DATA 621—Assignment no. 1

Critical Thinking Group 2: All of our names, Ben H.

September XX, 2019

Contents

Introduction

Load libraries:

```
library(corrplot)
library(dplyr)
library(ggplot2)
library(knitr)

library(kableExtra)
# library(psych)
library(reshape2)
library(gridExtra)
#library(psychometric)
#library(ggpubr)
#library(matlib)
#library(matrixcalc)
# library(psych)
# library(MASS)
# library(forecast)
```

Executive Overview

We present three multiple regression models to predict a professional baseball teams' performance.

Data Exploration

The training data has 17 columns and 2,276 rows.

The explanatory columns are broken down into four categories:

- Batting
- Base run
- Pitching
- Fielding

Below you will see a preview of the columns and the first few observations broken down into these four categories.

```
##   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
```

Response Variable: Yearly wins

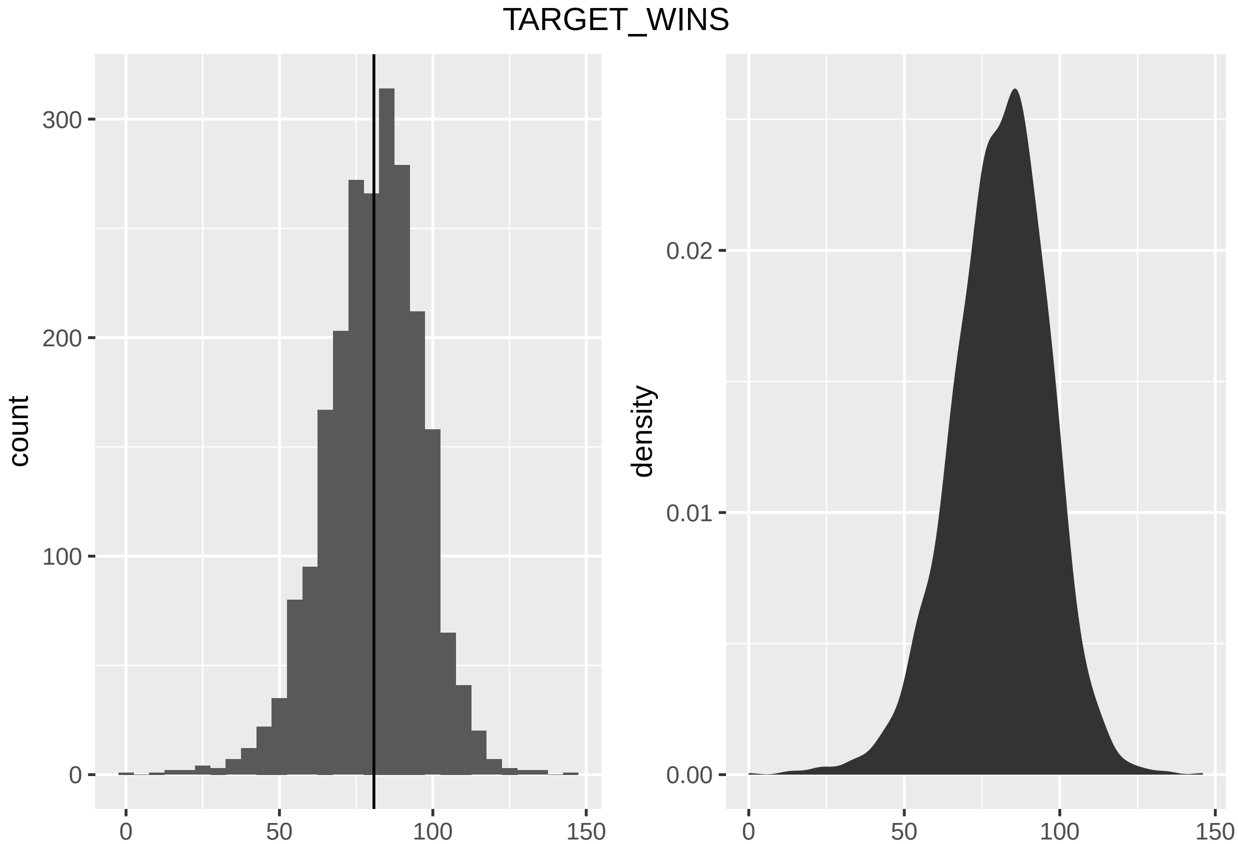
The variable `TARGET_WINS` is the number of wins of a professional baseball team for a given year. The year is not part of the data set.

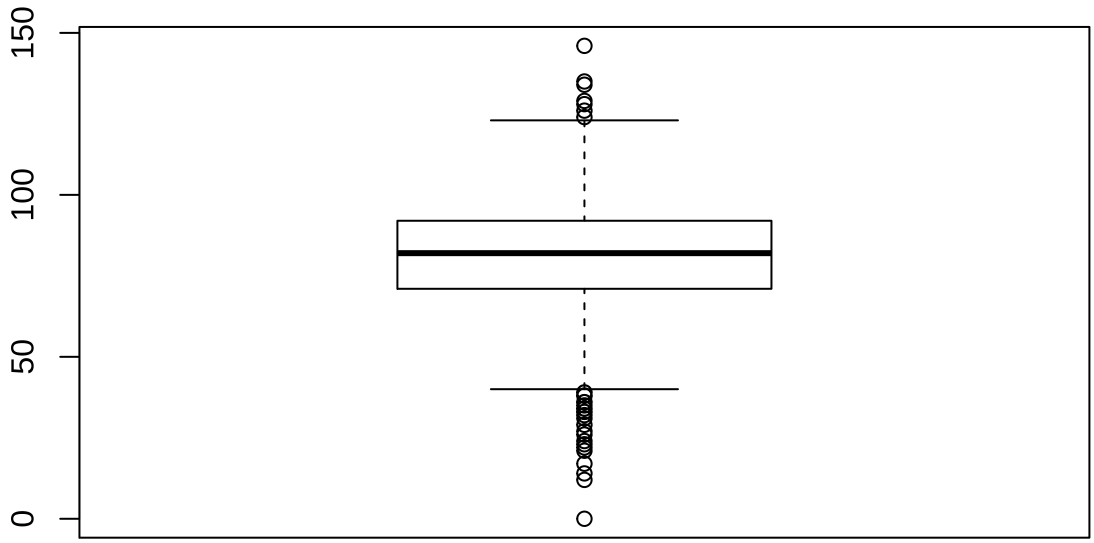
This is the dependent variable that our models will attempt to predict. It is characterized by:

	TARGET_WINS
	Min. : 0.00
	1st Qu.: 71.00
	Median : 82.00
	Mean : 80.79
	3rd Qu.: 92.00
	Max. :146.00

	n	sd	se
TARGET_WINS	2276	15.75215	0.3301823

the distribution of the number of wins is unimodal and skewed to the left with some outliers towards the tail. It looks approximately normal, though the boxplot shows there are quite a few outliers. The minimum number of wins for a team is 0 and the maximum is 146. The mean is 80.79.





