Moneyball

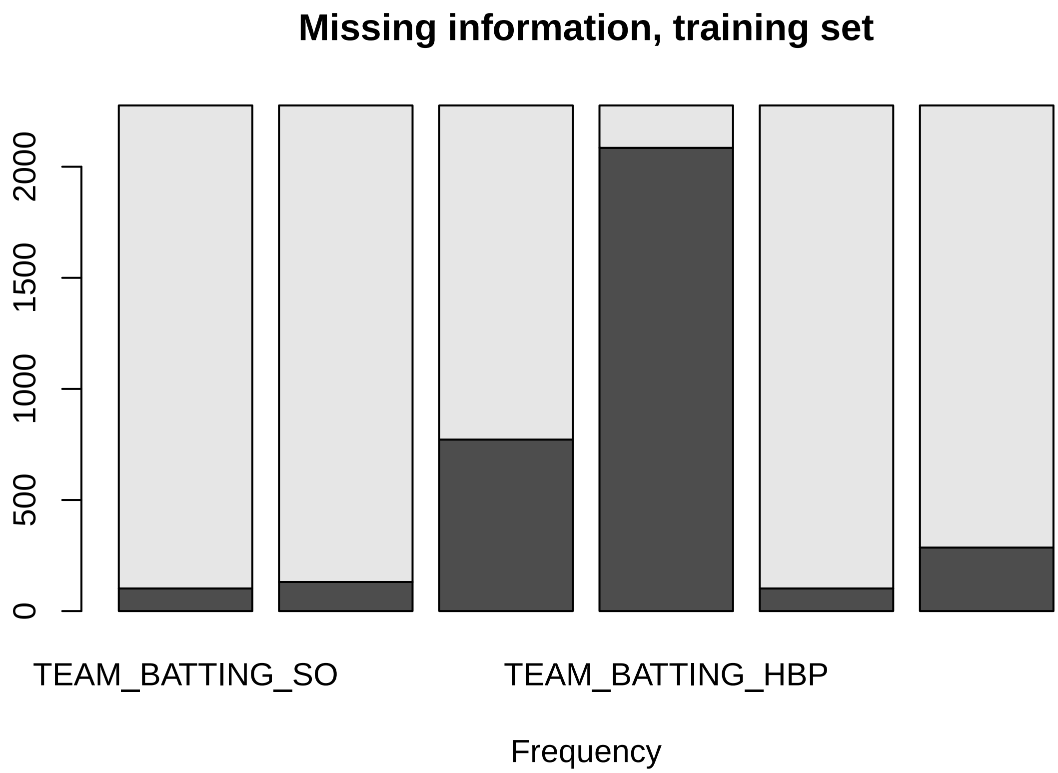
We are about to recreate the famous moneyball model.

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
INDEX	Identification Variable (do not use)	None
TARGET_WINS	Number of wins	
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
TEAM_BASERUN_SB	Stolen bases	Positive Impact on Wins
TEAM_BASERUN_CS	Caught stealing	Negative Impact on Wins
TEAM_FIELDING_E	Errors	Negative Impact on Wins
TEAM_FIELDING_DP	Double Plays	Positive Impact on Wins
TEAM_PITCHING_BB	Walks allowed	Negative Impact on Wins
TEAM_PITCHING_H	Hits allowed	Negative Impact on Wins
TEAM_PITCHING_HR	Homeruns allowed	Negative Impact on Wins
TEAM_PITCHING_SO	Strikeouts by pitchers	Positive Impact on Wins

DATA EXPLORATION and PREPARATION

Missing information

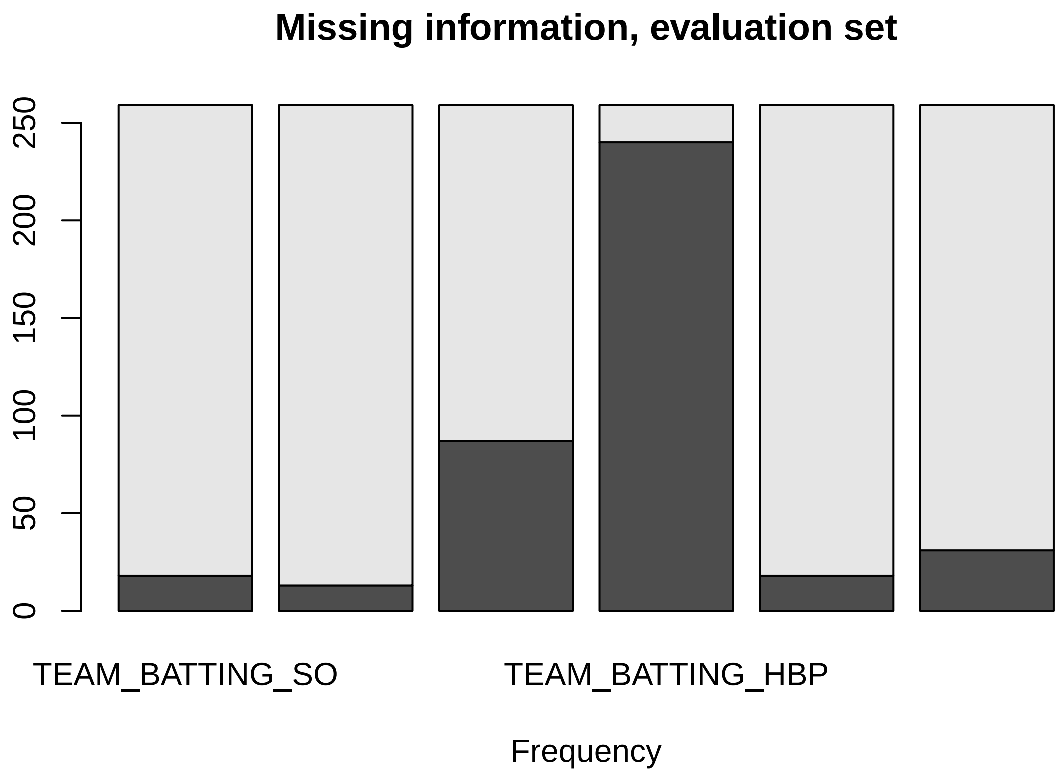
```
barplot(tr.binmat[,imp+1], main = "Missing information, training set",
        xlab = "Frequency")
```




```
colSums(is.na(tr))
```

```
##             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
```

```
barplot(ev.binmat[,imp], main = "Missing information, evaluation set",
        xlab = "Frequency")
```




```
colSums(is.na(ev))
```

