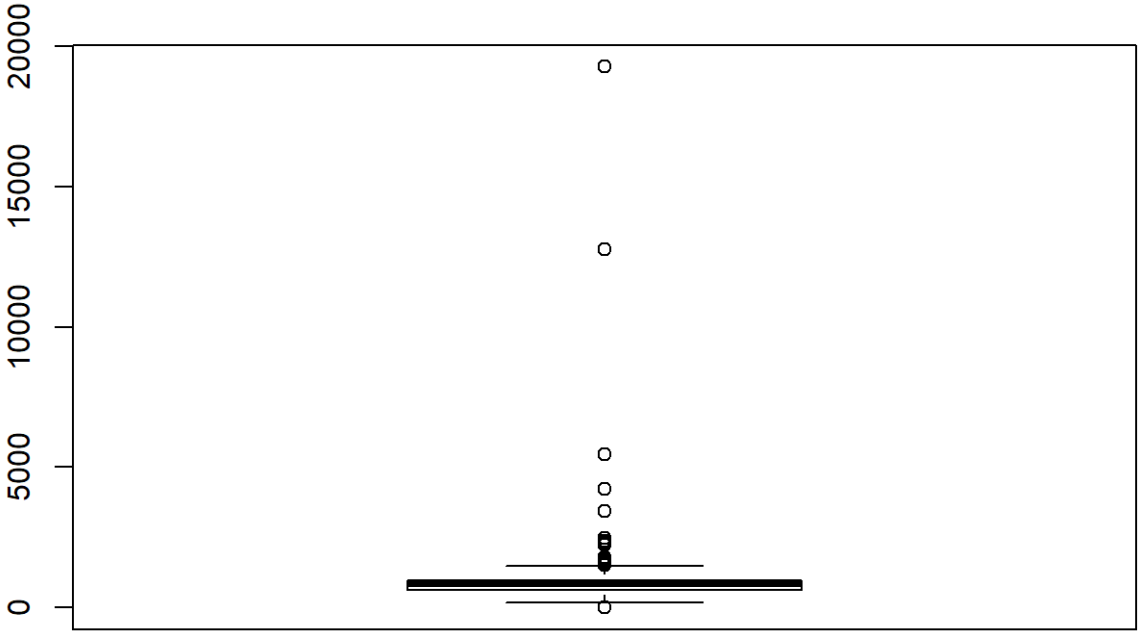
From the summary for this variable we see that the maximum is 3.013210^{4}, which is substantially far from the median of 1518. We can also see a wide spread between mean and median of 261. But we know the maximum is not a realistic number. We can compare the numbers of pitching hits, to the number of batting hits, summary for which is shown below:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     891    1383    1454    1469    1537    2554
```

For batting the maximum is 2554, much lower than 3.013210^{4}. We expect these two variables to somewhat equal values since one is the reciprocal of the other. As a sanity check of the distribution of the batting variable, we now see the mean and median much closer together with a spread of only 15.

TEAM_PITCHING_SO


mbTrain$TEAM_PITCHING_SO
0
5000
10000
15000
20000
0
500
1000
1500
2000
Index


0
5000
10000
15000
20000

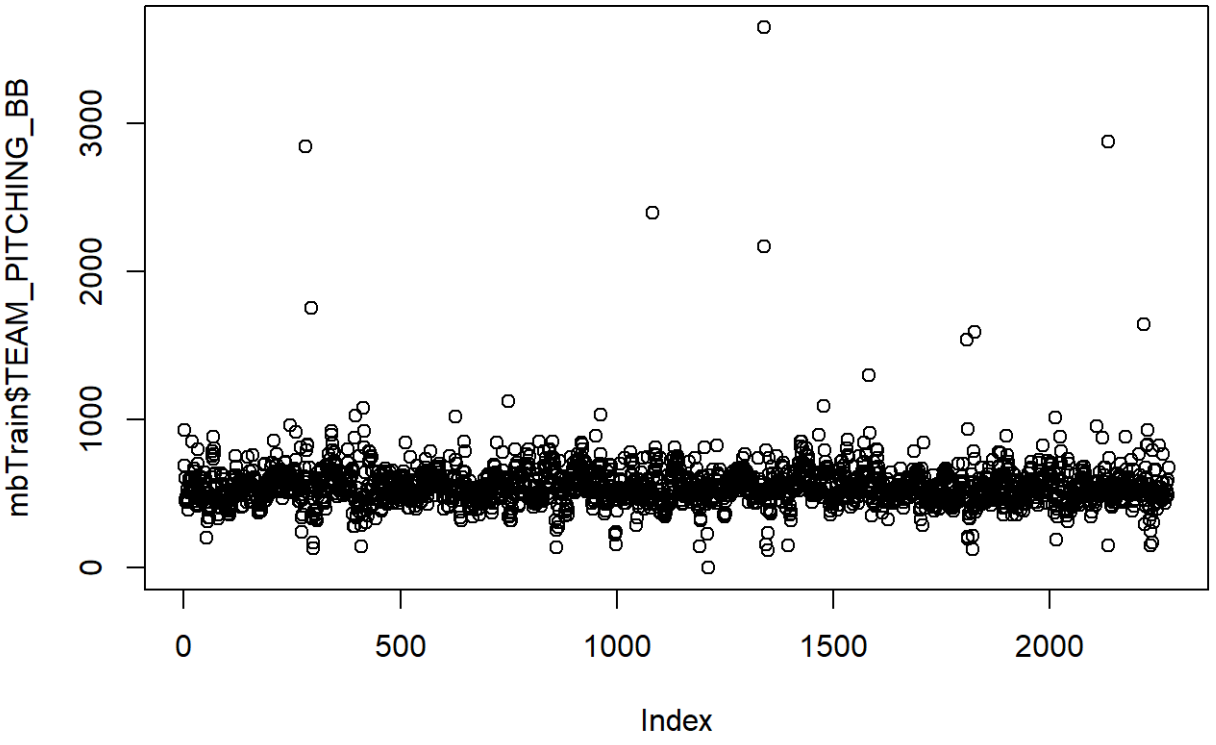
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0   615.0   813.5   817.7   968.0 19278.0     102
```

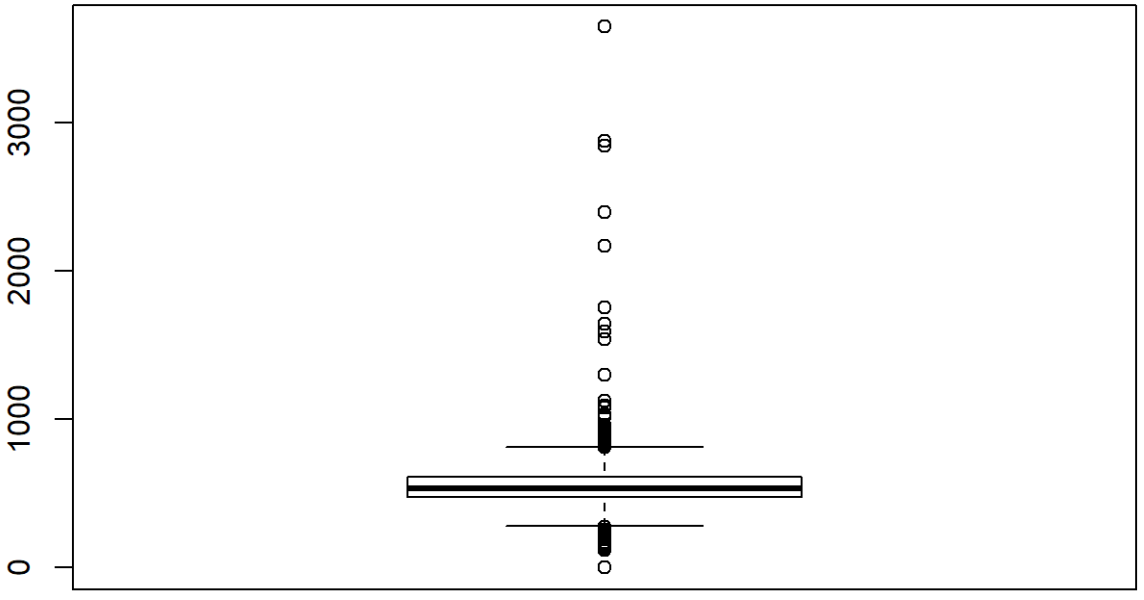
Again looking at the maximum we find it is unreasonably high at 1.927810^{4}, which again is substantially far from the median of 813.5. Again this variable has a reciprocal in TEAM_BATTING_SO. Comparing against it confirms the outliers.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0   548.0   750.0   735.6   930.0  1399.0     102
```

We see that or batting the maximum is 1399, much lower than 1.927810^{4}. As before as a sanity check of the distribution of the batting variable, we now see the mean and median much closer together with a spread of only 14.4.

TEAM_PITCHING_BB


mbTrain$TEAM_PITCHING_BB
0
1000
2000
3000
0
500
1000
1500
2000
Index


0
1000
2000
3000

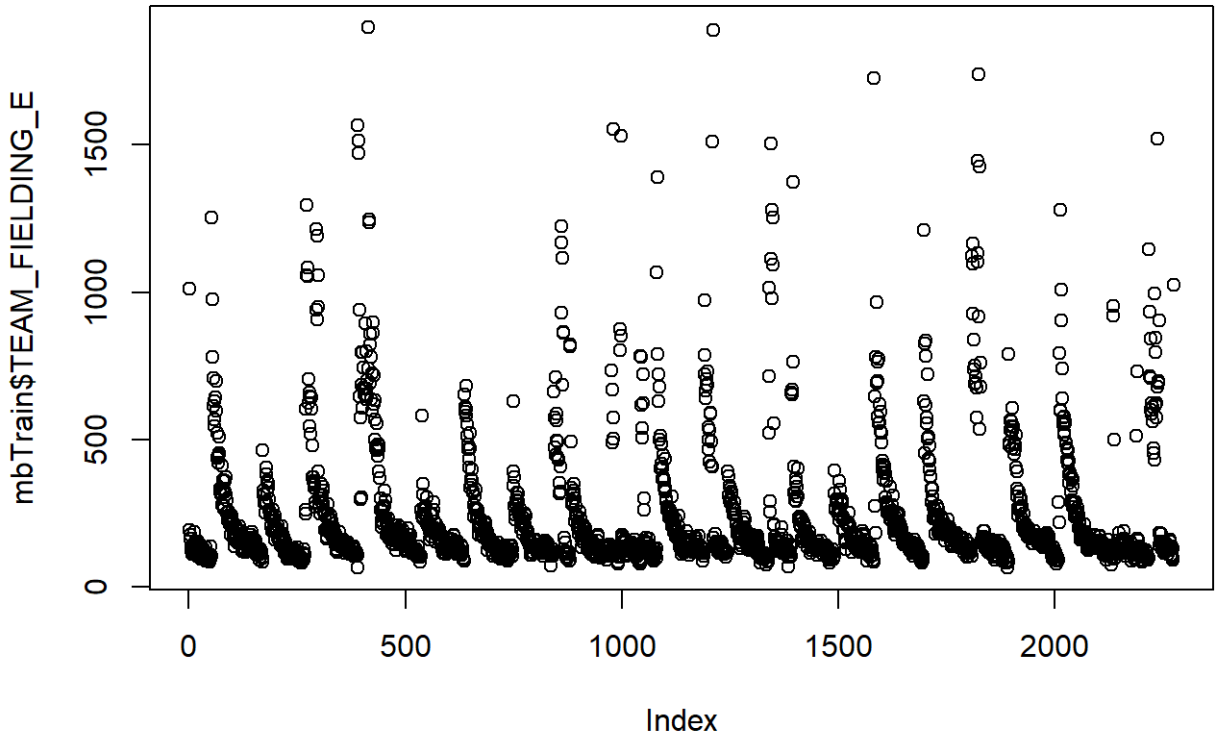
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   476.0   536.5   553.0   611.0  3645.0
```

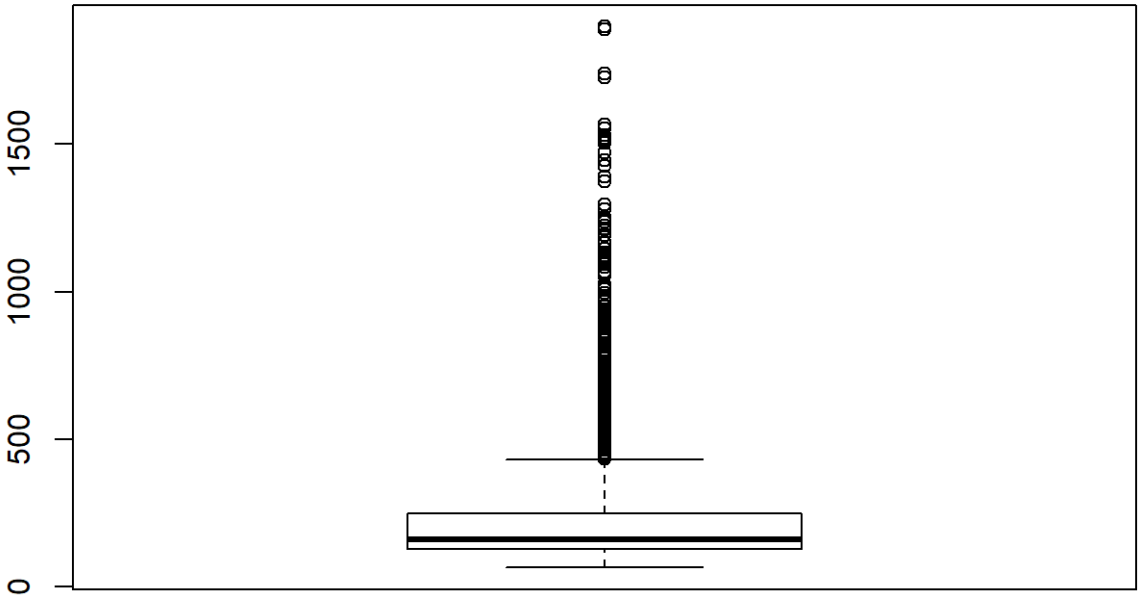
Again looking at the maximum we find it is unreasonably high at 3645, which again is substantially far from the median of 536.5. Again this variable has a reciprocal in TEAM_BATTING_SO. Comparing against it confirms the outliers.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   451.0   512.0   501.6   580.0   878.0
```

We see that or batting the maximum is 878, much lower than 3645. As before as a sanity check of the distribution of the batting variable, we now see the mean and median much closer together with a spread of only 10.4.

TEAM_FIELDING_E


mbTrain$TEAM_FIELDING_E
0
500
1000
1500
0
500
1000
1500
2000
Index


0
500
1000
1500

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    65.0   127.0   159.0   246.5   249.2  1898.0
```

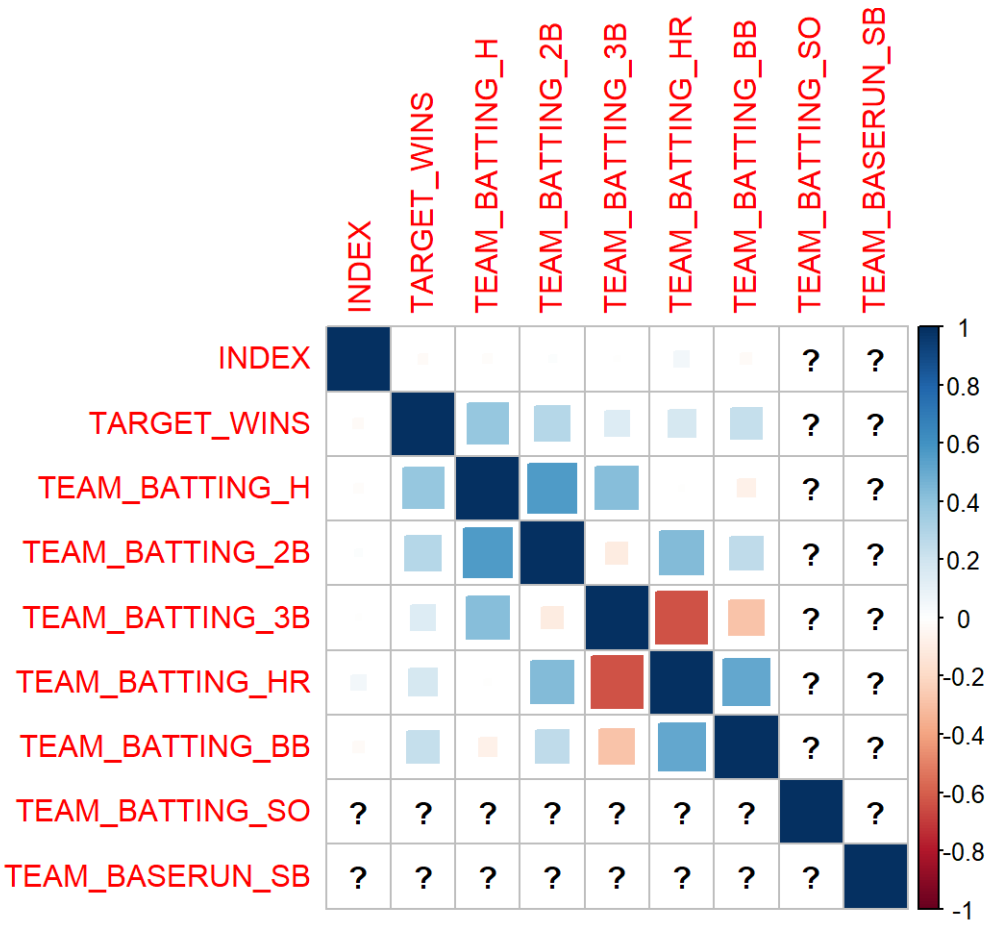
Again looking at the maximum we find it is unreasonably high at 1898, which again is substantially far from the median of 159.

2.3 Correlation Plot

Looking at correlation of variables to number of wins provides some interesting data. We find some correlations that make sense from what might assume with subject knowledge of base, e.g., the number of hits and number of variables both have significant positive correlation with Wins and other statistics like stolen bases, while still positive, are not so strongly related. What is surprising though, are the pitching statistics. We would assume that a team that allowed the opposing team more hits, would lose more games (and win less), but that is not what the data shows us. Perhaps there are outliers swaying the correlation.

Regardless, we can use some of these correlations to drive initial models later, in terms of likely fields to choose for an effective model.

```
##      TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## [1,]           1      0.4699467        0.312984      -0.1243459
##      TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## [1,]       0.4224168       0.4686879      -0.2288927      0.01483639
##      TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
## [1,]       -0.178756       0.07350424       0.4712343        0.4224668
##      TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## [1,]        0.4683988       -0.2293648       -0.386688        -0.195866
```



3 DATA PREPARATION

3.1 Variable Creation / Removal

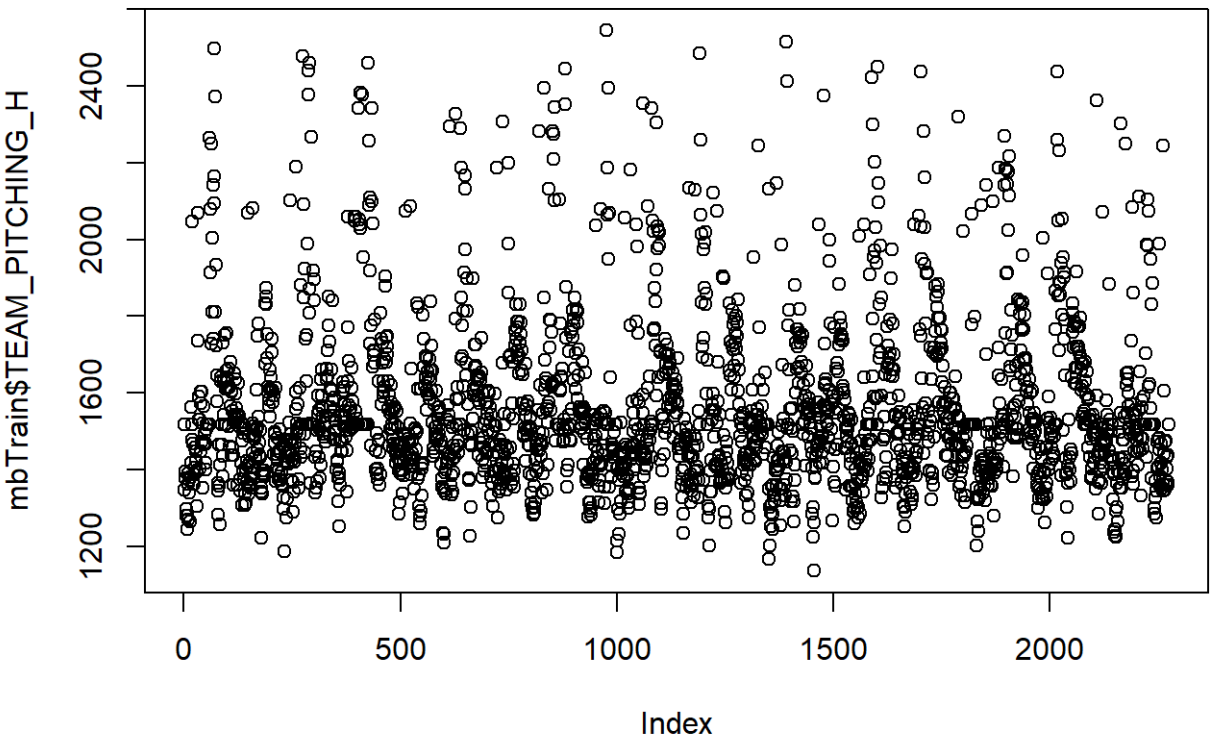
First task under data preparation will be to eliminate all missing data. In the Data Exploration section we found one variable, TEAM_BATTING_HBP with an exceptionaly high percentage of missing data, so we commence by eliminating this variable. We also removed the “INDEX” column as that is not used.

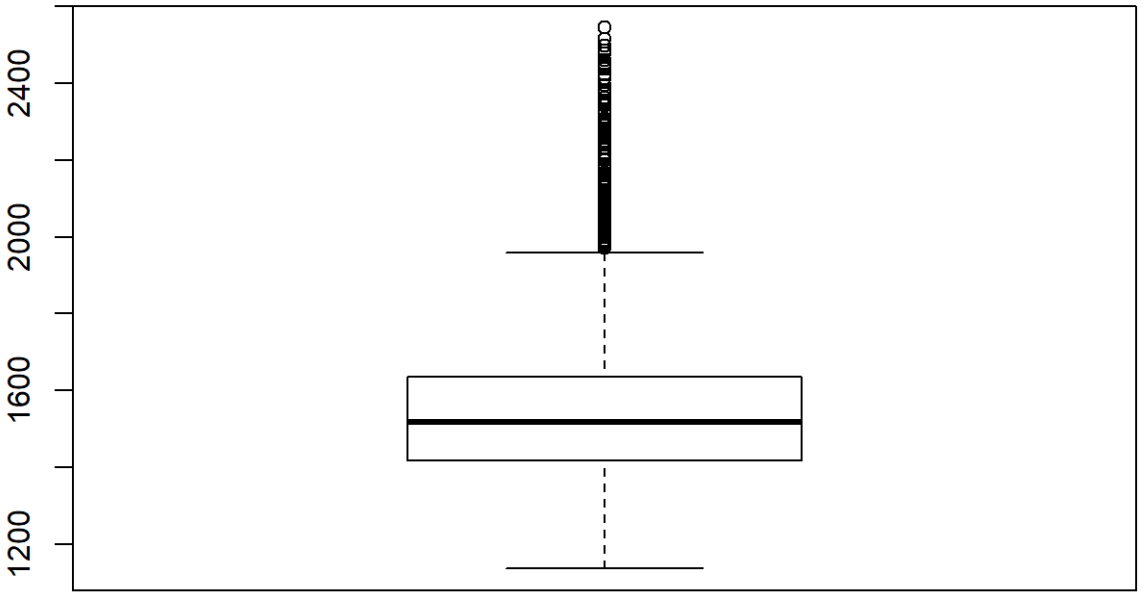
Next task is to handle missing data in the other variables. Here, because the percentages of missing data are lower, we can replace missing data with the median. We prefer replacing with median instead of mean because the latter is more sensitive to outliers. So we get a clean dataset without missing values.

Note, we also consider zeros to be missing data. Since each row is a season of data for a given baseball team, it would be extraordinarily unlikely that any of these statistics would have zero as an actual value. Therefore we are assuming zero is another indicator of missing value and we will transform them into a median value.

In the exploratory phase we also identified several variables with outliers. Outliers will be substituted with median. Again we choose median becouse it is less influenced by these outliers. What cut-off to use to tag an outlier reading could be a 3 standard deviation from the mean, or 1.5 time the inter quartile range, but in this case because these variables have reciprocals as seen in the exploratory phase, we will use the maximum reading of those variables.