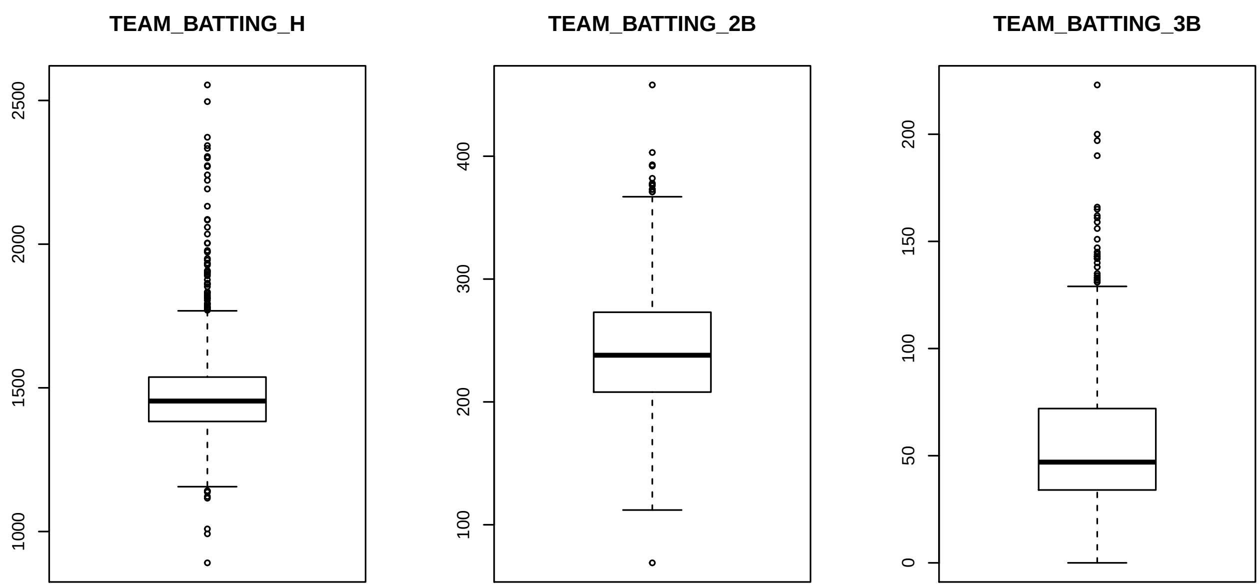
Explanatory Variables

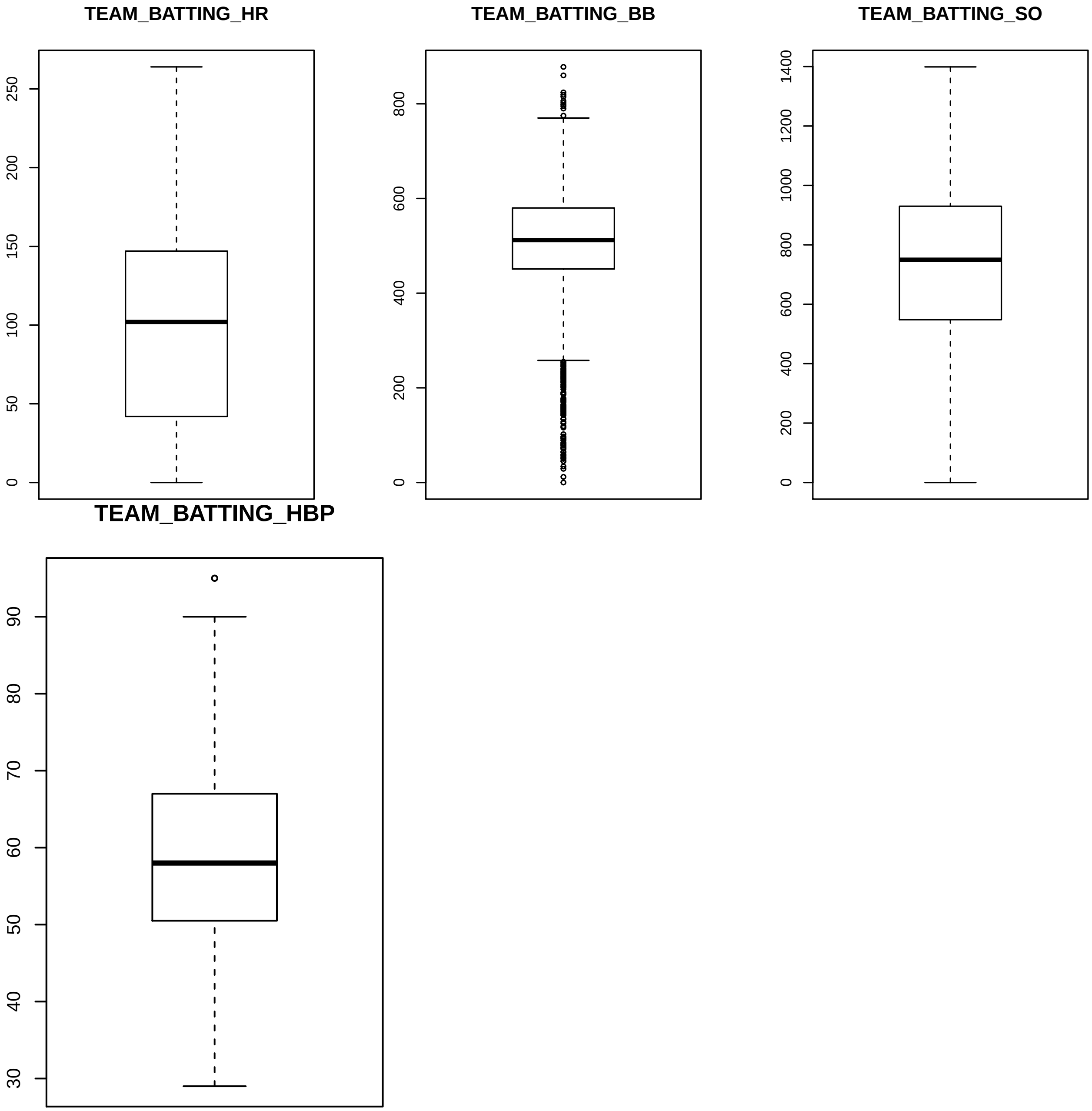
Batting variables

Below are our batting variables and their hypothesized effect on `TARGET_WINS`:

DESCRIPTION AND THEORETICAL EFFECT

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
TEAM_BATTING_H	Base Hits by batters (1B,2B,3B,HR)	Positive Impact on Wins
TEAM_BATTING_2B	Doubles by batters (2B)	Positive Impact on Wins
TEAM_BATTING_3B	Triples by batters (3B)	Positive Impact on Wins
TEAM_BATTING_HR	Homeruns by batters (4B)	Positive Impact on Wins
TEAM_BATTING_BB	Walks by batters	Positive Impact on Wins
TEAM_BATTING_HBP	Batters hit by pitch (get a free base)	Positive Impact on Wins
TEAM_BATTING_SO	Strikeouts by batters	Negative Impact on Wins





The boxplots hint that some of these variables are quite skewed, especially `TEAM_BATTING_BB` and `TEAM_BATTING_H`.

