DATA 621 Homework 1

Critical Thinking Group 1

September 22, 2021

Contents

Prepared for:

Prof. Dr. Nasrin Khansari

City University of New York, School of Professional Studies - Data 621

DATA 621 – Business Analytics and Data Mining

Prepared by:

Critical Thinking Group 1

Vic Chan

Gehad Gad

Evan McLaughlin

Bruno de Melo

Anjal Hussan

Zhouxin Shi

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.

Objective

The objective is 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 given to us (or variables that we derive from the variables provided).

#Data Exploration

Data Summery

```
# Import the data
Data <- read.csv("https://raw.githubusercontent.com/ahussan/DATA_621_Group1/main/HW1/moneyball-training
head(Data)
```

```
##   INDEX TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## 1     1          39           1445              194               39
## 2     2          70           1339              219               22
## 3     3          86           1377              232               35
## 4     4          70           1387              209               38
## 5     5          82           1297              186               27
## 6     6          75           1279              200               36
##   TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## 1              13             143              842              NA
## 2             190             685             1075              37
## 3             137             602              917              46
## 4              96             451              922              43
## 5             102             472              920              49
## 6              92             443              973             107
##   TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
## 1              NA               NA            9364               84
## 2              28               NA            1347              191
## 3              27               NA            1377              137
## 4              30               NA            1396               97
## 5              39               NA            1297              102
## 6              59               NA            1279               92
##   TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## 1              927             5456             1011               NA
## 2              689             1082              193              155
## 3              602              917              175              153
## 4              454              928              164              156
## 5              472              920              138              168
## 6              443              973              123              149
```

We can see that the data contain 17 columns and 2276 observations or records. The first column is the index which will be deleted as it is not useful.

```
# Remove the index
Data1 <- Data[-c(1)]
```

```
# Check the Summary
summary(Data1)
```

```
##   TARGET_WINS      TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
##  Min.   :  0.00   Min.   : 891   Min.   : 69.0   Min.   :  0.00
##  1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0   1st Qu.: 34.00
##  Median : 82.00   Median :1454   Median :238.0   Median : 47.00
##  Mean   : 80.79   Mean   :1469   Mean   :241.2   Mean   : 55.25
##  3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0   3rd Qu.: 72.00
##  Max.   :146.00   Max.   :2554   Max.   :458.0   Max.   :223.00
##
##  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_SO  TEAM_BASERUN_SB
##  Min.   :  0.00   Min.   :  0.0   Min.   :   0.0   Min.   :  0.0
##  1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 548.0   1st Qu.: 66.0
##  Median :102.00   Median :512.0   Median : 750.0   Median :101.0
##  Mean   : 99.61   Mean   :501.6   Mean   : 735.6   Mean   :124.8
##  3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 930.0   3rd Qu.:156.0
##  Max.   :264.00   Max.   :878.0   Max.   :1399.0   Max.   :697.0
##                                   NA's   :102      NA's   :131
##  TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
##  Min.   :  0.0   Min.   :29.00    Min.   : 1137   Min.   :  0.0
##  1st Qu.: 38.0   1st Qu.:50.50    1st Qu.: 1419   1st Qu.: 50.0
##  Median : 49.0   Median :58.00    Median : 1518   Median :107.0
##  Mean   : 52.8   Mean   :59.36    Mean   : 1779   Mean   :105.7
##  3rd Qu.: 62.0   3rd Qu.:67.00    3rd Qu.: 1682   3rd Qu.:150.0
##  Max.   :201.0   Max.   :95.00    Max.   :30132   Max.   :343.0
##  NA's   :772     NA's   :2085
##  TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E  TEAM_FIELDING_DP
##  Min.   :   0.0   Min.   :    0.0   Min.   :  65.0   Min.   : 52.0
##  1st Qu.: 476.0   1st Qu.:  615.0   1st Qu.: 127.0   1st Qu.:131.0
##  Median : 536.5   Median :  813.5   Median : 159.0   Median :149.0
##  Mean   : 553.0   Mean   :  817.7   Mean   : 246.5   Mean   :146.4
##  3rd Qu.: 611.0   3rd Qu.:  968.0   3rd Qu.: 249.2   3rd Qu.:164.0
##  Max.   :3645.0   Max.   :19278.0   Max.   :1898.0   Max.   :228.0
##                   NA's   :102                        NA's   :286
```

Summary of the data gives a useful information about each feature including the number of NA values. It is obvious that we have many NA values.

```
# Compute descriptive statistics
res <- stat.desc(Data1)
round(res, 2)
```

```
##              TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## nbr.val          2276.00        2276.00         2276.00         2276.00
## nbr.null            1.00           0.00            0.00            2.00
## nbr.na              0.00           0.00            0.00            0.00
```

```
## min                0.00        891.00          69.00           0.00
## max              146.00       2554.00         458.00         223.00
## range            146.00       1663.00         389.00         223.00
## sum           183880.00    3344058.00      549078.00      125749.00
## median            82.00       1454.00         238.00          47.00
## mean              80.79       1469.27         241.25          55.25
## SE.mean            0.33          3.03           0.98           0.59
## CI.mean.0.95       0.65          5.94           1.92           1.15
## var              248.13      20906.61        2190.37         780.56
## std.dev           15.75        144.59          46.80          27.94
## coef.var           0.19          0.10           0.19           0.51
##             TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## nbr.val             2276.00         2276.00         2174.00         2145.00
## nbr.null              15.00            1.00           20.00            2.00
## nbr.na                 0.00            0.00          102.00          131.00
## min                    0.00            0.00            0.00            0.00
## max                  264.00          878.00         1399.00          697.00
## range                264.00          878.00         1399.00          697.00
## sum               226717.00      1141548.00      1599206.00       267614.00
## median               102.00          512.00          750.00          101.00
## mean                  99.61          501.56          735.61          124.76
## SE.mean                1.27            2.57            5.33            1.90
## CI.mean.0.95           2.49            5.04           10.45            3.72
## var                 3665.92        15048.14        61765.38         7707.29
## std.dev               60.55          122.67          248.53           87.79
## coef.var               0.61            0.24            0.34            0.70
##             TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H TEAM_PITCHING_HR
## nbr.val             1504.00          191.00         2276.00         2276.00
## nbr.null               1.00            0.00            0.00           15.00
## nbr.na               772.00         2085.00            0.00            0.00
## min                    0.00           29.00         1137.00            0.00
## max                  201.00           95.00        30132.00          343.00
## range                201.00           66.00        28995.00          343.00
## sum                79417.00        11337.00      4049483.00       240570.00
## median                49.00           58.00         1518.00          107.00
## mean                  52.80           59.36         1779.21          105.70
## SE.mean                0.59            0.94           29.49            1.28
## CI.mean.0.95           1.16            1.85           57.83            2.52
## var                  526.99          168.15      1979207.03         3757.54
## std.dev               22.96           12.97         1406.84           61.30
## coef.var               0.43            0.22            0.79            0.58
##             TEAM_PITCHING_BB TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## nbr.val             2276.00         2174.00         2276.00         1990.00
## nbr.null               1.00           20.00            0.00            0.00
## nbr.na                 0.00          102.00            0.00          286.00
## min                    0.00            0.00           65.00           52.00
## max                 3645.00        19278.00         1898.00          228.00
## range               3645.00        19278.00         1833.00          176.00
## sum              1258646.00      1777746.00       560990.00       291312.00
## median               536.50          813.50          159.00          149.00
## mean                 553.01          817.73          246.48          146.39
## SE.mean                3.49           11.86            4.77            0.59
## CI.mean.0.95           6.84           23.26            9.36            1.15
## var                27674.77       305903.05        51879.62          687.82
```

```
## std.dev                166.36          553.09          227.77           26.23
## coef.var                 0.30            0.68            0.92            0.18
```

```
# The mean for each column in the data
colMeans(Data1)
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##         80.79086       1469.26977        241.24692         55.25000
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##         99.61204        501.55888               NA               NA
##  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR
##               NA               NA       1779.21046        105.69859
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##        553.00791               NA        246.48067               NA
```

```
# The Standard Deviation for each column in the data
sapply(Data1, sd)
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##         15.75215        144.59120         46.80141         27.93856
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##         60.54687        122.67086               NA               NA
##  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR
##               NA               NA       1406.84293         61.29875
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##        166.35736               NA        227.77097               NA
```

```
# The median for each column in the data
apply(Data1, 2, median)
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##             82.0           1454.0            238.0             47.0
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##            102.0            512.0               NA               NA
##  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR
##               NA               NA           1518.0            107.0
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##            536.5               NA            159.0               NA
```

Missing Vlaues

```
# Search if there are any NA values

sum(is.na(Data1))
```

```
## [1] 3478
```

```
# We are not able to delete the NA values. We will replace NA values.

Data2 = replace(Data1, TRUE, lapply(Data1, na.aggregate))
```

```r
# Confirm the all NA values were replaced by the mean.
sum(is.na(Data2))
```

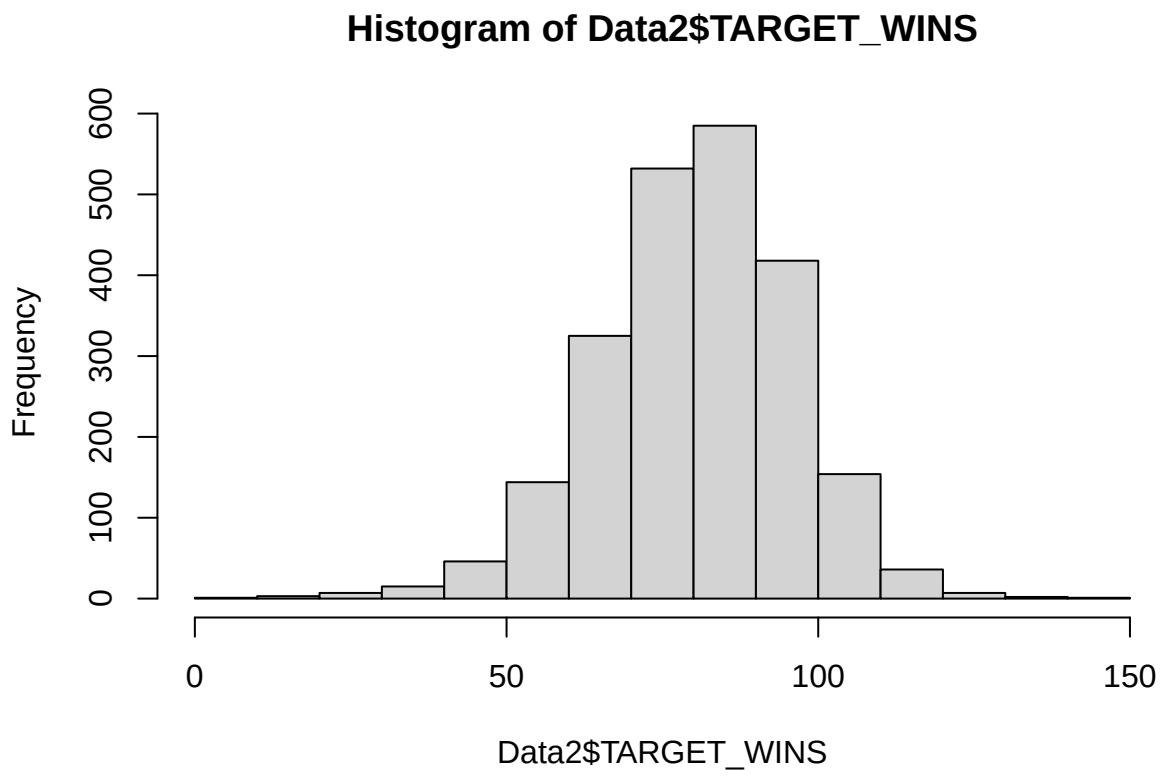
```
## [1] 0
```

```r
# Confirm that data is numeric
sapply(Data2, is.numeric)
```

```
##      TARGET_WINS   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##             TRUE             TRUE             TRUE             TRUE
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##             TRUE             TRUE             TRUE             TRUE
##  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR
##             TRUE             TRUE             TRUE             TRUE
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##             TRUE             TRUE             TRUE             TRUE
```