TEAM_PITCHING_H

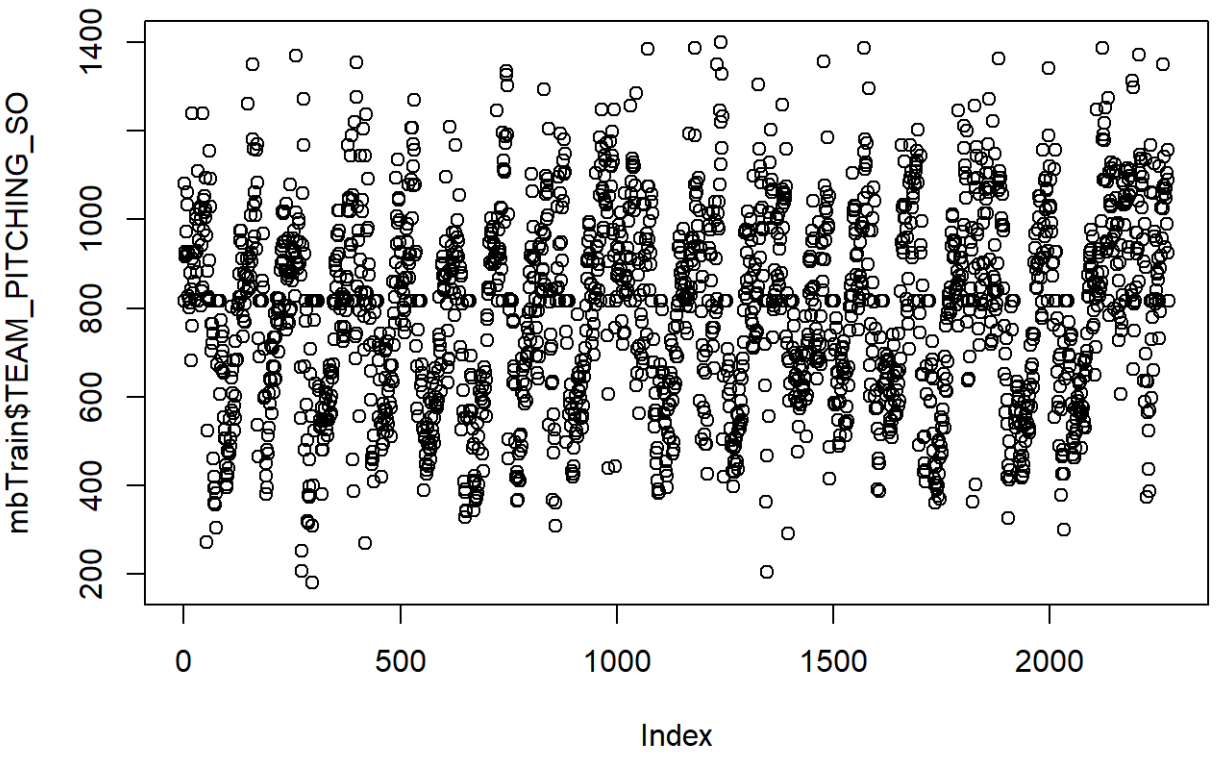

mbTrain$TEAM_PITCHING_H
1200
1600
2000
2400
0
500
1000
1500
2000
Index

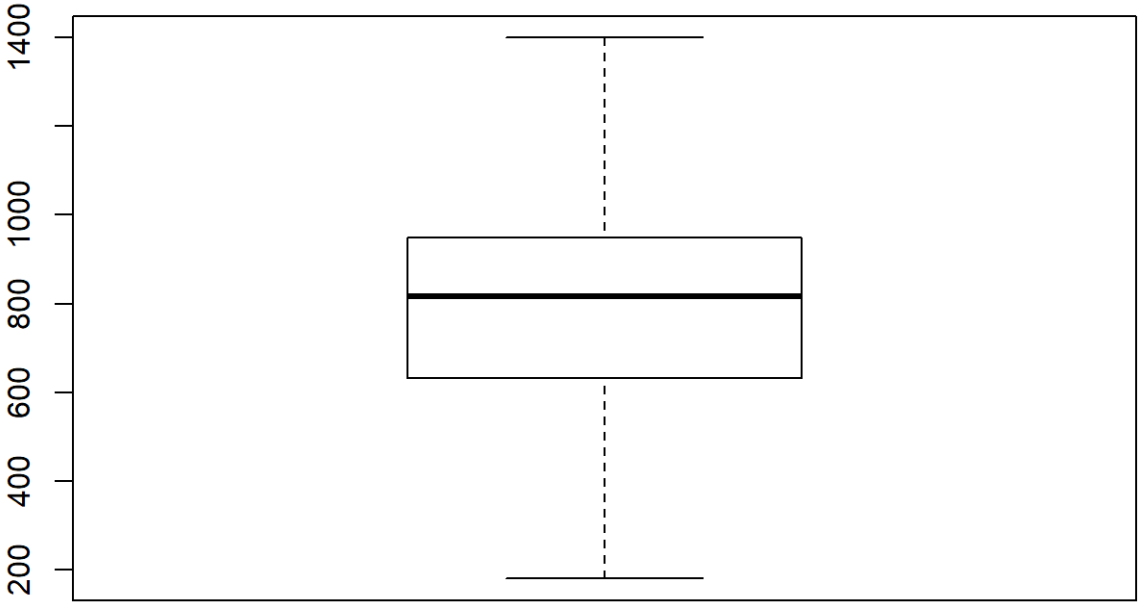

1200
1600
2000
2400

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1137    1419    1518    1562    1636    2544
```

From the summary we now see that the maximum is 2544, which is a much more reasonable number. We can also see a wide spread between mean and median of 44, indicating a more normal distribution than before.

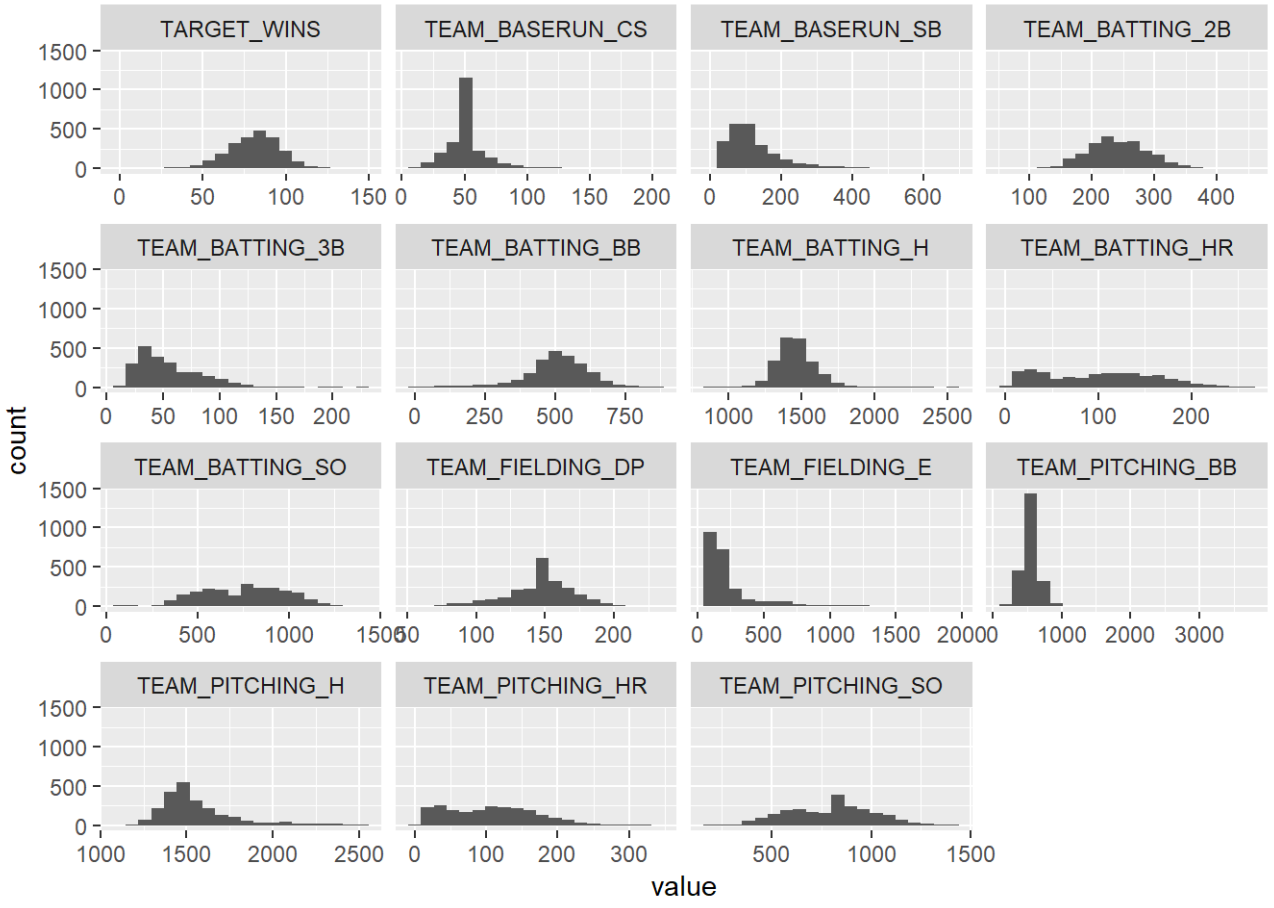
TEAM_PITCHING_SO


mbTrain$TEAM_PITCHING_SO
200
400
600
800
1000
1400
0
500
1000
1500
2000
Index


200
400
600
800
1000
1400

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   181.0   633.0   816.0   796.8   948.2  1399.0
```

From the summary we now see that the maximum is 1399, which is a much more reasonable number. We can also see a wide spread between mean and median of -19, indicating a more normal distribution than before.



New Variables

With a clean dataset, we can now start looking at what predictor variables can be combined and what new statistics can be derived.

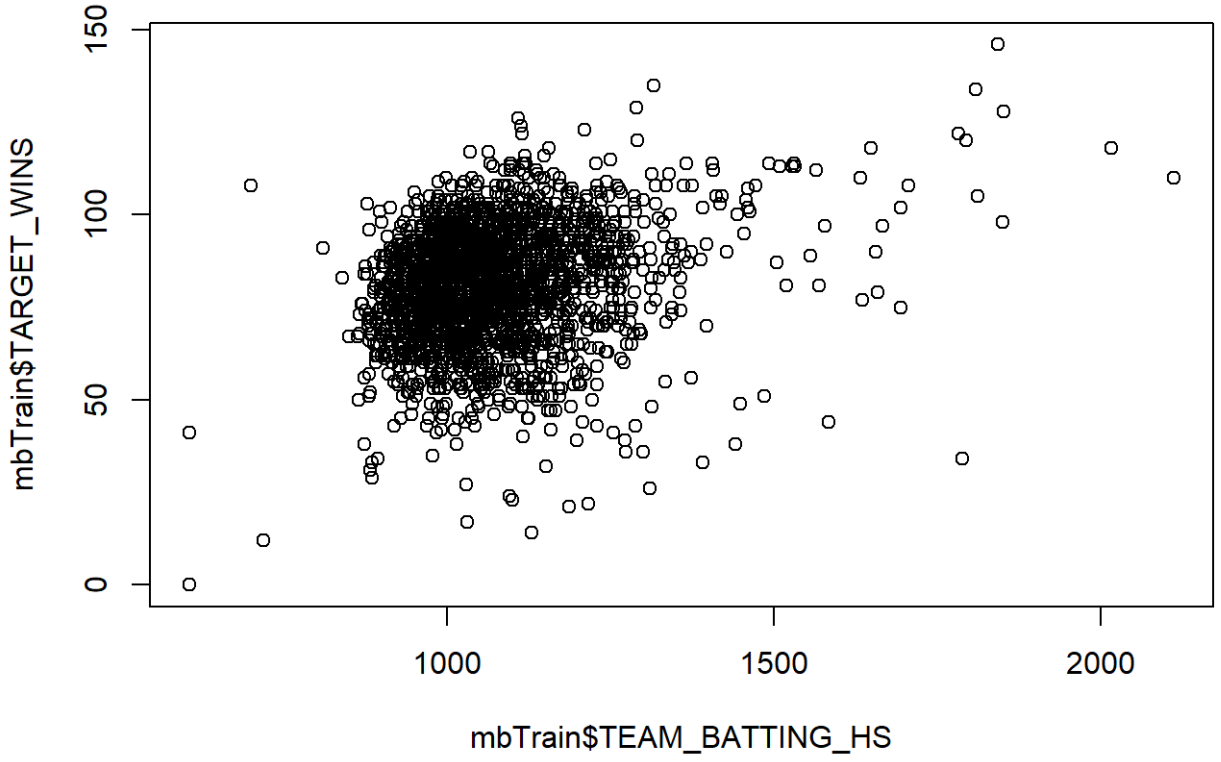
Batting Hit Singles

On the batting side we can start by adding a variable for single hits since the dataset has a variable for all 4 kinds of hits.

TEAM_BATTING_HS = TEAM_BATTING_H - (TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR)

TEAM_BATTING_HS Summary

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     606     990    1050    1072    1129    2112
```



There are other popular baseball statistics which are regularly calculated. The data given is limited, so it won't be possible to make all these calculations. But we can use the data given to calculate some statistics that resemble some of the common baseball measurements.

The number of times a batter reaches base can be calculated as Times On Base:

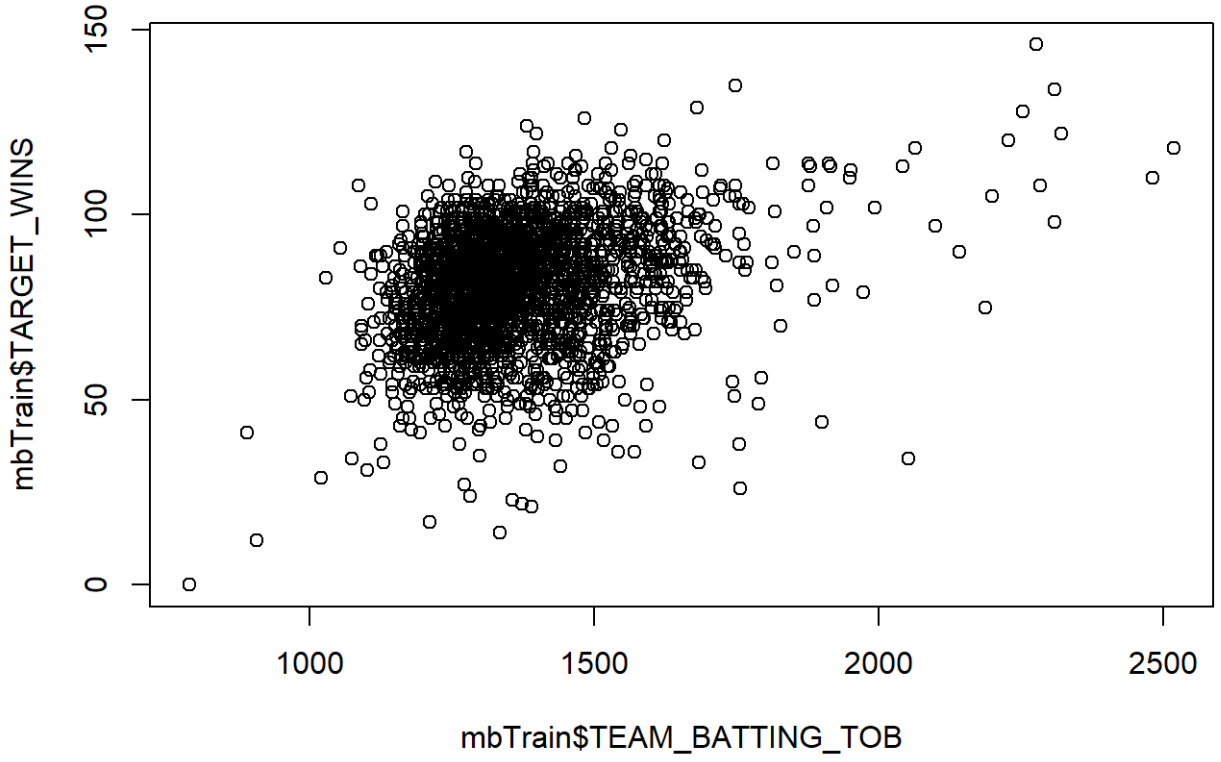
Times On Base

TOB = Base Hits + Walks + Hits by Pitch TOB = (TEAM_BATTING_H - TEAM_BATTING_HR) + TEAM_BATTING_BB + TEAM_BATTING_HBP TOB = TEAM_BATTING_TOB

In our case we do not have TEAM_BATTING_HBP. We deleted this predictor since it didn't contain enough data, so we will not include this term in calculating TOB.

TEAM_BATTING_TOB Summary:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     788    1269    1338    1369    1442    2518
```



On Base Percentage

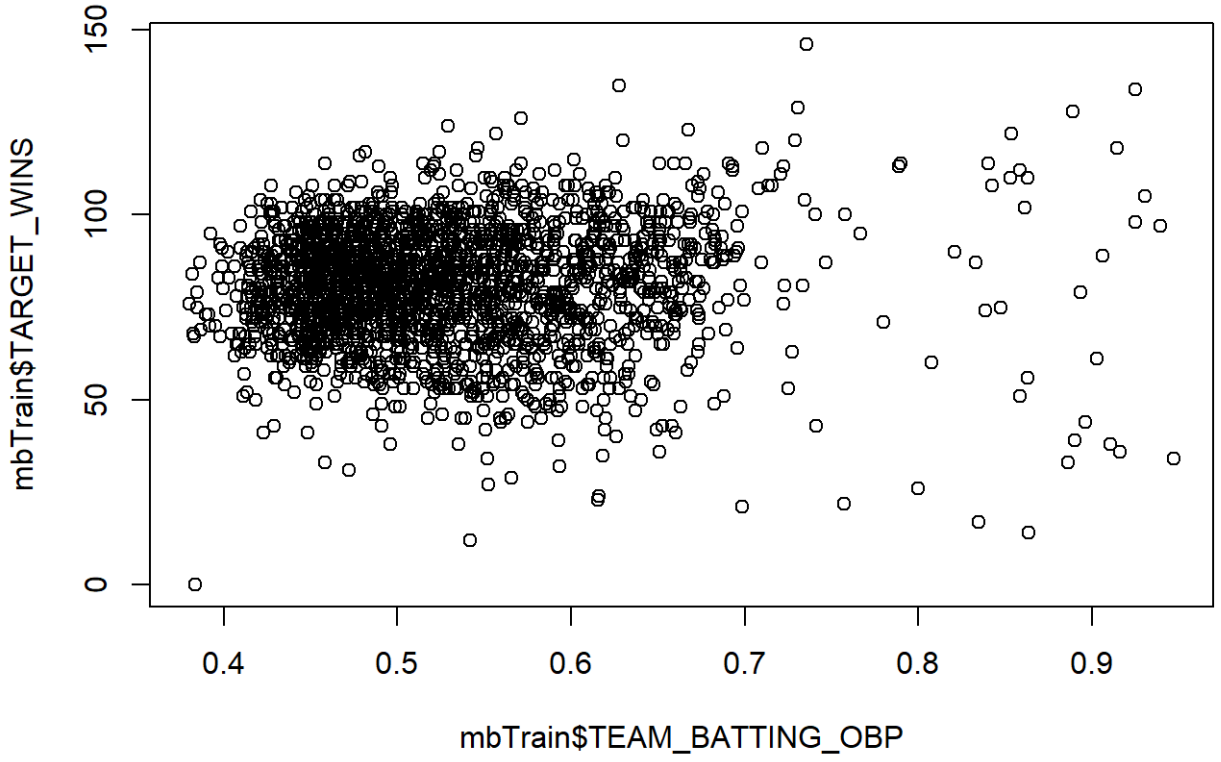
If we divide this statistic by the times a batter appears on plate, we have a ratio for On Base Percentage. Batter appearances on plate is not a statistic that was given, but we can assumes it would the similar to the number of times a batter produces a hit and the times of strikeouts.

OBP = TOB / (Base Hits + Walks + Hits by Pitch + Strikeouts) OBP = TEAM_BATTING_TOB / ((TEAM_BATTING_H - TEAM_BATTING_HR) + TEAM_BATTING_BB + TEAM_BATTING_HBP + TEAM_BATTING_SO) OBP = TEAM_BATTING_OBP

Same as before TEAM_BATTING_HBP is missing so we do not include it.

TEAM_BATTING_OBP Summary

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3801  0.4658  0.5122  0.5287  0.5743  0.9469
```



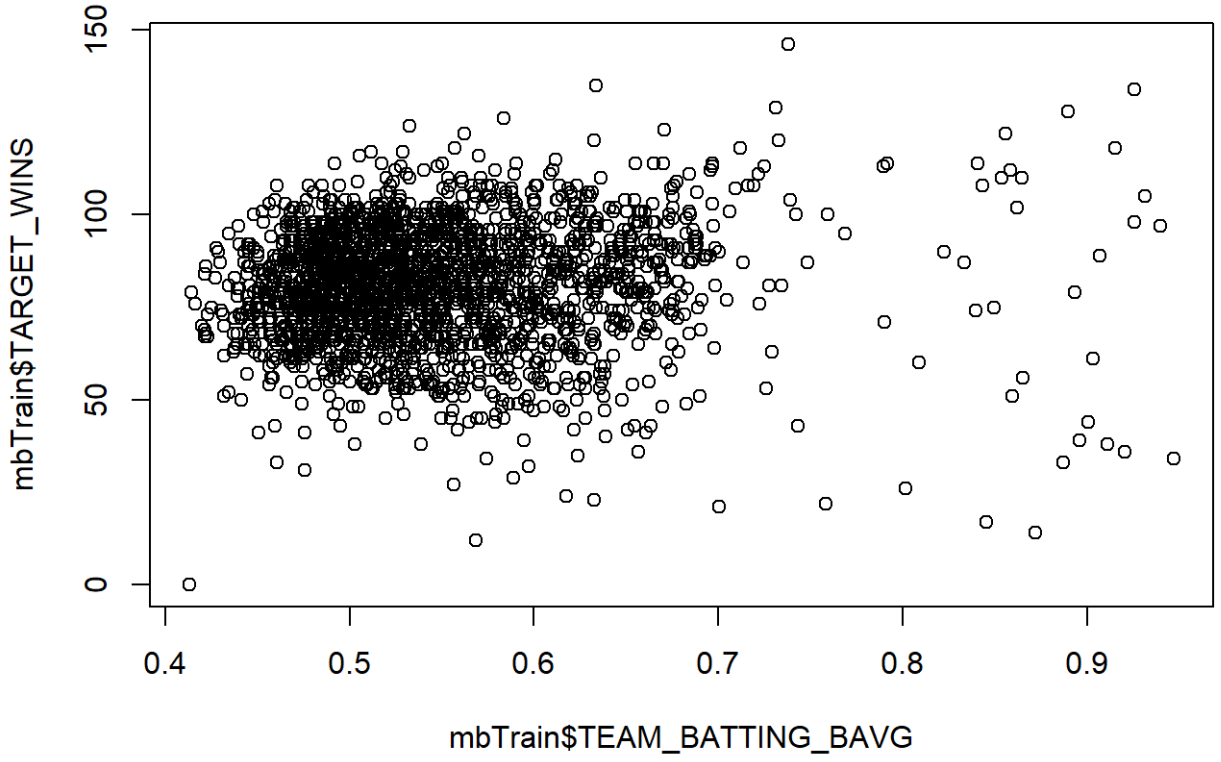
Batting Average

This statistics is calculated as the number of batter hits divided by times at bat or on plate. With our dataset we will compute times at bat as the sum of a batters hits and strike out, same as we did on the previous calculated variable since the number of Hits by Pitch is not available:

AVG = Hits / (Hits + Walks + Strikeouts) AVG = TEAM_BATTING_H / (TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_BATTING_SO) AVG = TEAM_BATTING_BAVG

TEAM_BATTING_BAVG Summary

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4131  0.4923  0.5290  0.5464  0.5846  0.9471
```



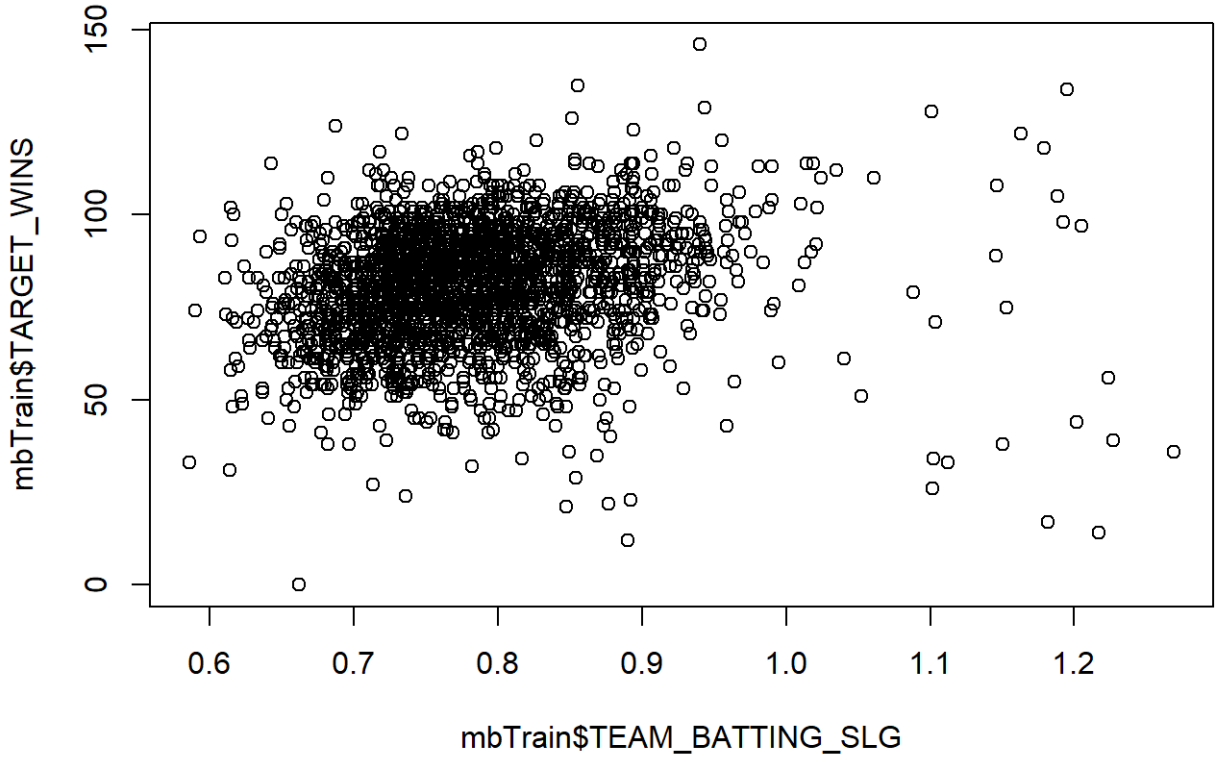
Slugging Percentage

A shortcoming of the previous statistic is that it weights any kind of hits equally. To account for the fact that some hits are more beneficial or carry higher weight we can calculate a slugging percentage by multiplying each kind of hit by an increasing number.