Since all of these variables relate to the same thing, batting, we expect at least some of them to be correlated. This has implications on later modeling:

	TARGET_WINS	TEAM_BATTING_H	TEAM_BATTING_2B	TEAM_BATTING_3B	TEAM_BATTING_HR	TEAM_BATTING_BB	TEAM_BATTING_SO	TEAM_BATTING_HBP
TARGET_WINS	1.0000000	0.4699467	0.3129840	-0.1243459	0.4224168	0.4686879	-0.2288927	0.0735042
TEAM_BATTING_H	0.4699467	1.0000000	0.5617729	0.2139188	0.3962759	0.1973523	-0.3417433	-0.0291122
TEAM_BATTING_2B	0.3129840	0.5617729	1.0000000	0.0420344	0.2509905	0.1974926	-0.0641512	0.0460848

	TARGET_WINS	TEAM_BATTING_H	TEAM_BATTING_2B	TEAM_BATTING_3B	TEAM_BATTING_HR	TEAM_BATTING_BB	TEAM_BATTING_SO	TEAM_BATTING_HBP
TEAM_BATTING_3B	-0.1243459	0.2139188	0.0420344	1.0000000	-0.2187993	-0.2058439	-0.1929184	-0.1742472
TEAM_BATTING_HR	0.1424168	0.3962759	0.2509905	-0.2187993	1.0000000	0.4563816	0.2104544	0.1061812
TEAM_BATTING_BB	0.1468793	0.1973523	0.1974926	-0.2058439	0.4563816	1.0000000	0.2183387	0.0474601
TEAM_BATTING_SO	-0.2288927	-0.3417433	-0.0641512	-0.1929184	0.2104544	0.2183387	1.0000000	0.2209422
TEAM_BATTING_HBP	0.0735042	-0.0291122	0.0460848	-0.1742472	0.1061812	0.0474601	0.2209422	1.0000000

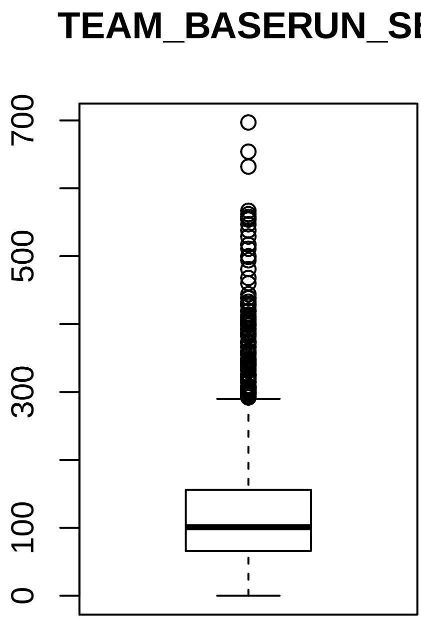
Baserun Variables

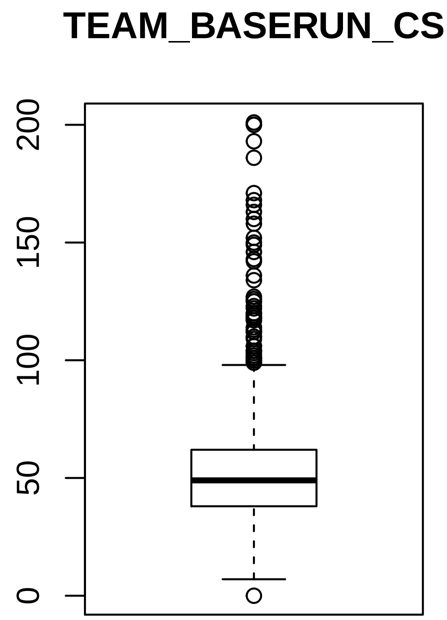
Description and theoretical effects:

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
TEAM_BASERUN_SB	Stolen bases	Positive Impact on Wins
TEAM_BASERUN_CS	Caught stealing	Negative Impact on Wins

As you can see, both variables have some missing values:

TEAM_BASERUN_SB	TEAM_BASERUN_CS
Min. : 0.0	Min. : 0.0
1st Qu.: 66.0	1st Qu.: 38.0
Median :101.0	Median : 49.0
Mean :124.8	Mean : 52.8
3rd Qu.:156.0	3rd Qu.: 62.0
Max. :697.0	Max. :201.0
NA's :131	NA's :772





	TARGET_WINS	TEAM_BASERUN_SB	TEAM_BASERUN_CS
TARGET_WINS	1.0000000	0.1539213	0.0224041
TEAM_BASERUN_SB	0.1539213	1.0000000	0.6552448
TEAM_BASERUN_CS	0.0224041	0.6552448	1.0000000

Pitching Variables

Description:

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
TEAM_PITCHING_BB	Walks allowed	Negative Impact on Wins
TEAM_PITCHING_H	Hits allowed	Negative Impact on Wins
TEAM_PITCHING_HR	Homeruns allowed	Negative Impact on Wins
TEAM_PITCHING_SO	Strikeouts by pitchers	Positive Impact on Wins

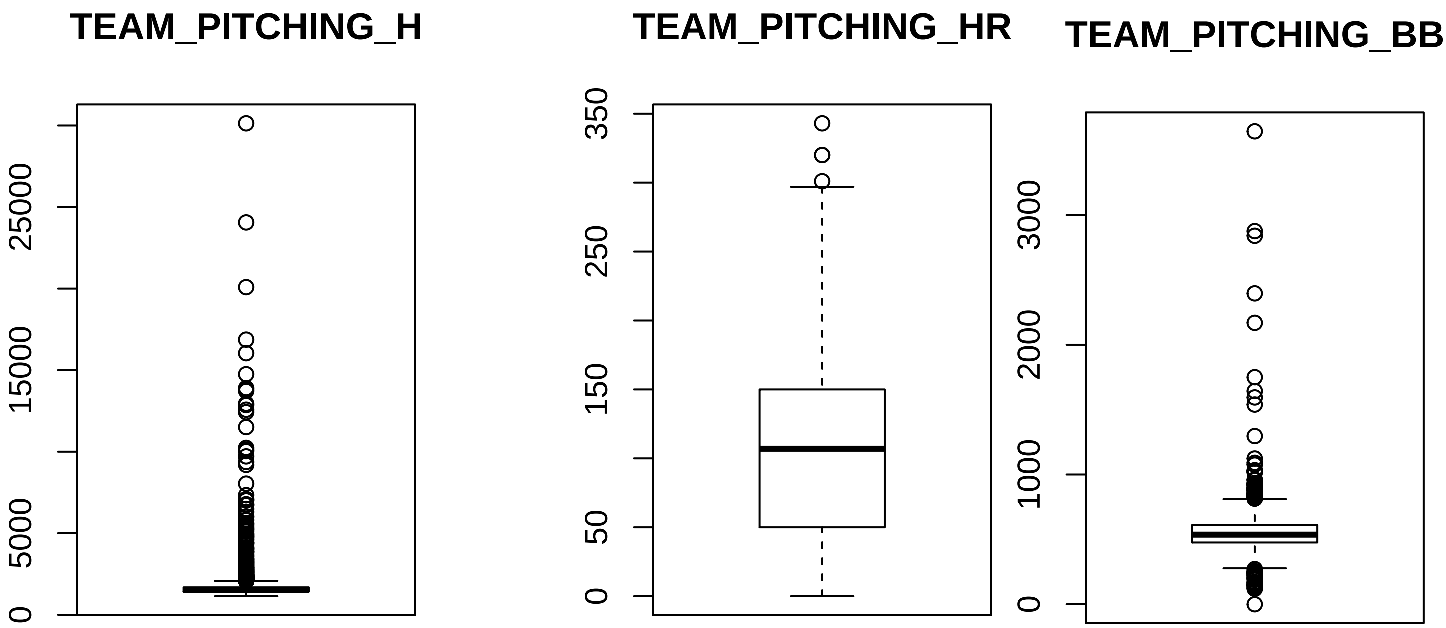
```
kable(head(data[c(12:15)]), format='markdown')
```