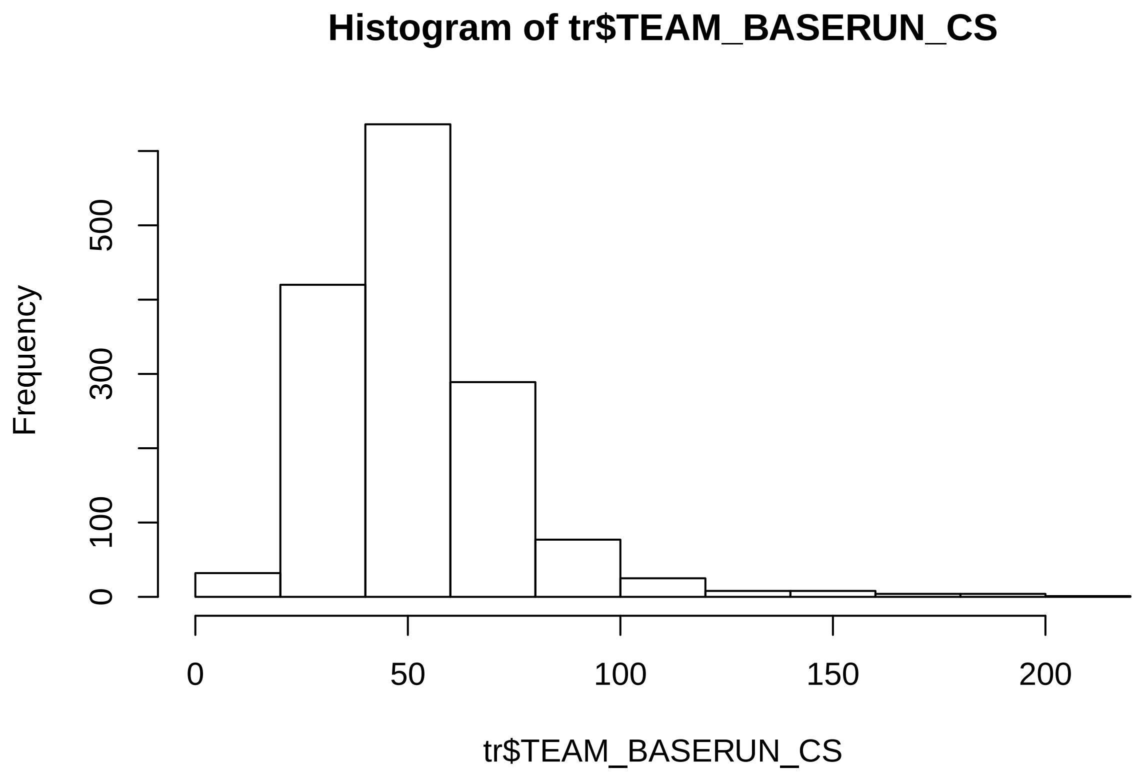
```
##            INDEX   TEAM_BATTING_H  TEAM_BATTING_2B  TEAM_BATTING_3B
##                0                0                0                0
##  TEAM_BATTING_HR  TEAM_BATTING_BB  TEAM_BATTING_SO  TEAM_BASERUN_SB
##                0                0               18               13
##  TEAM_BASERUN_CS TEAM_BATTING_HBP  TEAM_PITCHING_H TEAM_PITCHING_HR
##               87              240                0                0
## TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E TEAM_FIELDING_DP
##                0               18                0               31
```

TEAM_BATTING_HBP is missing almost every value, so we may have to destroy this variable. For now we will add a flag and impute the values... After the other things are filled. This variable represents batters hit by a pitch, and none of the values are zero, so these are truly missing.

The main question for this dataset, is whether or not imputing this huge missing column will help or hurt the models.

Apart from that, TEAM_BASERUN_CS is missing about a third of its values, and it seems like a good idea to look at the distribution of known values and add a flag for it, too. This is players caught stealing bases. This information also makes sense. This appears to be a skewed normal distribution ranging from zero off near 200. A good candidate for imputation.

```
hist(tr$TEAM_BASERUN_CS)
```



The other variables have relatively low amounts of missing data. These two exhibit very cooperative distributions, and we will impute around them.

The distributions in the training and evaluation sets are nearly identical, so we cn make a generic function for this.