SLG = (Single Hits + 2 * Double Hits + 3 * Tripple Hits + 4 * Home Runs) / (Hits + Walks + Strikeouts) TEAM_BATTING_SLG = ((TEAM_BATTING_H - TEAM_BATTING_2B - TEAM_BATTING_3B - TEAM_BATTING_HR) + 2 * TEAM_BATTING_2B + 3 * TEAM_BATTING_3B + 4 * TEAM_BATTING_HR) / (mbTrain$TEAM_BATTING_H + mbTrain$TEAM_BATTING_BB + mbTrain$TEAM_BATTING_SO)

TEAM_BATTING_SLG Summary

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5861  0.7288  0.7731  0.7842  0.8291  1.2690
```



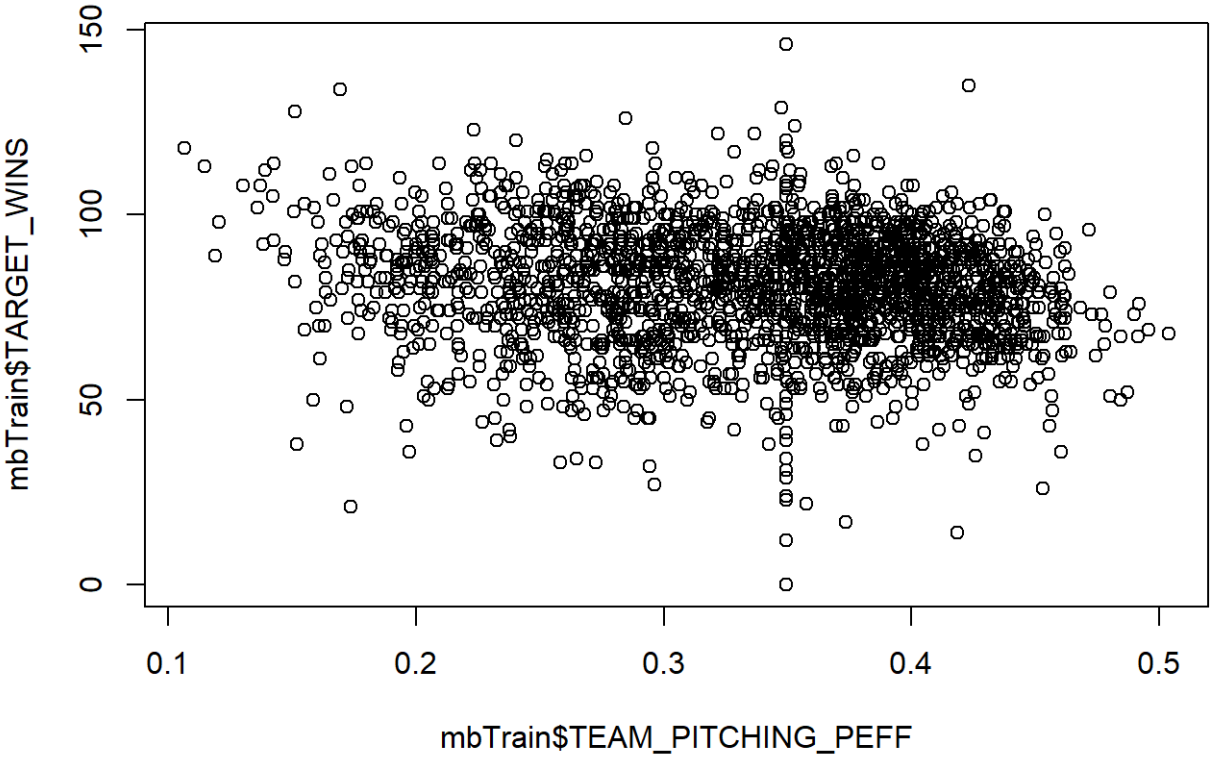
Strikeout Efficiency

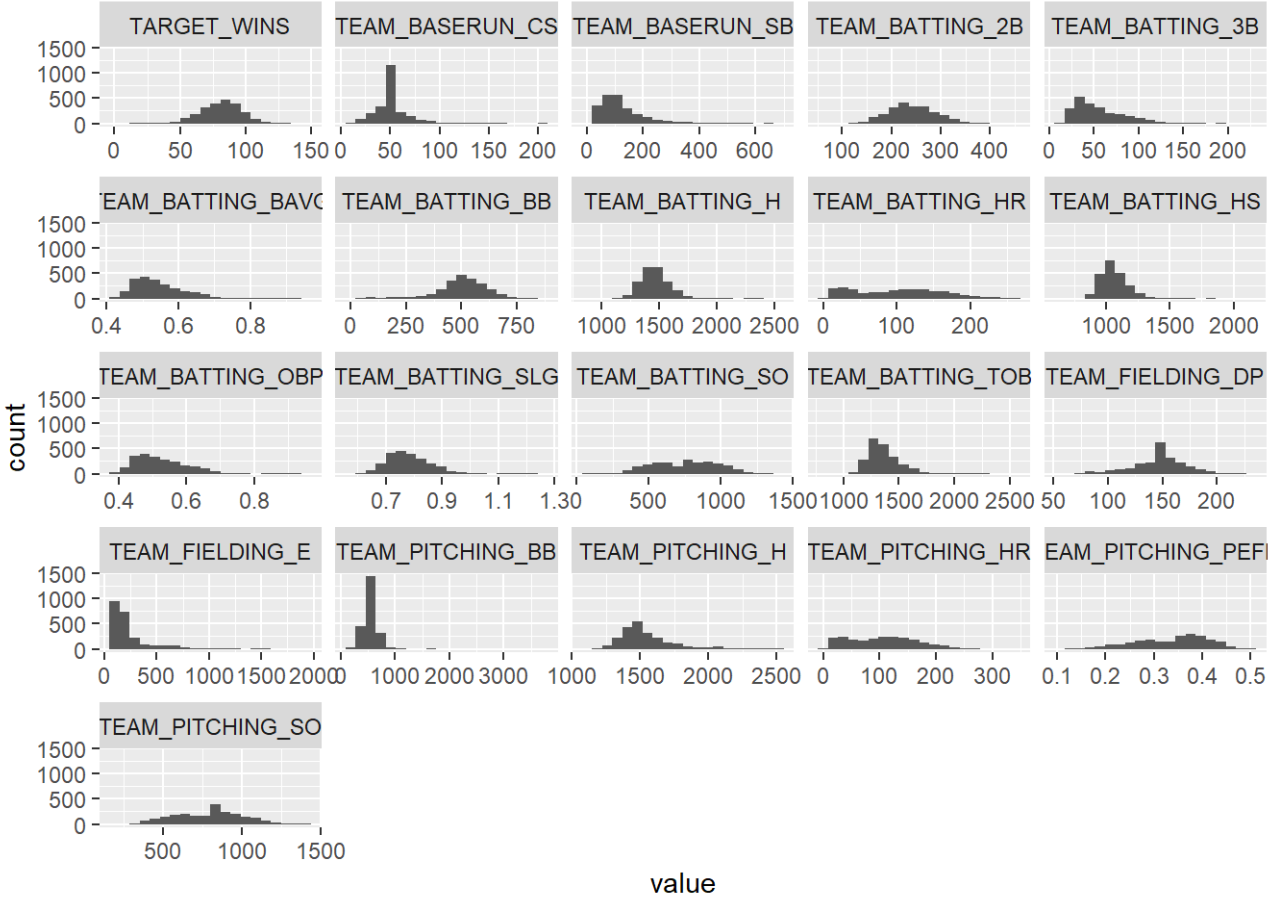
Measures how successful a pitches is at striking out batters:

PEFF = Strike Outs (Hits + Strike Outs) TEAM_PITCHING_PEFF = TEAM_PITCHING_SO (TEAM_PITCHING_H + TEAM_PITCHING_SO)

TEAM_PITCHING_PEFF Summary

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1065  0.2813  0.3503  0.3358  0.3944  0.5038
```





Training and Test

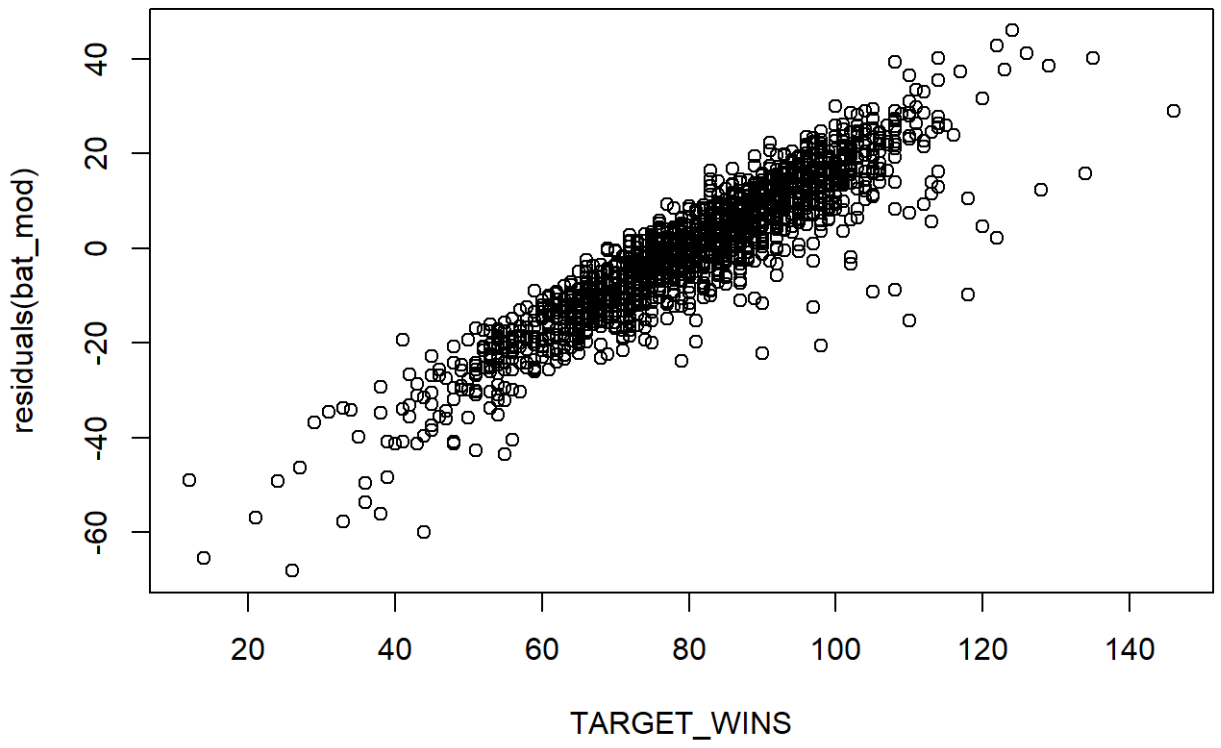
Lastly, before we create models, let's divide data into test and training sets, with 80% for training, 20% for test. This way we have a method to validate our models.

4 BUILD MODELS

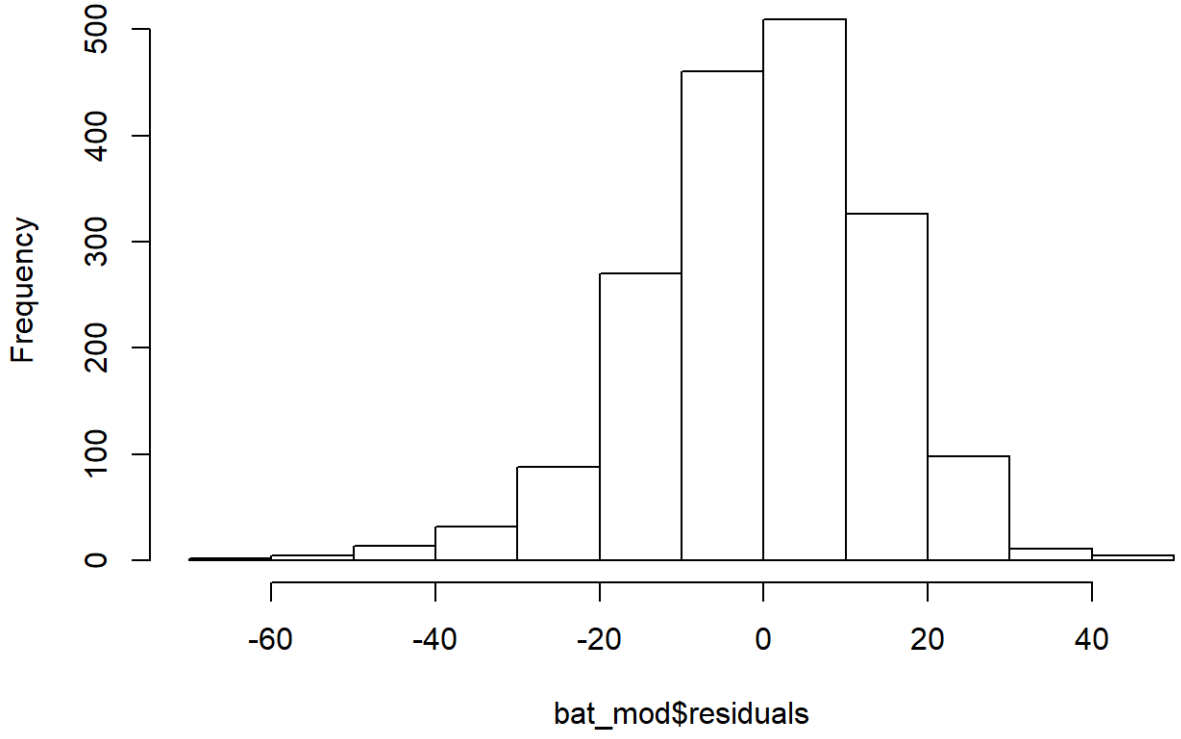
4.1 Batting only model

Combine all batting variables.

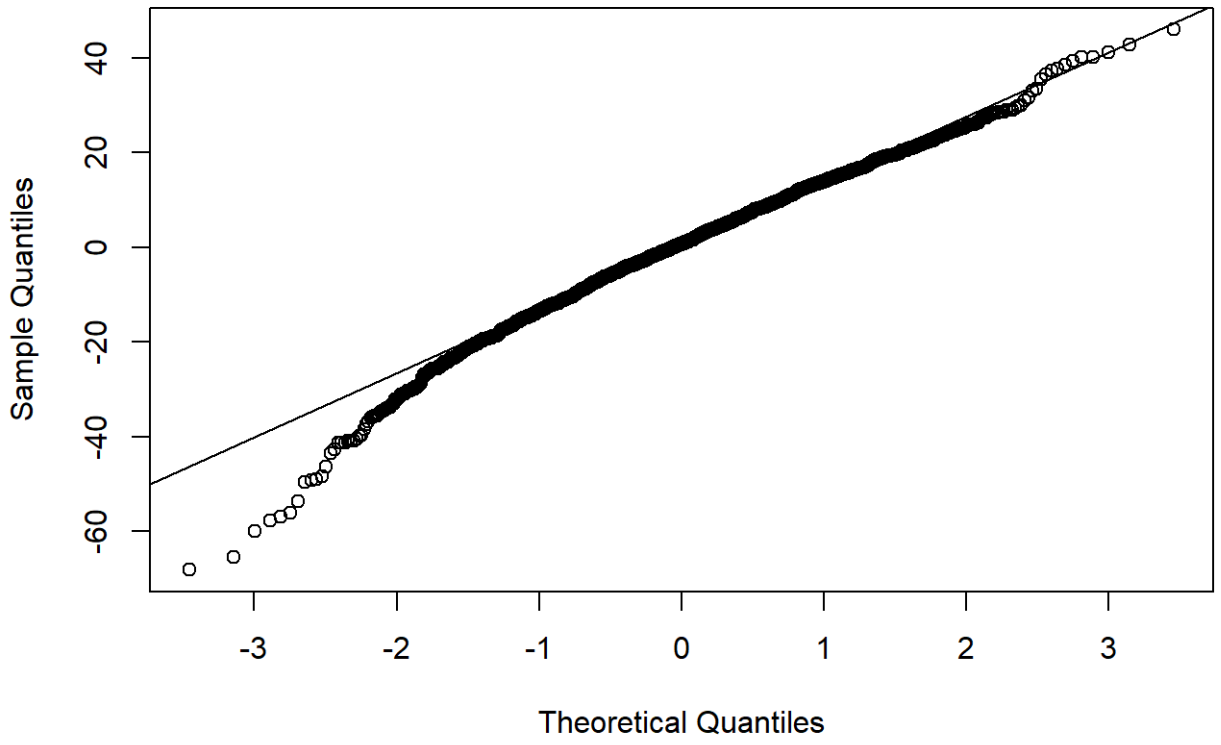
```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.157  -8.688   0.679   9.599  45.949
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    17.472589   3.369651   5.185  2.4e-07 ***
## TEAM_BATTING_H  0.043178   0.002281  18.927  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.4 on 1818 degrees of freedom
## Multiple R-squared:  0.1646, Adjusted R-squared:  0.1642
## F-statistic: 358.2 on 1 and 1818 DF,  p-value: < 2.2e-16
```