TEAM_PITCHING_H	TEAM_PITCHING_HR	TEAM_PITCHING_BB	TEAM_PITCHING_SO
9364	84	927	5456
1347	191	689	1082
1377	137	602	917
1396	97	454	928
1297	102	472	920
1279	92	443	973

```
kable(summary(data[c(12:15)]), format='markdown')
```

TEAM_PITCHING_H	TEAM_PITCHING_HR	TEAM_PITCHING_BB	TEAM_PITCHING_SO
Min. : 1137	Min. : 0.0	Min. : 0.0	Min. : 0.0

TEAM_PITCHING_H	TEAM_PITCHING_HR	TEAM_PITCHING_BB	TEAM_PITCHING_SO
1st Qu.: 1419	1st Qu.: 50.0	1st Qu.: 476.0	1st Qu.: 615.0
Median : 1518	Median :107.0	Median : 536.5	Median : 813.5
Mean : 1779	Mean :105.7	Mean : 553.0	Mean : 817.7
3rd Qu.: 1682	3rd Qu.:150.0	3rd Qu.: 611.0	3rd Qu.: 968.0
Max. :30132	Max. :343.0	Max. :3645.0	Max. :19278.0
NA	NA	NA	NA's :102




```
kable(cor(data[c(2, 12:15)]), format='markdown')
```

	TARGET_WINS	TEAM_PITCHING_H	TEAM_PITCHING_HR	TEAM_PITCHING_BB	TEAM_PITCHING_SO
TARGET_WINS	1.0000000	-0.1099371	0.1890137	0.1241745	NA
TEAM_PITCHING_H	-0.1099371	1.0000000	-0.1416128	0.3206762	NA
TEAM_PITCHING_HR	0.1890137	-0.1416128	1.0000000	0.2219375	NA
TEAM_PITCHING_BB	0.1241745	0.3206762	0.2219375	1.0000000	NA
TEAM_PITCHING_SO	NA	NA	NA	NA	1

Fielding Variables

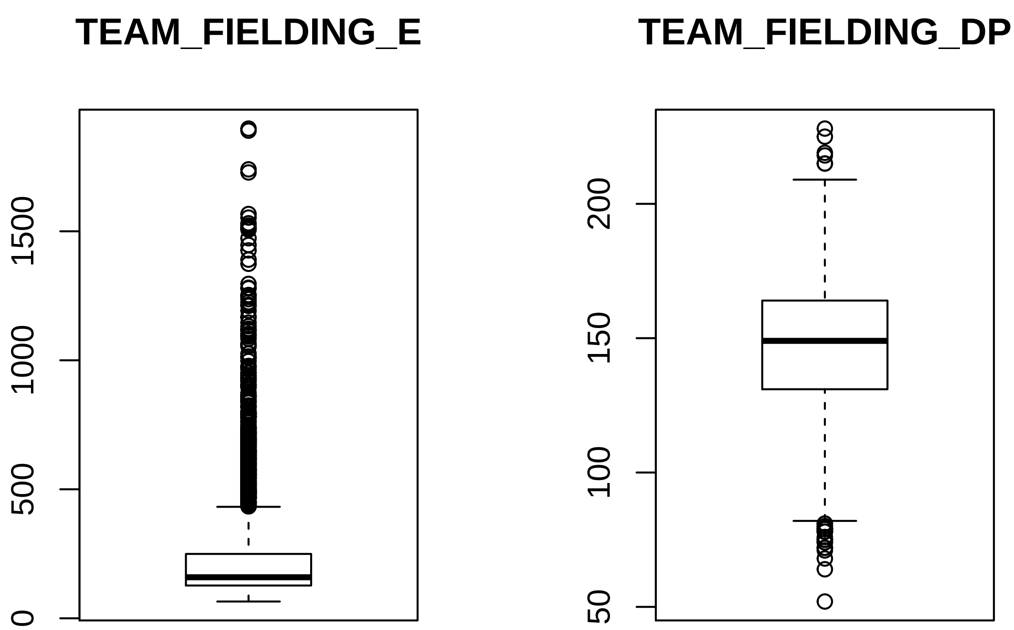
Description:

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
TEAM_FIELDING_E	Errors	Negative Impact on Wins
TEAM_FIELDING_DP	Double Plays	Positive Impact on Wins

```
kable(summary(data[c(16:17)]), format='markdown')
```

TEAM_FIELDING_E	TEAM_FIELDING_DP
Min. : 65.0	Min. : 52.0

TEAM_FIELDING_E	TEAM_FIELDING_DP
1st Qu.: 127.0	1st Qu.:131.0
Median : 159.0	Median :149.0
Mean : 246.5	Mean :146.4
3rd Qu.: 249.2	3rd Qu.:164.0
Max. :1898.0	Max. :228.0
NA	NA's :286




```r
kable(cor(data[c(2, 16:17)]))
```

	TARGET_WINS	TEAM_FIELDING_E	TEAM_FIELDING_DP
TARGET_WINS	1.0000000	-0.1764848	NA
TEAM_FIELDING_E	-0.1764848	1.0000000	NA
TEAM_FIELDING_DP	NA	NA	1

Data Preparation

There are missing data in this dataset:

```r
sort(sapply(data, FUN=function(x) mean(is.na(x))), decreasing=TRUE)
```

```
## TEAM_BATTING_HBP  TEAM_BASERUN_CS TEAM_FIELDING_DP  TEAM_BASERUN_SB
##       0.91608084       0.33919156       0.12565905       0.05755712
##  TEAM_BATTING_SO TEAM_PITCHING_SO            INDEX      TARGET_WINS
##       0.04481547       0.04481547       0.00000000       0.00000000
##   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B  TEAM_BATTING_HR
##       0.00000000       0.00000000       0.00000000       0.00000000
##  TEAM_BATTING_BB  TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB
##       0.00000000       0.00000000       0.00000000       0.00000000
##  TEAM_FIELDING_E
##       0.00000000
```

The `TEAM_BATTING_HBP` variable is almost entirely missing. We chose to omit it from the rest of the analysis.