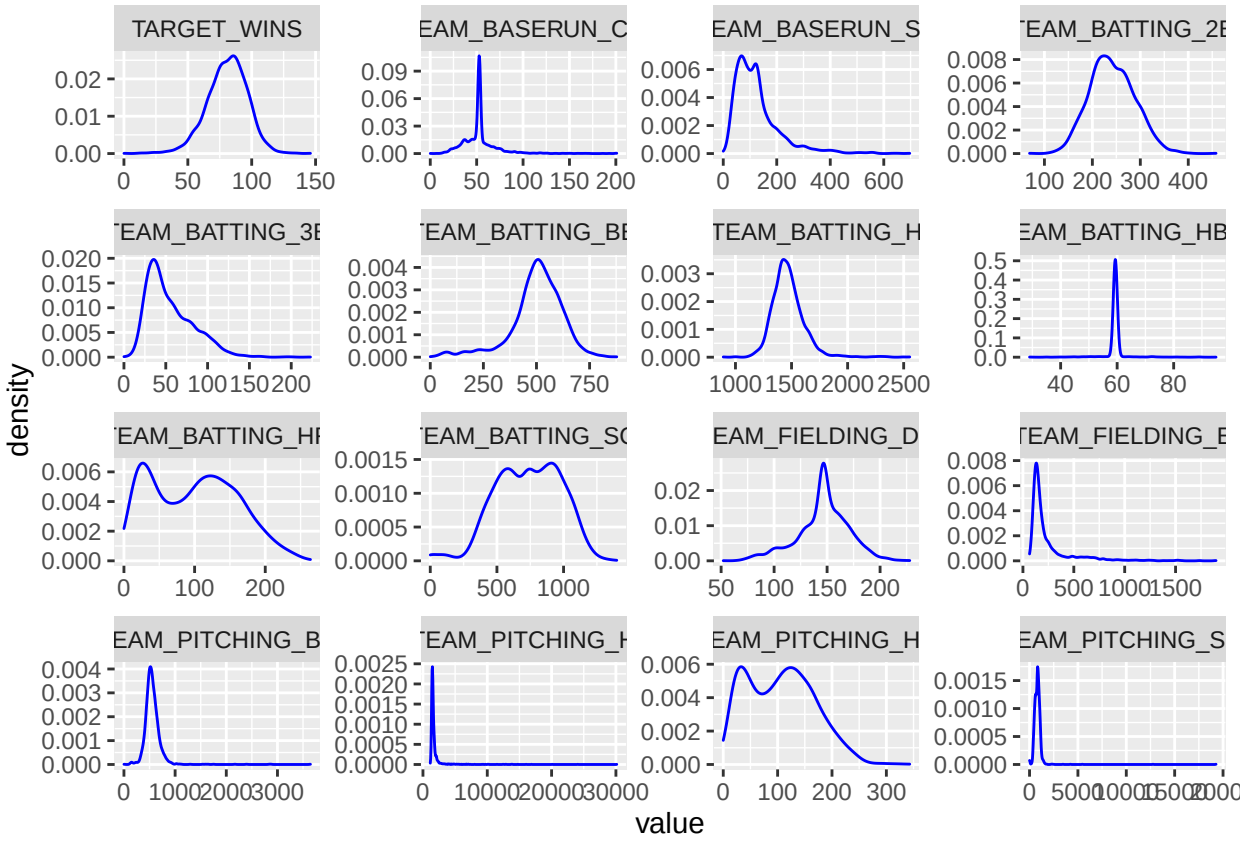
Graphs

```r
hist(Data2$TARGET_WINS)
```

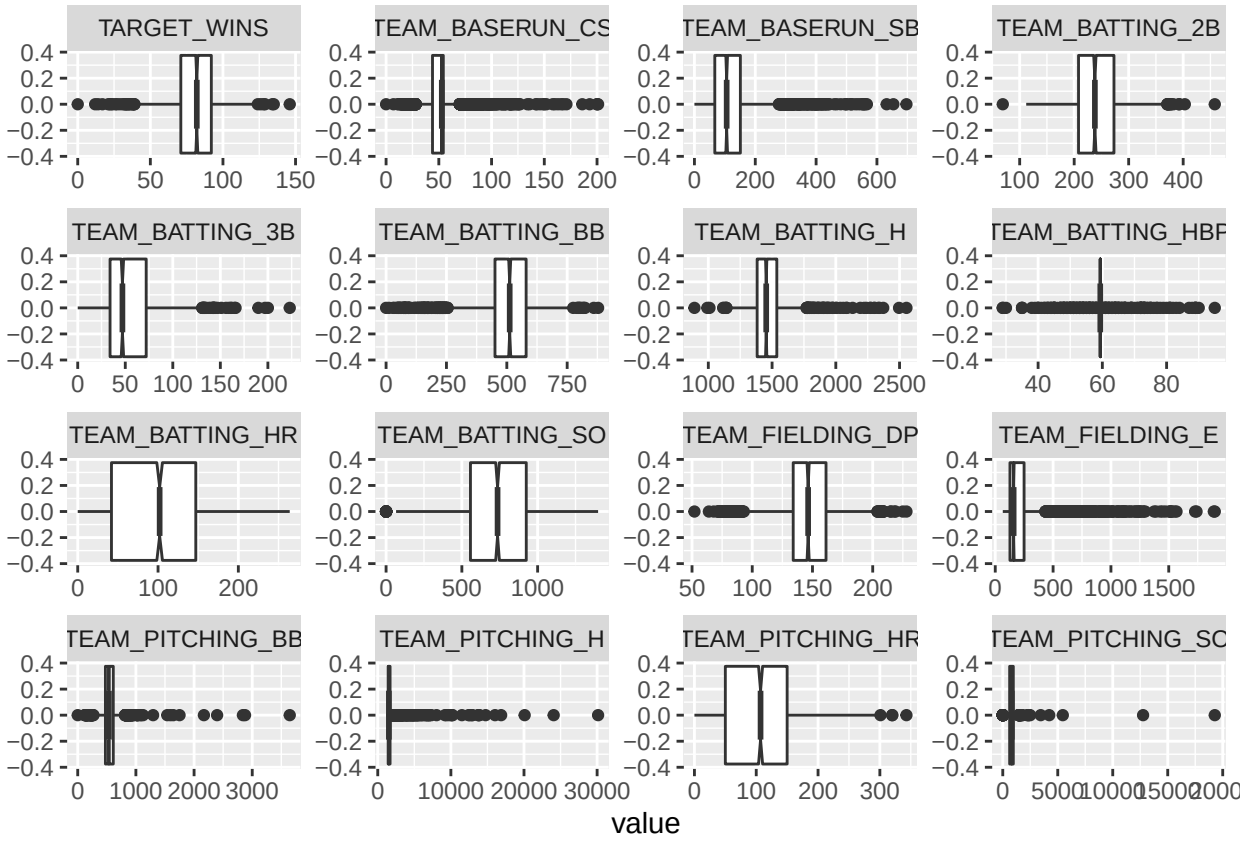


The histogram of the target_wins column is normally distributed.

```
Data2 %>%
    gather(var, value, TARGET_WINS:TEAM_FIELDING_DP) %>%
    ggplot(., aes(value)) + geom_density(color = "blue") + facet_wrap(~var, scales = "free",
    ncol = 4)
```




```
Data2 %>%
    gather(var, value, TARGET_WINS:TEAM_FIELDING_DP) %>%
    ggplot(., aes(value)) + geom_boxplot(notch = TRUE) + facet_wrap(~var, scales = "free",
    ncol = 4)
```



Correlation

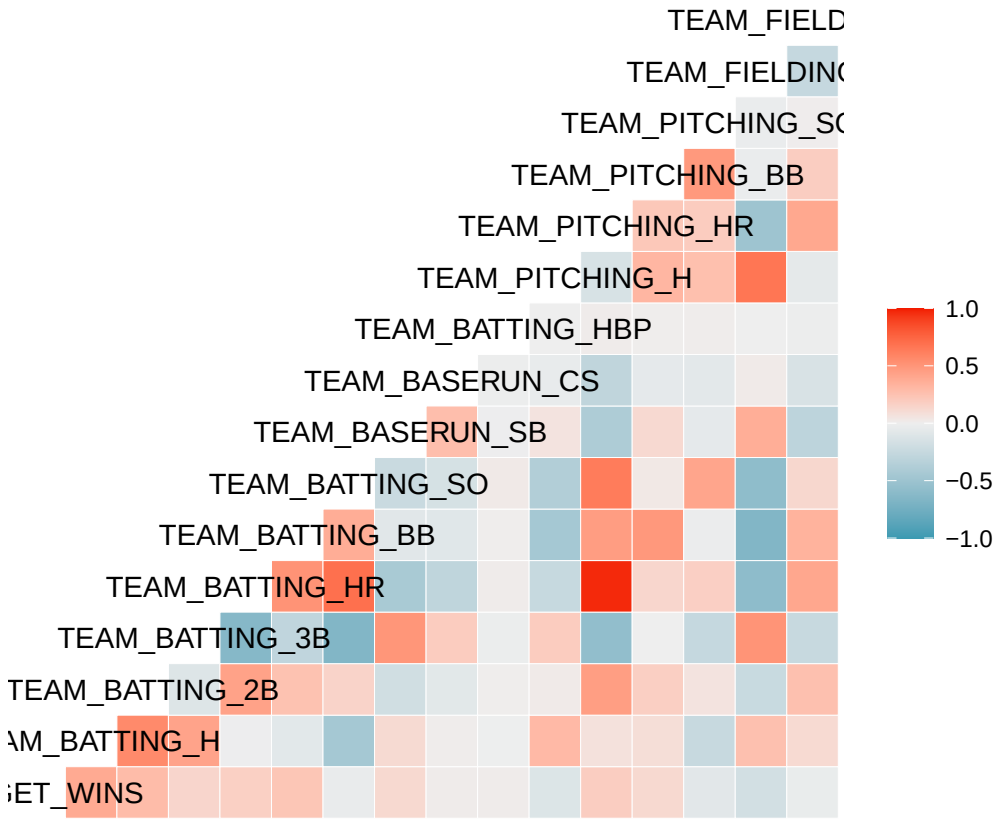
```
# Use pearson correlation
corrr::correlate(Data2, method = "pearson")
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

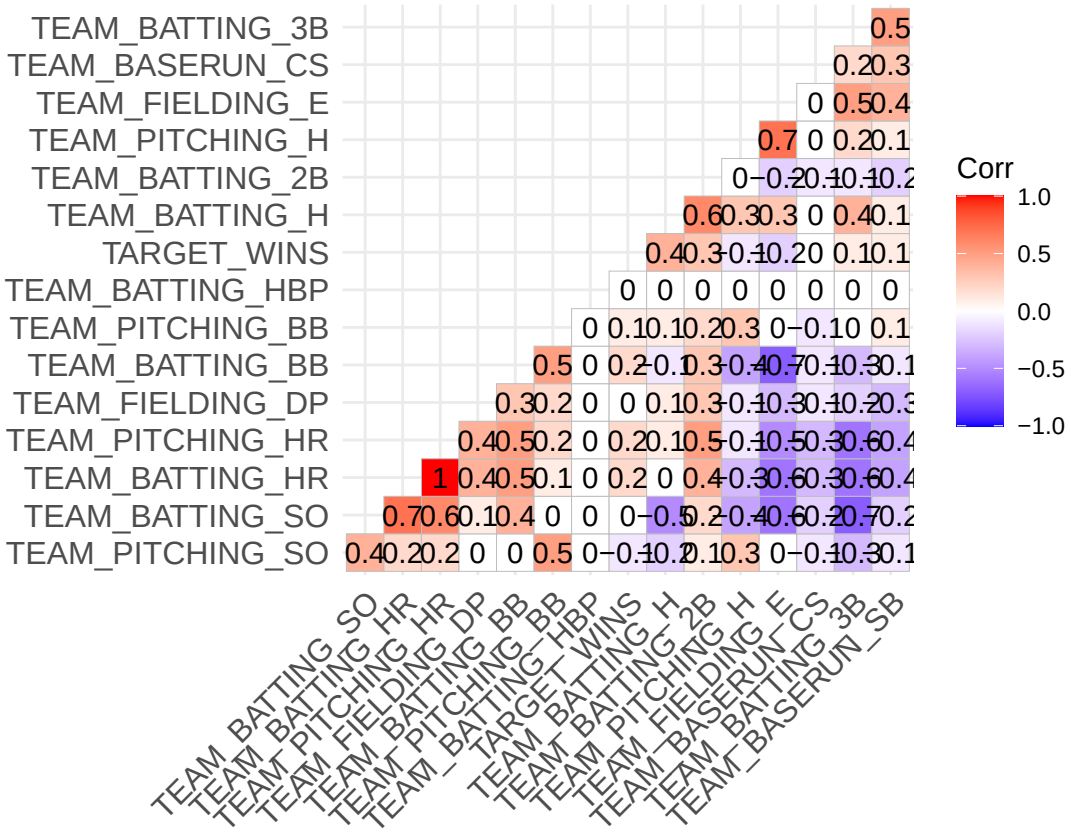
```
## # A tibble: 16 x 17
##    term  TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
##    <chr>       <dbl>          <dbl>           <dbl>           <dbl>
##  1 TARG~     NA             0.389           0.289           0.143
##  2 TEAM~      0.389        NA               0.563           0.428
##  3 TEAM~      0.289         0.563          NA              -0.107
##  4 TEAM~      0.143         0.428          -0.107          NA
##  5 TEAM~      0.176        -0.00654         0.435          -0.636
##  6 TEAM~      0.233        -0.0725          0.256          -0.287
##  7 TEAM~     -0.0307       -0.451           0.155          -0.657
##  8 TEAM~      0.123         0.114          -0.190           0.501
##  9 TEAM~      0.0156        0.0116         -0.0739          0.195
## 10 TEAM~      0.0163       -0.00443         0.00749        -0.0163
## 11 TEAM~     -0.110         0.303           0.0237          0.195
```