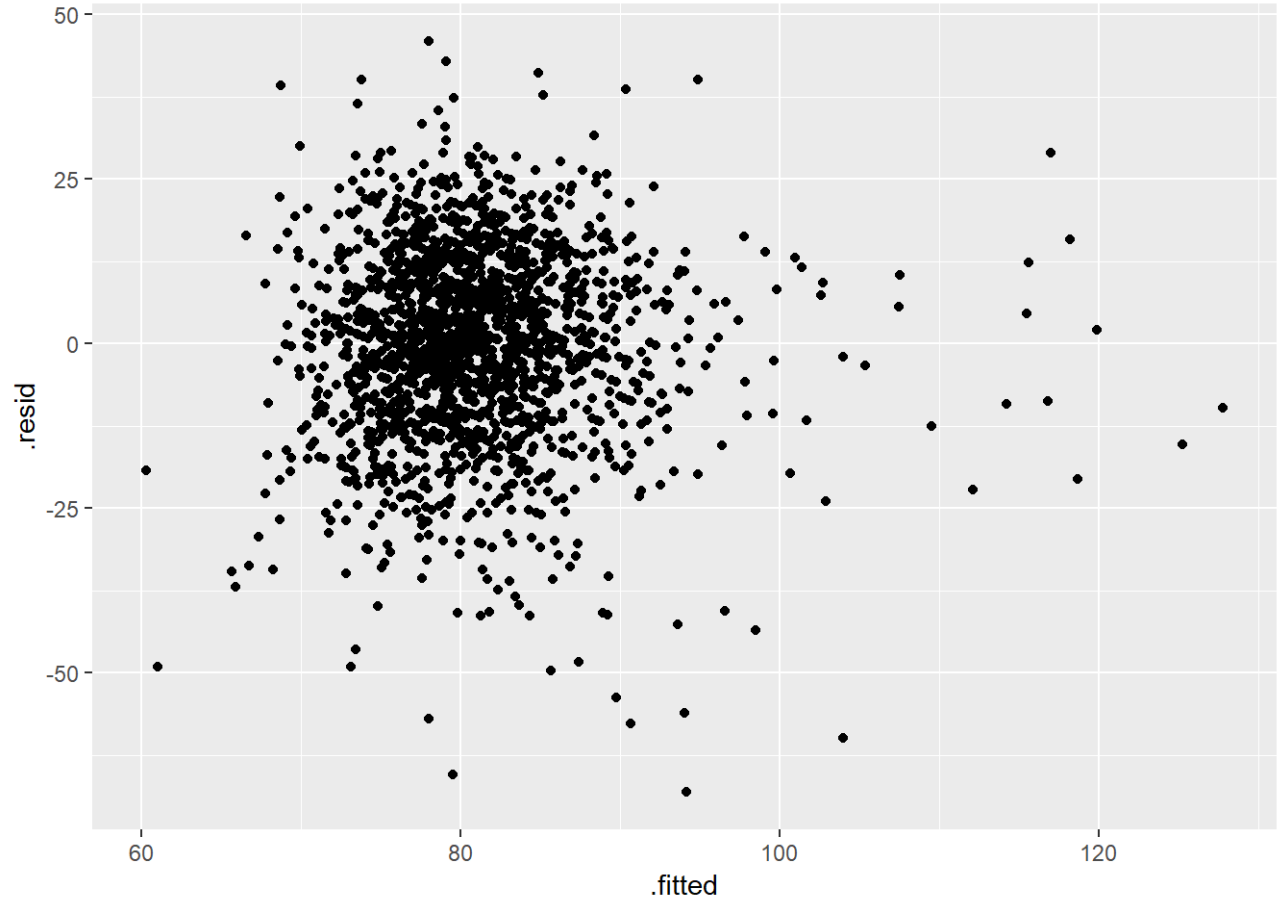


Histogram of bat_mod$residuals



Normal Q-Q Plot

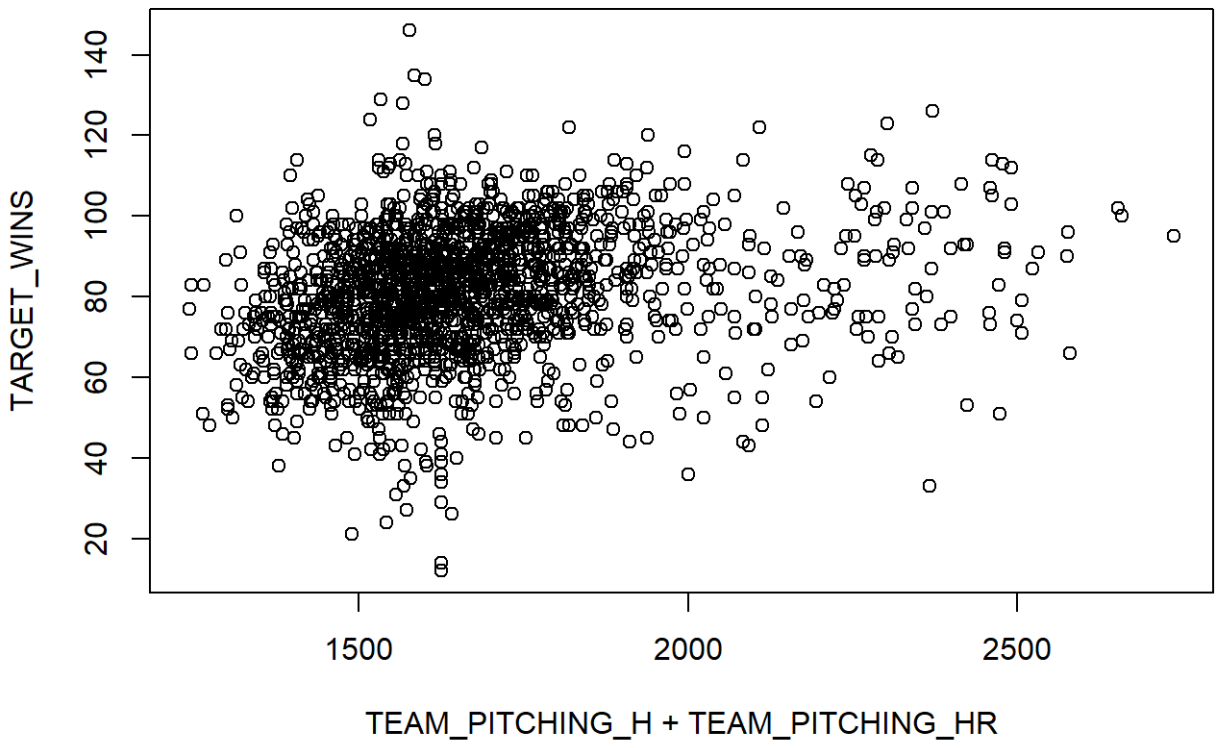


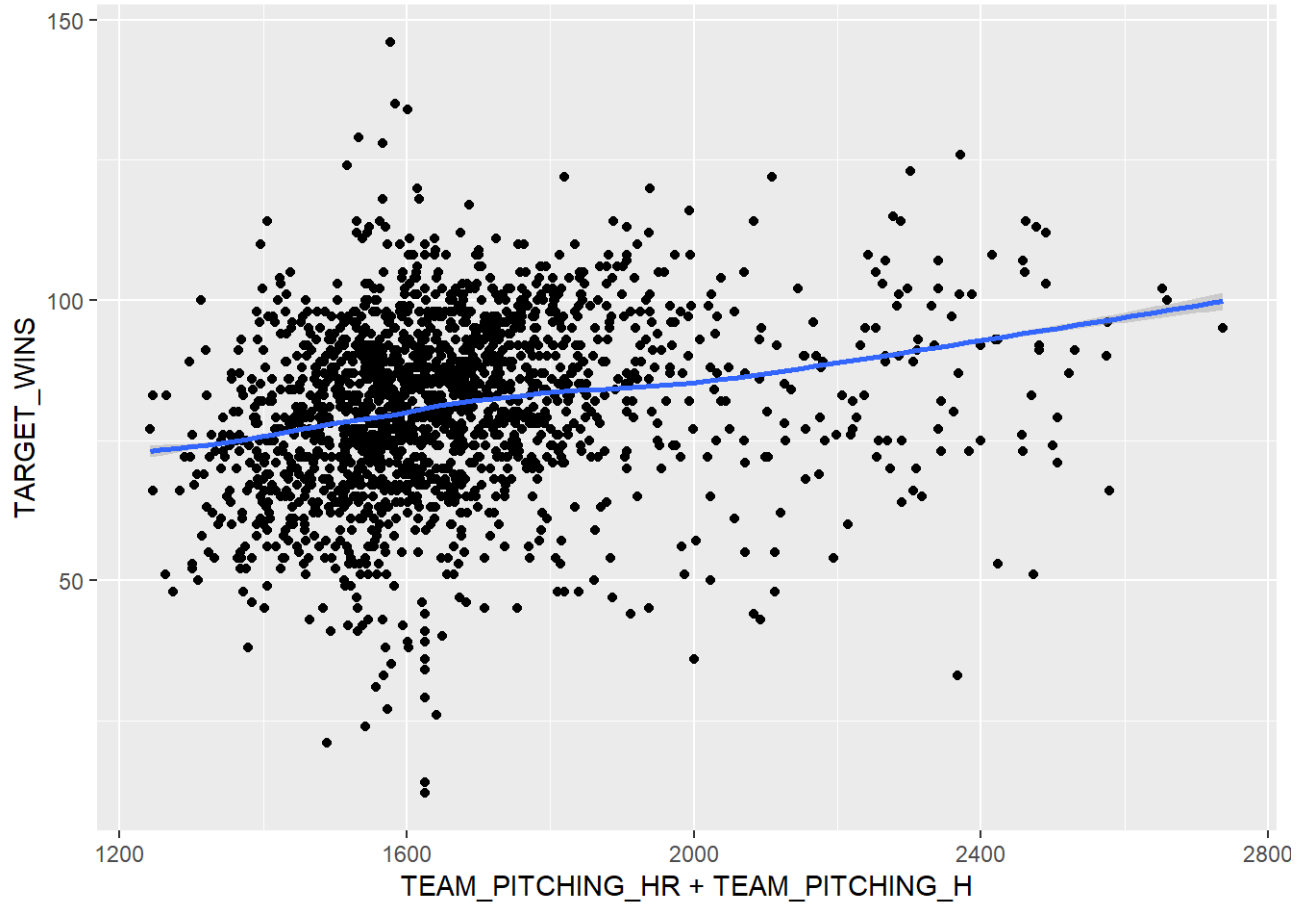


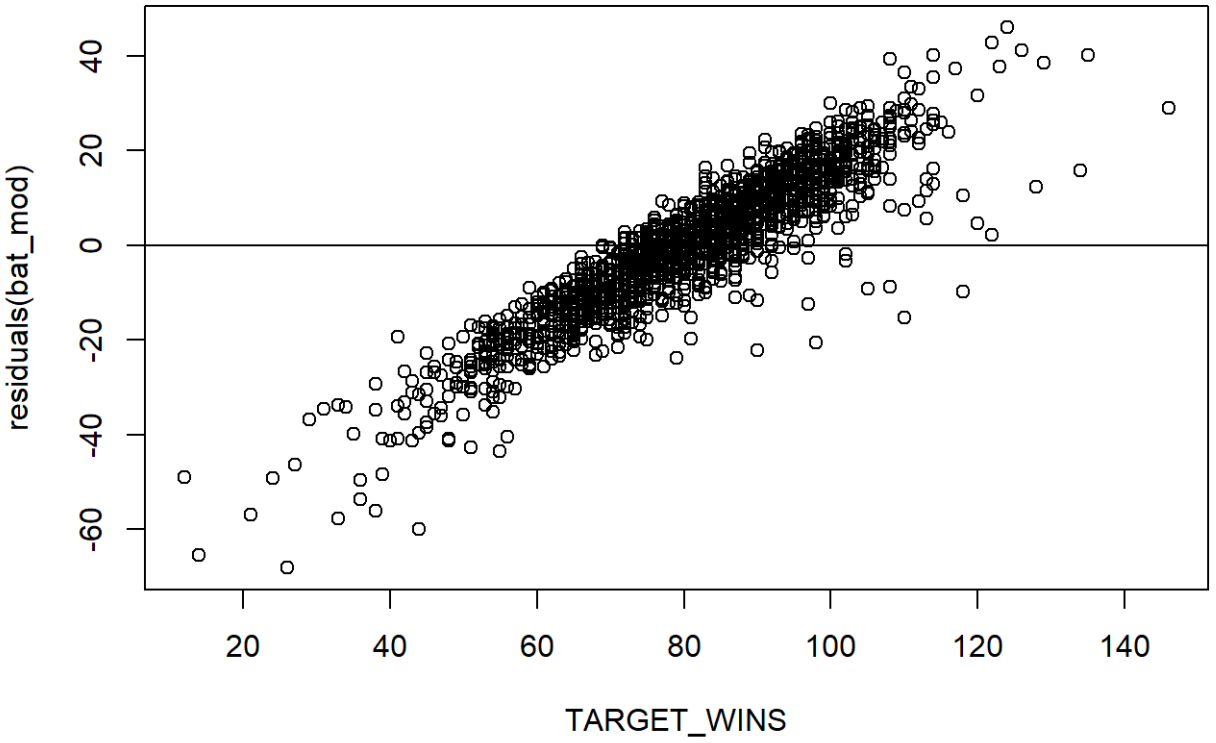
4.2 Pitching only model

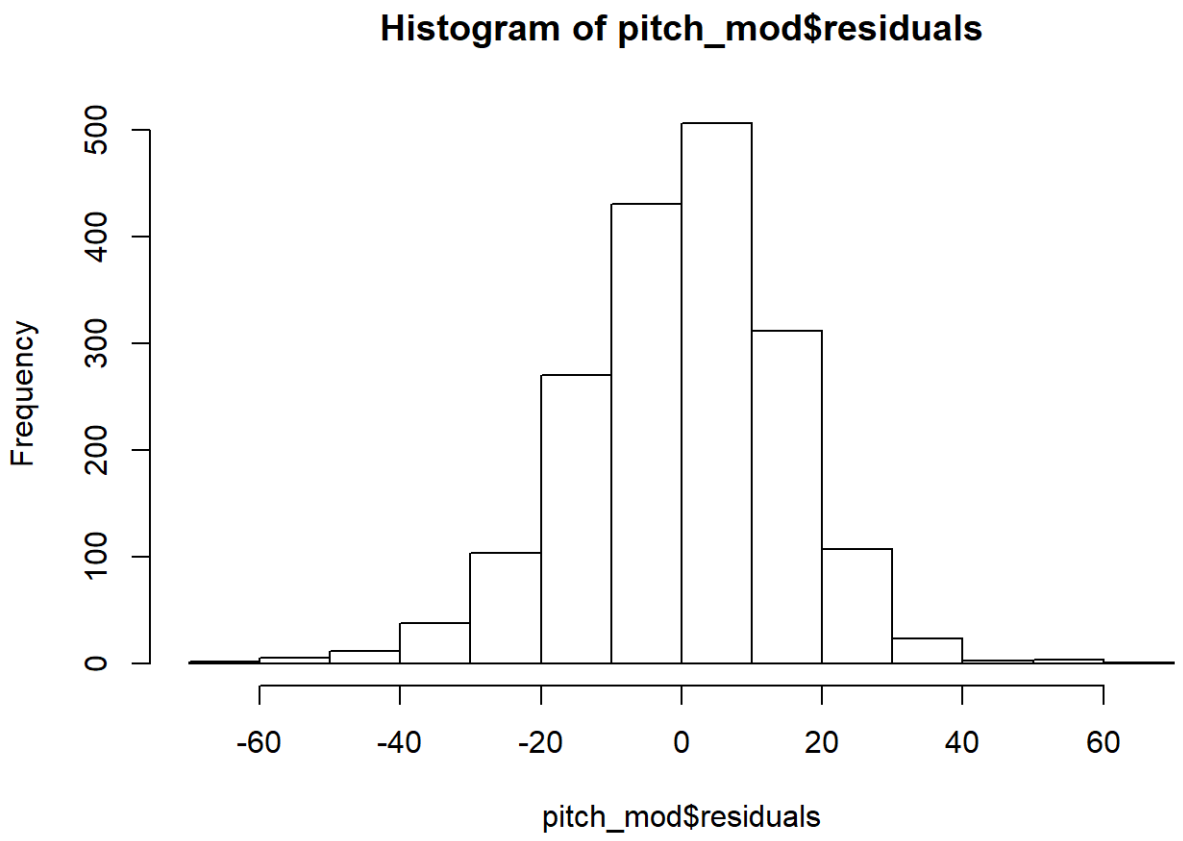
Combine all pitching variables.

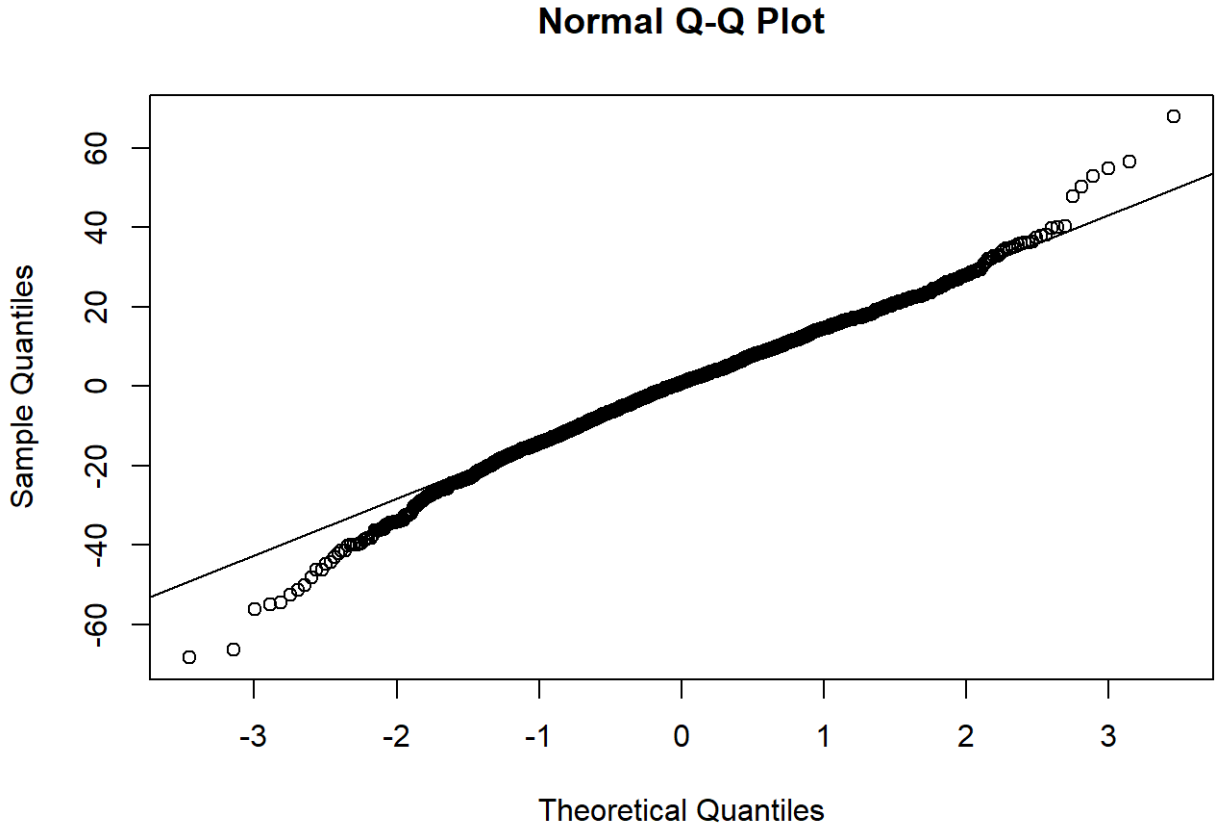
```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_PITCHING_H + TEAM_PITCHING_HR,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.379  -9.340   0.787   9.917  67.847
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      52.476611   2.691696  19.496  < 2e-16 ***
## TEAM_PITCHING_H   0.015148   0.001626   9.319  < 2e-16 ***
## TEAM_PITCHING_HR  0.045438   0.005863   7.750 1.51e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.2 on 1817 degrees of freedom
## Multiple R-squared:  0.06883,    Adjusted R-squared:  0.06781
## F-statistic: 67.16 on 2 and 1817 DF,  p-value: < 2.2e-16
```

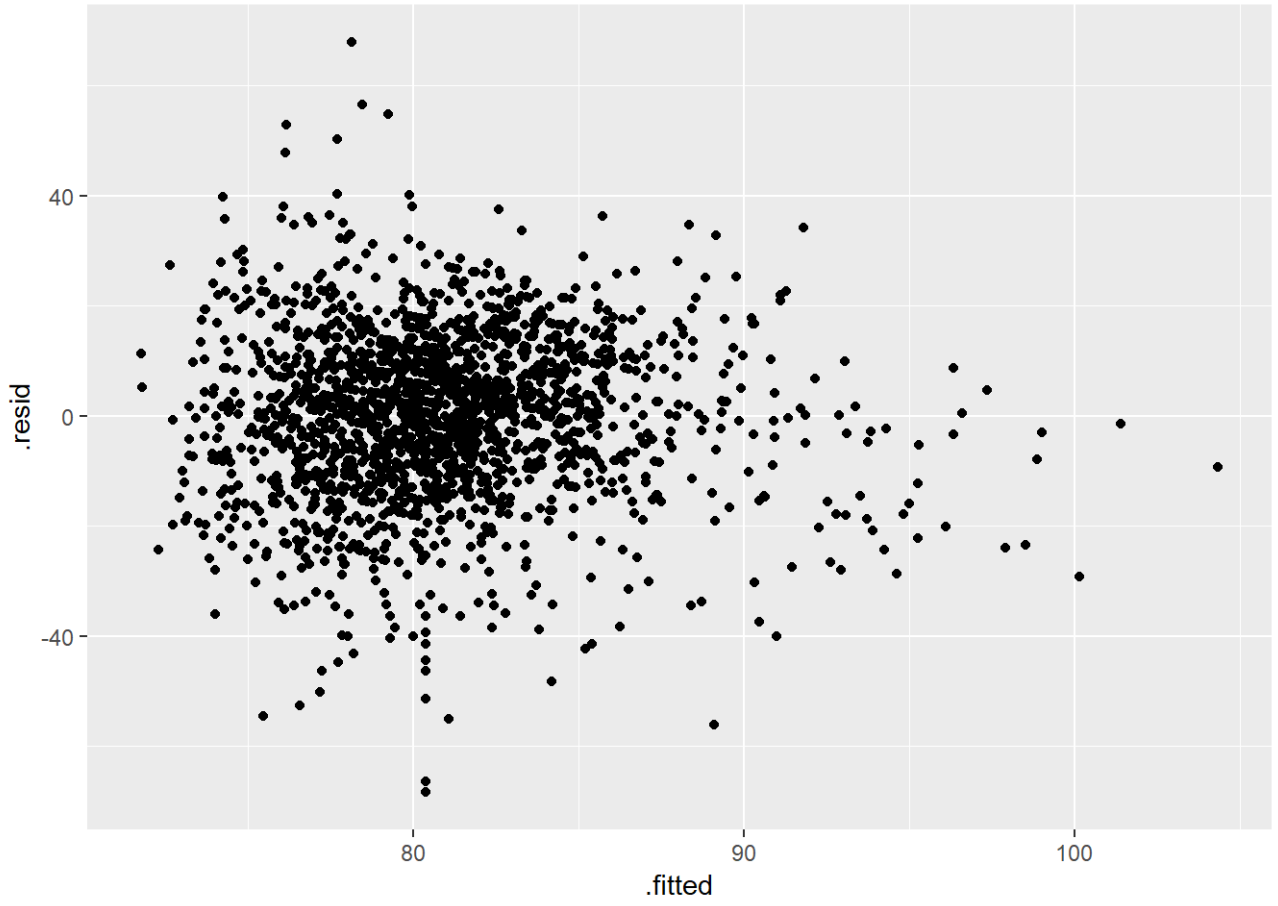

TARGET_WINS
20
40
60
80
100
120
140
1500
2000
2500
TEAM_PITCHING_H + TEAM_PITCHING_HR


TARGET_WINS
150
100
50
1200
1600
2000
2400
2800
TEAM_PITCHING_HR + TEAM_PITCHING_H

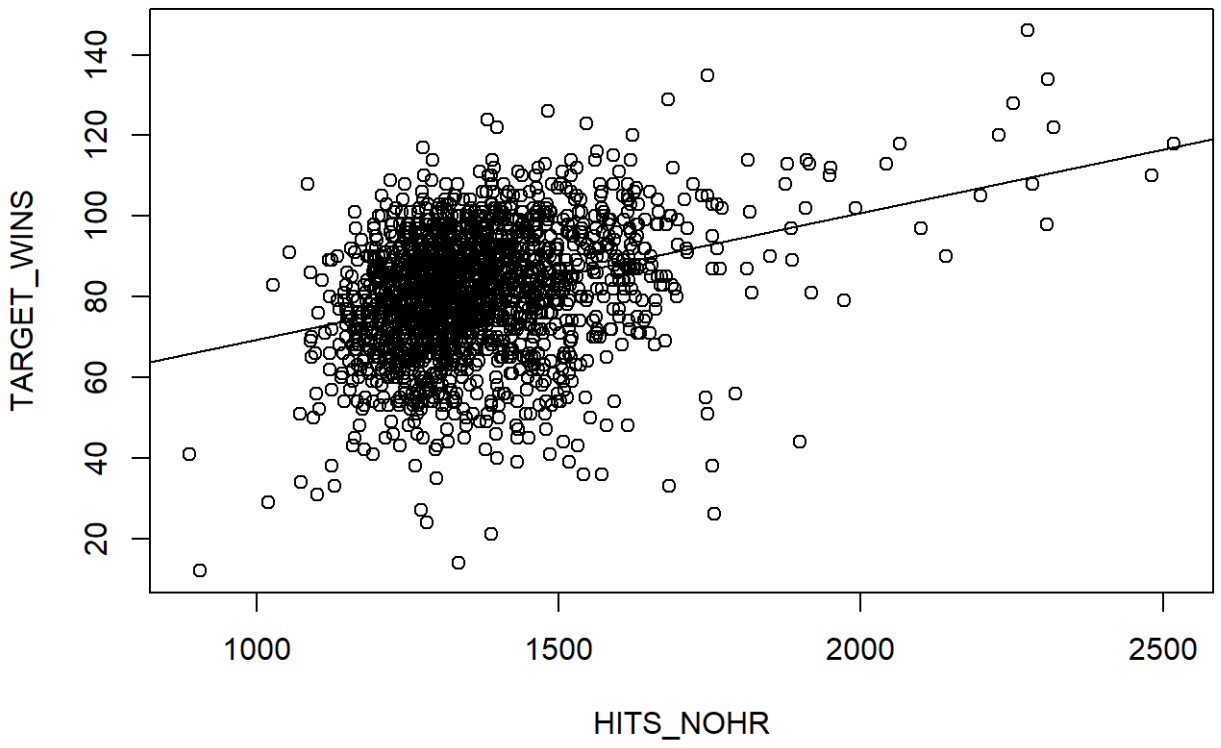

residuals(bat_mod)
40
20
0
-20
-40
-60
20
40
60
80
100
120
140
TARGET_WINS

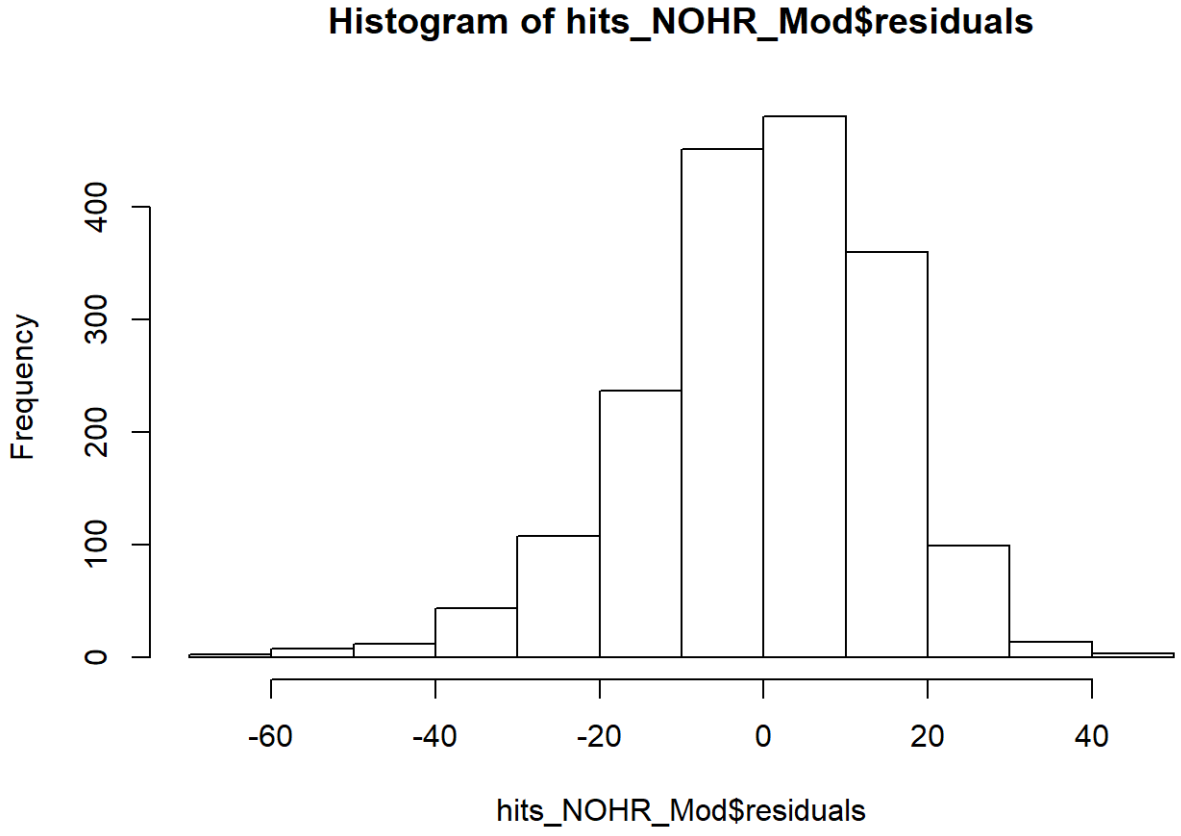

Histogram of pitch_mod$residuals
Frequency
0
100
200
300
400
500
-60
-40
-20
0
20
40
60
pitch_mod$residuals


Normal Q-Q Plot
Sample Quantiles
-60
-40
-20
0
20
40
60
-3
-2
-1
0
1
2
3
Theoretical Quantiles

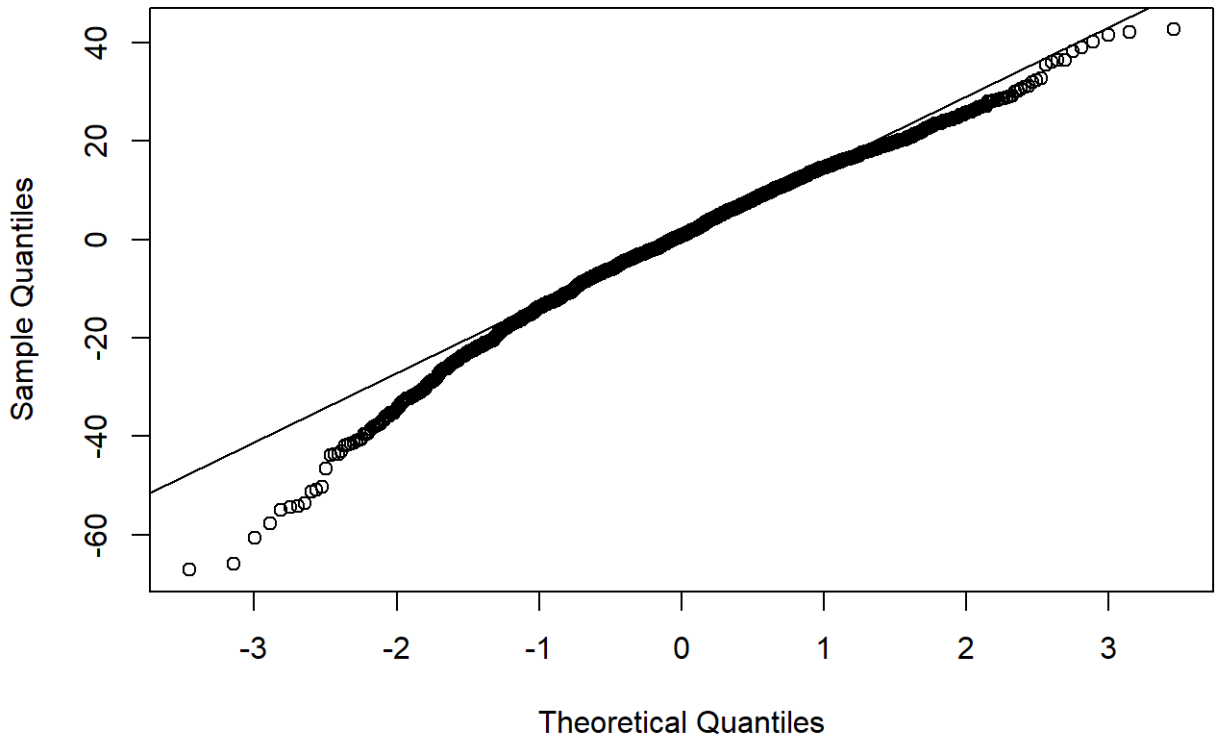


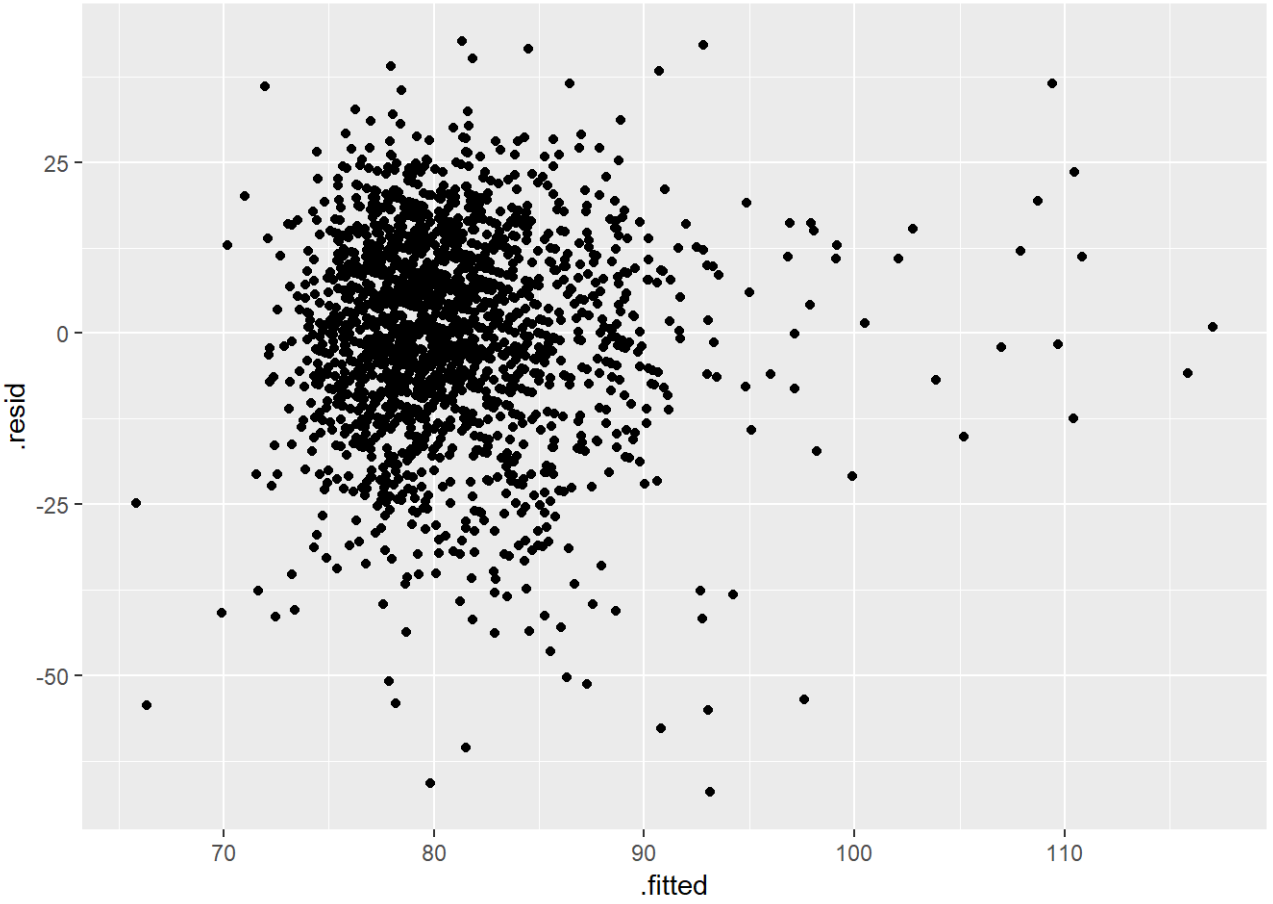

```
##
## Call:
## lm(formula = TARGET_WINS ~ HITS_NOHR)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -67.111  -8.461   0.808  10.497  42.679
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.86903    3.00594   12.60   <2e-16 ***
## HITS_NOHR    0.03144    0.00218   14.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.92 on 1818 degrees of freedom
## Multiple R-squared:  0.1027, Adjusted R-squared:  0.1022
## F-statistic:   208 on 1 and 1818 DF,  p-value: < 2.2e-16
```