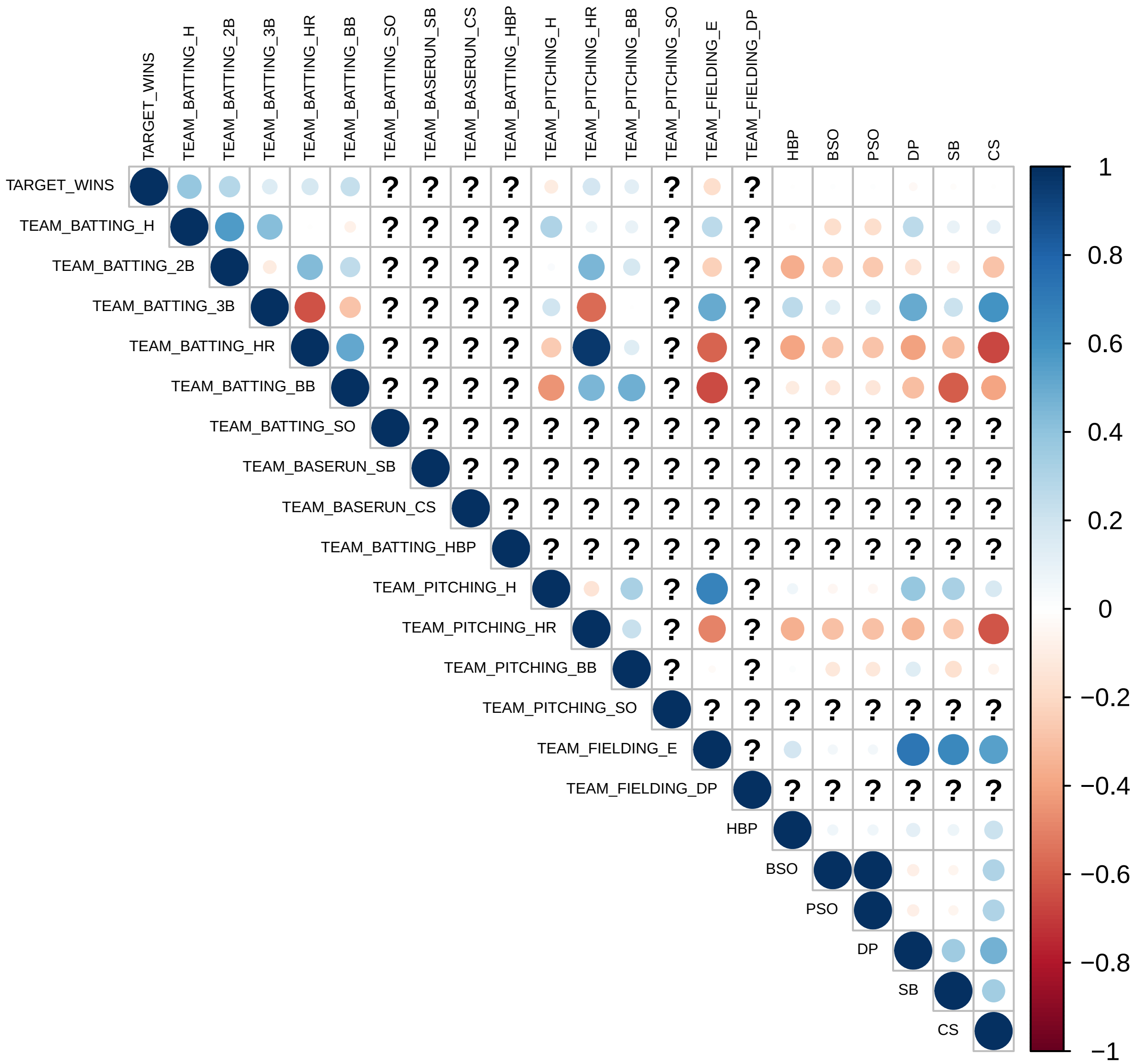
Munging

Our data munging process mostly just deals with adding flags for missing information and imputing these points based on normal distributions. The information presented is 100% numeric, all integers, so we don't have to transform it much.

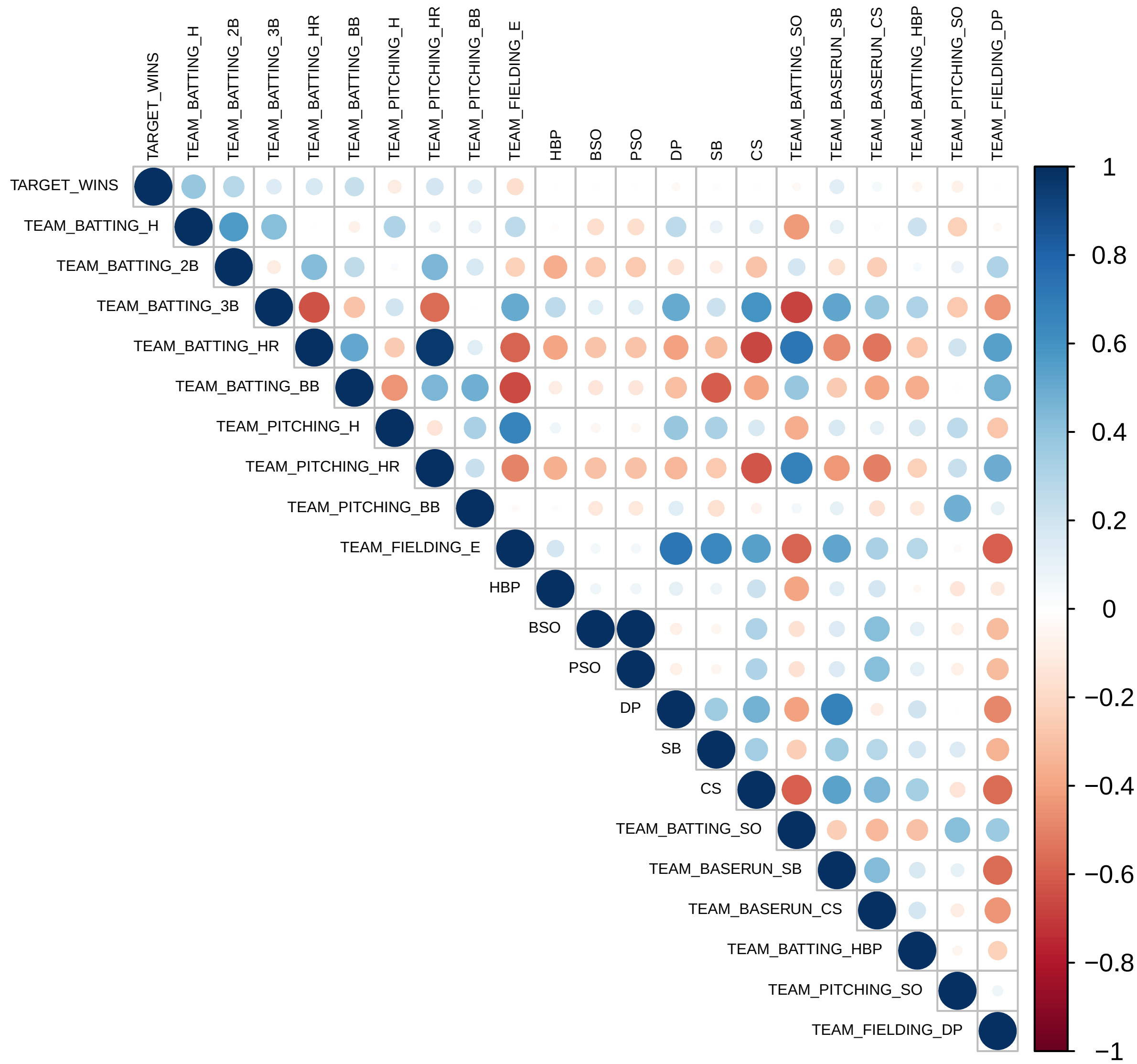
We will start by imputing the data in the training set alone because it contains the target variable. Then we impute the data from the evaluation set using the filled training set.