```
## 12 TEAM~      0.189         0.0729          0.455          -0.568
## 13 TEAM~      0.124         0.0942          0.178          -0.00222
## 14 TEAM~     -0.0758       -0.245           0.0617         -0.254
## 15 TEAM~     -0.176         0.265          -0.235           0.510
## 16 TEAM~     -0.0288        0.115           0.263          -0.246
## # ... with 12 more variables: TEAM_BATTING_HR <dbl>, TEAM_BATTING_BB <dbl>,
## #   TEAM_BATTING_SO <dbl>, TEAM_BASERUN_SB <dbl>, TEAM_BASERUN_CS <dbl>,
## #   TEAM_BATTING_HBP <dbl>, TEAM_PITCHING_H <dbl>, TEAM_PITCHING_HR <dbl>,
## #   TEAM_PITCHING_BB <dbl>, TEAM_PITCHING_SO <dbl>, TEAM_FIELDING_E <dbl>,
## #   TEAM_FIELDING_DP <dbl>
```

```
ggcorr(Data2)
```




```
# Add correlation coefficients
corr <- round(cor(Data2), 1)
ggcorrplot(corr, hc.order = TRUE, type = "lower", lab = TRUE)
```



Data Preperation

In this section we will be looking at the different ways to prepare the data for modeling. We will show the different steps that we took and the reasoning why we did certain transformations, replacement and creation of columns.

```
moneyball_training_data = read.csv("https://raw.githubusercontent.com/ahussan/DATA_621_Group1/main/HW1/m
```

```
na_count = sapply(moneyball_training_data, function(y) sum(is.na(y)))
na_count = data.frame(na_count)
na_count %>%
    arrange(desc(na_count)) %>%
    mutate(total_rows = nrow(moneyball_training_data)) %>%
    mutate(percent_missing = na_count/total_rows)
```

```
##                  na_count total_rows percent_missing
## TEAM_BATTING_HBP     2085       2276      0.91608084
## TEAM_BASERUN_CS       772       2276      0.33919156
## TEAM_FIELDING_DP      286       2276      0.12565905
## TEAM_BASERUN_SB       131       2276      0.05755712
## TEAM_BATTING_SO       102       2276      0.04481547
## TEAM_PITCHING_SO      102       2276      0.04481547
## INDEX                   0       2276      0.00000000
```

```
## TARGET_WINS             0       2276      0.00000000
## TEAM_BATTING_H          0       2276      0.00000000
## TEAM_BATTING_2B         0       2276      0.00000000
## TEAM_BATTING_3B         0       2276      0.00000000
## TEAM_BATTING_HR         0       2276      0.00000000
## TEAM_BATTING_BB         0       2276      0.00000000
## TEAM_PITCHING_H         0       2276      0.00000000
## TEAM_PITCHING_HR        0       2276      0.00000000
## TEAM_PITCHING_BB        0       2276      0.00000000
## TEAM_FIELDING_E         0       2276      0.00000000
```

Initially when looking at the data we can see that **TEAM_BATTING_HBP** is missing 91% of its data and **TEAM_BASERUN_CS** is missing around 34% of its data. This is a lot of data missing which is why those columns will be removing these. Based on online reading there is no definite cut of for how much data one should be missing before removing a column, but it is always better to have more data. The columns **TEAM_FIELDING_DP**, **TEAM_BASERUN_SB**, **TEAM_BATTING_SO**, and **TEAM_PITCHING_SO** are missing around 12% - 4% of its data and can fill those in with using mean and median. In the next section we will look at to see whether using the mean or median would be the better choice in filling the missing data.

```
moneyball_subset = subset(moneyball_training_data, select = -c(TEAM_BATTING_HBP,
    TEAM_BASERUN_CS, INDEX))
```

Replacing NA with Mean or Median

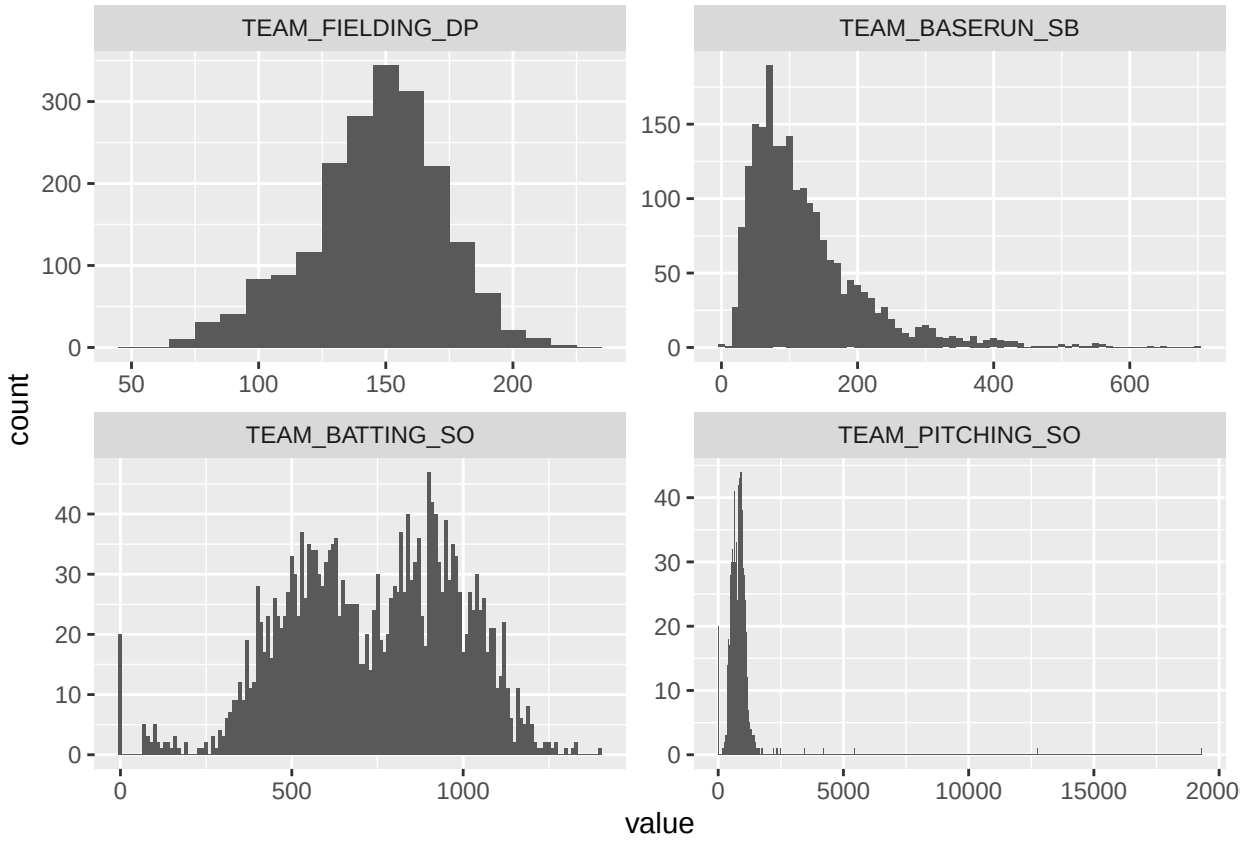
In this section we will need to decide whether to fill the missing data using the mean or median. We will need to look at the distribution of each of the columns with missing data in order to decide if we will be using the median or mean to fill in the missing data

```
missing_data = subset(moneyball_subset, select = c(TEAM_FIELDING_DP, TEAM_BASERUN_SB,
    TEAM_BATTING_SO, TEAM_PITCHING_SO))
missing_data = melt(missing_data)
```

```
## No id variables; using all as measure variables
```

```
ggplot(missing_data, aes(x = value)) + geom_histogram(binwidth = 10) + facet_wrap(~variable,
    scale = "free")
```

```
## Warning: Removed 621 rows containing non-finite values (stat_bin).
```



Looking at the above graphs we can see that not all the distribution are uniform distribution. We can see that **TEAM_BATTING_SO** is a bimodal distribution, **TEAM_BASERUN_SB** is skewed to the right, and **TEAM_PITCHING_SO** has very large outliers. For this reason we will be using the median to replace all the missing data as the median is less susceptible to outliers and non-uniform distributions.

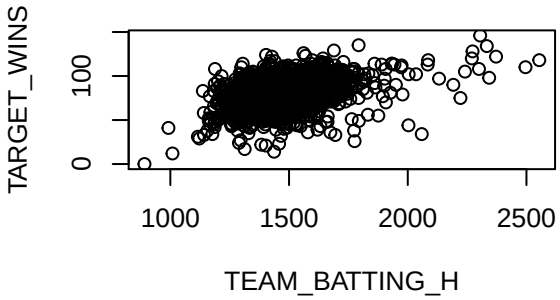
```
replace_na_with_median = function(x) {
    x[is.na(x)] = median(x, na.rm = TRUE)
    return(x)
}