As for the other variables, we are simply dropping them, further analyzing only the complete cases.

```
data$TEAM_BATTING_HBP <- NULL
data2 <- data[complete.cases(data), ]
```

Some variables can be transformed to appromximately normal distribution by logging them. We will test this below:

```
train <- data
train.df <- data
train.trans <- train.df
# cols.nomiss = names(train.df)[!names(train.df)%in%(cols.miss)]
train.trans$TEAM_PITCHING_HR = apply(train.trans['TEAM_PITCHING_HR'], 1, function(x) if(x==0){return(0)}
train.trans$TEAM_PITCHING_BB = apply(train.trans['TEAM_PITCHING_BB'], 1, function(x) if(x==0){return(0)}
train.trans$TEAM_FIELDING_E = apply(train.trans['TEAM_FIELDING_E'], 1, function(x) if(x==0){return(0)}
```

Modeling

```
# SAMRITA's prep ?
train.trans$TEAM_BATTING_H = train.trans$TEAM_BATTING_H  + train.trans$TEAM_BATTING_2B + train.trans$TEA
train.trans$TEAM_FIELDING_E = train.trans$TEAM_FIELDING_E + train.trans$TEAM_PITCHING_H
train.trans = train.trans%>%dplyr::select(-TEAM_BATTING_2B, -TEAM_BATTING_3B, -TEAM_BATTING_HR, -TEAM_P
```

Model 1

The first model `model` regresses `target_wins` on the five variables identified above:

```
model1 = lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB + TEAM_PITCHING_HR +
                TEAM_PITCHING_BB + TEAM_FIELDING_E, data=train.trans)
summary(model1)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_BB +
##     TEAM_PITCHING_HR + TEAM_PITCHING_BB + TEAM_FIELDING_E, data = train.trans)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.087  -9.176   0.455   9.110  59.641
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.8223840  8.3128468   0.340    0.734
## TEAM_BATTING_H    0.0383705  0.0017159  22.362  < 2e-16 ***
## TEAM_BATTING_BB   0.0167647  0.0041036   4.085 4.55e-05 ***
## TEAM_PITCHING_HR -1.9009586  0.4477722  -4.245 2.27e-05 ***
## TEAM_PITCHING_BB  1.5483525  1.5160674   1.021    0.307
## TEAM_FIELDING_E  -0.0018848  0.0002602  -7.244 5.95e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.77 on 2270 degrees of freedom
## Multiple R-squared:  0.2377, Adjusted R-squared:  0.236
```

```
## F-statistic: 141.6 on 5 and 2270 DF,  p-value: < 2.2e-16
```

All but one of the variables are extremely significant, including the intercept parameter. The F-statistic suggests the model is not just picking up random noise.

It estimates that base hits and walks by batters are positively related to team wins, and that walks allowed and errors and homeruns allowed are negatively related. Curiously, it finds `TEAM_PITCHING_BB` is positively related, although it was theorized to be negatively related.

Mean squared error, as a baseline for evaluation:

```r
mean(resid(model1)^2)
```

```
## [1] 189.0616
```

Model 2

The second model uses the same varibles, but some of them have been log-transformed, and we use squared functions of some of them, rather than the variable itself.

```r
train.trans = train.df
train.trans$TEAM_BATTING_H = train.trans$TEAM_BATTING_H  + train.trans$TEAM_BATTING_2B + train.trans$TEA
train.trans$TEAM_FIELDING_E = train.trans$TEAM_FIELDING_E + train.trans$TEAM_PITCHING_H
train.trans = train.trans%>%dplyr::select(-TEAM_BATTING_2B, -TEAM_BATTING_3B, -TEAM_BATTING_HR, -TEAM_PI
summary(train.trans)
```

```
##      INDEX         TARGET_WINS     TEAM_BATTING_H TEAM_BATTING_BB
##  Min.   :   1.0   Min.   :  0.00   Min.   :1026   Min.   :  0.0  
##  1st Qu.: 630.8   1st Qu.: 71.00   1st Qu.:1739   1st Qu.:451.0  
##  Median :1270.5   Median : 82.00   Median :1862   Median :512.0  
##  Mean   :1268.5   Mean   : 80.79   Mean   :1865   Mean   :501.6  
##  3rd Qu.:1915.5   3rd Qu.: 92.00   3rd Qu.:1978   3rd Qu.:580.0  
##  Max.   :2535.0   Max.   :146.00   Max.   :3092   Max.   :878.0  
##                                                                  
##  TEAM_BATTING_SO  TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_PITCHING_HR
##  Min.   :   0.0   Min.   :  0.0   Min.   :  0.0   Min.   :  0.0   
##  1st Qu.: 548.0   1st Qu.: 66.0   1st Qu.: 38.0   1st Qu.: 50.0   
##  Median : 750.0   Median :101.0   Median : 49.0   Median :107.0   
##  Mean   : 735.6   Mean   :124.8   Mean   : 52.8   Mean   :105.7   
##  3rd Qu.: 930.0   3rd Qu.:156.0   3rd Qu.: 62.0   3rd Qu.:150.0   
##  Max.   :1399.0   Max.   :697.0   Max.   :201.0   Max.   :343.0   
##  NA's   :102      NA's   :131     NA's   :772                     
##  TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##  Min.   :   0.0   Min.   :    0.0   Min.   : 1276   Min.   : 52.0   
##  1st Qu.: 476.0   1st Qu.:  615.0   1st Qu.: 1566   1st Qu.:131.0   
##  Median : 536.5   Median :  813.5   Median : 1679   Median :149.0   
##  Mean   : 553.0   Mean   :  817.7   Mean   : 2026   Mean   :146.4   
##  3rd Qu.: 611.0   3rd Qu.:  968.0   3rd Qu.: 1922   3rd Qu.:164.0   
##  Max.   :3645.0   Max.   :19278.0   Max.   :31860   Max.   :228.0   
##                   NA's   :102                       NA's   :286     
```

```r
train.trans$TEAM_PITCHING_HR = apply(train.trans['TEAM_PITCHING_HR'], 1, function(x) if(x==0){return(0)
train.trans$TEAM_PITCHING_BB = apply(train.trans['TEAM_PITCHING_BB'], 1, function(x) if(x==0){return(0)
train.trans$TEAM_FIELDING_E = apply(train.trans['TEAM_FIELDING_E'], 1, function(x) if(x==0){return(0)}

model2 = lm(TARGET_WINS ~ TEAM_BATTING_H + poly(TEAM_BATTING_BB,2) +
                TEAM_PITCHING_HR + poly(TEAM_PITCHING_BB,2) +
```

```r
                poly(TEAM_FIELDING_E,2), data=train.trans)
summary(model2)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + poly(TEAM_BATTING_BB,
##     2) + TEAM_PITCHING_HR + poly(TEAM_PITCHING_BB, 2) + poly(TEAM_FIELDING_E,
##     2), data = train.trans)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.787  -8.938   0.192   9.165  49.336
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.848e+00  3.764e+00   0.757    0.449
## TEAM_BATTING_H             4.945e-02  2.586e-03  19.122  < 2e-16 ***
## poly(TEAM_BATTING_BB, 2)1 -3.781e+02  6.711e+01  -5.634 1.98e-08 ***
## poly(TEAM_BATTING_BB, 2)2  2.057e+02  2.629e+01   7.826 7.65e-15 ***
## TEAM_PITCHING_HR          -3.246e+00  5.333e-01  -6.088 1.34e-09 ***
## poly(TEAM_PITCHING_BB, 2)1  2.847e+02  4.062e+01   7.009 3.16e-12 ***
## poly(TEAM_PITCHING_BB, 2)2  1.759e+02  2.884e+01   6.100 1.24e-09 ***
## poly(TEAM_FIELDING_E, 2)1 -5.730e+02  6.423e+01  -8.920  < 2e-16 ***
## poly(TEAM_FIELDING_E, 2)2 -1.043e+02  1.606e+01  -6.496 1.01e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.58 on 2267 degrees of freedom
## Multiple R-squared:  0.259,  Adjusted R-squared:  0.2563
## F-statistic: 99.02 on 8 and 2267 DF,  p-value: < 2.2e-16
```