Correlations

```
corrplot(cor(tr),
         type = 'upper',
         tl.col = 'black',
         tl.cex = 0.5)
```

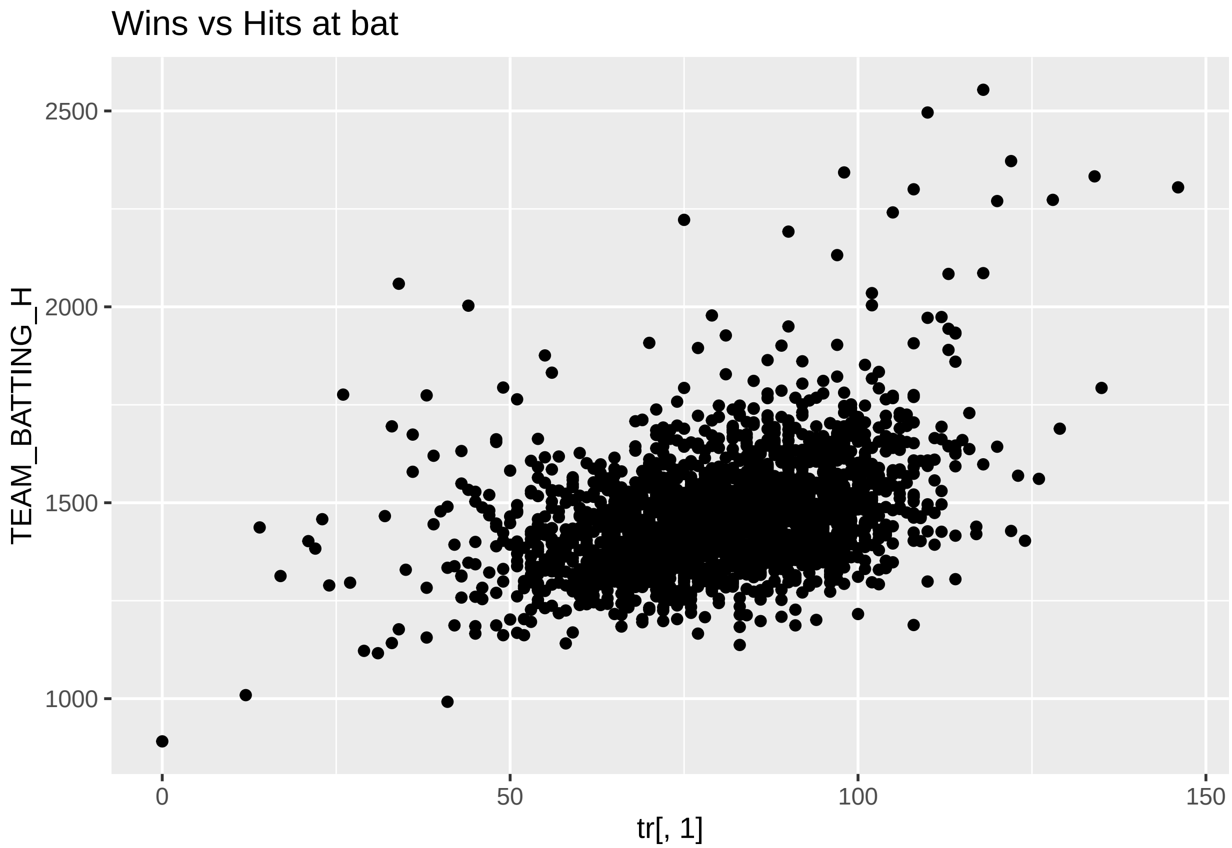



```r
corrplot(cor(tr.imp),
         type = 'upper',
         tl.col = 'black',
         tl.cex = 0.5)
```