moneyball_fill = apply(moneyball_subset, 2, replace_na_with_median)
moneyball_fill = as.data.frame(moneyball_fill)
```

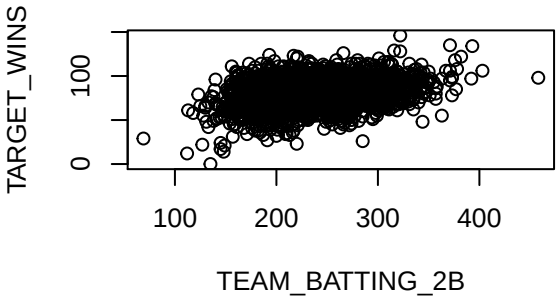
Transformation

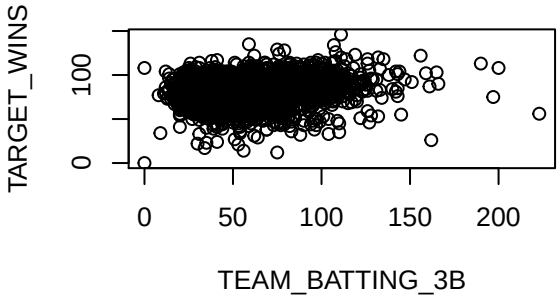
We will also be needing to check all of the columns to see if they will need any type of transformation in order to create a linear line. We will be be graphing all the columns with **TARGET_WINS** as the response variable. This will allow us to see if there are any columns that can be transformed in order to improve the model.

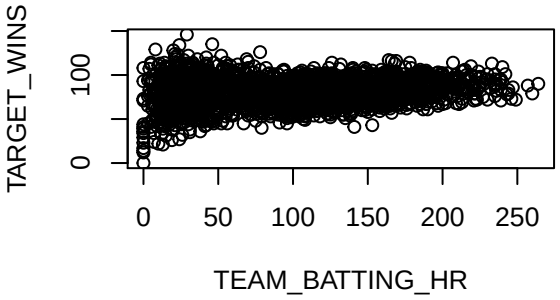
```
par(mfrow = c(2, 2))


for (i in 2:ncol(moneyball_fill)) {
```

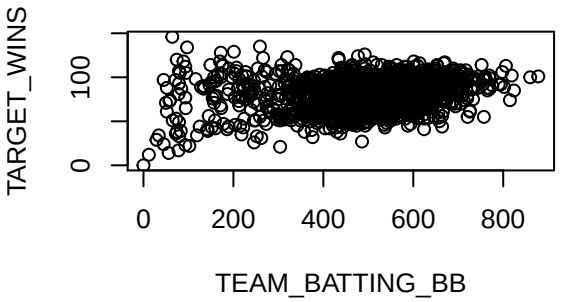
```
    y = moneyball_subset[, 1]
    x = moneyball_subset[, i]
    plot(x, y, ylab = "TARGET_WINS", xlab = names(moneyball_fill)[i])
}
```

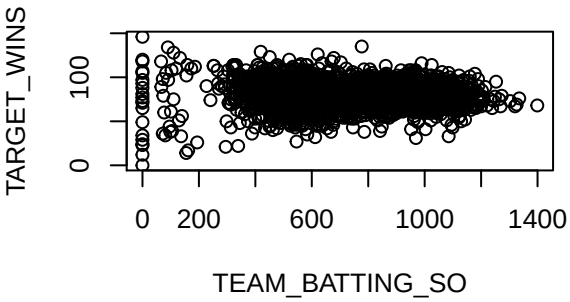


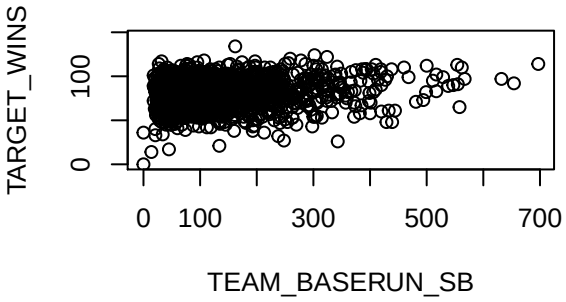


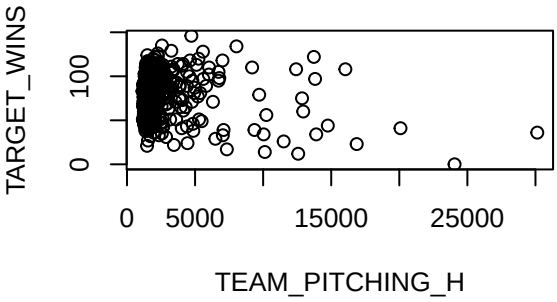


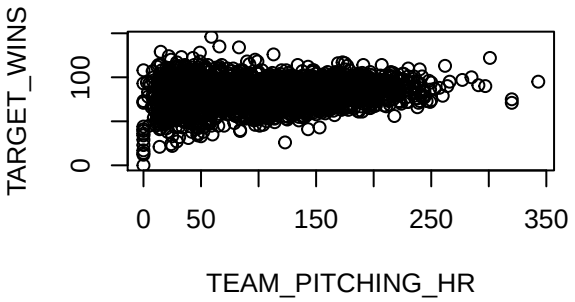


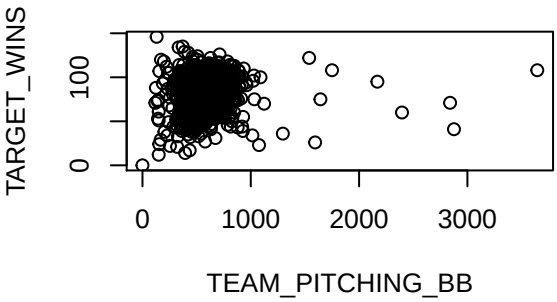

TARGET_WINS
0
100
0
200
400
600
800
TEAM_BATTING_BB

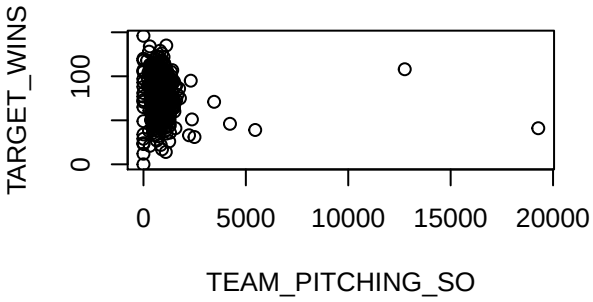

TARGET_WINS
0
100
0
200
600
1000
1400
TEAM_BATTING_SO

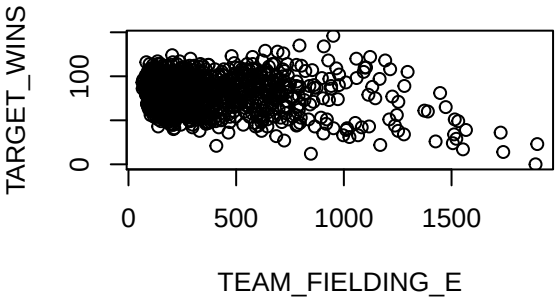

TARGET_WINS
0
100
0
100
300
500
700
TEAM_BASERUN_SB

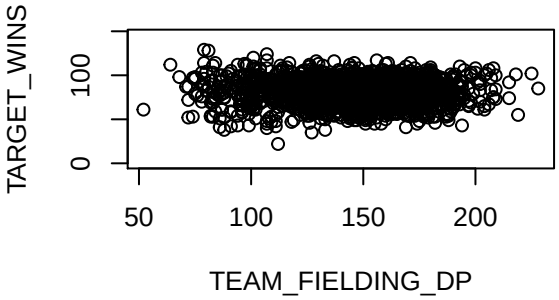

TARGET_WINS
0
100
0
5000
15000
25000
TEAM_PITCHING_H


TARGET_WINS
0
100
0
50
150
250
350
TEAM_PITCHING_HR


TARGET_WINS
0
100
0
1000
2000
3000
TEAM_PITCHING_BB


TARGET_WINS
0
100
0
5000
10000
15000
20000
TEAM_PITCHING_SO


TARGET_WINS
0
100
0
500
1000
1500
TEAM_FIELDING_E



Looking at the graphs above we can see that none of the columns are real good candidates for transformation.

Putting Teams Into Buckets

We will be putting the dataset into buckets based on the teams winning score as this will allow us to see if there is any patterns between weak and strong teams. The teams will be split into two groups **Strong** and **Weak** based on the **TARGET_WINS** column.

```
moneyball_fill$TEAM_TYPE = cut(moneyball_subset[, "TARGET_WINS"], breaks = c(0, 73,
    146), include.lowest = TRUE, labels = c("Weak", "Strong"))
```

Creating Total Hits

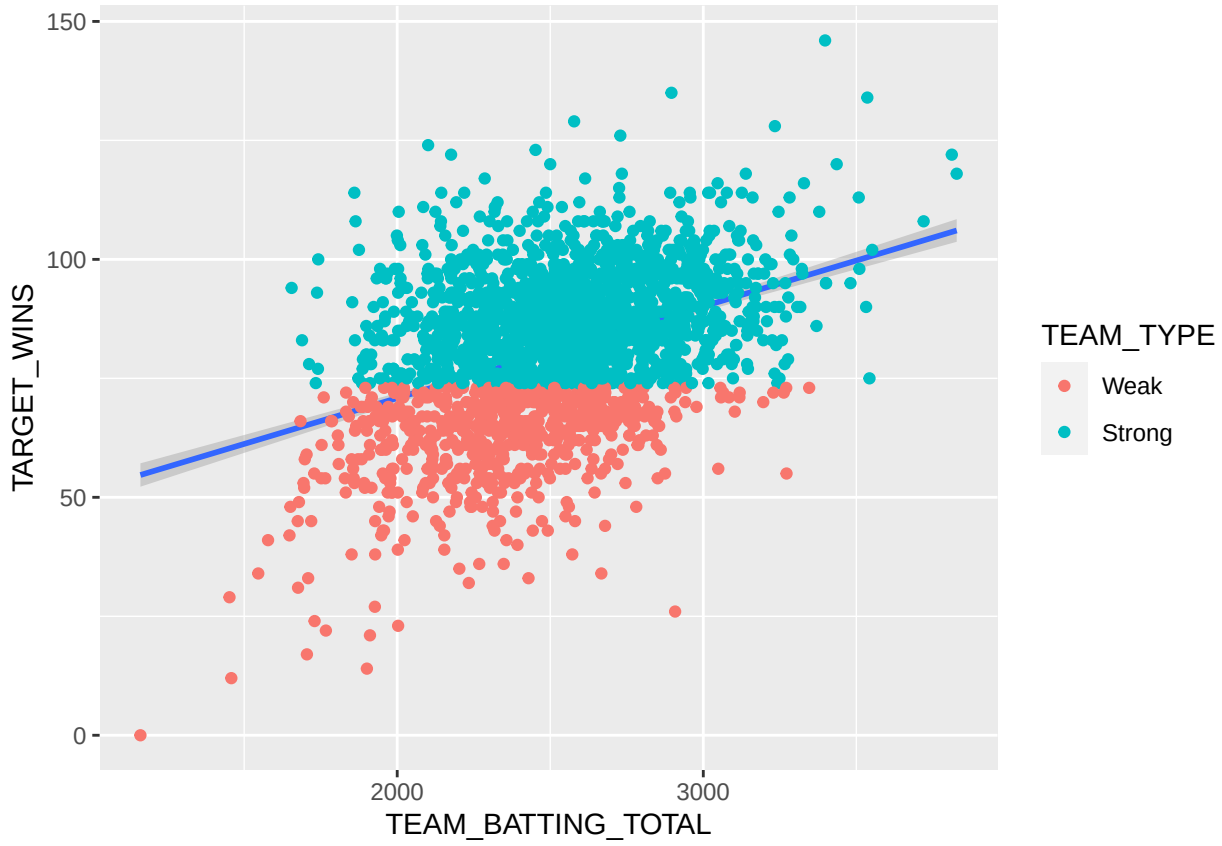
Creating a column which includes the total amount of hits a team has

```
createTEAM_BATTING_TOTAL = function(x) {
    x$TEAM_BATTING_TOTAL = (x$TEAM_BATTING_H + (2 * x$TEAM_BATTING_2B) + (3 * x$TEAM_BATTING_3B) +
        (4 * x$TEAM_BATTING_HR))
    return(x)
}
```

```
moneyball_fill$TEAM_BATTING_TOTAL = (moneyball_fill$TEAM_BATTING_H + (2 * moneyball_fill$TEAM_BATTING_2
    (3 * moneyball_fill$TEAM_BATTING_3B) + (4 * moneyball_fill$TEAM_BATTING_HR))
```