Normal Q-Q Plot

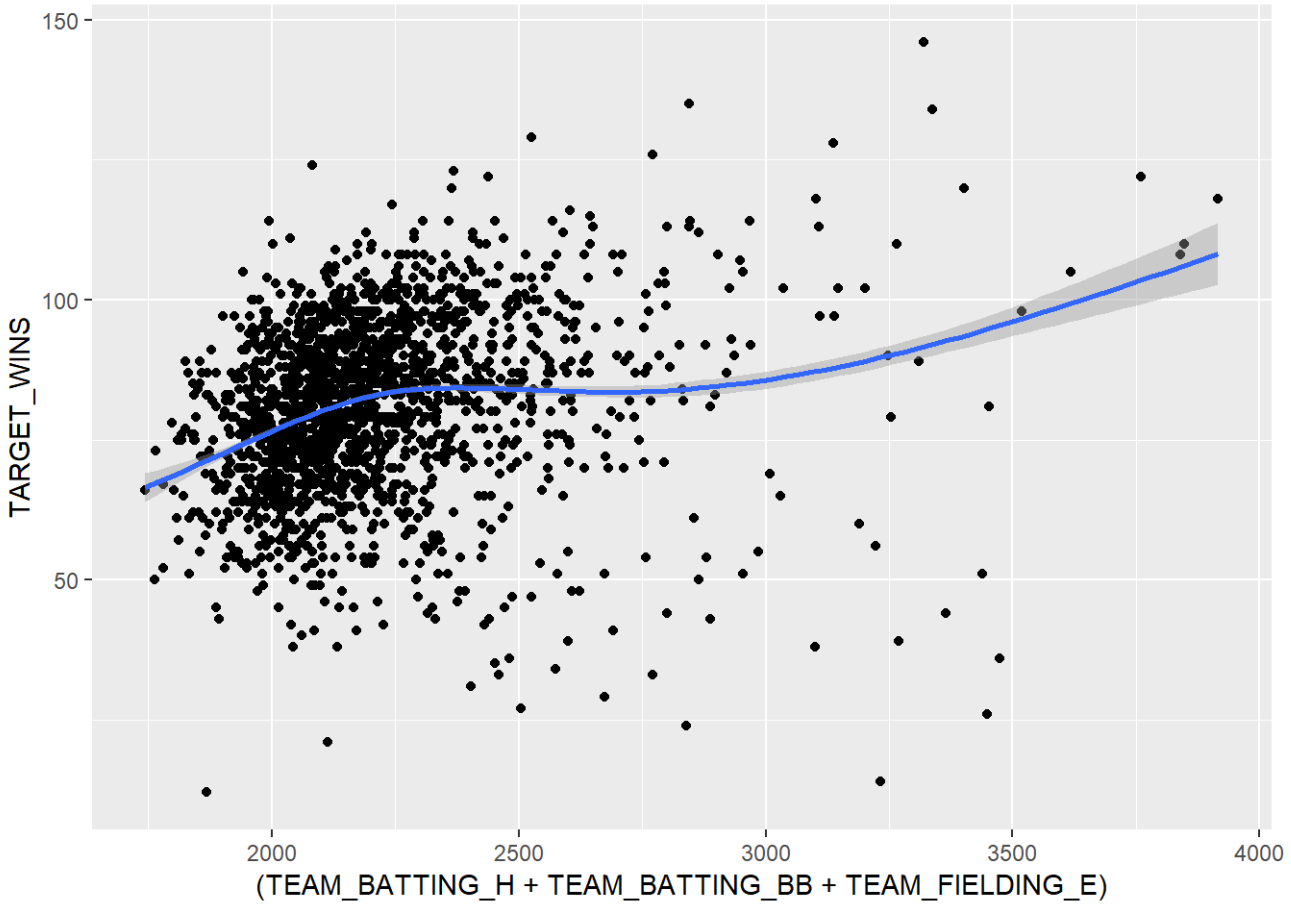


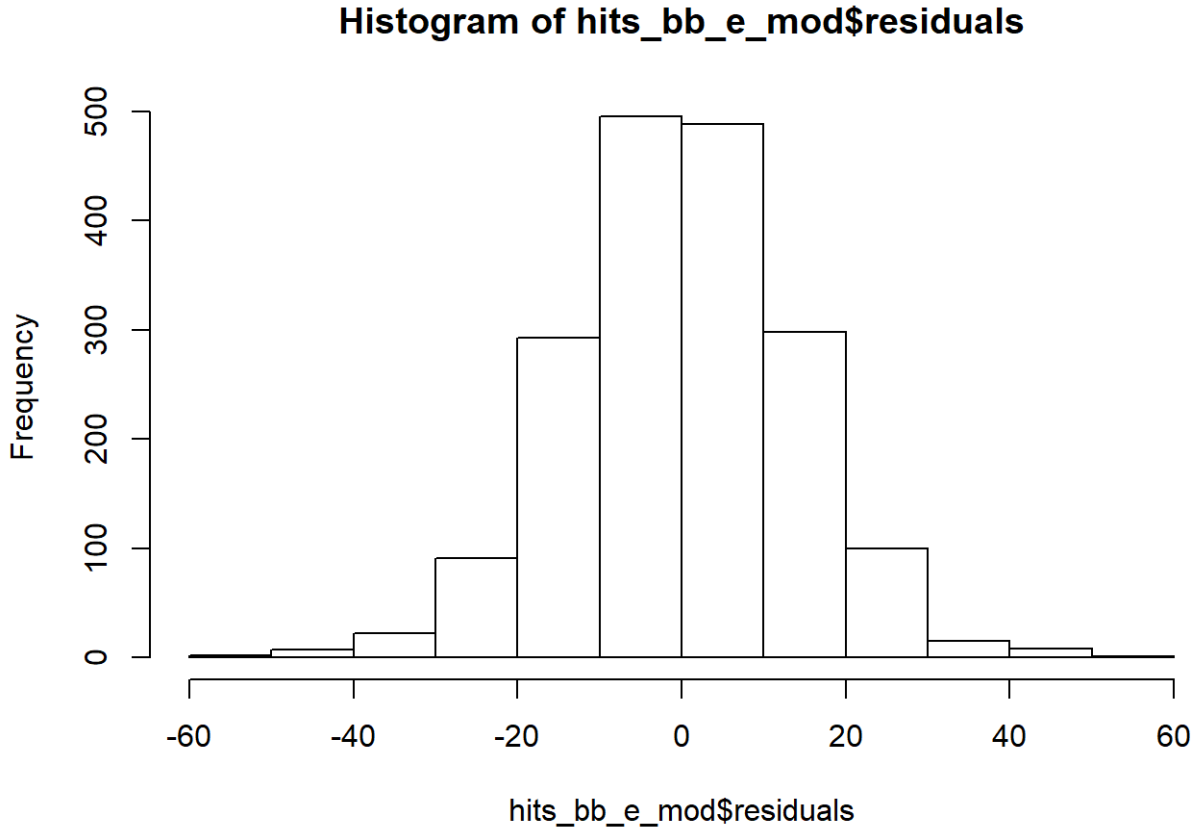


4.3 Model with Hits without Home Runs, BB's and Errors

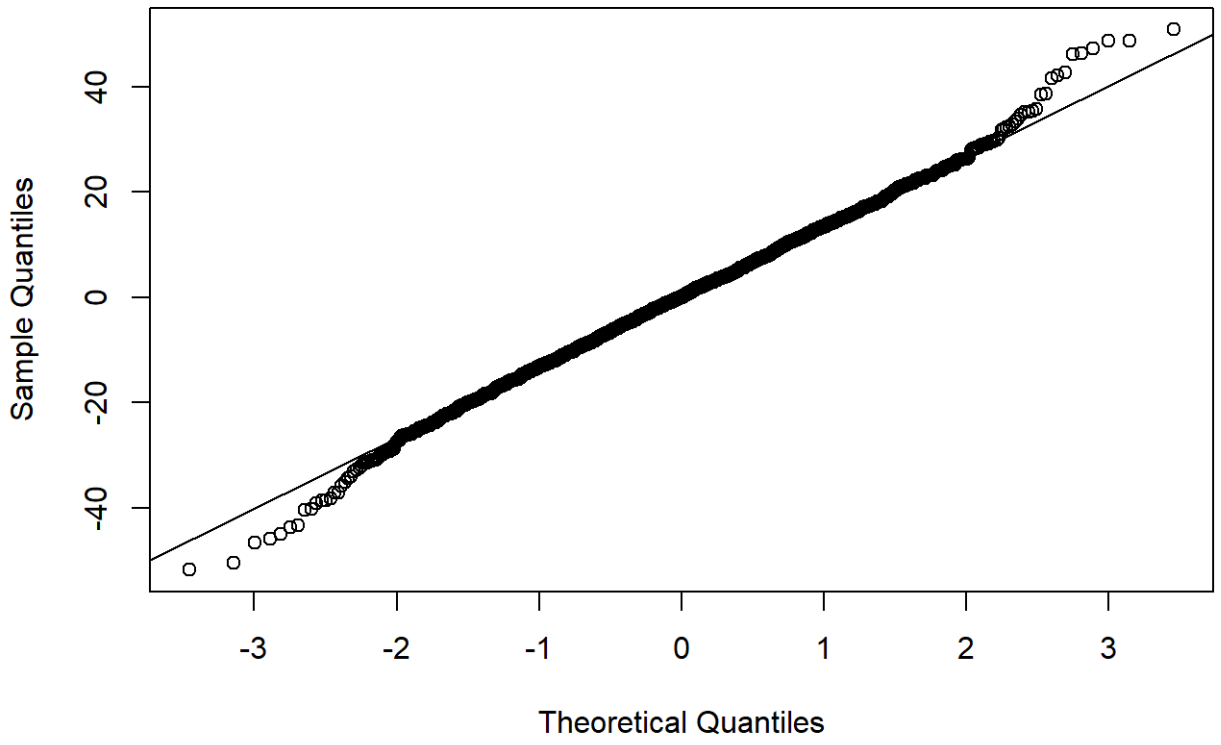
This model uses batting hits without homeruns, base batting on balls and fielding errors. PlotS look good except for short tailed issues in the QQ plot. The residuals are normal and there is no trend in the residual scatter plot or unusual skewness– the values look random.

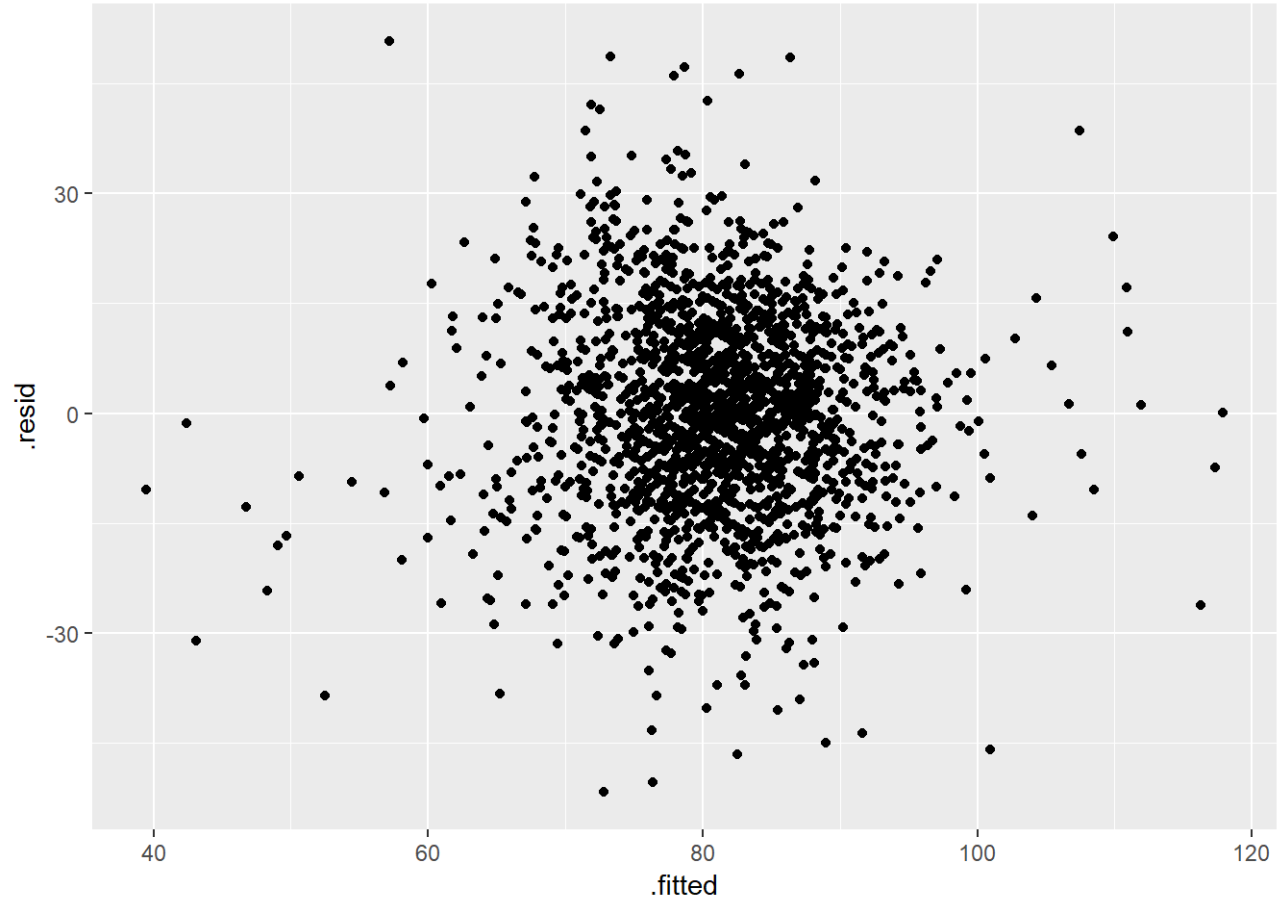
```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB +
##     TEAM_FIELDING_E, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.767  -8.959   0.003   9.071  50.788
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.277272   3.639218   1.175 0.240018
## TEAM_BATTING_H   0.050269   0.002305  21.813  < 2e-16 ***
## TEAM_BATTING_BB  0.012509   0.003507   3.567 0.000371 ***
## TEAM_FIELDING_E -0.014193   0.002022  -7.018 3.17e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.77 on 1816 degrees of freedom
## Multiple R-squared:  0.237,  Adjusted R-squared:  0.2357
## F-statistic:   188 on 3 and 1816 DF,  p-value: < 2.2e-16
```





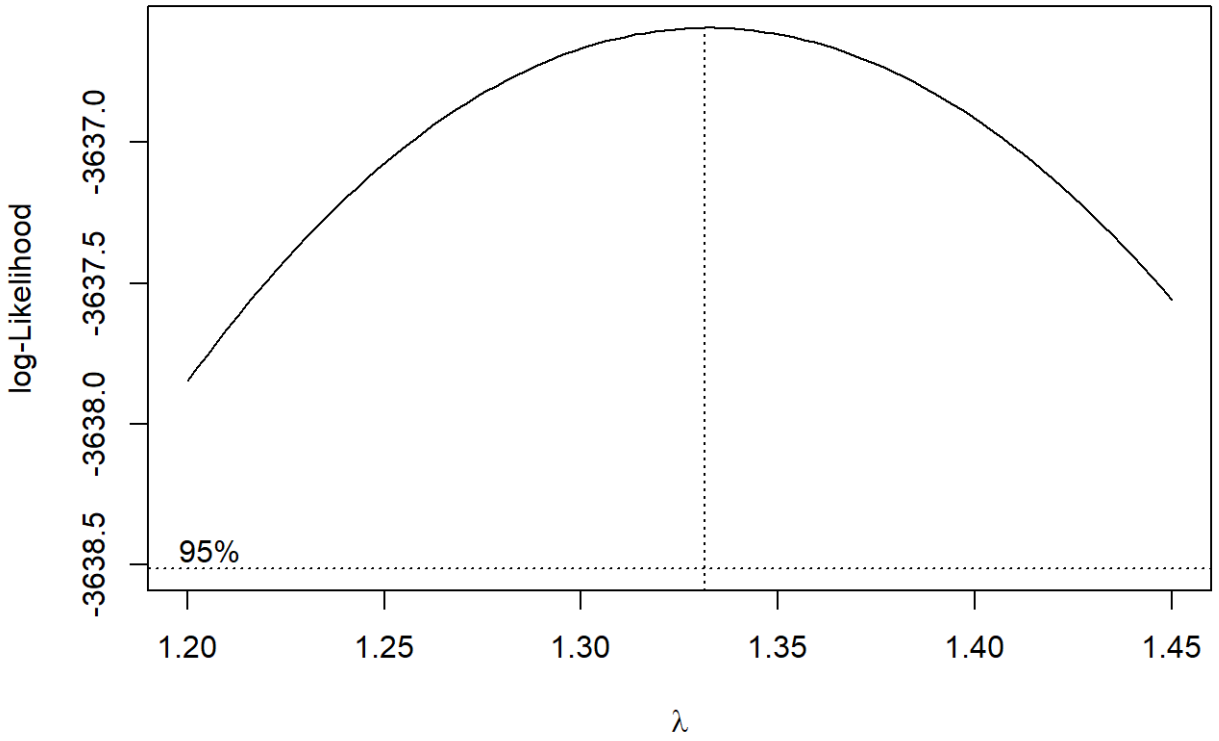
Normal Q-Q Plot



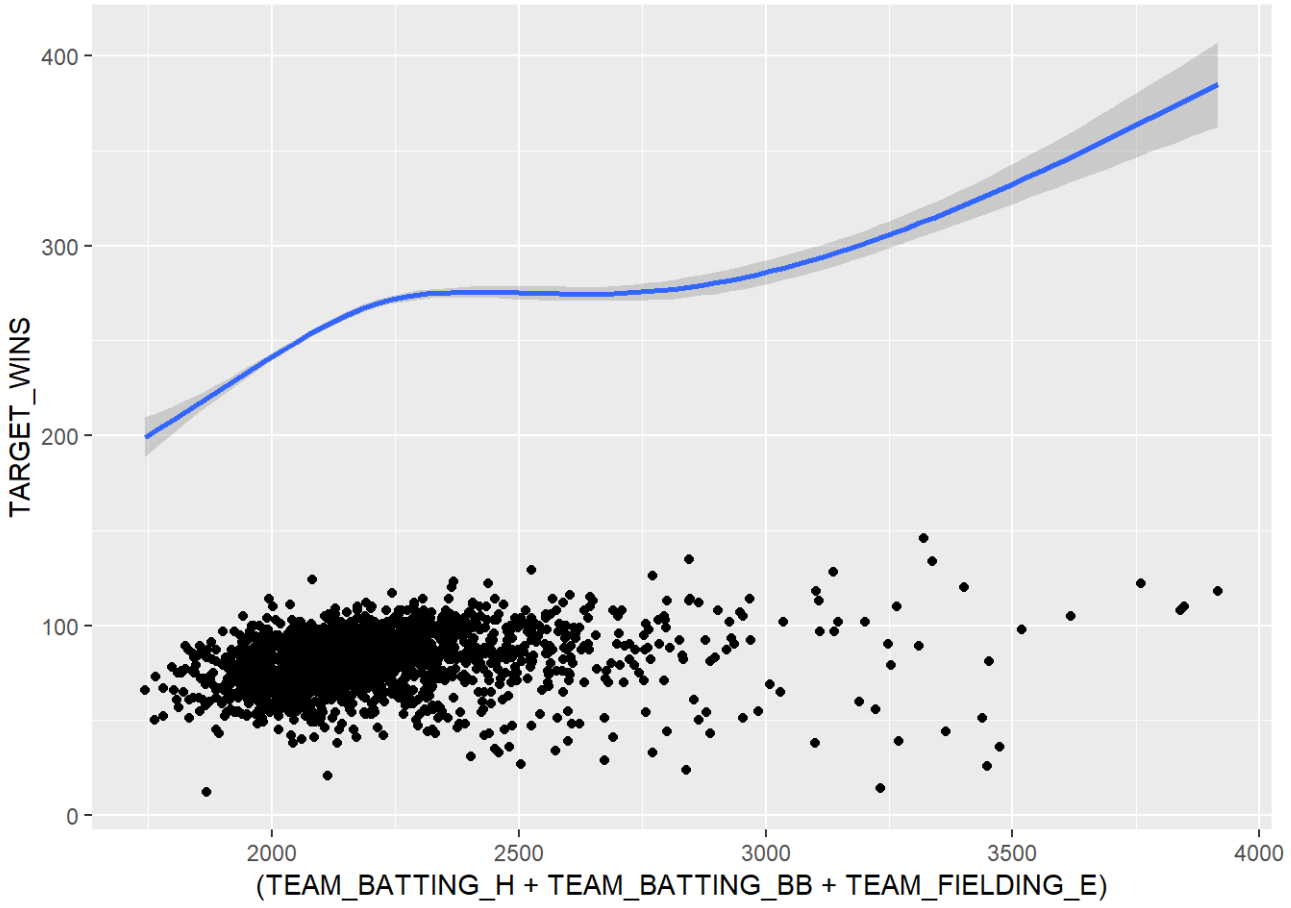


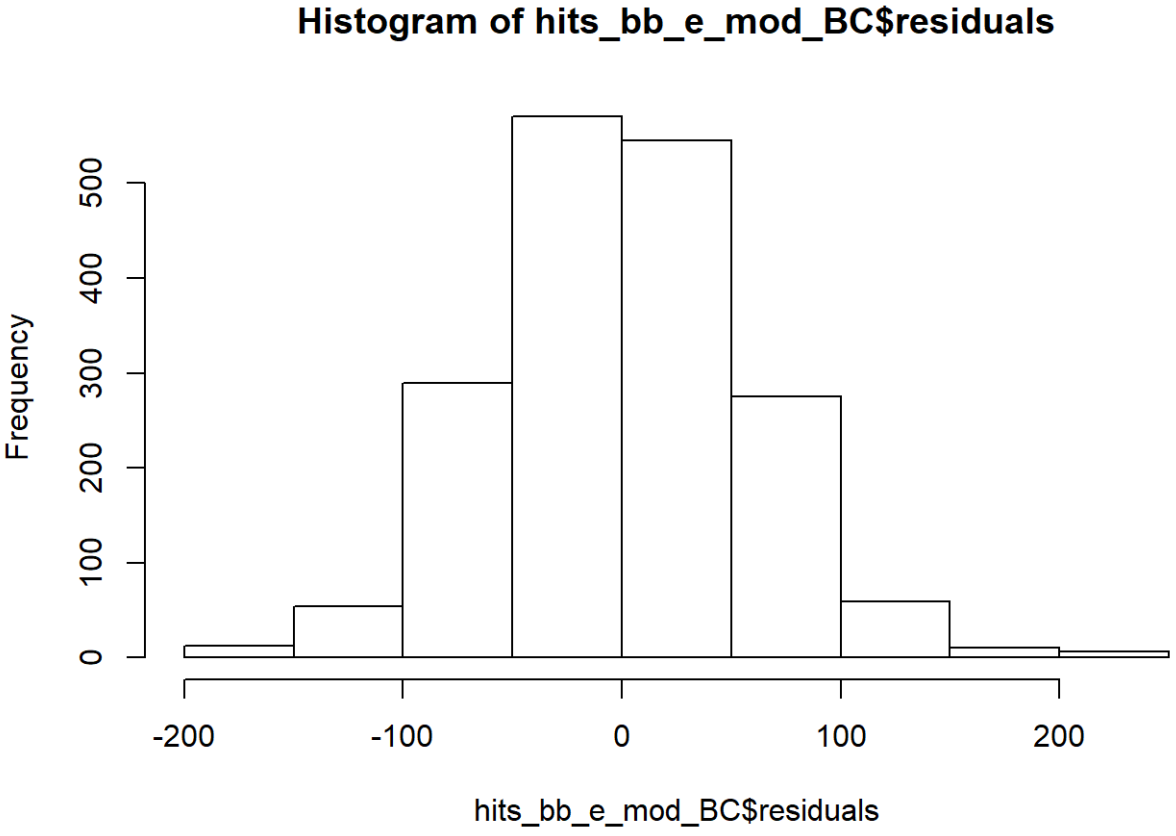
4.4 BoxCox Model

Attempt at boxcox, didn't achieve better results in R squared (not SE not directly comparable due to adjustment with boxcox)
The QQ plot seems to look a bit better as the negative quantiles are much closer to the line. Box Cox