This is a little bit scary. Not a single one of the dependent variables is strongly correlated to the target. After imputation, there are some interesting pairs, TEAM_PITCHING_HR and TEAM_BATTING_HR and the flags for TEAM_PITCHING_SO and TEAM_BATTING_SO, which could be an artifact of the imputation. The correlations look pretty fake and full of bad information.

```
ggplot(tr.imp, aes(x = tr[, 1], y = TEAM_BATTING_H)) +
  geom_point() + ggtitle(label = "Wins vs Hits at bat")
```

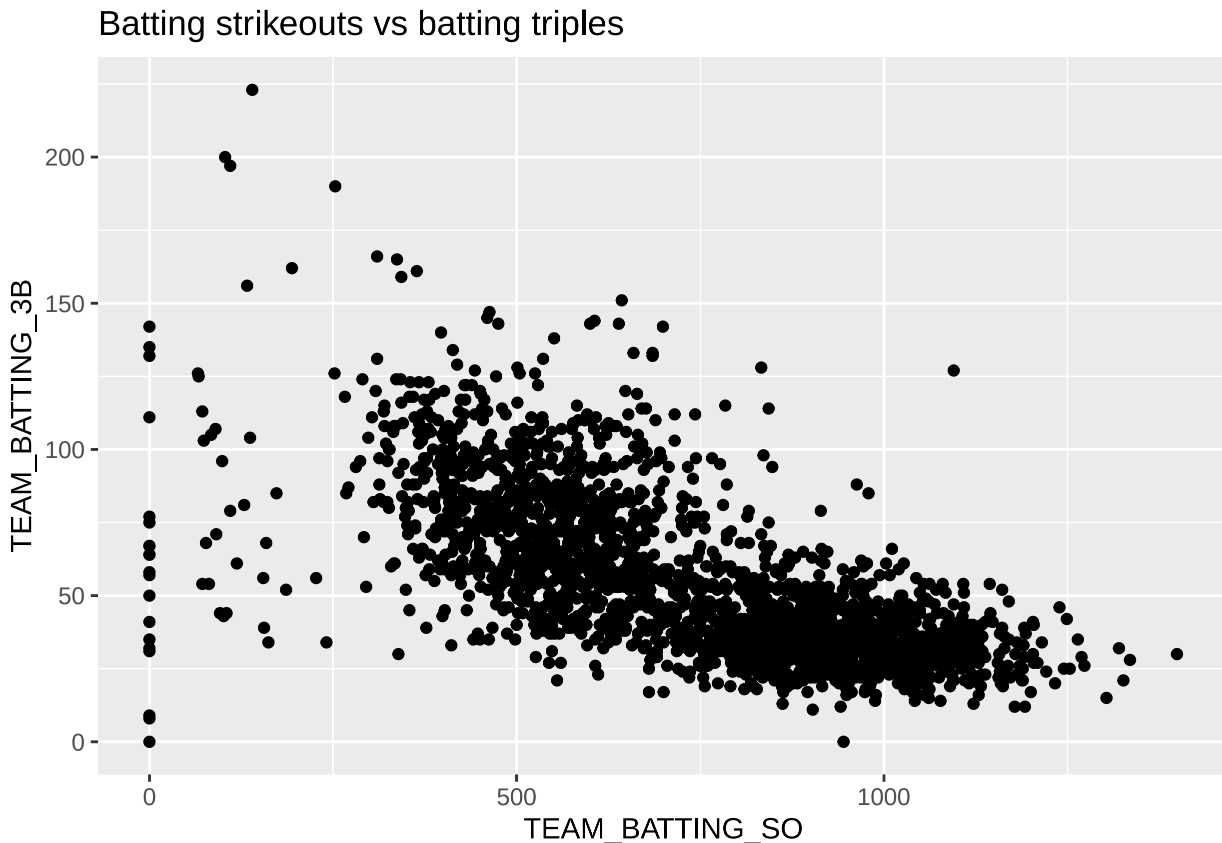


Batting hits is the only datapoint that correlates noticeably with wins. There is a lot more to the story.