All of our parameters are significant at $p < 0.001$ level, although the intercept term is no longer as significant as it was. Adjusted R^2 has improved, as has MSE:

```r
mean(resid(model1)^2)
```

```
## [1] 189.0616
```

Model 3

`model3` attempts to more precisely model `TEAM_FIELDING_E` by cubing it.

```r
model3 = lm(TARGET_WINS ~ TEAM_BATTING_H + poly(TEAM_BATTING_BB,2) +
                TEAM_PITCHING_HR + poly(TEAM_PITCHING_BB,2) +
                poly(TEAM_FIELDING_E,3), data=train.trans)
summary(model3)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TEAM_BATTING_H + poly(TEAM_BATTING_BB,
##     2) + TEAM_PITCHING_HR + poly(TEAM_PITCHING_BB, 2) + poly(TEAM_FIELDING_E,
##     3), data = train.trans)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -51.800  -8.957   0.291   9.101  49.593
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.836e+00  3.756e+00   0.755  0.45028
## TEAM_BATTING_H             5.083e-02  2.614e-03  19.443  < 2e-16 ***
## poly(TEAM_BATTING_BB, 2)1  -3.519e+02  6.744e+01  -5.218 1.97e-07 ***
## poly(TEAM_BATTING_BB, 2)2   1.826e+02  2.716e+01   6.722 2.26e-11 ***
## TEAM_PITCHING_HR          -3.827e+00  5.607e-01  -6.826 1.12e-11 ***
## poly(TEAM_PITCHING_BB, 2)1  2.781e+02  4.058e+01   6.855 9.19e-12 ***
## poly(TEAM_PITCHING_BB, 2)2  1.952e+02  2.937e+01   6.646 3.77e-11 ***
## poly(TEAM_FIELDING_E, 3)1  -5.658e+02  6.413e+01  -8.823  < 2e-16 ***
## poly(TEAM_FIELDING_E, 3)2  -1.051e+02  1.602e+01  -6.561 6.61e-11 ***
## poly(TEAM_FIELDING_E, 3)3  -5.528e+01  1.682e+01  -3.286  0.00103 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.55 on 2266 degrees of freedom
## Multiple R-squared:  0.2625, Adjusted R-squared:  0.2595
## F-statistic:  89.6 on 9 and 2266 DF,  p-value: < 2.2e-16
```

All the model parameters are highly significant, although the intercept is no longer signficicant. This is fine, however; arguably, a team with scores of 0 across all variables could expect to have zero wins on average.

Adjusted R^2 and MSE tick up slightly compared to previous models:

```
mean(resid(model3)^2)
```

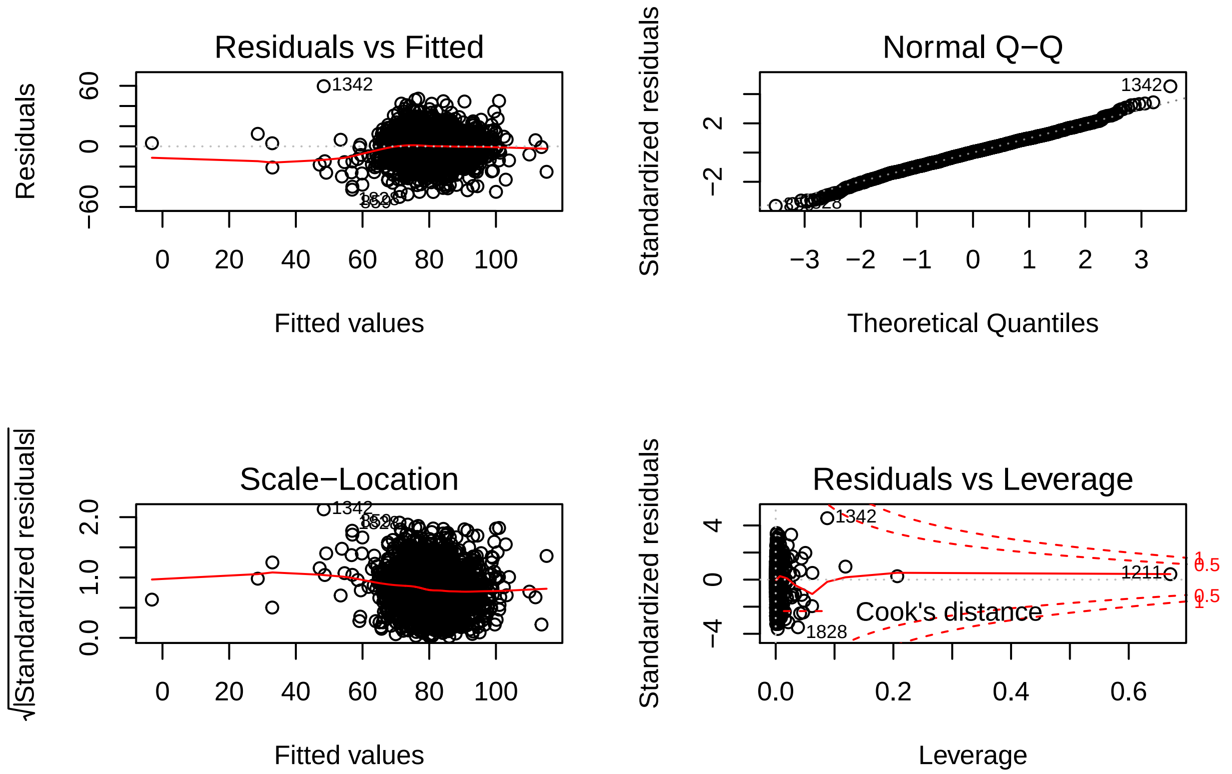
```
## [1] 182.9232
```

Evaluation

This section evaluates the above models, particularly focusing on residuals analysis.

Model 1

```
par(mfrow=c(2,2))
plot(model1)
```



Although the Q-Q and histogram plots show the residuals aren't too bad, there does appear to be some systematic bias. Lower fitted values from approximately 20-60 are systematically negative.