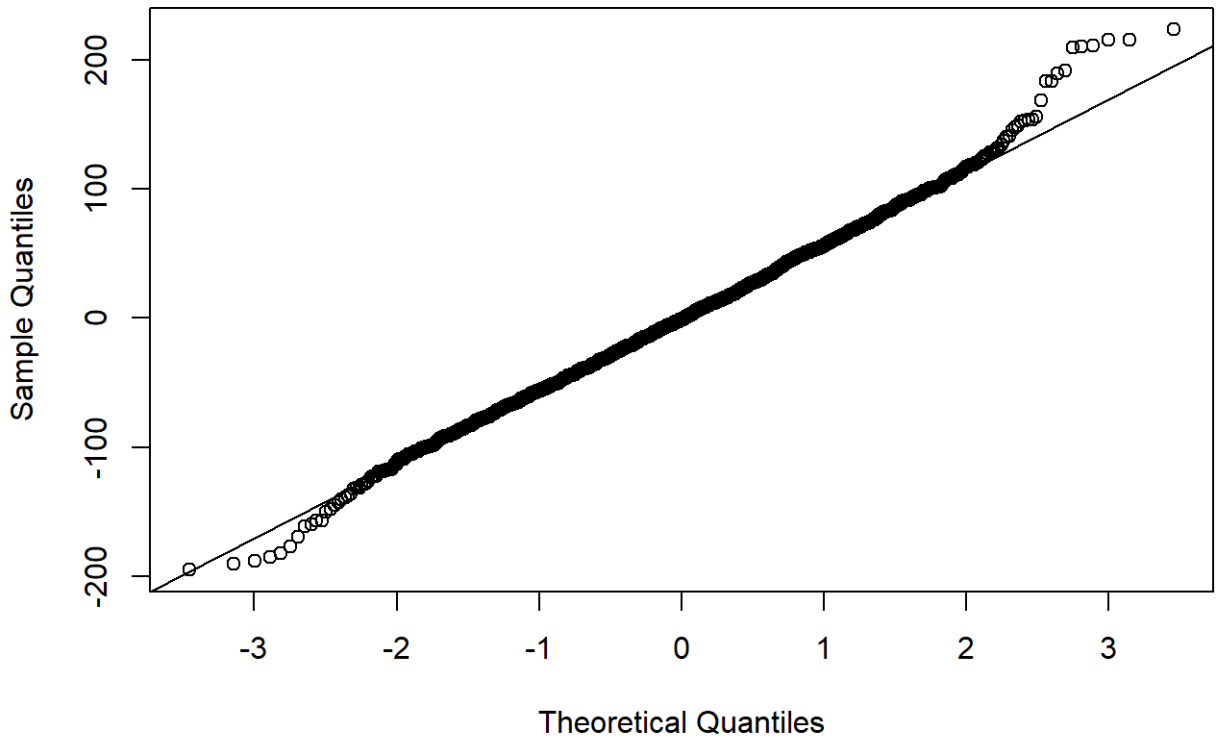


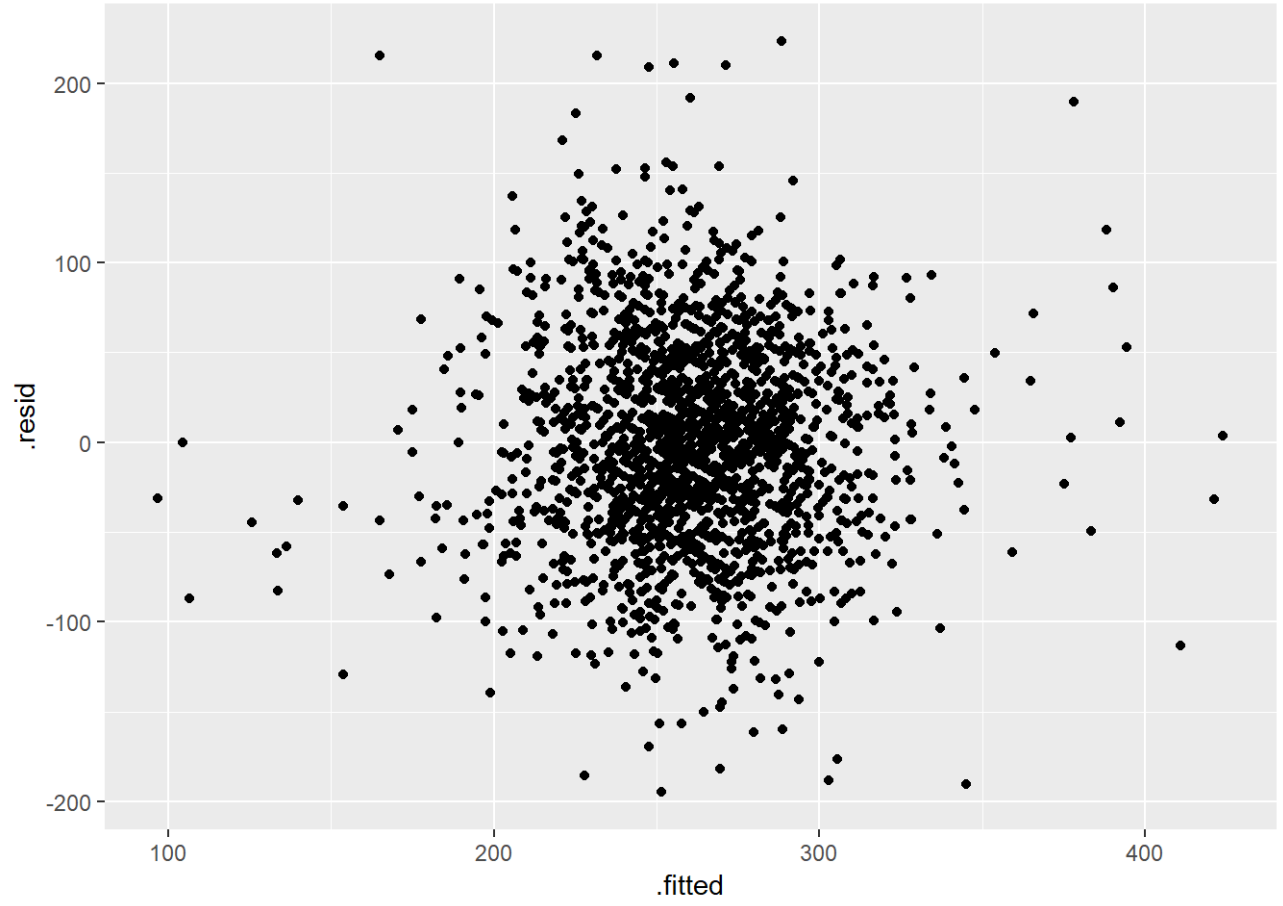
```
##
## Call:
## lm(formula = TARGET_WINS_BC ~ TEAM_BATTING_H + TEAM_BATTING_BB +
##     TEAM_FIELDING_E)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -194.979  -38.736   -1.184   37.665  223.257
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -64.017547  15.331986  -4.175 3.12e-05 ***
## TEAM_BATTING_H    0.211530   0.009709  21.787  < 2e-16 ***
## TEAM_BATTING_BB   0.053146   0.014776   3.597 0.000331 ***
## TEAM_FIELDING_E  -0.051346   0.008520  -6.026 2.03e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 57.99 on 1816 degrees of freedom
## Multiple R-squared:  0.2317, Adjusted R-squared:  0.2305
## F-statistic: 182.6 on 3 and 1816 DF,  p-value: < 2.2e-16
```





Normal Q-Q Plot





4.5 Run against Test DataSet

We developed models against the “Train” dataset (a random 80% from the entire training csv file), and now let’s run against the “test” dataset (being the other 20%). We see comparatively similar results in this cross validation of our models, with the model we picked originally (Hits without hr’s, BB’s, and errors) performing best.

Batting:

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H, data = test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.737  -8.935   1.556  10.264  38.579
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    24.386164   7.942929   3.070  0.00227 **
## TEAM_BATTING_H  0.038053   0.005391   7.059 6.32e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15 on 454 degrees of freedom
## Multiple R-squared:  0.0989, Adjusted R-squared:  0.09691
## F-statistic: 49.83 on 1 and 454 DF,  p-value: 6.324e-12
```

Pitching

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_PITCHING_H + TEAM_PITCHING_HR,
##     data = test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -79.667 -10.231   1.105  10.841  41.895
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      56.063991   5.616953   9.981  < 2e-16 ***
## TEAM_PITCHING_H   0.011492   0.003315   3.466 0.000578 ***
## TEAM_PITCHING_HR  0.057024   0.012299   4.636 4.65e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.33 on 453 degrees of freedom
## Multiple R-squared:  0.06053,    Adjusted R-squared:  0.05638
## F-statistic: 14.59 on 2 and 453 DF,  p-value: 7.217e-07
```

Hits Without HomeRuns

```
##
## Call:
## lm(formula = TARGET_WINS ~ HITS_NOHR, data = test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.828  -9.885   2.126  10.988  36.153
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 48.568743   6.916120   7.023 8.00e-12 ***
## HITS_NOHR    0.023172   0.005033   4.604 5.39e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.44 on 454 degrees of freedom
## Multiple R-squared:  0.0446, Adjusted R-squared:  0.0425
## F-statistic:  21.2 on 1 and 454 DF,  p-value: 5.393e-06
```

Hits (no HR's), BB, Errors:

```
##
## Call:
## lm(formula = TARGET_WINS ~ HITS_NOHR + TEAM_BATTING_BB + TEAM_FIELDING_E,
##     data = test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.312  -8.808  -0.024   9.322  43.261
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      8.462942   7.439747   1.138    0.256
## HITS_NOHR        0.042862   0.004644   9.229  < 2e-16 ***
## TEAM_BATTING_BB  0.036533   0.006750   5.412 1.01e-07 ***
## TEAM_FIELDING_E -0.021295   0.003304  -6.445 2.97e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.32 on 452 degrees of freedom
## Multiple R-squared:  0.2921, Adjusted R-squared:  0.2874
## F-statistic: 62.16 on 3 and 452 DF,  p-value: < 2.2e-16
```

5 SELECT MODELS

5.1 Compare Model Statistics

Metric	Batting Model	Pitching Model	Hits BB Errors Model	BoxCox Model
RSE	14.40	15.20	13.77	57.99
R^2	0.1646	0.06883	0.237	0.2317
Adj. R^2	0.1642	0.06781	0.2357	0.2305
F Stat.	358.2	67.16	188	182.6

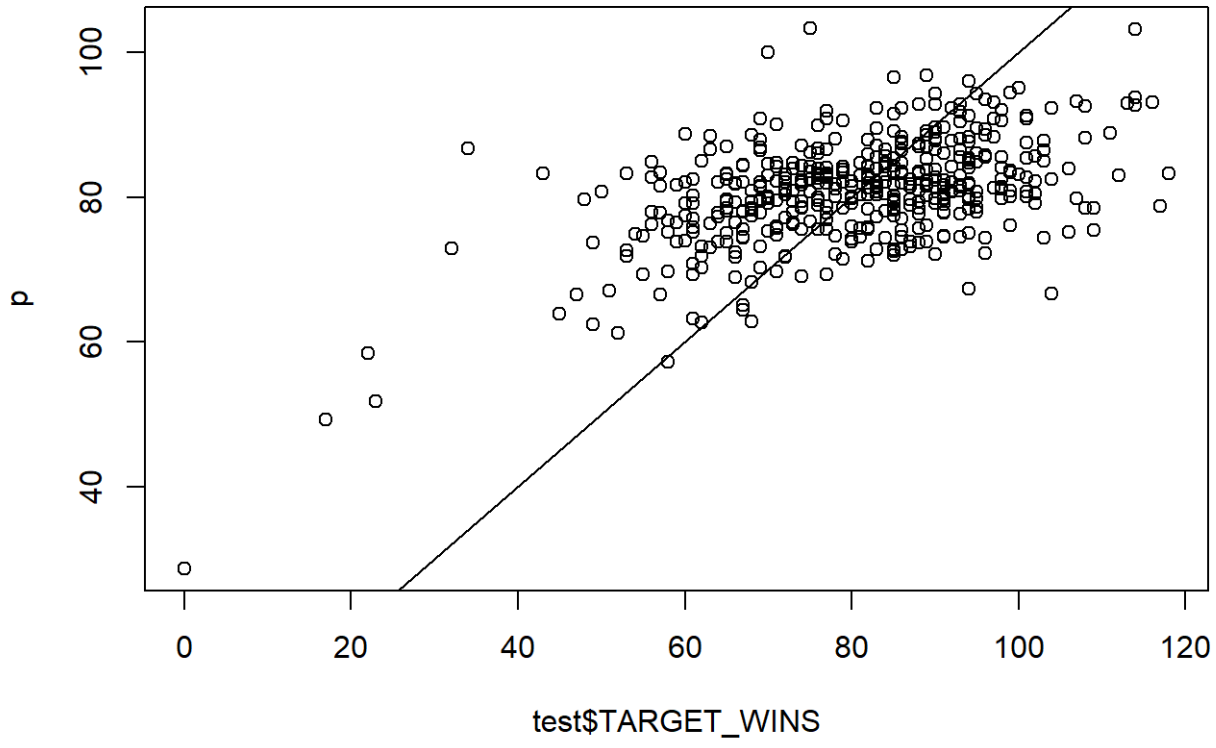
When comparing models we can see how they all describe a rather small amount of the variation of the output data TARGET_WINS. Although all model calculated were statistically significant, showing small p-values for the F statistic, the majority of them showed R squared and adjusted R squared of less than 20%. This included models using the calculated variables added to the dataset. A BoxCox transformation was also performed in an attempt to generate a better model, but while its results of around 23% was higher than most models, at least one simpler model was found to have an equal R square of ~23%.

Important to note is the fact that the RSE for the transformation BoxCox model can't really be used as a comparison with other models. this is because the underlaying variables have been changed or transformed in the process.

Residual plots were performed for all models and in general results were fairly similar and didn't have as much effect in selecting a model, but were useful in validating models.

5.2 Pick the best regression model

The chosen regression model was the Batting and Fielding Errors Model. It was the model with the highest achieved R and adjusted R square. It was chosen over the BoxCox transformation model because it is a simpler model. A plot of predicted test data was plotted against real test data to understand how well the model performs. If the model were perfect, all point would fall on the plotted line, that is predicted and true values would be the same.



5.3 Conclusion

In general all models generated in the analysis explained rather small amounts of the variation of the output Target_WINS variables. The transformation model showed the highest, but simpler models also explained about the same amount of variation of the TARGET_WINS data. One of these simpler models was selected over the transformation, mainly becouse of its simplicity.

The model selected was generated using 80% of the training dataset. The remaining 20% was also used in cross validation to generate a similar model using the same predictor variables. Both models showed similar results, with the R square of the 20% data actually showing slightly higher. This is a good indication of lack of bias in the model, that is overfitting the model to the dataset given since a similar model was generated using different data.

Selected model was used to predict TARGET_WINS values from data in Evaluation data file, results were written to a csv file: predicted_eval_values.csv

6 APPENDIX

Code used in analysis

Find counts of na's Notice that #hbp and cs (hit by pitch and caught stealing) have very large numbers of missing values, probably shouldn't use, and knowing baseball, probably minor impact on wins regardless, so remove those columns…

```
mbTrain <- read.csv("moneyball-training-data.csv")
sapply(mbTrain, function(x) sum(is.na(x)))
```

```
##            INDEX      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B
##                0                0                0                0
##  TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO
##                0                0                0              102
##  TEAM_BASERUN_SB  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H
##              131              772             2085                0
## TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E
##                0                0              102                0
## TEAM_FIELDING_DP
##              286
```

```
#remove those two columns, plus the index colume which we don't need
mbTrain <- mbTrain[, -which(names(mbTrain) %in% c("INDEX", "TEAM_BATTING_HBP", "TEAM_BASERUN_CS"))]
```

convert NA's to -1 for now so we can run test on zero's as zero's for any of these stats clearly is inaccurate and eqivalent to an NA

Number of zeros fields is not so bad, but lets not replace them until we check for outliers and convert NA's back to -1

```
mbTrain[is.na(mbTrain)] <- -1
sapply(mbTrain, function(x) sum(x == 0))
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##                1                0                0                2
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##               15                1               20                2
##  TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO
##                0               15                1               20
##  TEAM_FIELDING_E TEAM_FIELDING_DP
##                0                0
```

Running summary we can see that there are some really big outliers (e.g., 19278 Strikeouts in one year?, that would be 119 per game….) at initial glance (can always revisit), looks like Pitching_Hits, Pitching_BB, Pitching_SO and Fielding Errors all have outliers that are not realistic values

```
summary(mbTrain)
```

```
##   TARGET_WINS     TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
##  Min.   :  0.00   Min.   : 891   Min.   : 69.0   Min.   :  0.00
##  1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0   1st Qu.: 34.00
##  Median : 82.00   Median :1454   Median :238.0   Median : 47.00
##  Mean   : 80.79   Mean   :1469   Mean   :241.2   Mean   : 55.25
##  3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0   3rd Qu.: 72.00
##  Max.   :146.00   Max.   :2554   Max.   :458.0   Max.   :223.00
##  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB
##  Min.   :  0.00   Min.   :  0.0   Min.   :  -1.0   Min.   : -1.0
##  1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 524.0   1st Qu.: 60.0
##  Median :102.00   Median :512.0   Median : 728.0   Median : 97.0
##  Mean   : 99.61   Mean   :501.6   Mean   : 702.6   Mean   :117.5
##  3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 925.0   3rd Qu.:151.0
##  Max.   :264.00   Max.   :878.0   Max.   :1399.0   Max.   :697.0
##  TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO
##  Min.   : 1137   Min.   :  0.0    Min.   :   0.0   Min.   :   -1.0
##  1st Qu.: 1419   1st Qu.: 50.0    1st Qu.: 476.0   1st Qu.:  587.8
##  Median : 1518   Median :107.0    Median : 536.5   Median :  797.0
##  Mean   : 1779   Mean   :105.7    Mean   : 553.0   Mean   :  781.0
##  3rd Qu.: 1682   3rd Qu.:150.0    3rd Qu.: 611.0   3rd Qu.:  957.0
##  Max.   :30132   Max.   :343.0    Max.   :3645.0   Max.   :19278.0
##  TEAM_FIELDING_E  TEAM_FIELDING_DP
##  Min.   :  65.0   Min.   : -1.0
##  1st Qu.: 127.0   1st Qu.:118.0
##  Median : 159.0   Median :145.0
##  Mean   : 246.5   Mean   :127.9
##  3rd Qu.: 249.2   3rd Qu.:161.2
##  Max.   :1898.0   Max.   :228.0
```

```
#looking at top values we can see that there maybe a number of years that are unrealistic
head(sort(mbTrain$TEAM_PITCHING_H, decreasing = TRUE))
```

```
## [1] 30132 24057 20088 16871 16038 14749
```

```
head(sort(mbTrain$TEAM_PITCHING_BB, decreasing = TRUE))
```

```
## [1] 3645 2876 2840 2396 2169 1750
```

```
head(sort(mbTrain$TEAM_PITCHING_SO, decreasing = TRUE))
```

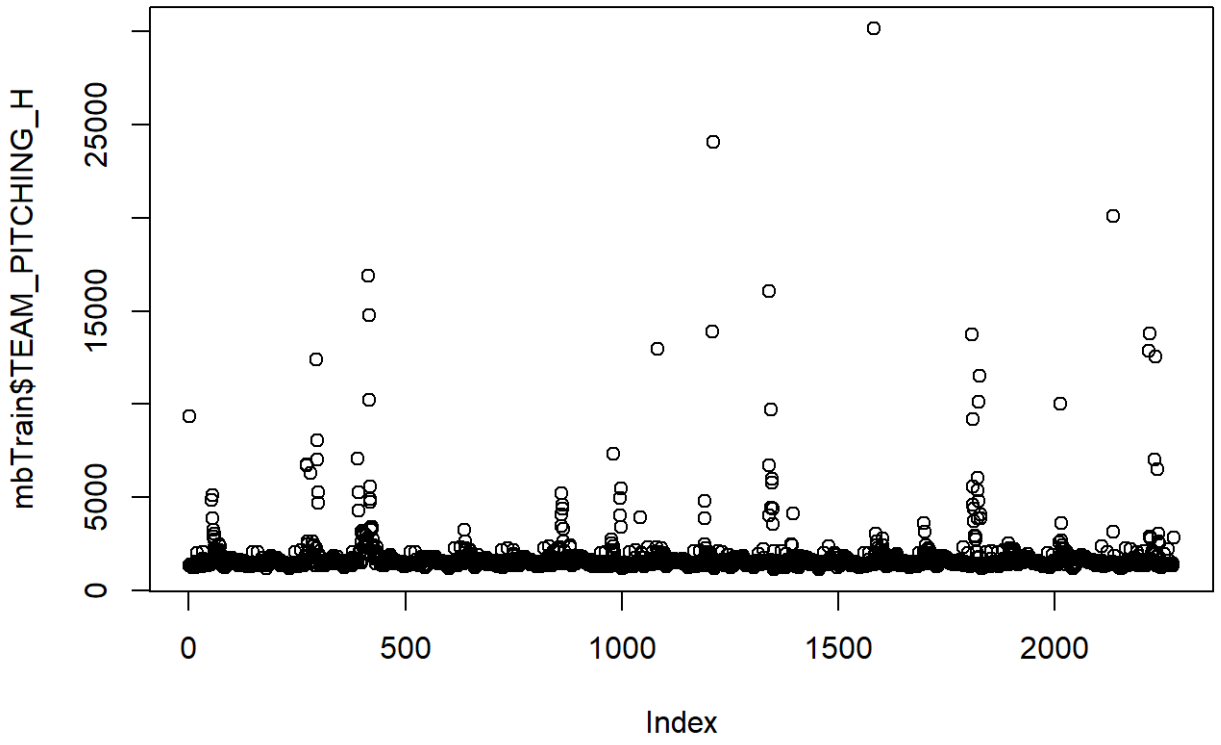
```
## [1] 19278 12758  5456  4224  3450  2492
```

```
head(sort(mbTrain$TEAM_FIELDING_E, decreasing = TRUE))
```

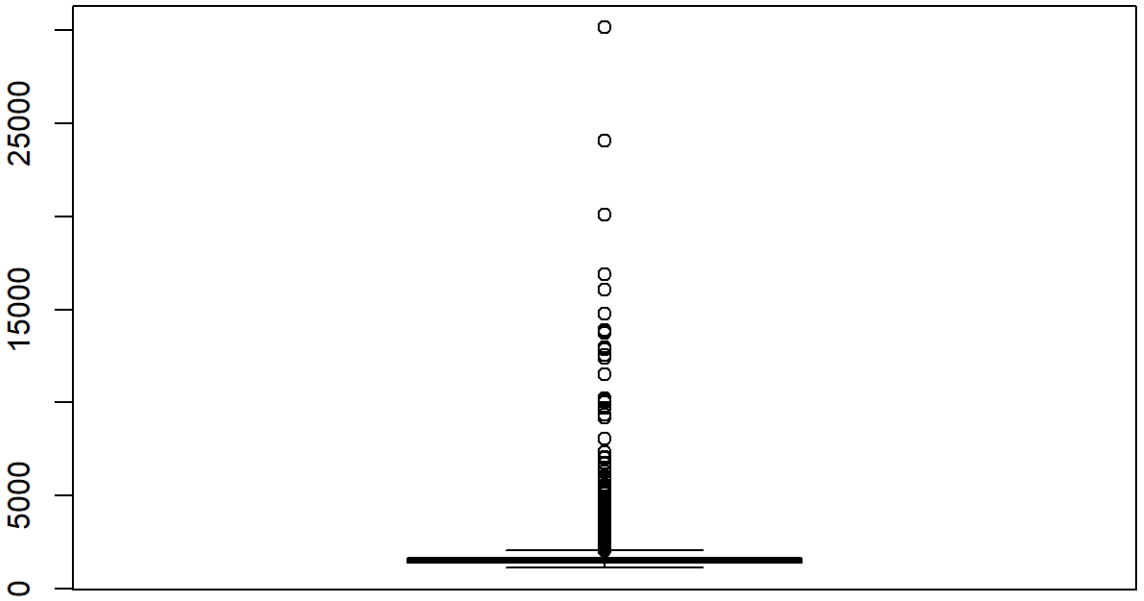
```
## [1] 1898 1890 1740 1728 1567 1553
```