Some key distributions

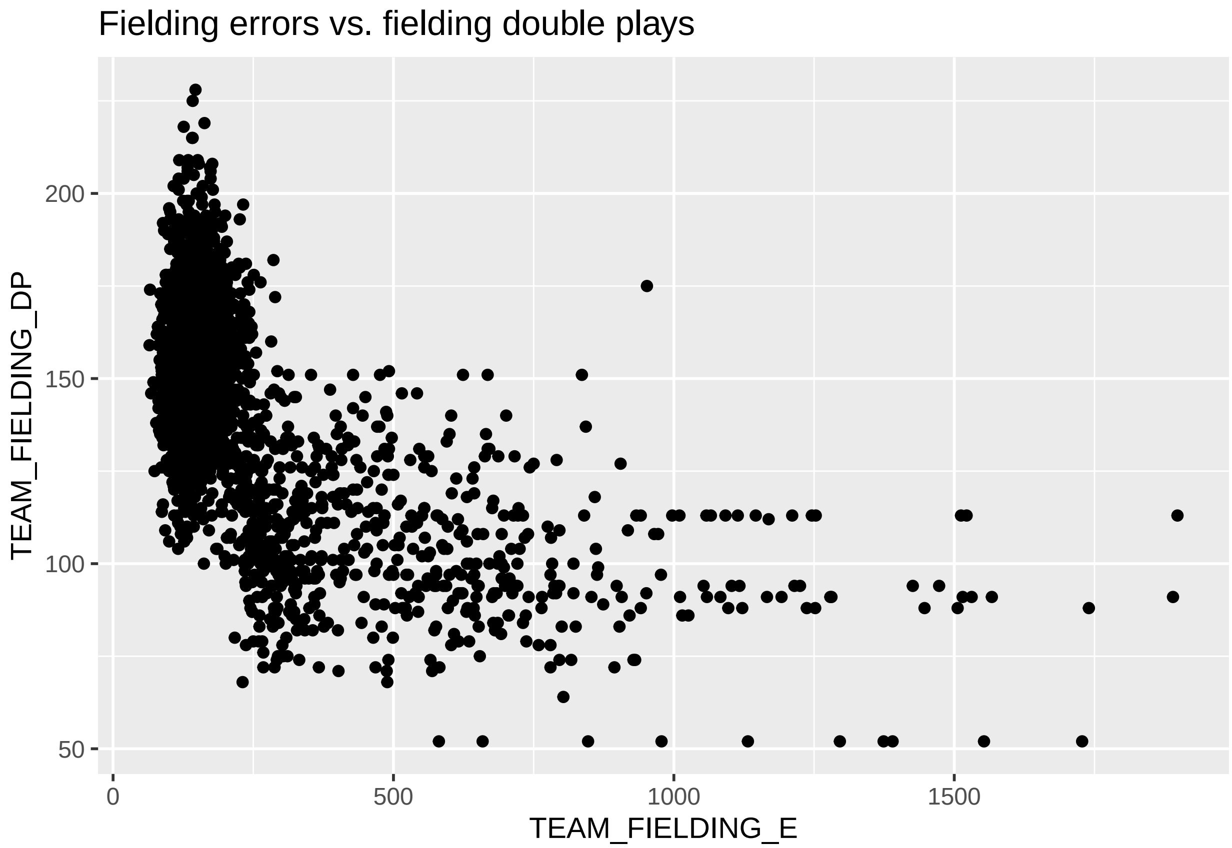
Of the most correlated and anticorrelated, here is a selection:

```
ggplot(tr.imp, aes(x = TEAM_BATTING_SO, y = TEAM_BATTING_3B)) +
  geom_point() + ggtitle("Batting strikeouts vs batting triples")
```



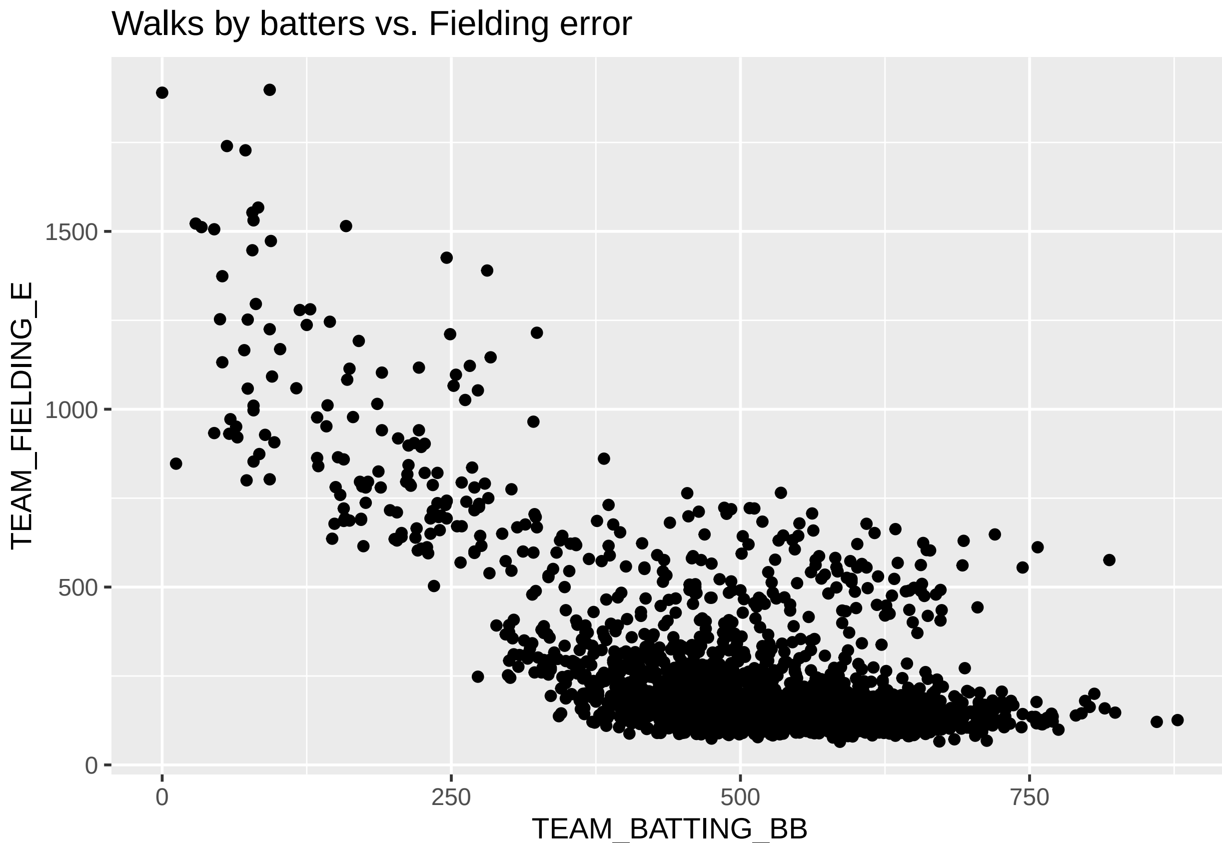
It makes a lot of sense that strikeouts and triples would be anticorrelated. This is good information. There is changing variance in this relationship, so it may be a good feature.

```
ggplot(tr.imp, aes(x = TEAM_FIELDING_E, y = TEAM_FIELDING_DP)) +
  geom_point() + ggtitle("Fielding errors vs. fielding double plays")
```



This may be a very strong feature because the relationship between errors and double plays tells a lot about a team's ability to respond to wild hits.

```
ggplot(tr.imp, aes(x = TEAM_BATTING_BB, y = TEAM_FIELDING_E)) +
  geom_point() + ggtitle("Walks by batters vs. Fielding error")
```



It makes a lot of sense that teams that walk a lot at bat are also less likely to make fielding errors. This shows a defensive mentality in some teams, and an offensive one in others.