There's a strange point in the residuals plot, around $\hat{y} = 15$. Examining this point more closely, we don't see any obvious cause for it:

```
train[which.min(resid(model1)), ]
```

```
##     INDEX TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## 859   950          21           1402             149              53
##     TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## 859              13             304             295             134
##     TEAM_BASERUN_CS TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB
## 859              NA            1475               14              320
##     TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## 859              310             408               NA
```

This point has extremely high leverage:

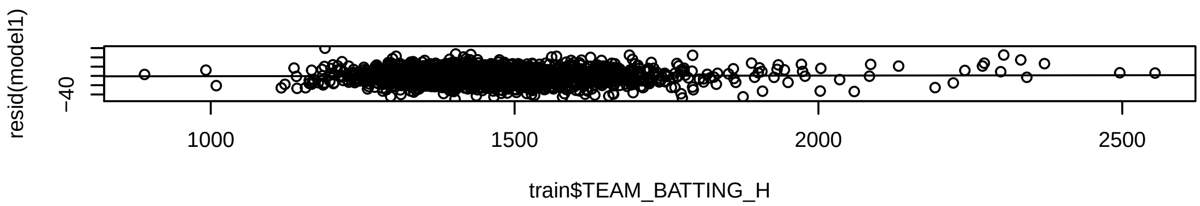
```
train[1211, ]
```

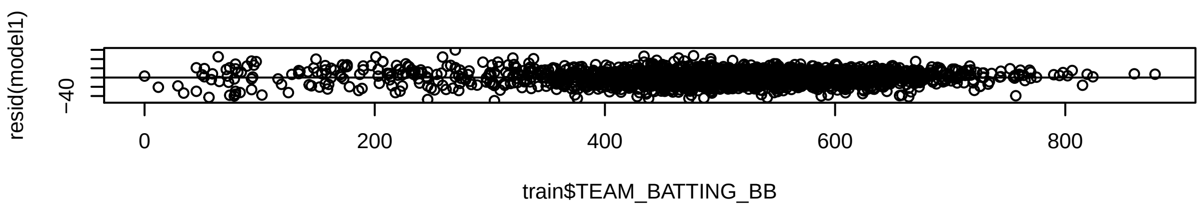
```
##      INDEX TARGET_WINS TEAM_BATTING_H TEAM_BATTING_2B TEAM_BATTING_3B
## 1211  1347           0            891             135               0
##      TEAM_BATTING_HR TEAM_BATTING_BB TEAM_BATTING_SO TEAM_BASERUN_SB
## 1211               0               0               0               0
##      TEAM_BASERUN_CS TEAM_PITCHING_H TEAM_PITCHING_HR TEAM_PITCHING_BB
## 1211               0           24057                0                0
##      TEAM_PITCHING_SO TEAM_FIELDING_E TEAM_FIELDING_DP
## 1211                0            1890               NA
```

It is the only team with 0 wins in the entire dataset, and naturally the model could not accurately estimate wins for this data point. Future modeling should consider excluding this unusual case.

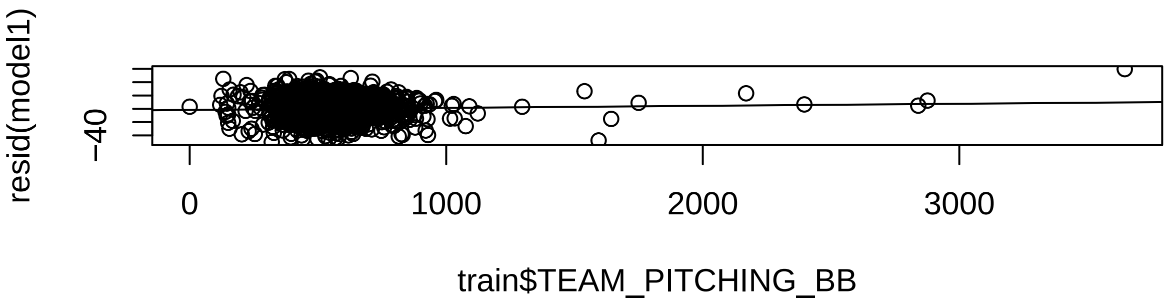
Look at each variable plotted against the model's residuals to see if we can understand the source of some of this bias:

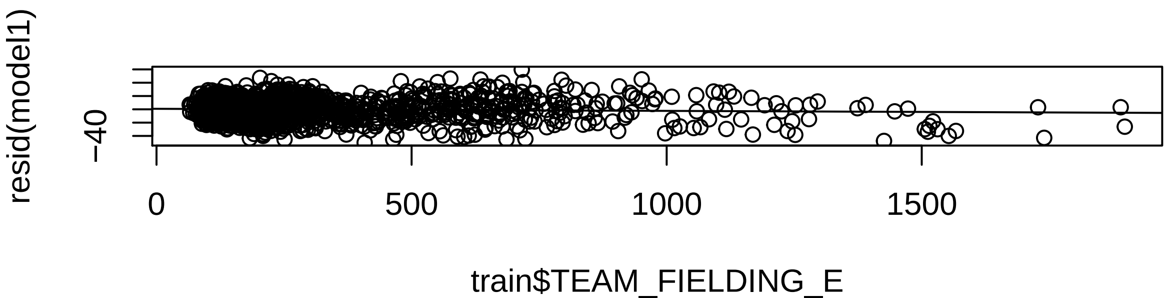
```
par(mfrow=c(3,1))
plot(train$TEAM_BATTING_H, resid(model1))
abline(lm( resid(model1) ~ train$TEAM_BATTING_H))

plot(train$TEAM_BATTING_BB, resid(model1))
abline(lm( resid(model1) ~ train$TEAM_BATTING_BB))

plot(train$TEAM_PITCHING_HR, resid(model1))
abline(lm( resid(model1) ~ train$TEAM_PITCHING_HR))
```






```
par(mfrow=c(2,1))
plot(train$TEAM_PITCHING_BB, resid(model1))
abline(lm( resid(model1) ~ train$TEAM_PITCHING_BB))

plot(train$TEAM_FIELDING_E, resid(model1))
abline(lm( resid(model1) ~ train$TEAM_FIELDING_E))
```