```
ggplot(moneyball_fill, aes(x = TEAM_BATTING_TOTAL, y = TARGET_WINS)) + geom_smooth(method = "lm") +
    geom_point(aes(color = TEAM_TYPE))
```

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



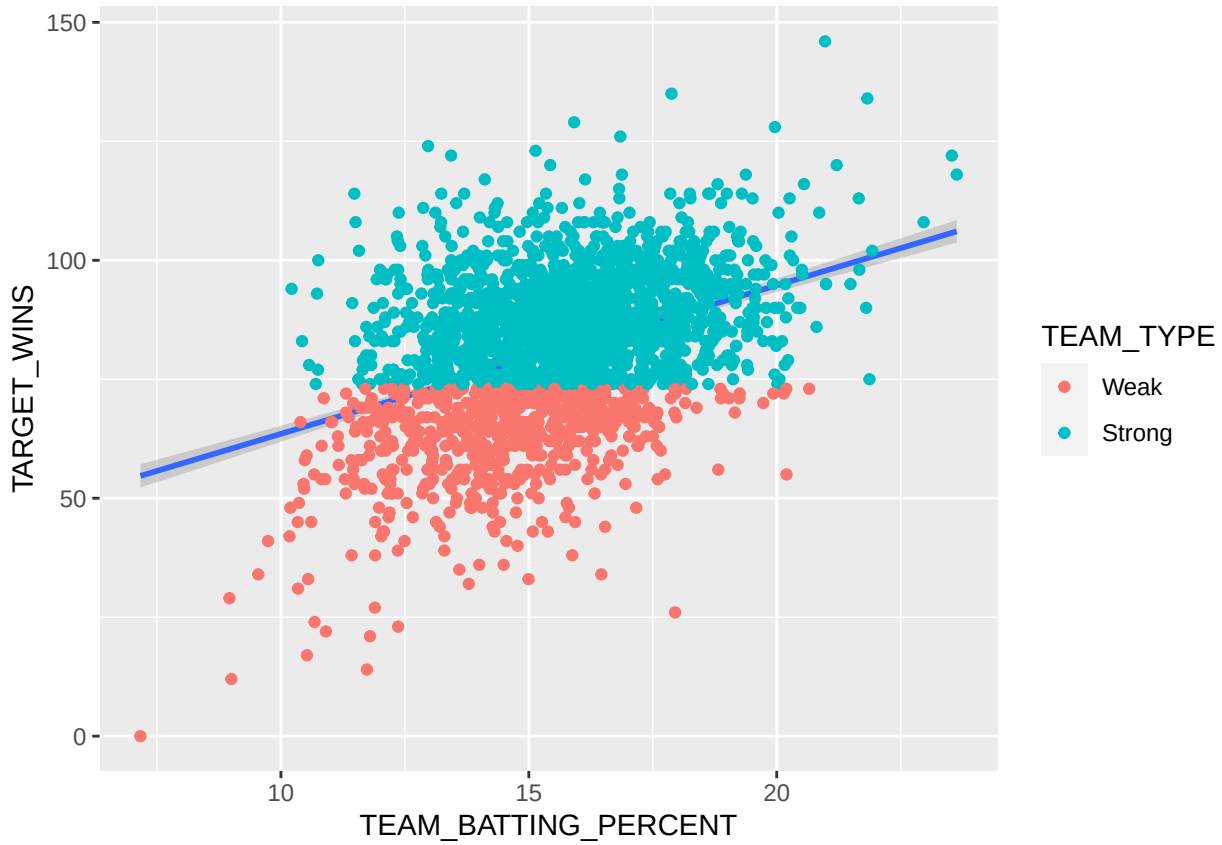
Hit Percentage

We would like to create a column which states what is the teams hit/base they get per game. This will be calculated by summing the total amount of hits a team gets and dividing 162 game season.

```
moneyball_fill$TEAM_BATTING_PERCENT = moneyball_fill$TEAM_BATTING_TOTAL/162
```

```
ggplot(moneyball_fill, aes(x = TEAM_BATTING_PERCENT, y = TARGET_WINS)) + geom_smooth(method = "lm") +
    geom_point(aes(color = TEAM_TYPE))
```

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



Model Building

At the beginning, we were presented with 16 independent variables. It makes sense to exclude index since it is not relevant. It also makes sense to exclude team_batting_hbp and team_batting_cs since they are comprised of so many N/As. We are thus able to concentrate on the 13 remaining variables, pursuing continuous incremental model improvement.

Our models are outlined below:

lmodel1 - an "all-in" model that includes all 13 remaining variables lmodel2 - a model that strips out outliers lmodel3 - a model that eliminates impertinent attributes

```
names(moneyball_fill) <- tolower(names(moneyball_fill))
# let's strip out the team type since it doesn't enhance the model
train1 <- subset(moneyball_fill, select = -c(team_type))
head(train1)
```

```
##   target_wins team_batting_h team_batting_2b team_batting_3b team_batting_hr
## 1          39           1445             194              39              13
## 2          70           1339             219              22             190
## 3          86           1377             232              35             137
## 4          70           1387             209              38              96
## 5          82           1297             186              27             102
## 6          75           1279             200              36              92
##   team_batting_bb team_batting_so team_baserun_sb team_pitching_h
## 1             143             842             101            9364
## 2             685            1075              37            1347
## 3             602             917              46            1377
```

```
## 4             451             922              43            1396
## 5             472             920              49            1297
## 6             443             973             107            1279
##   team_pitching_hr team_pitching_bb team_pitching_so team_fielding_e
## 1               84              927             5456            1011
## 2              191              689             1082             193
## 3              137              602              917             175
## 4               97              454              928             164
## 5              102              472              920             138
## 6               92              443              973             123
##   team_fielding_dp team_batting_total team_batting_percent
## 1              149               2002              12.35802
## 2              155               2603              16.06790
## 3              153               2494              15.39506
## 4              156               2303              14.21605
## 5              168               2158              13.32099
## 6              149               2155              13.30247
```

We'll start with the all-in model

```
lmodel1 <- lm(target_wins ~ ., data = train1)
summary(lmodel1)
```

```
##
## Call:
## lm(formula = target_wins ~ ., data = train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.827  -8.580   0.103   8.432  58.544
##
## Coefficients: (2 not defined because of singularities)
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         22.9775583  5.3046349   4.332 1.54e-05 ***
## team_batting_h       0.0488787  0.0036941  13.232  < 2e-16 ***
## team_batting_2b     -0.0212136  0.0091699  -2.313 0.020791 *
## team_batting_3b      0.0649302  0.0167897   3.867 0.000113 ***
## team_batting_hr      0.0545602  0.0273630   1.994 0.046279 *
## team_batting_bb      0.0105502  0.0058352   1.808 0.070734 .
## team_batting_so     -0.0084176  0.0025457  -3.307 0.000959 ***
## team_baserun_sb      0.0247806  0.0042572   5.821 6.69e-09 ***
## team_pitching_h     -0.0008598  0.0003668  -2.344 0.019147 *
## team_pitching_hr     0.0123395  0.0243703   0.506 0.612672
## team_pitching_bb     0.0008863  0.0041539   0.213 0.831065
## team_pitching_so     0.0028087  0.0009218   3.047 0.002338 **
## team_fielding_e     -0.0191590  0.0024016  -7.978 2.35e-15 ***
## team_fielding_dp    -0.1219877  0.0129372  -9.429  < 2e-16 ***
## team_batting_total          NA         NA      NA       NA
## team_batting_percent        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.07 on 2262 degrees of freedom
```

```
## Multiple R-squared:  0.3152, Adjusted R-squared:  0.3113
## F-statistic:  80.1 on 13 and 2262 DF,  p-value: < 2.2e-16
```

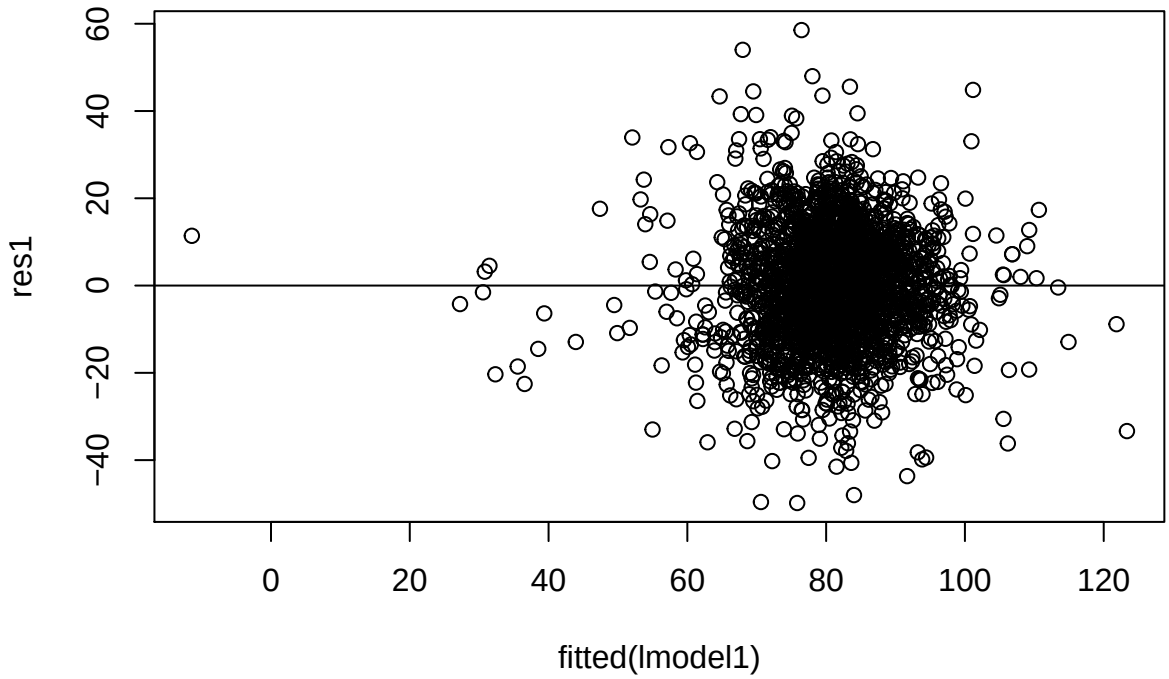
So in looking at the all-in model, we can identify how the model behaves intuitively and not-so-intuitively. For example, we see the following variables as having positive coefficients: team_batting_h, team_batting_3b, team_baserun_sb, and team_pitching_strikeouts. These make sense, as you'd expect a team to win games that gets hits, hits triples, steals bases efficiently, and strikes out opponents. However, some of the positive coefficients don't make as much sense. For example, we would expect teams whose pitchers give up lots of home runs to not win very many games. This certainly warrants further analysis.

For negative coefficients, we'd obviously expect teams whose players make a lot of errors to not win at a high rate. However, hitting doubles and fielding double plays have negative coefficients as well, which are not intuitive at all.

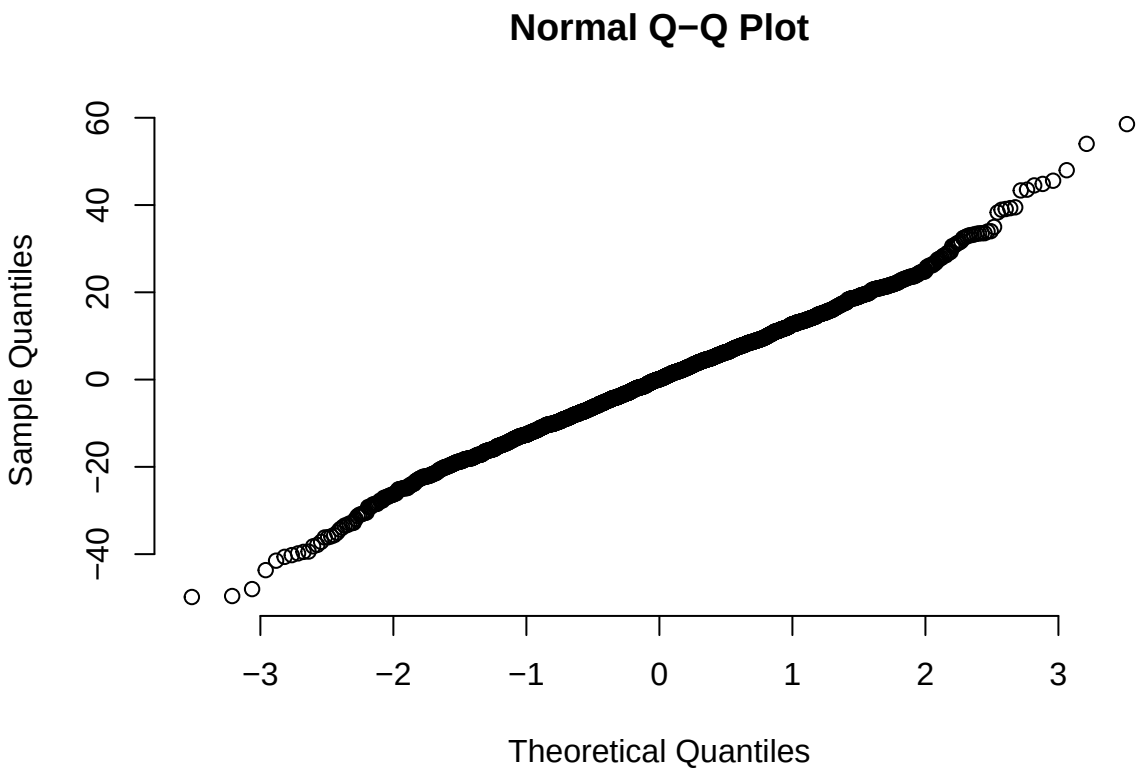
A majority of the variables that we are assessing appear to contribute to predicting wins. We can gain some comfort in our model due to the low RSE (13.07) and satisfactory F-statistic (80.1), and we should feel ok about the overall efficacy of our model. However, the Adjusted R-square well under 1 is cause for some concern, but we can look to improve that in future iterations of the model.

What else can we do to improve our model? Well, its predictive value might be enhanced by eliminating some problematic outliers. So let's take a look at if it makes sense to do so.