Let's plot to see where outliers are

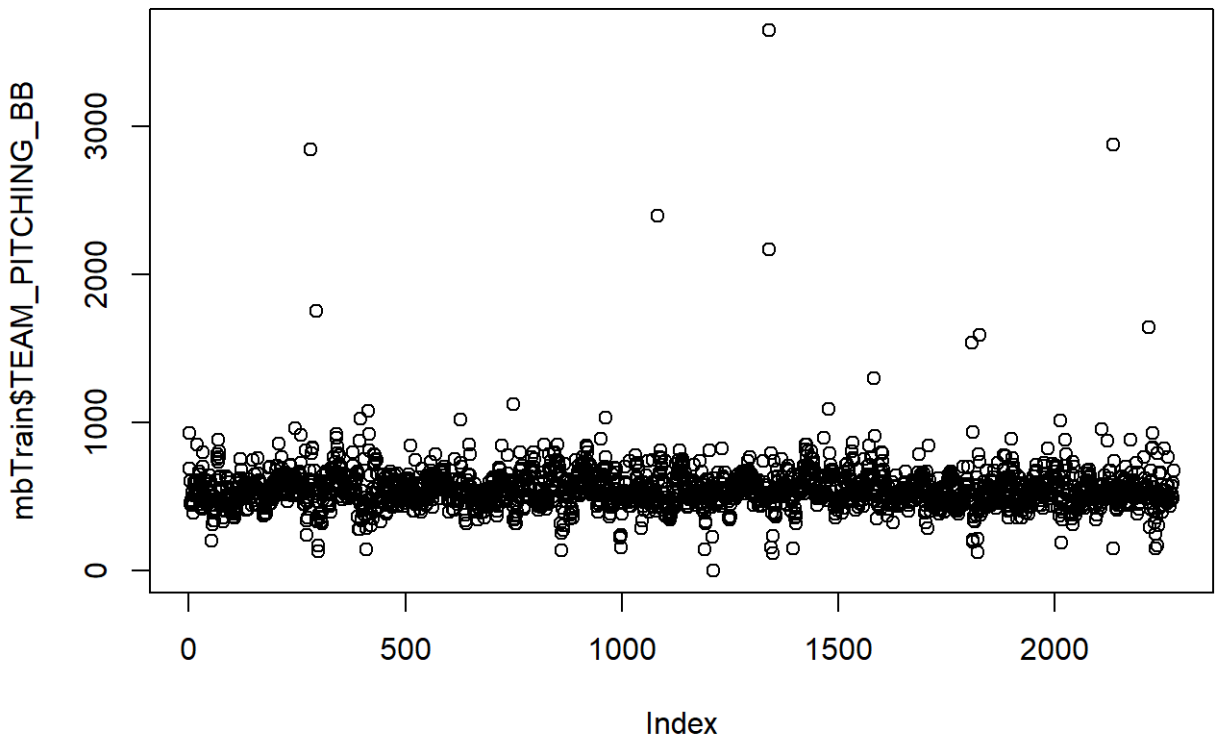
```
plot(mbTrain$TEAM_PITCHING_H)
```



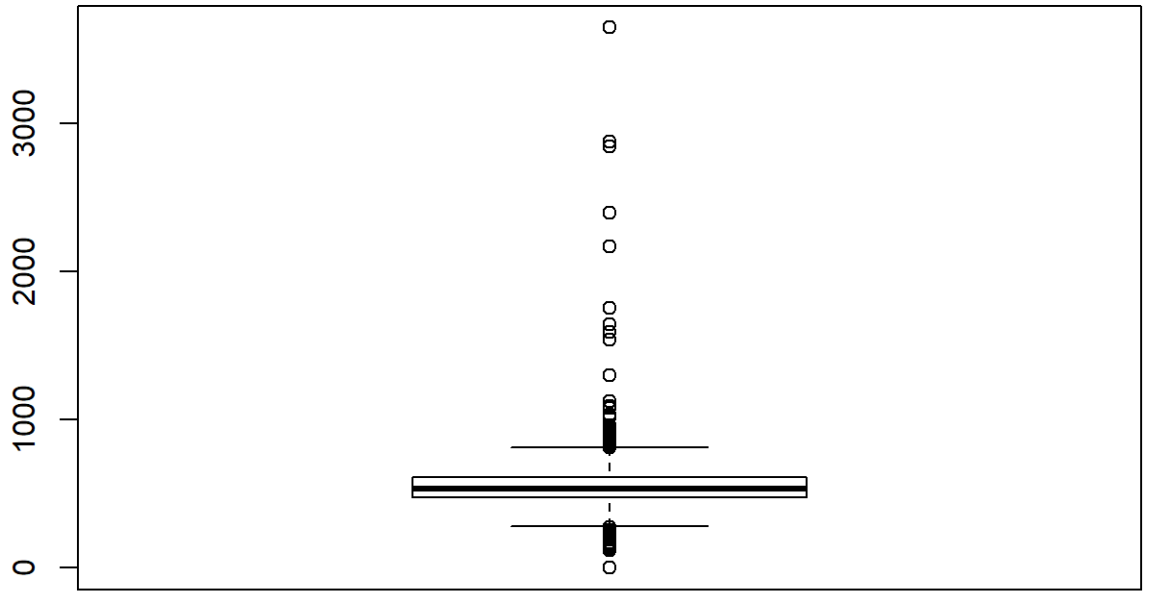

```
boxplot(mbTrain$TEAM_PITCHING_H)
```



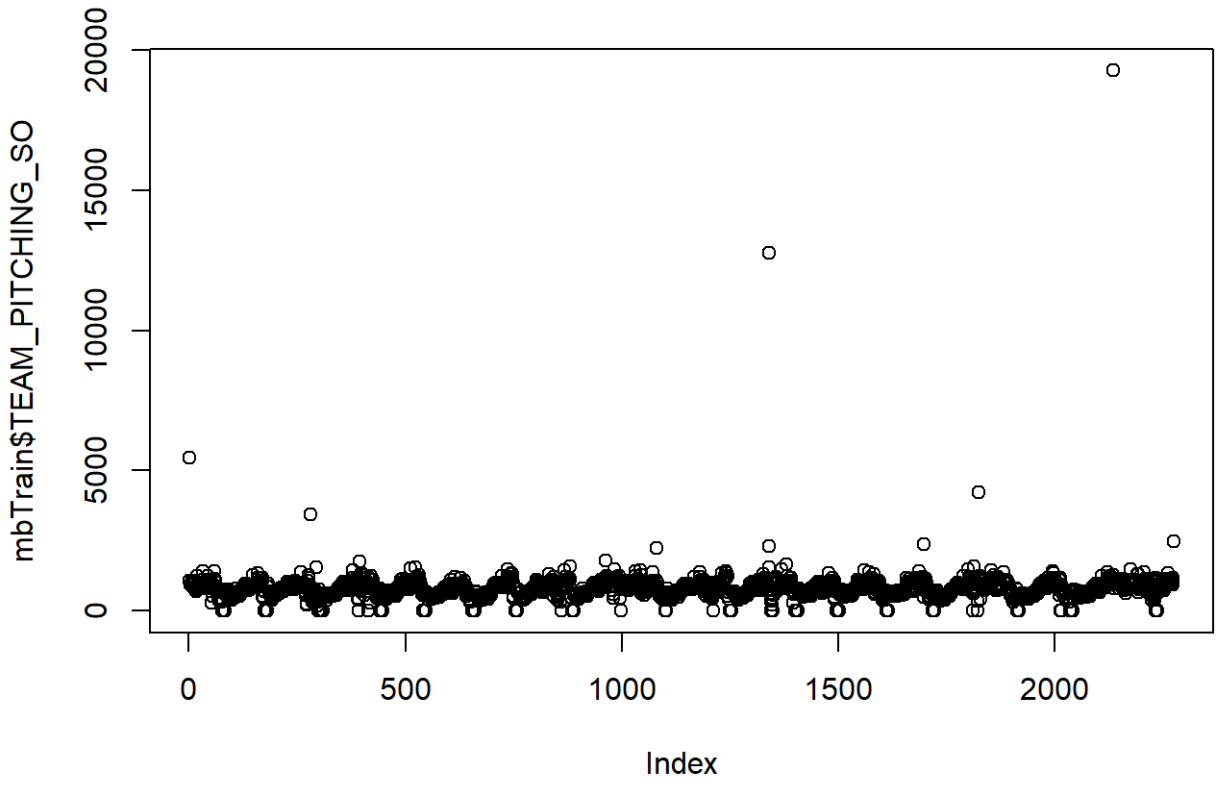

```
plot(mbTrain$TEAM_PITCHING_BB)
```



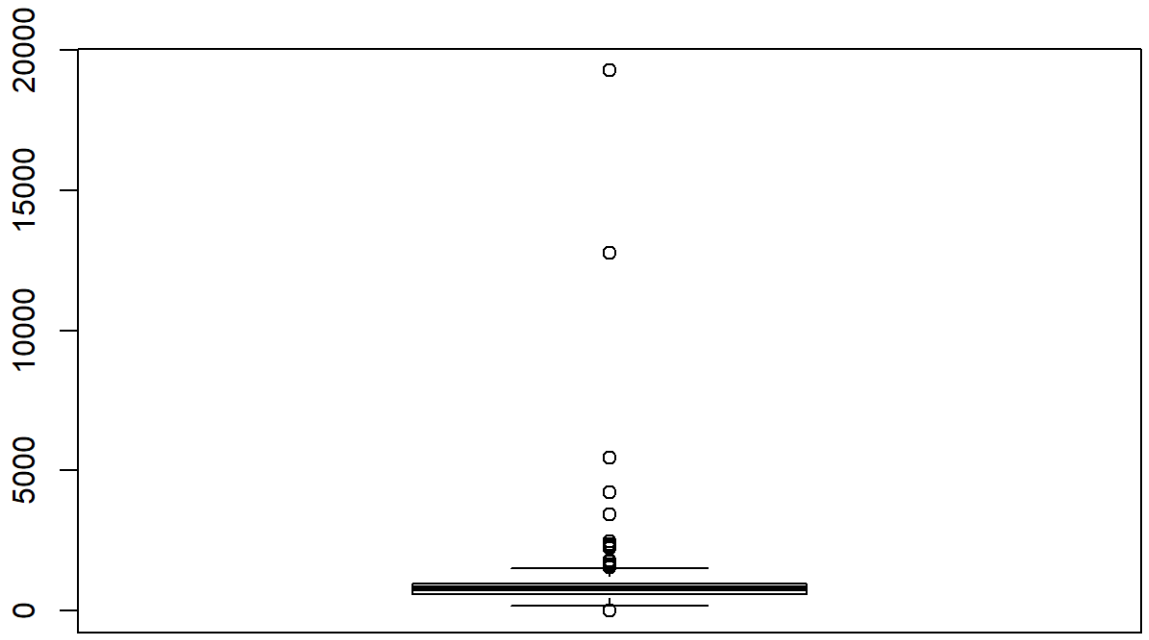

```
boxplot(mbTrain$TEAM_PITCHING_BB)
```



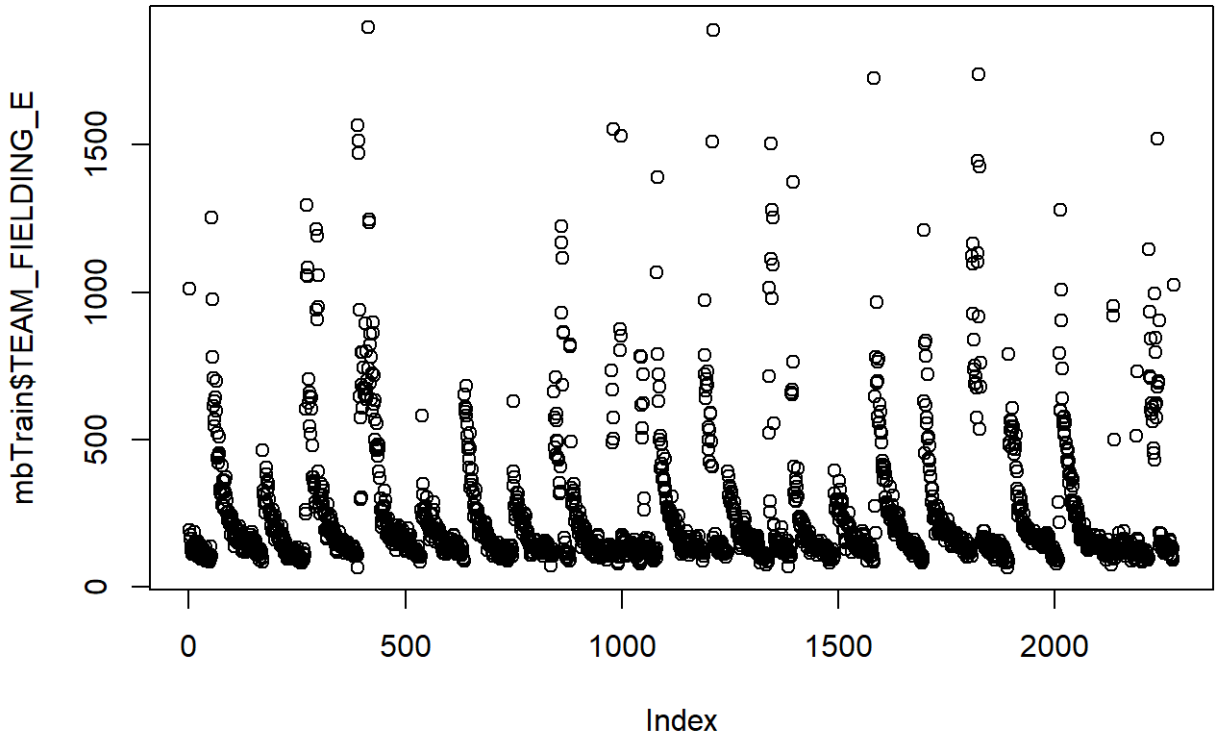

```
plot(mbTrain$TEAM_PITCHING_SO)
```



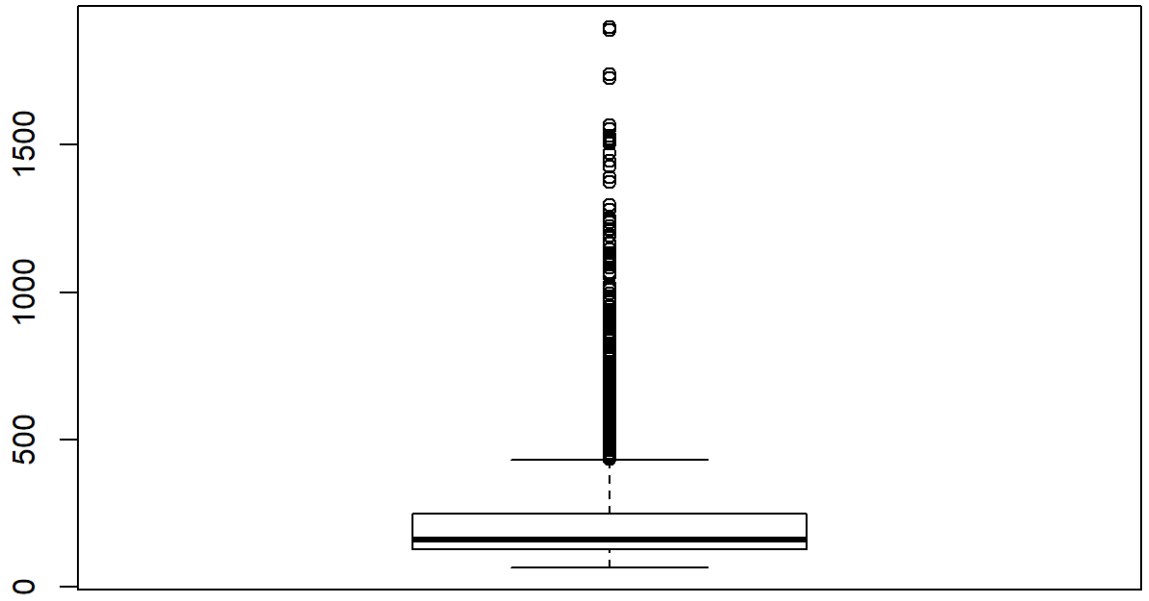

```
boxplot(mbTrain$TEAM_PITCHING_SO)
```




```
plot(mbTrain$TEAM_FIELDING_E)
```




```
boxplot(mbTrain$TEAM_FIELDING_E)
```



There are a lot of rows with pitching hits greater than 3000 (there was max of 2554 hits by batting team, so a limit of 3000 hits (18.5 avg per game) by a pitching team doesn't seem so unreasonable as max)we see 86, these should probably be tossed and replaced with median (mean?)

```
sum(mbTrain$TEAM_PITCHING_H > 3000)
```

```
## [1] 86
```

Using same idea for base on balls, max for batting was 878, so perhaps a 1200 max for pitching (7.5 avg a game) might be reasonable, so 10 rows like this.

```
sum(mbTrain$TEAM_PITCHING_BB > 1200)
```

```
## [1] 10
```

for strikeouts, batting max of 1399, so perhaps 1800 for pitching max, and we have 9 rows like this.

```
## [1] 9
```

For fielding errors, there is no corresponding number to validate against (like with pitching compared to batting), but if we use a plausible limit of about 10 per game (which is still really high we can eliminate clear outliers).

```
## [1] 4
```

Now that we have identified them, let's replace all these values

```
mbTrain$TEAM_PITCHING_H[mbTrain$TEAM_PITCHING_H>max(mbTrain$TEAM_BATTING_H)] <- median(mbTrain$TEAM_PITCHI
NG_H, na.rm=TRUE)
mbTrain$TEAM_PITCHING_SO[mbTrain$TEAM_PITCHING_SO>max(mbTrain$TEAM_BATTING_SO)] <- median(mbTrain$TEAM_PIT
CHING_SO, na.rm=TRUE)
```

Do some correlation

```
##      TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B TEAM_BATTING_HR
## [1,]      0.3887675       0.2891036       0.1426084       0.1761532
##      TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB TEAM_PITCHING_H
## [1,]       0.2325599     -0.03009449       0.1207944       0.1808751
##      TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDING_E
## [1,]        0.1890137        0.1241745      -0.05078415      -0.1764848
##      TEAM_FIELDING_DP
## [1,]       0.01904895
```

Do some basic plots to start looking at data:

```
attach(mbTrain)
```

```
## The following objects are masked from test:
##
##     TARGET_WINS, TEAM_BASERUN_SB, TEAM_BATTING_2B,
##     TEAM_BATTING_3B, TEAM_BATTING_BB, TEAM_BATTING_H,
##     TEAM_BATTING_HR, TEAM_BATTING_SO, TEAM_FIELDING_DP,
##     TEAM_FIELDING_E, TEAM_PITCHING_BB, TEAM_PITCHING_H,
##     TEAM_PITCHING_HR, TEAM_PITCHING_SO
```

```
## The following objects are masked from train (pos = 4):
##
##     TARGET_WINS, TEAM_BASERUN_SB, TEAM_BATTING_2B,
##     TEAM_BATTING_3B, TEAM_BATTING_BB, TEAM_BATTING_H,
##     TEAM_BATTING_HR, TEAM_BATTING_SO, TEAM_FIELDING_DP,
##     TEAM_FIELDING_E, TEAM_PITCHING_BB, TEAM_PITCHING_H,
##     TEAM_PITCHING_HR, TEAM_PITCHING_SO
```

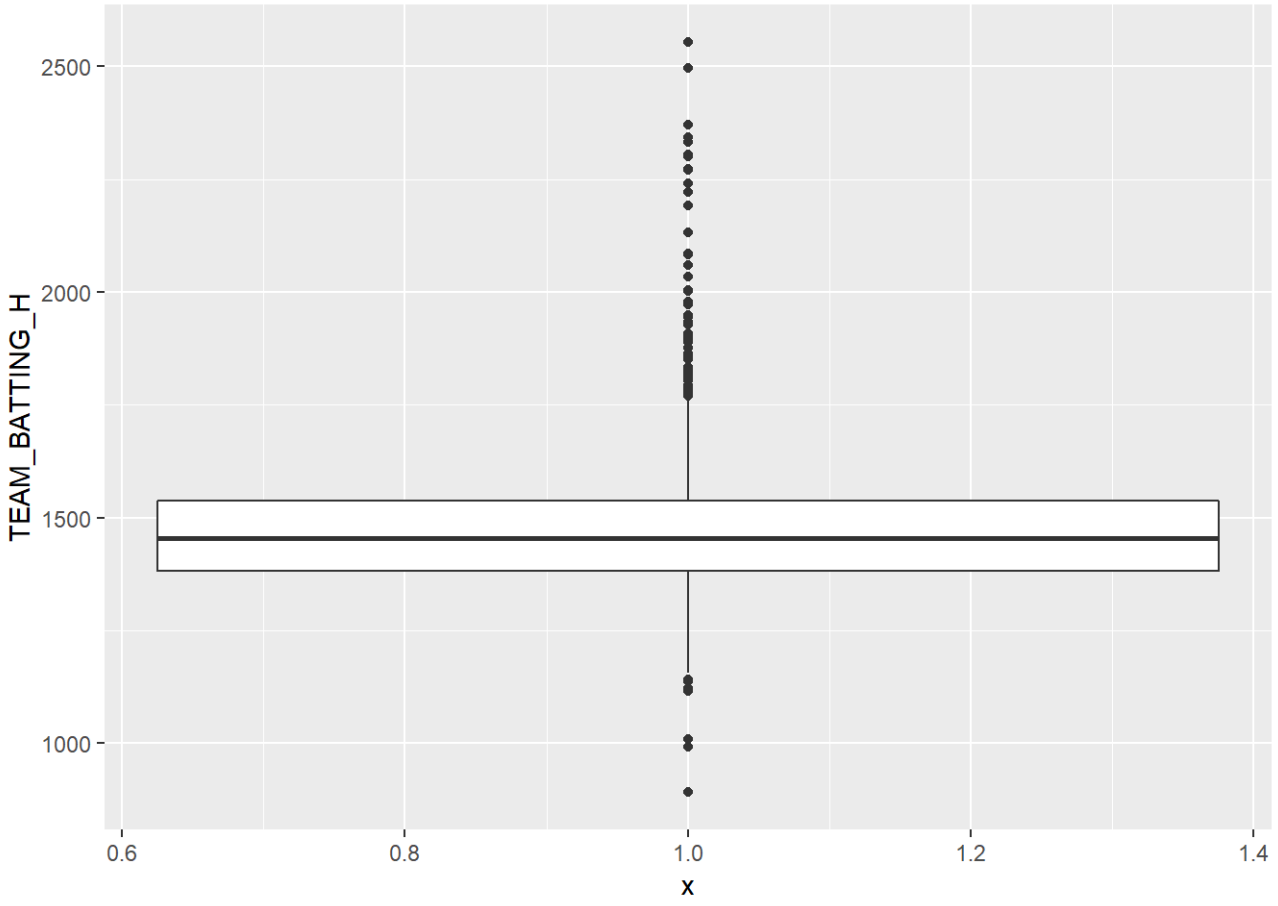
```
## The following objects are masked from train (pos = 5):
##
##     TARGET_WINS, TEAM_BASERUN_SB, TEAM_BATTING_2B,
##     TEAM_BATTING_3B, TEAM_BATTING_BB, TEAM_BATTING_H,
##     TEAM_BATTING_HR, TEAM_BATTING_SO, TEAM_FIELDING_DP,
##     TEAM_FIELDING_E, TEAM_PITCHING_BB, TEAM_PITCHING_H,
##     TEAM_PITCHING_HR, TEAM_PITCHING_SO
```

```
## The following objects are masked from mbTrain (pos = 6):
##
##     TARGET_WINS, TEAM_BASERUN_SB, TEAM_BATTING_2B,
##     TEAM_BATTING_3B, TEAM_BATTING_BB, TEAM_BATTING_H,
##     TEAM_BATTING_HR, TEAM_BATTING_SO, TEAM_FIELDING_DP,
##     TEAM_FIELDING_E, TEAM_PITCHING_BB, TEAM_PITCHING_H,
##     TEAM_PITCHING_HR, TEAM_PITCHING_SO
```

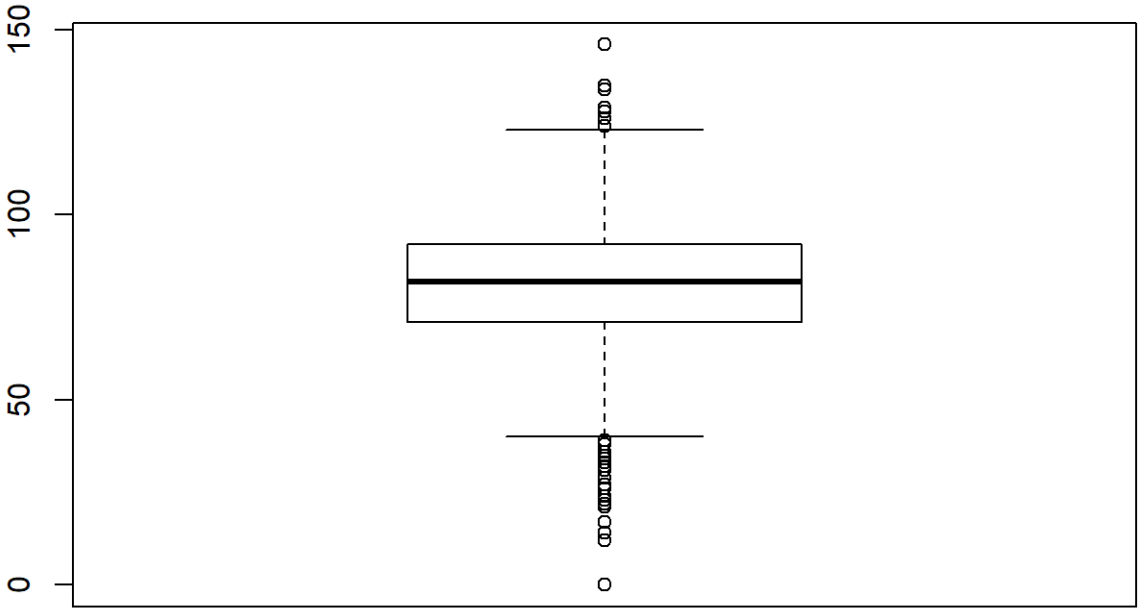
```
## The following objects are masked from mbTrain (pos = 7):
##
##     TARGET_WINS, TEAM_BASERUN_SB, TEAM_BATTING_2B,
##     TEAM_BATTING_3B, TEAM_BATTING_BB, TEAM_BATTING_H,
##     TEAM_BATTING_HR, TEAM_BATTING_SO, TEAM_FIELDING_DP,
##     TEAM_FIELDING_E, TEAM_PITCHING_BB, TEAM_PITCHING_H,
##     TEAM_PITCHING_HR, TEAM_PITCHING_SO
```

```
## The following objects are masked from mbTrain[, -1]:
##
##     TARGET_WINS, TEAM_BASERUN_SB, TEAM_BATTING_2B,
##     TEAM_BATTING_3B, TEAM_BATTING_BB, TEAM_BATTING_H,
##     TEAM_BATTING_HR, TEAM_BATTING_SO, TEAM_FIELDING_DP,
##     TEAM_FIELDING_E, TEAM_PITCHING_BB, TEAM_PITCHING_H,
##     TEAM_PITCHING_HR, TEAM_PITCHING_SO
```

```
ggplot(mbTrain, aes(y=TEAM_BATTING_H, x= 1)) + geom_boxplot()
```