BUILD MODELS

We need to split the data for training and test sets internally, since we don't know the target values in the evaluation set.

The all-in model

```
summary(m1 <- lm(TARGET_WINS ~ . -BSO,
                 tr.tr))
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - BSO, data = tr.tr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.846  -7.907   0.122   7.894  47.618
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      36.5626531  6.2037256   5.894 4.43e-09 ***
## TEAM_BATTING_H    0.0439312  0.0036300  12.102  < 2e-16 ***
## TEAM_BATTING_2B  -0.0363220  0.0095546  -3.802 0.000148 ***
## TEAM_BATTING_3B   0.0602248  0.0165415   3.641 0.000279 ***
## TEAM_BATTING_HR   0.0925965  0.0278921   3.320 0.000917 ***
## TEAM_BATTING_BB   0.0283302  0.0063653   4.451 9.03e-06 ***
## TEAM_PITCHING_H   0.0022912  0.0004139   5.535 3.52e-08 ***
## TEAM_PITCHING_HR  0.0064541  0.0245412   0.263 0.792584
## TEAM_PITCHING_BB -0.0043780  0.0046049  -0.951 0.341862
## TEAM_FIELDING_E  -0.0612890  0.0038399 -15.961  < 2e-16 ***
## HBP               1.6541100  1.2007350   1.378 0.168488
## PSO               5.4467803  1.4903467   3.655 0.000264 ***
## DP                6.4219765  2.8904709   2.222 0.026411 *
## SB               25.5356871  1.8946146  13.478  < 2e-16 ***
## CS               -0.5122250  0.9095147  -0.563 0.573373
## TEAM_BATTING_SO  -0.0169384  0.0029988  -5.648 1.85e-08 ***
## TEAM_BASERUN_SB   0.0315303  0.0071293   4.423 1.03e-05 ***
## TEAM_BASERUN_CS   0.0472224  0.0142089   3.323 0.000905 ***
## TEAM_BATTING_HBP -0.0928193  0.0271539  -3.418 0.000643 ***
## TEAM_PITCHING_SO  0.0001206  0.0015467   0.078 0.937876
## TEAM_FIELDING_DP -0.1348988  0.0135187  -9.979  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.75 on 1979 degrees of freedom
## Multiple R-squared:  0.4346, Adjusted R-squared:  0.4288
## F-statistic: 76.05 on 20 and 1979 DF,  p-value: < 2.2e-16
```

Trimmed feature model

```r
summary(m2 <- lm(TARGET_WINS ~ . -DP -CS -TEAM_PITCHING_SO
                 -TEAM_PITCHING_HR -TEAM_PITCHING_BB -BSO
                 -TEAM_BASERUN_CS,
                 tr.tr))
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ . - DP - CS - TEAM_PITCHING_SO - TEAM_PITCHING_HR -
##     TEAM_PITCHING_BB - BSO - TEAM_BASERUN_CS, data = tr.tr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -43.936  -8.073   0.307   7.977  47.167
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      39.2837912  5.8705312   6.692 2.86e-11 ***
## TEAM_BATTING_H    0.0452726  0.0035624  12.708  < 2e-16 ***
## TEAM_BATTING_2B  -0.0408293  0.0093725  -4.356 1.39e-05 ***
## TEAM_BATTING_3B   0.0671148  0.0161877   4.146 3.53e-05 ***
## TEAM_BATTING_HR   0.0952072  0.0095460   9.973  < 2e-16 ***
## TEAM_BATTING_BB   0.0218761  0.0035031   6.245 5.18e-10 ***
## TEAM_PITCHING_H   0.0019805  0.0003127   6.333 2.96e-10 ***
## TEAM_FIELDING_E  -0.0574001  0.0029547 -19.426  < 2e-16 ***
## HBP               1.5039621  1.1772743   1.277 0.201577
## PSO               6.5875635  1.4003633   4.704 2.72e-06 ***
## SB               25.1630581  1.7376595  14.481  < 2e-16 ***
## TEAM_BATTING_SO  -0.0169548  0.0023885  -7.098 1.75e-12 ***
## TEAM_BASERUN_SB   0.0447468  0.0041338  10.825  < 2e-16 ***
## TEAM_BATTING_HBP -0.0949992  0.0266385  -3.566 0.000371 ***
## TEAM_FIELDING_DP -0.1400173  0.0132387 -10.576  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.78 on 1985 degrees of freedom
## Multiple R-squared:  0.4303, Adjusted R-squared:  0.4263
## F-statistic: 107.1 on 14 and 1985 DF,  p-value: < 2.2e-16
```