```
res1 <- resid(lmodel1)
plot(fitted(lmodel1), res1)

abline(0, 0)
```

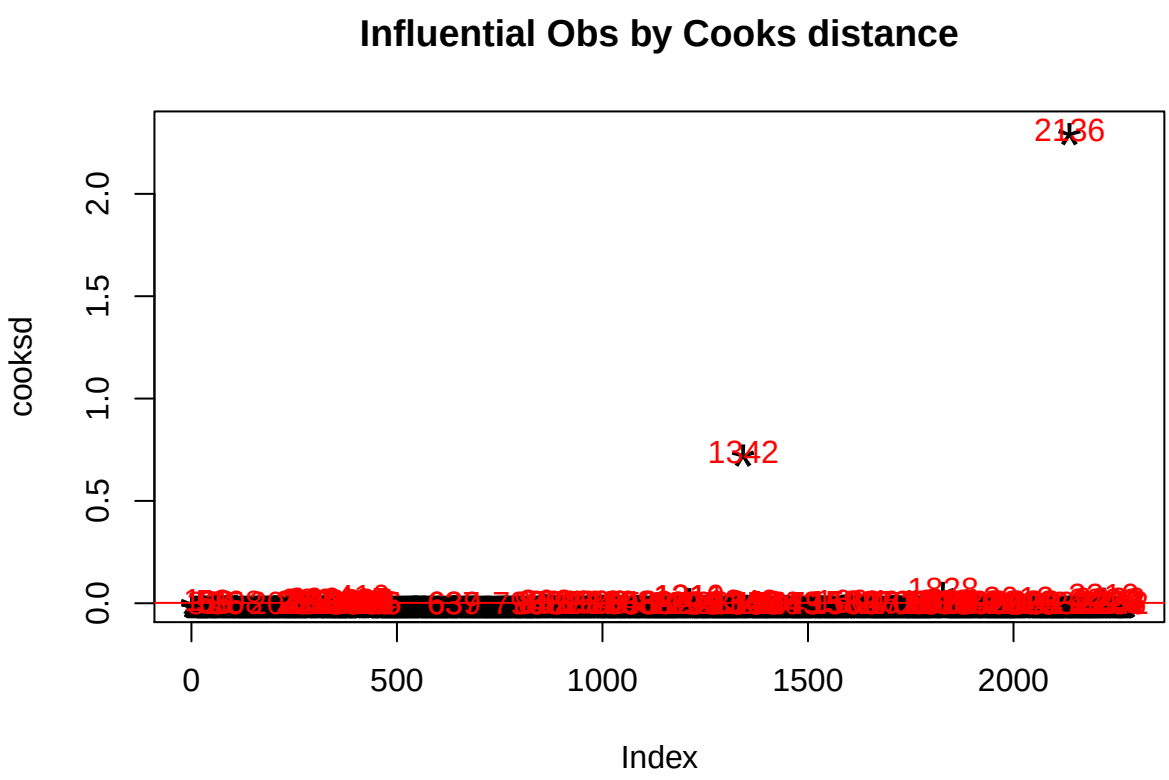


```r
qqnorm(res1, pch = 1, frame = FALSE)
```



The data is not evenly scattered but we don't detect any unexpected non-linear patter. The normal QQ looks good as well with a relatively straight line. We can spot some outliers that we should drill down on using Cook's Distance. Then, we can then attempt to strip them out to improve our model somewhat.

```r
cooksd <- cooks.distance(lmodel1)
sample_size <- nrow(train1)
plot(cooksd, pch = "*", cex = 2, main = "Influential Obs by Cooks distance")  # plot cook's distance
abline(h = 4/sample_size, col = "red")  # add cutoff line
text(x = 1:length(cooksd) + 1, y = cooksd, labels = ifelse(cooksd > 4/sample_size,
    names(cooksd), ""), col = "red")
```



We can spot two that breach our threshold, so now we set about removing them. Next, we can re-run our initial all-in model to see if dropping the outliers has any impact on improving the model.

```
influential <- as.numeric(names(cooksd)[(cooksd > (4/sample_size))])
train1_strip <- train1[-influential, ]

lmodel2 <- lm(target_wins ~ ., data = train1_strip)
summary(lmodel2)
```

```
##
## Call:
## lm(formula = target_wins ~ ., data = train1_strip)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.469  -7.796   0.246   7.405  34.488
##
## Coefficients: (2 not defined because of singularities)
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          30.440842   4.948518   6.152 9.13e-10 ***
## team_batting_h        0.031941   0.003887   8.218 3.55e-16 ***
## team_batting_2b      -0.038693   0.008314  -4.654 3.46e-06 ***
## team_batting_3b       0.110678   0.016304   6.788 1.46e-11 ***
## team_batting_hr       0.079148   0.043669   1.812   0.0701 .
## team_batting_bb       0.099555   0.012268   8.115 8.08e-16 ***
## team_batting_so      -0.047069   0.005498  -8.561  < 2e-16 ***
```

```
## team_baserun_sb       0.050541   0.004185  12.076  < 2e-16 ***
## team_pitching_h       0.008201   0.001069   7.670 2.59e-14 ***
## team_pitching_hr      0.008620   0.040704   0.212   0.8323
## team_pitching_bb     -0.069371   0.010792  -6.428 1.59e-10 ***
## team_pitching_so      0.034187   0.004537   7.536 7.13e-14 ***
## team_fielding_e      -0.040194   0.003121 -12.880  < 2e-16 ***
## team_fielding_dp     -0.119672   0.011334 -10.559  < 2e-16 ***
## team_batting_total          NA         NA      NA       NA
## team_batting_percent        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.98 on 2140 degrees of freedom
## Multiple R-squared:  0.3925, Adjusted R-squared:  0.3888
## F-statistic: 106.4 on 13 and 2140 DF,  p-value: < 2.2e-16
```

This looks like good news. Our RSE is down, and our F-statistic is up. Even our Adjusted R-Squared value is up slightly from .31. Nevertheless, the explanatory value of our model remains limited without this last number increasing significantly. And we can clearly see some variables with high p-values that ought to be removed in order to improve our model. Let's proceed with removing team_batting_hr and team_picthing_hr.

```
train3 <- subset(train1_strip, select = -c(team_batting_hr, team_pitching_hr))
lmodel3 <- lm(target_wins ~ ., data = train3)
summary(lmodel3)
```