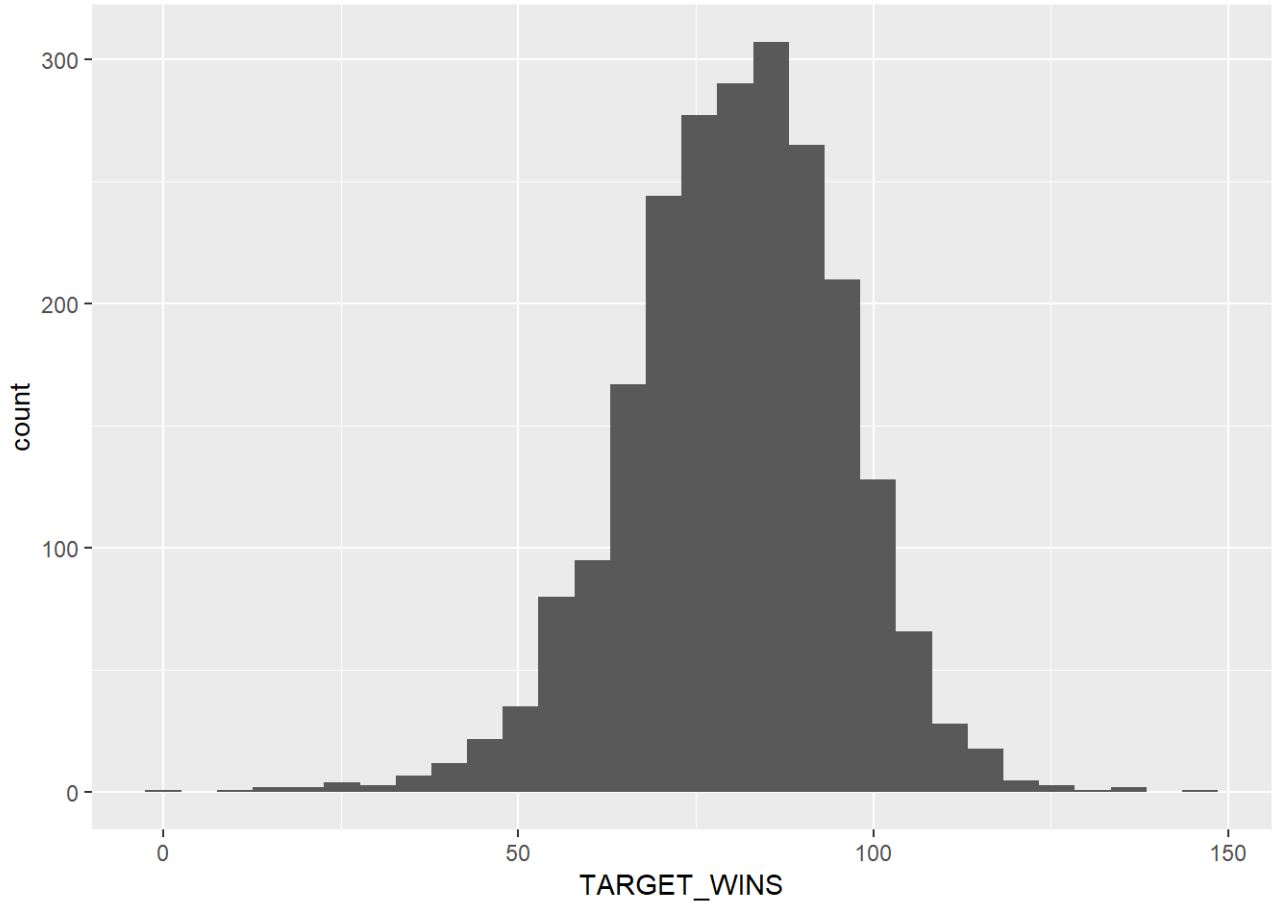



```
boxplot(TARGET_WINS)
```



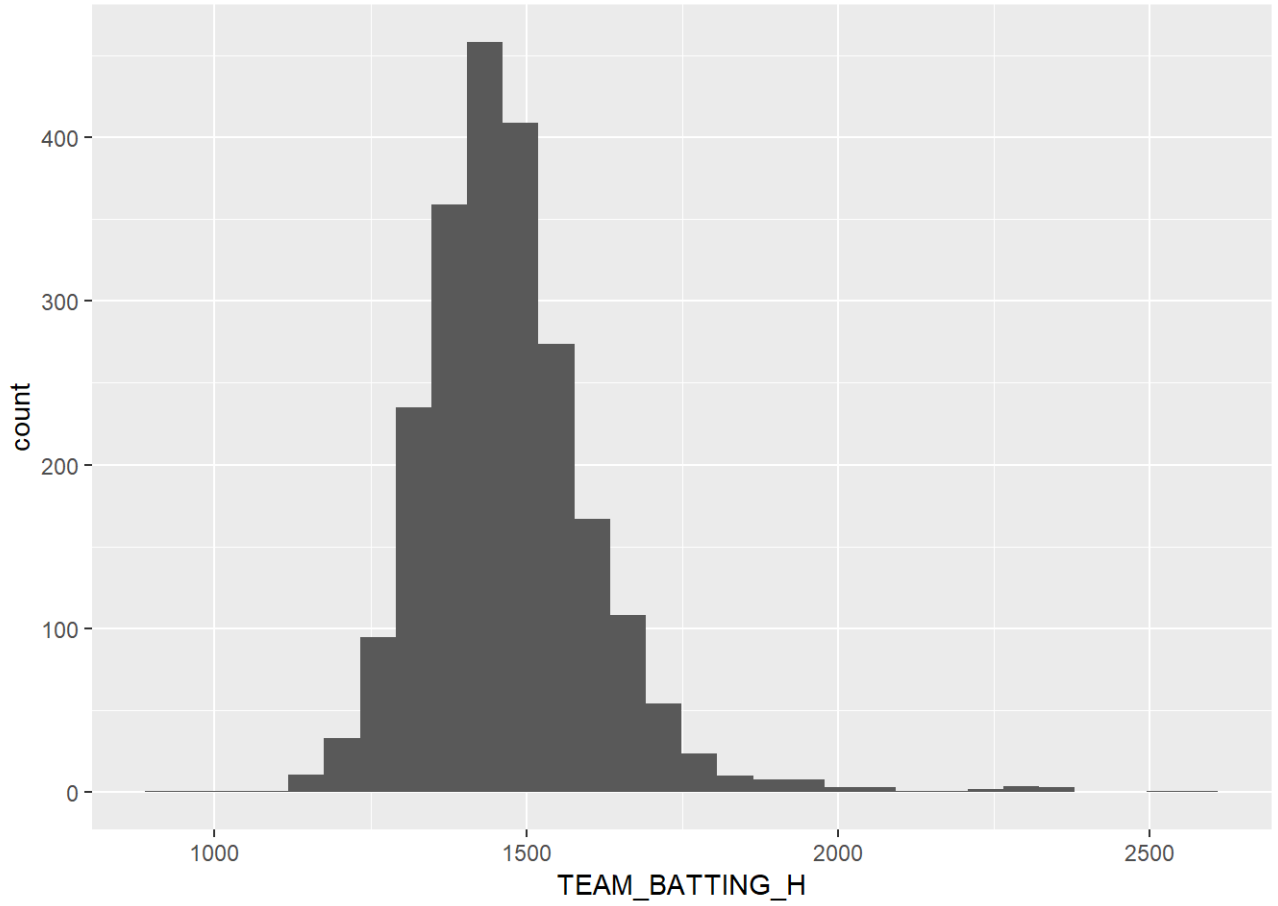

```
#note appears reasonably normally distributed
ggplot() + geom_histogram(aes(x = TARGET_WINS))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

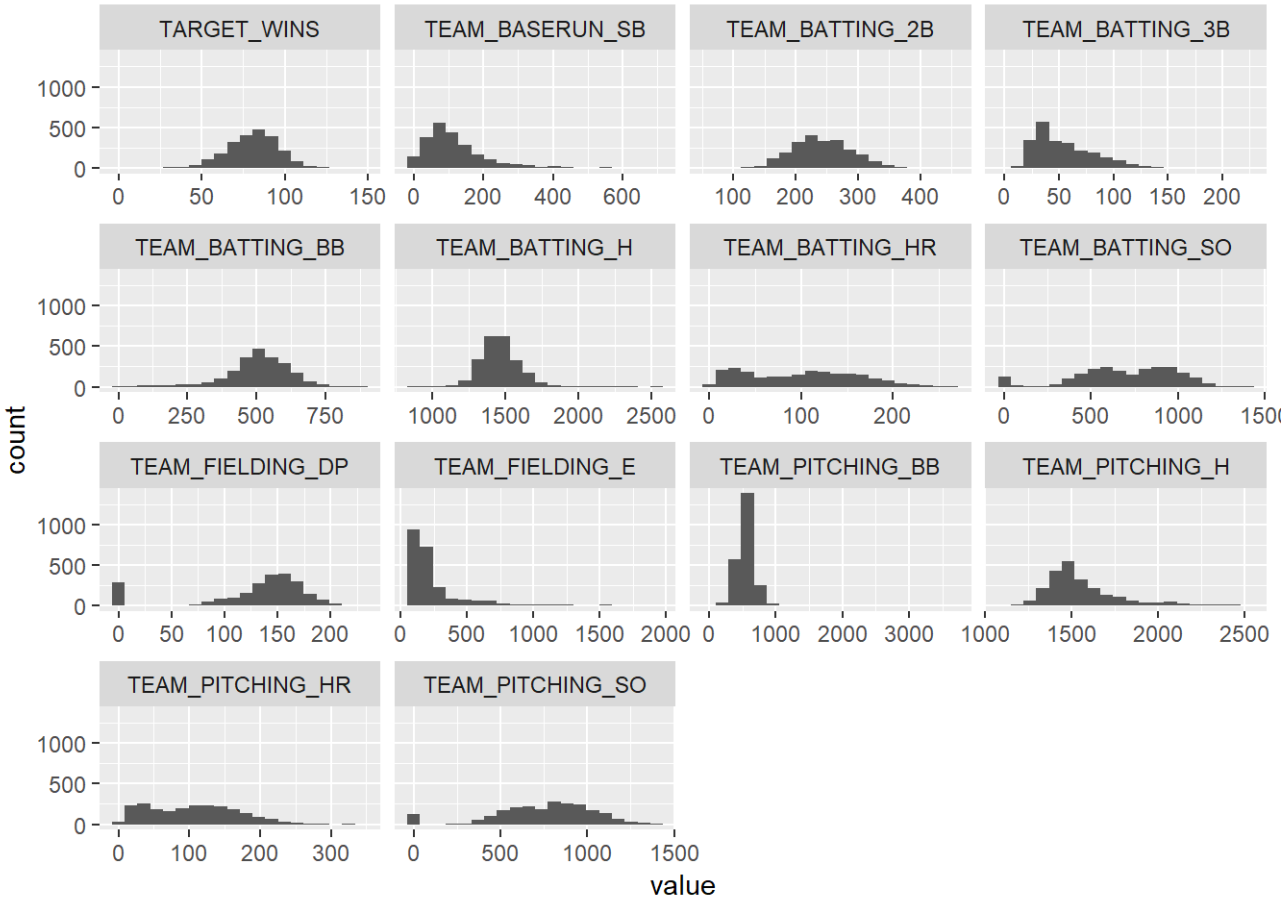



```
ggplot() + geom_histogram(aes(x = TEAM_BATTING_H))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(gather(mbTrain), aes(value)) +
    geom_histogram(bins = 20) +
    facet_wrap(~key, scales = "free_x")
```



Fit a couple really simple models just to explore, one with everything, one with just home runs against wins.

```
lmFit <- lm(TARGET_WINS ~., data = mbTrain)
summary(lmFit)
```

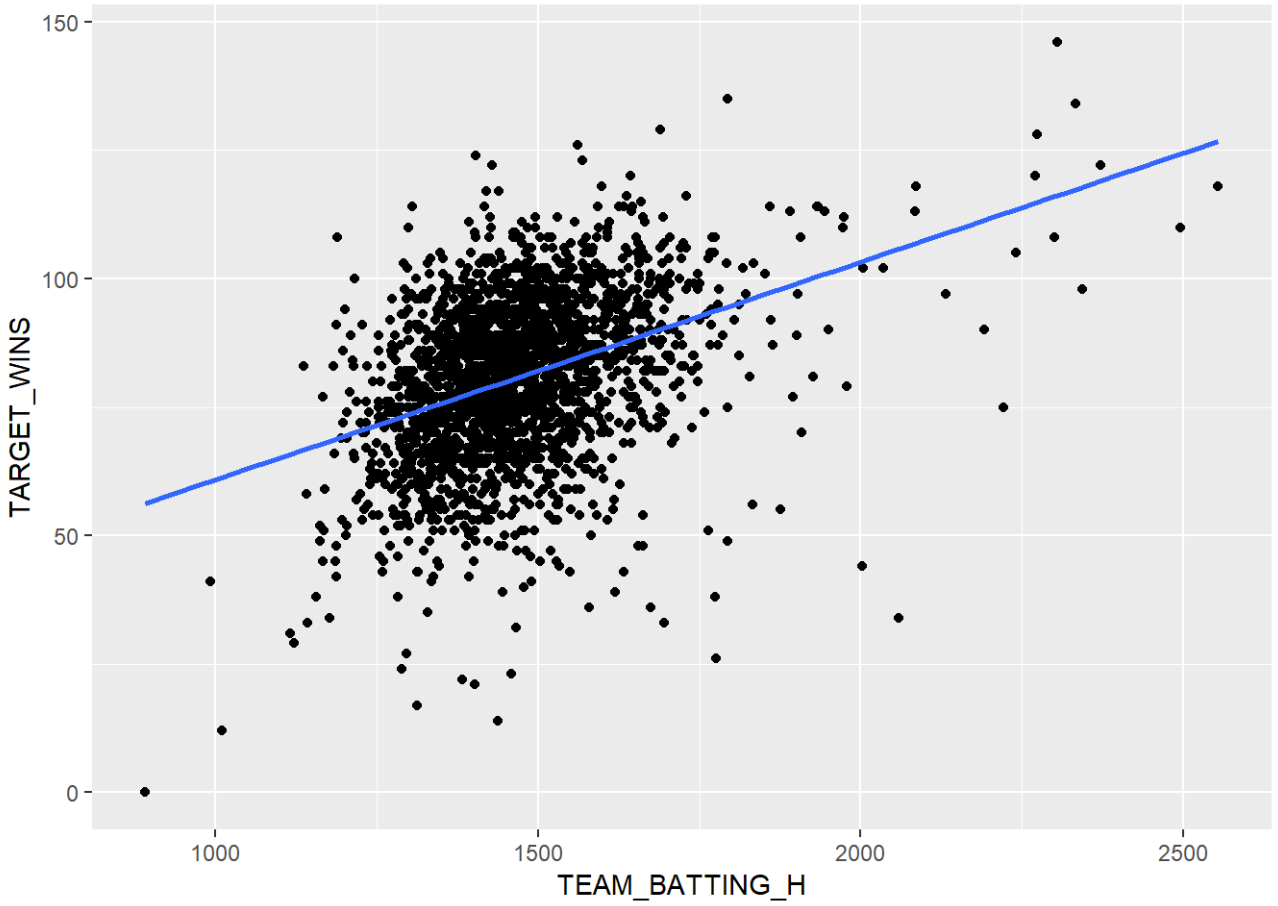
```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = mbTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.173  -8.324   0.168   8.670  51.804
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      37.397579   4.732471   7.902 4.24e-15 ***
## TEAM_BATTING_H    0.044540   0.003455  12.890  < 2e-16 ***
## TEAM_BATTING_2B  -0.006759   0.009279  -0.728 0.466427
## TEAM_BATTING_3B   0.078149   0.016958   4.608 4.29e-06 ***
## TEAM_BATTING_HR   0.072884   0.028139   2.590 0.009656 **
## TEAM_BATTING_BB   0.004194   0.004366   0.961 0.336758
## TEAM_BATTING_SO  -0.022270   0.003805  -5.853 5.54e-09 ***
## TEAM_BASERUN_SB   0.002304   0.004127   0.558 0.576626
## TEAM_PITCHING_H  -0.006469   0.001765  -3.665 0.000253 ***
## TEAM_PITCHING_HR -0.011394   0.024897  -0.458 0.647261
## TEAM_PITCHING_BB  0.004409   0.002462   1.791 0.073500 .
## TEAM_PITCHING_SO  0.013304   0.003213   4.141 3.58e-05 ***
## TEAM_FIELDING_E  -0.036860   0.002788 -13.223  < 2e-16 ***
## TEAM_FIELDING_DP -0.084309   0.010084  -8.360  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.21 on 2262 degrees of freedom
## Multiple R-squared:  0.3012, Adjusted R-squared:  0.2971
## F-statistic: 74.98 on 13 and 2262 DF,  p-value: < 2.2e-16
```

```
lmFit <- lm(TARGET_WINS ~ TEAM_BATTING_HR, data = mbTrain)
summary(lmFit)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_HR, data = mbTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -76.226  -9.909   0.520  10.218  68.445
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     76.22576    0.62599 121.768   <2e-16 ***
## TEAM_BATTING_HR  0.04583    0.00537   8.534   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.51 on 2274 degrees of freedom
## Multiple R-squared:  0.03103,    Adjusted R-squared:  0.0306
## F-statistic: 72.82 on 1 and 2274 DF,  p-value: < 2.2e-16
```

Let's plot hits against wins as that field should highest correlation.

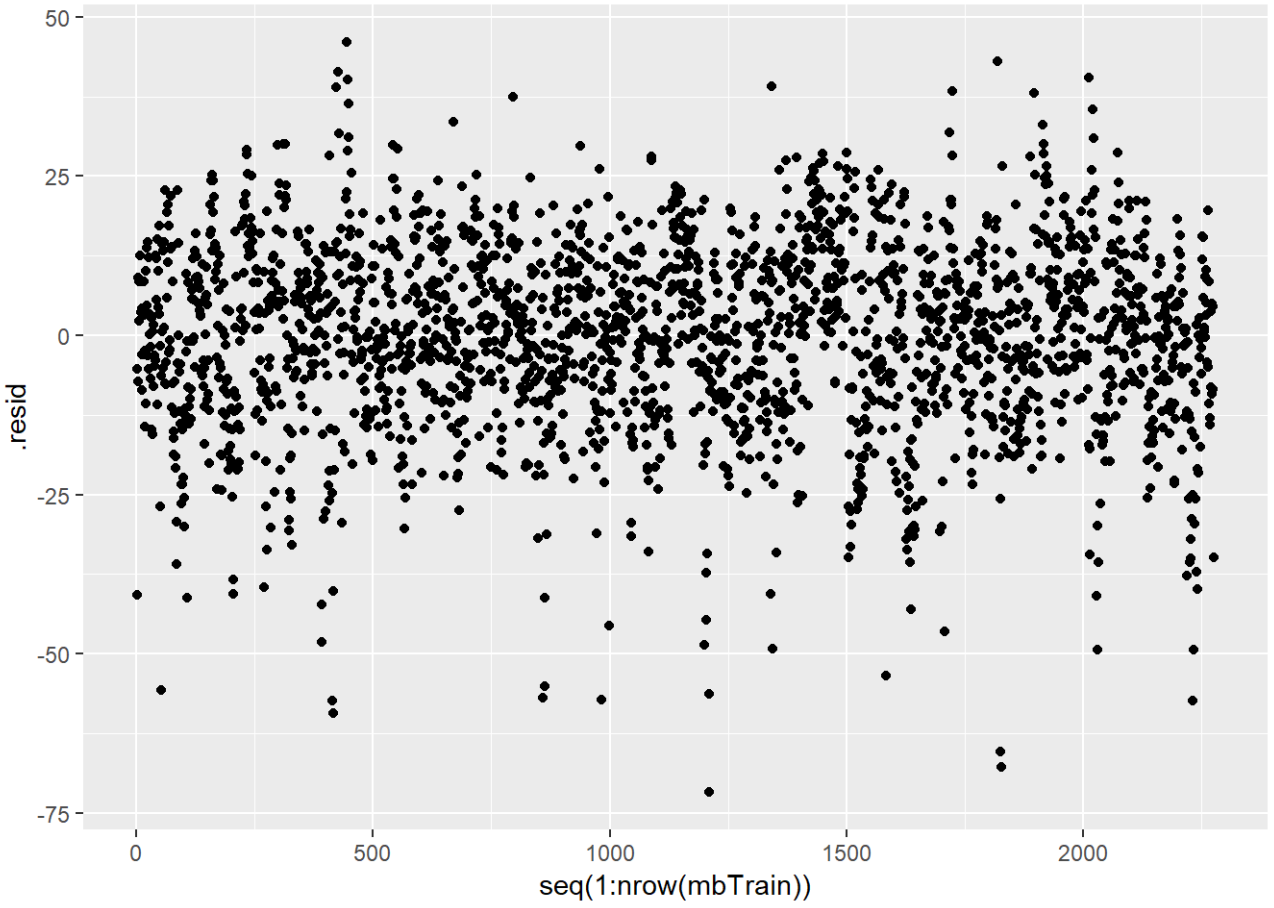
```
lmFitH <- lm(TARGET_WINS ~ TEAM_BATTING_H)
ggplot(mbTrain, aes(x= TEAM_BATTING_H, y = TARGET_WINS)) +
    geom_point() +
    geom_smooth(method = "lm", se= FALSE)
```



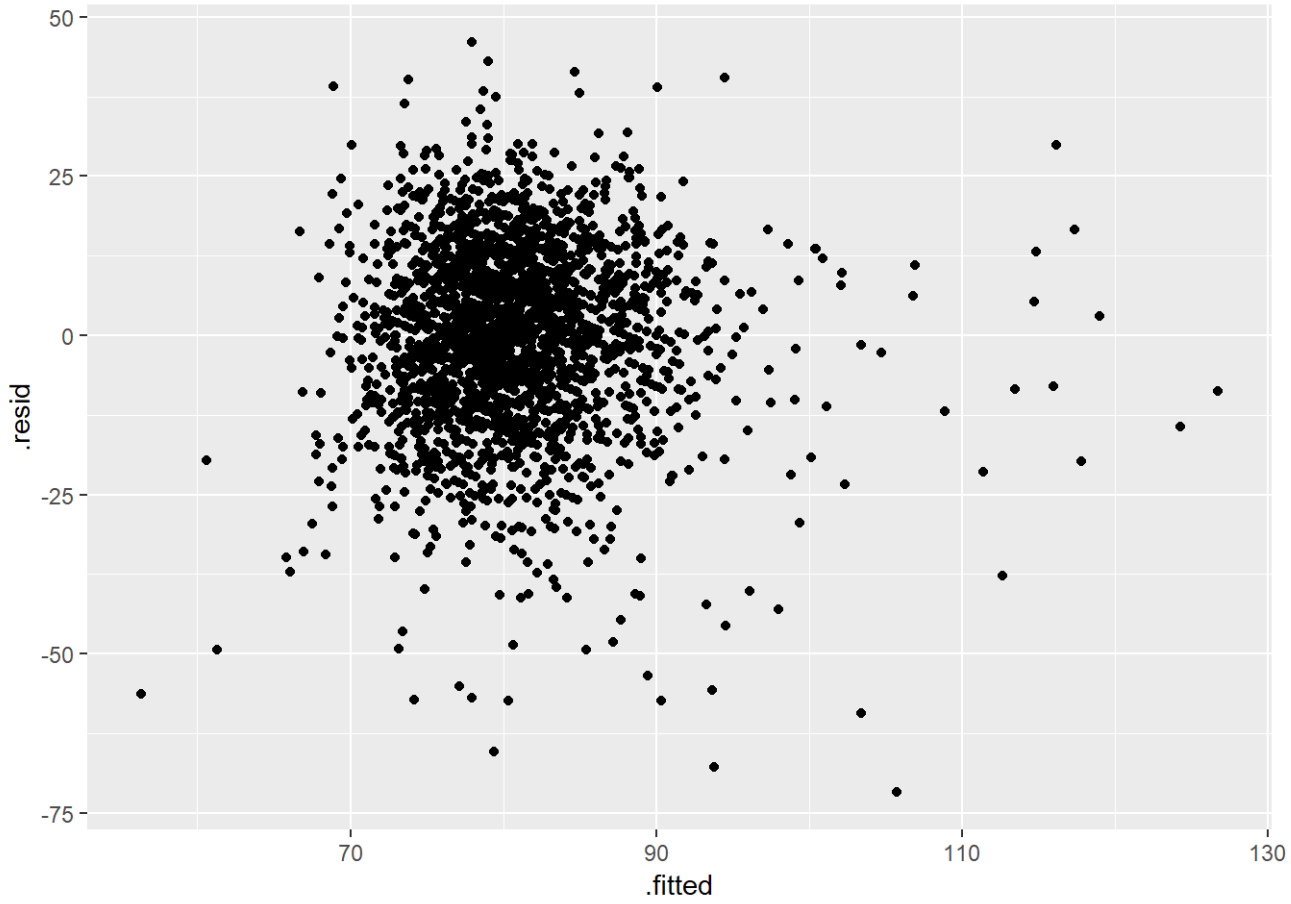
Plot

Residuals

```
#vs index
ggplot(lmFitH, aes(x= seq(1:nrow(mbTrain)), y= .resid)) +geom_point()
```




```
#vs fitted value
ggplot(lmFitH, aes(x= .fitted, y= .resid)) +geom_point()
```


50
25
0
-25
-50
-75
.resid
70
90
110
130
.fitted