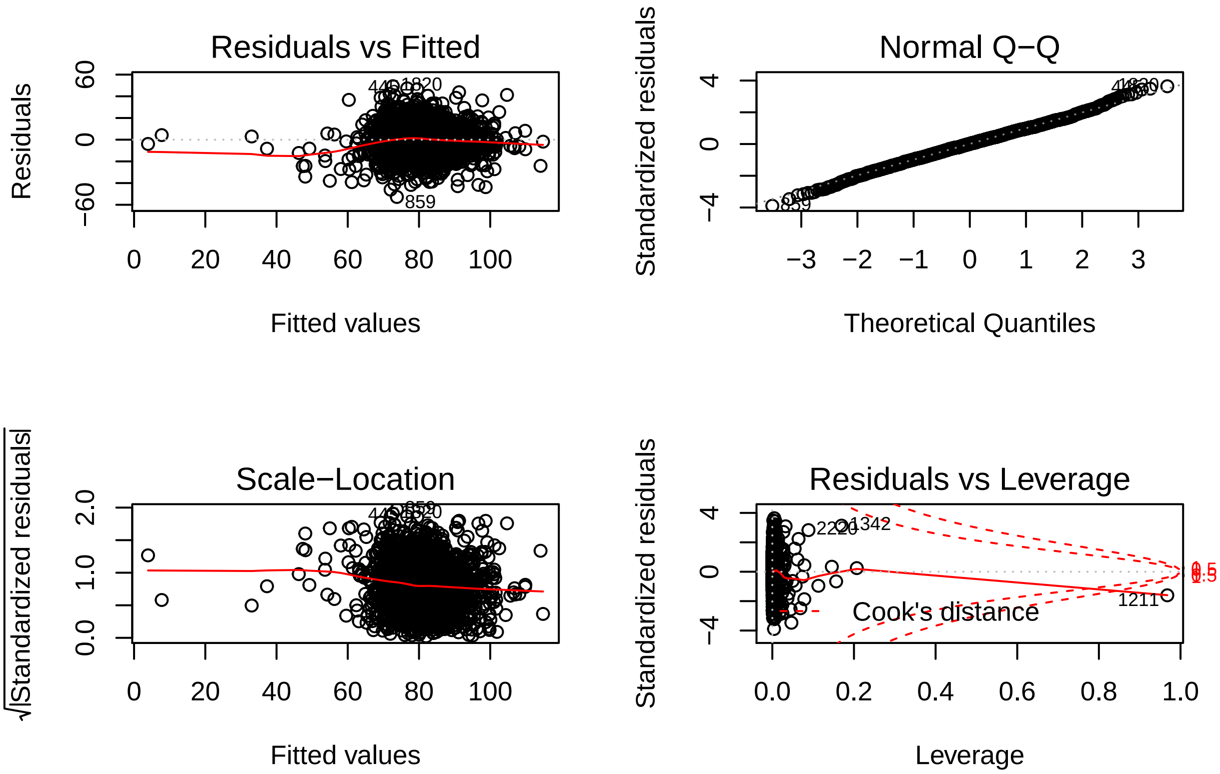


The trend lines all look fairly level, the least squares' assumption of constant variance is violated in most of these plots. Additionally, the mode consistently underestimates when `TEAM_FIELDING_E` is around or greater than 1500.

Model 2

From above, we know that this model performs better than `model1`. Examine its residuals:

```
par(mfrow=c(2,2))
plot(model2)
```



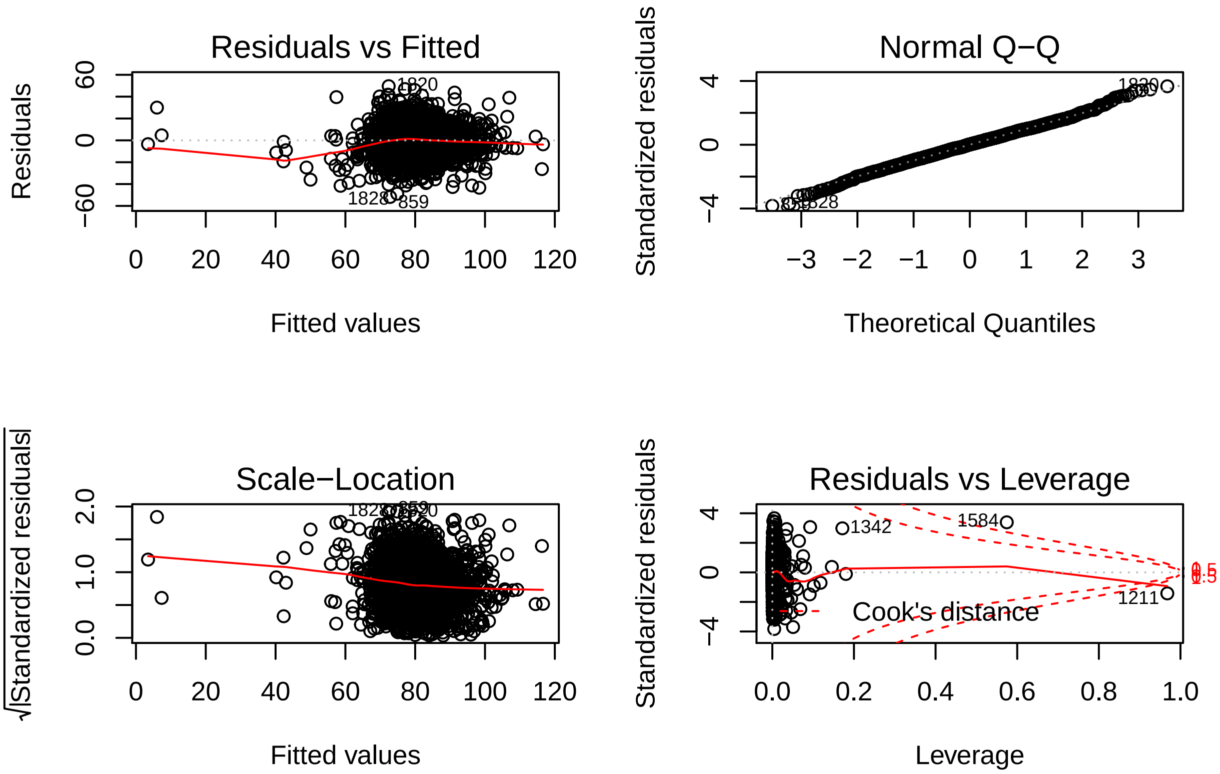
Thanks to the transformations, the Q-Q plot indicates these residuals are somewhat more normal than before. However, despite the transformations, data point 1211 is still an issue.

Systematic bias in the model remains, although it appears to be on the other side: as the predicted `TARGET_WINS` increases, the residuals are more strongly negative. I.e., this model is under-predicting better performing teams.

Model 3

Finally, model three:

```
par(mfrow=c(2,2))
plot(model3)
```



Although this model had the best adjusted R^2 and MSE, it appears even more biased than the previous model, but in the same way.

The plots of each variable against the model residual suggests the same problems plague `model3` as the others: Unusual outlying points and lack of constant variance.

Predictions

```
df.test = read.table("moneyball-evaluation-data.csv", sep=',', header = TRUE, stringsAsFactors = FALSE)
#2] SELECT Non Null columns
df.test = df.test%>%dplyr::select(-INDEX)
#cols.nomiss = names(train.df)[!names(train.df)%in%(cols.miss)]
#df.test = df.test%>%dplyr::select(cols.nomiss)
#3] Transform Data
test.trans = df.test
test.trans$TEAM_BATTING_H = test.trans$TEAM_BATTING_H  + test.trans$TEAM_BATTING_2B + test.trans$TEAM_BA
test.trans$TEAM_FIELDING_E = test.trans$TEAM_FIELDING_E + test.trans$TEAM_PITCHING_H
test.trans = test.trans%>%dplyr::select(-TEAM_BATTING_2B, -TEAM_BATTING_3B, -TEAM_BATTING_HR, -TEAM_PITC
test.trans$TEAM_PITCHING_HR = apply(test.trans['TEAM_PITCHING_HR'], 1, function(x) if(x==0){return(0)} e
test.trans$TEAM_PITCHING_BB = apply(test.trans['TEAM_PITCHING_BB'], 1, function(x) if(x==0){return(0)} e
test.trans$TEAM_FIELDING_E = apply(test.trans['TEAM_FIELDING_E'], 1, function(x) if(x==0){return(0)} els
Target_Wins = predict(model3, newdata = test.trans)
test.trans$TARGET_WINS = Target_Wins
df.test1 = read.table("moneyball-evaluation-data.csv", sep=',', header = TRUE, stringsAsFactors = FALSE)
df.test1$TARGET_WINS = Target_Wins
write.csv(df.test1, "Target_Prediction.csv", row.names = FALSE)
```