```
##
## Call:
## lm(formula = target_wins ~ ., data = train3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.462  -7.810   0.229   7.392  34.504
##
## Coefficients: (1 not defined because of singularities)
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          30.418573   4.946297   6.150 9.23e-10 ***
## team_batting_h        0.010015   0.005093   1.966   0.0494 *
## team_batting_2b      -0.082926   0.009262  -8.953  < 2e-16 ***
## team_batting_3b       0.044487   0.015645   2.844   0.0045 **
## team_batting_bb       0.098449   0.011100   8.869  < 2e-16 ***
## team_batting_so      -0.047460   0.005179  -9.164  < 2e-16 ***
## team_baserun_sb       0.050534   0.004184  12.077  < 2e-16 ***
## team_pitching_h       0.008148   0.001039   7.842 6.92e-15 ***
## team_pitching_bb     -0.068337   0.009620  -7.103 1.65e-12 ***
## team_pitching_so      0.034524   0.004248   8.127 7.36e-16 ***
## team_fielding_e      -0.040318   0.003064 -13.159  < 2e-16 ***
## team_fielding_dp     -0.119690   0.011331 -10.563  < 2e-16 ***
## team_batting_total    0.022054   0.002145  10.282  < 2e-16 ***
## team_batting_percent        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 10.98 on 2141 degrees of freedom
## Multiple R-squared:  0.3925, Adjusted R-squared:  0.3891
## F-statistic: 115.3 on 12 and 2141 DF,  p-value: < 2.2e-16
```

We've improved the model incrementally by removing variables with high p-values, and our RSE and F-stat look better The explanatory power of our model, however, remains in doubt due to the Adjusted R-Squared value that remains low, even though it's improved from the previous model. What stands out here is that triples hit, bases stolen, and gaining walks remain the overall strongest positive coefficients, while team_fielding_dp remains the largest negative coefficient, which is counter-intuitive at first blush. However, one thing necessary for a double play is at least one opponent runner on base. Those teams that earn a high number of double plays are only able to do so because their pitchers are allowing runners on base to begin with.

Select Models

In order to select on model we need to consider few aspects of the models:

Degrees of Freedom: Number of observations minus the number of coefficients (including intercepts). The larger this number is the better and if it's close to 0, your model is seriously over fit.

R-squared:

R-squared evaluates the scatter of the data points around the fitted regression line. It is also called the coefficient of determination. For the same data set, higher R-squared values represent smaller differences between the observed data and the fitted values.

R-squared is the percentage of the dependent variable variation that a linear model explains.

R-squared is always between 0 and 100%:

0% represents a model that does not explain any of the variation in the response variable around its mean. The mean of the dependent variable predicts the dependent variable as well as the regression model. 100% represents a model that explains all the variation in the response variable around its mean. Usually, the larger the R2, the better the regression model fits your observations.

Adjusted R-squared:

The adjusted R-squared is a modified version of R-squared that adjusts for predictors. when we compare Adj-R-Squared among the input variables, sa lower adjusted R-squared indicates that the additional input variables are not adding value to the model. A higher adjusted R-squared indicates that the changes in input variables are adding value to the model.s

Residuals:

Residuals are estimates of experimental error obtained by subtracting the observed responses from the predicted responses. The predicted response is calculated from the chosen model. Since this is a form of error, the same general assumptions apply to the group of residuals that we typically use for errors in general: one expects them to be (roughly) normal and (approximately) independently distributed with a mean of 0 and some constant variance.in other words, we should not see any pattern in the residuals when plotting. A simple plot is suitable for displaying the normality of the distribution of a group of residuals

p-value:

we need to evaluate p-value and if the p-value is much less than 0.05 then we can reject the null hypothesis. That will indicate that there is a significant relationship between the variables in the linear regression model of the data set faithful.

Model comparison

Now, before we select our model, let's find out the statistics of our models so that we can compare the models using the statistics i.e. R2, MSE, F-statistic, Number of Variables (K), Number of Observations (N), and number of observations in the original training set that were excluded from the model.

name	rsquared	adjustedR2	mse	f	pvalue	k	n
model1	0.3152383	0.3113029	169.8355	80.10299	0.0247806	13	2262
model2	0.3924991	0.3888087	119.7359	106.35604	0.0505407	13	2140
model3	0.3924864	0.3890813	119.7384	115.26673	0.0081481	12	2141

From the table above, the r-Squared of Model2 and Model3 are almost similar. The adjusted R2 is little higher in the model 3. In addition the p-value is significant in all three models but the Model3 has lowest p-value. Based on the statistics and diagnostics, we can select Model3 for our final model.

Final Model Review

Let's take a look to some other aspects of our final model.

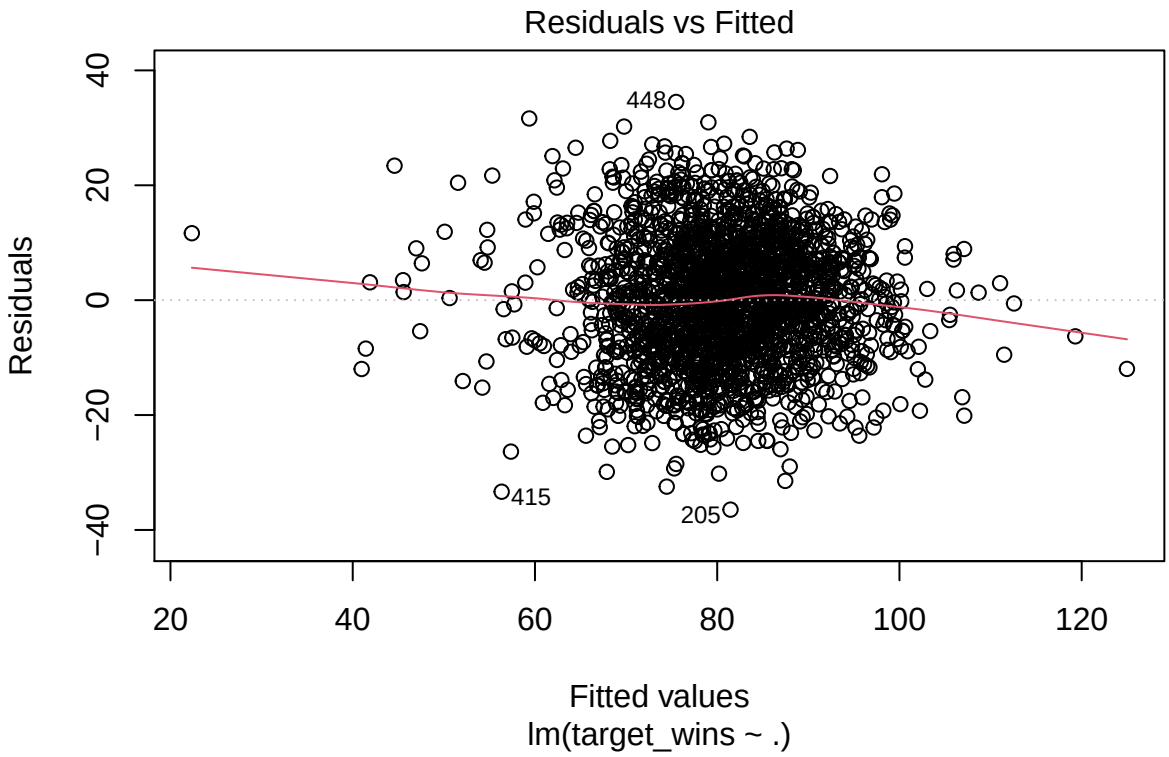
Our final model is Model3(lmodel3):

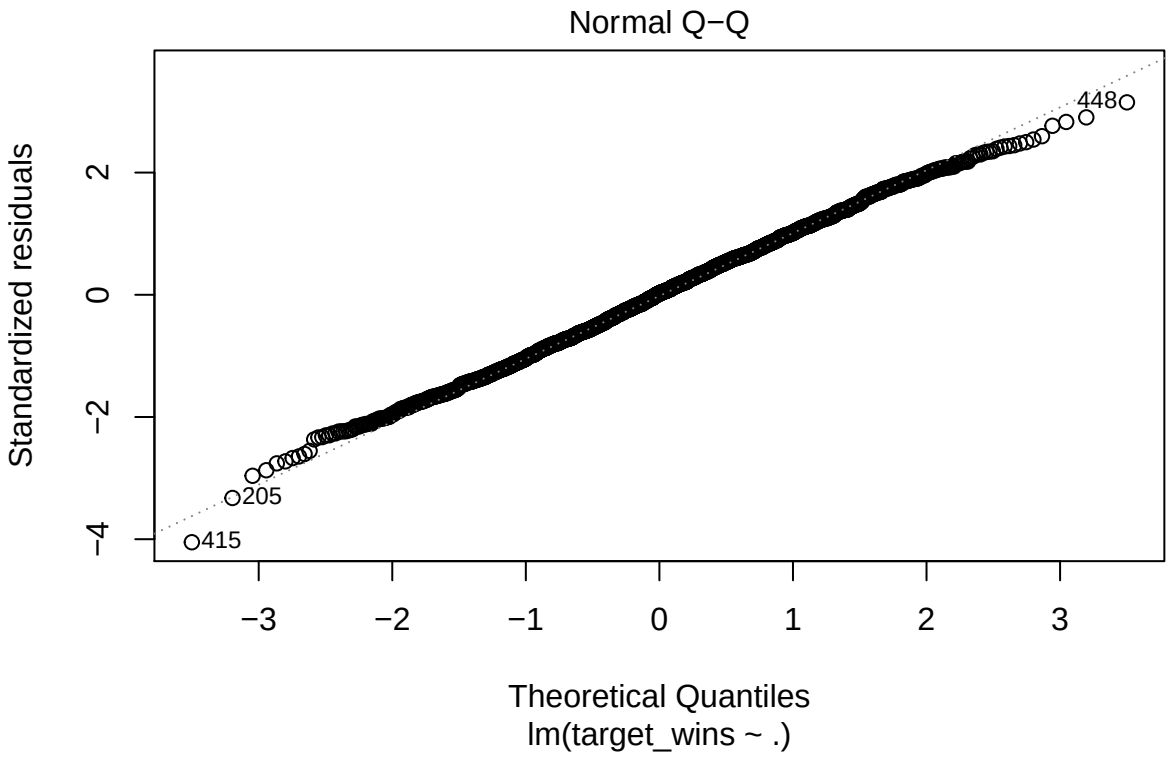
And the dependents columns are:

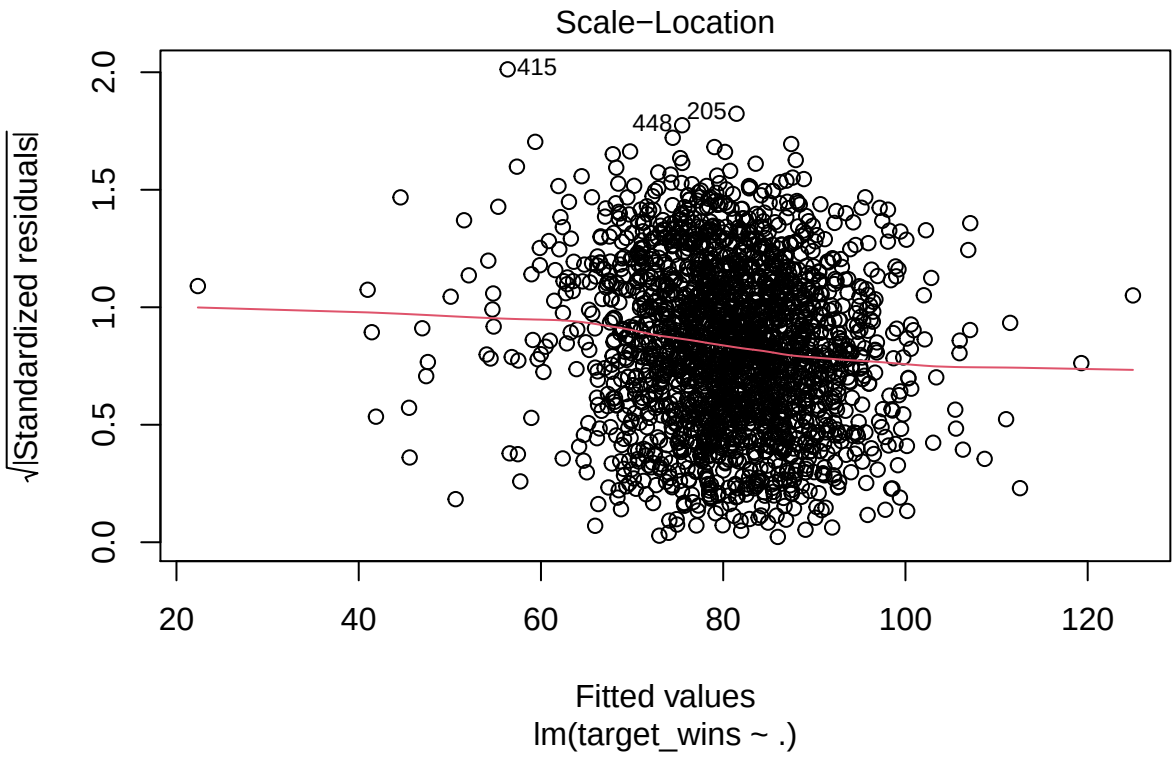
```
print(names(train3))
```

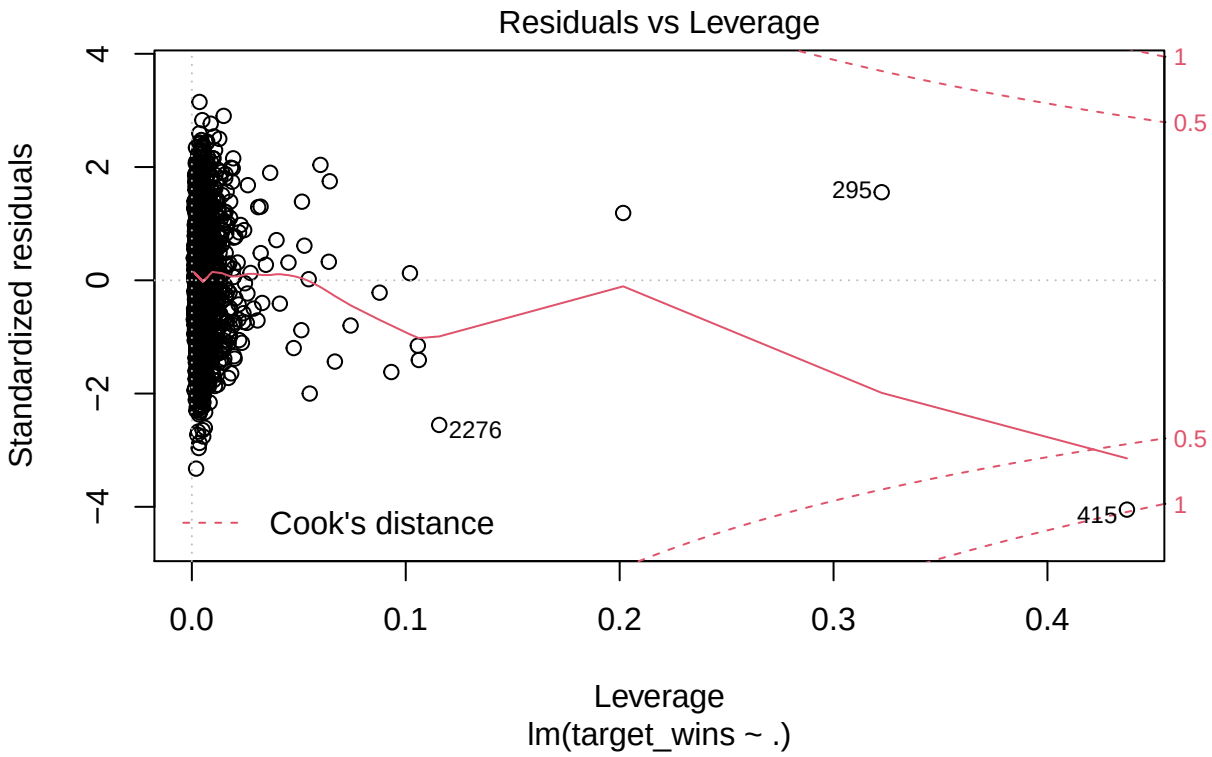
```
##  [1] "target_wins"          "team_batting_h"       "team_batting_2b"
##  [4] "team_batting_3b"      "team_batting_bb"      "team_batting_so"
##  [7] "team_baserun_sb"      "team_pitching_h"      "team_pitching_bb"
## [10] "team_pitching_so"     "team_fielding_e"      "team_fielding_dp"
## [13] "team_batting_total"   "team_batting_percent"
```

Let's review the diagnostic plots and a plot of the residuals.

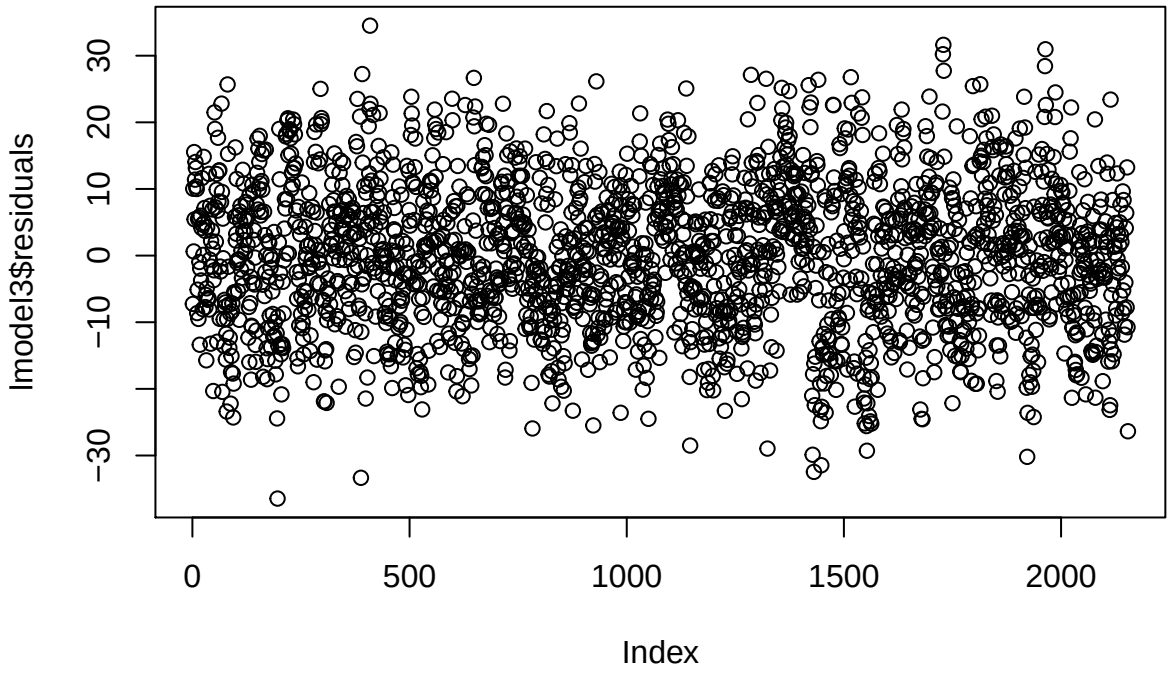

Residuals vs Fitted
Residuals
Fitted values
lm(target_wins ~ .)
448
415
205
40
20
0
−20
−40
20
40
60
80
100
120


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(target_wins ~ .)
448
205
415
−4
−2
0
2
−3
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
415
448
205
20
40
60
80
100
120
Fitted values
lm(target_wins ~ .)

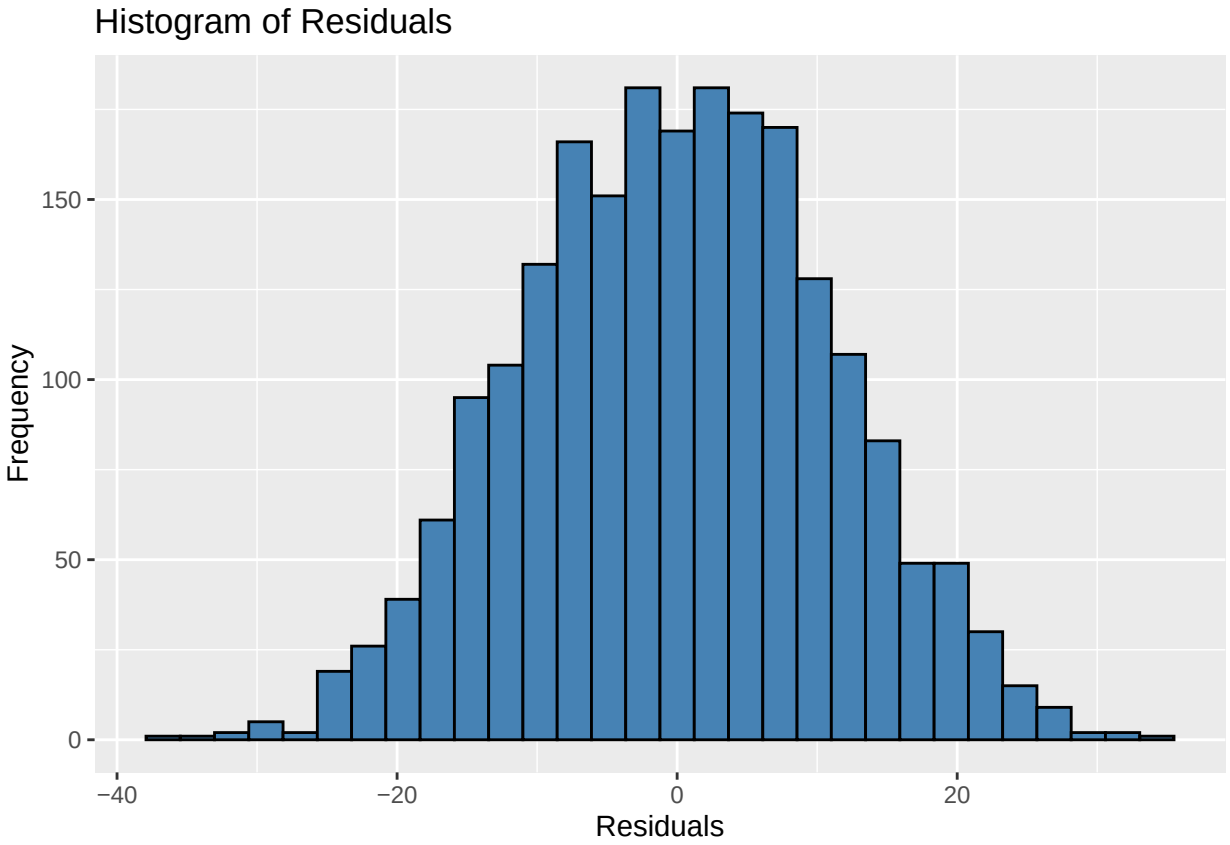

Residuals vs Leverage
Standardized residuals
4
2
0
−2
−4
0.0
0.1
0.2
0.3
0.4
Leverage
lm(target_wins ~ .)
Cook's distance
1
0.5
0.5
1
295
2276
415

Plot the residual of selected model



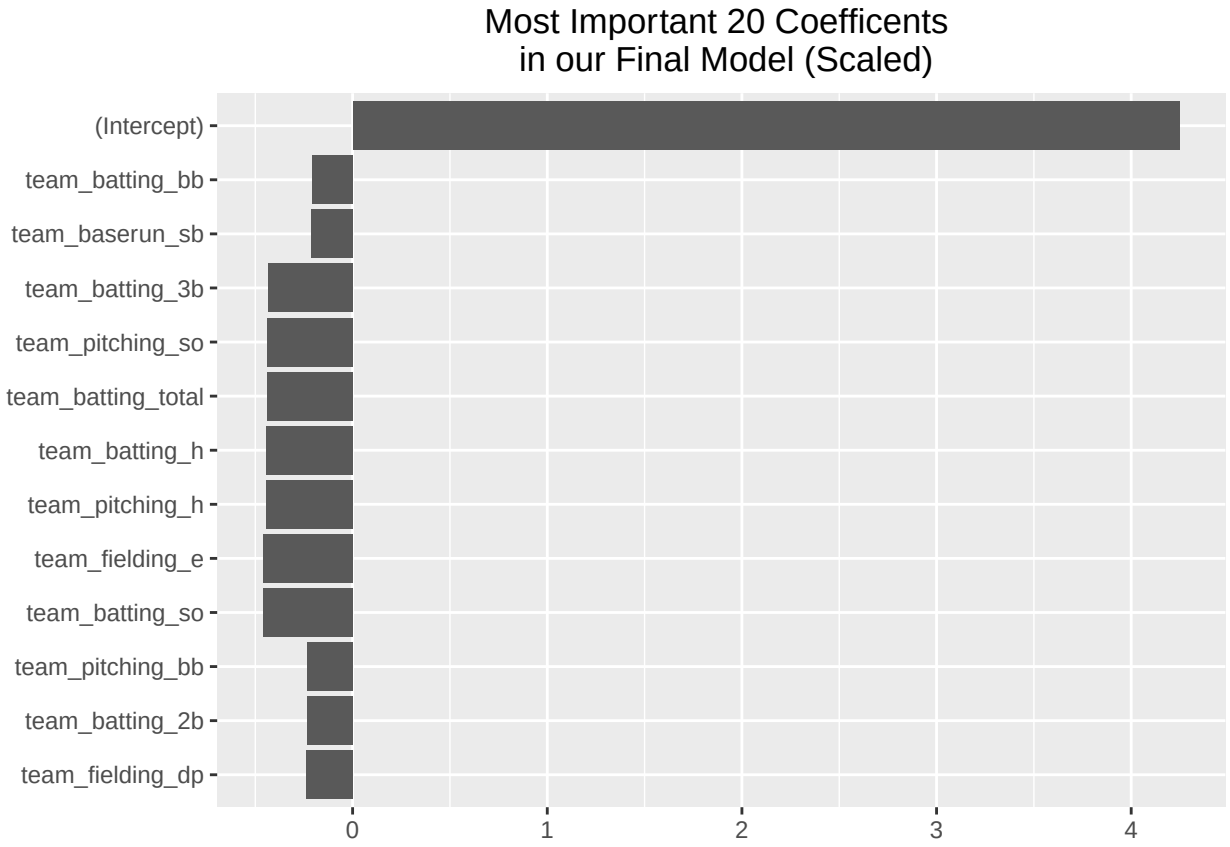
From this plot, we can conclude that there is no obvious pattern in this plot.

Create histogram of the residuals of selected model



From this histogram we can visualize that the distribution is pretty normal.

Plot the top Coefficients of our model



Predictions using evaluation data

Let's use our evaluation data to predict and evaluate our selected model.

Prediction explanation will be added by Zhouxin Shi(Joe)