Trimmed model with generated features

We add the three new relationships from the data exploration section as features to the trimmed model.

```r
summary(m3 <- lm(TARGET_WINS ~ . -DP -CS -TEAM_PITCHING_SO
                 -TEAM_PITCHING_HR -TEAM_PITCHING_BB -BSO -TEAM_BASERUN_CS
                 +TEAM_BATTING_BB*TEAM_FIELDING_E,
                 tr.tr))
```

```
##
## Call:
```

```
## lm(formula = TARGET_WINS ~ . - DP - CS - TEAM_PITCHING_SO - TEAM_PITCHING_HR -
##     TEAM_PITCHING_BB - BSO - TEAM_BASERUN_CS + TEAM_BATTING_BB *
##     TEAM_FIELDING_E, data = tr.tr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.258  -7.771   0.174   7.648  47.432
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      2.984e+01  5.911e+00   5.048 4.87e-07 ***
## TEAM_BATTING_H                   4.458e-02  3.511e-03  12.695  < 2e-16 ***
## TEAM_BATTING_2B                 -4.196e-02  9.236e-03  -4.543 5.89e-06 ***
## TEAM_BATTING_3B                  1.021e-01  1.657e-02   6.160 8.80e-10 ***
## TEAM_BATTING_HR                  8.680e-02  9.468e-03   9.168  < 2e-16 ***
## TEAM_BATTING_BB                  4.481e-02  4.541e-03   9.869  < 2e-16 ***
## TEAM_PITCHING_H                  1.042e-03  3.309e-04   3.149  0.00166 **
## TEAM_FIELDING_E                 -3.617e-02  3.991e-03  -9.062  < 2e-16 ***
## HBP                              1.875e+00  1.161e+00   1.615  0.10646
## PSO                              5.615e+00  1.386e+00   4.053 5.26e-05 ***
## SB                               2.100e+01  1.794e+00  11.704  < 2e-16 ***
## TEAM_BATTING_SO                 -1.660e-02  2.354e-03  -7.053 2.41e-12 ***
## TEAM_BASERUN_SB                  5.949e-02  4.493e-03  13.241  < 2e-16 ***
## TEAM_BATTING_HBP                -6.891e-02  2.646e-02  -2.604  0.00928 **
## TEAM_FIELDING_DP                -1.449e-01  1.306e-02 -11.096  < 2e-16 ***
## TEAM_BATTING_BB:TEAM_FIELDING_E -8.121e-05  1.044e-05  -7.775 1.20e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.61 on 1984 degrees of freedom
## Multiple R-squared:  0.4472, Adjusted R-squared:  0.443
## F-statistic:   107 on 15 and 1984 DF,  p-value: < 2.2e-16
```

SELECT MODELS

To select our models, we should make predictions on the internal evaluation set, where we know the target values.

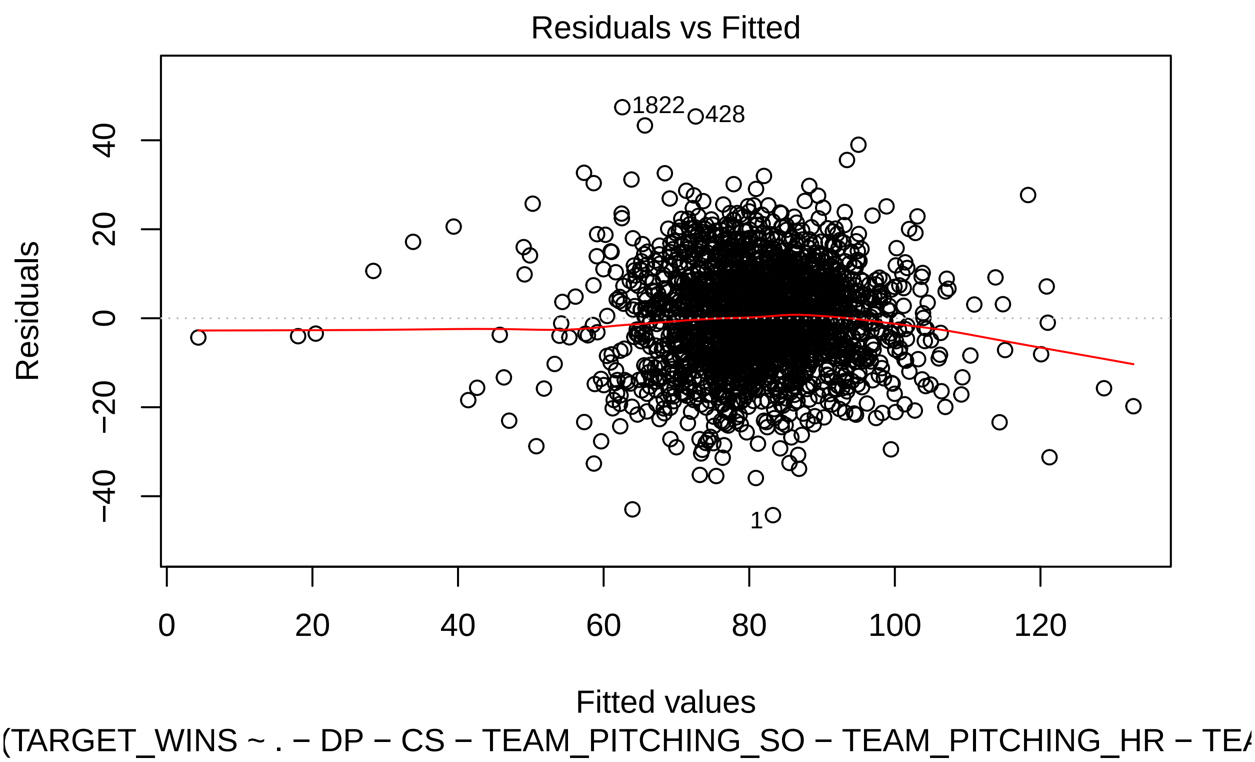
Mean squared error

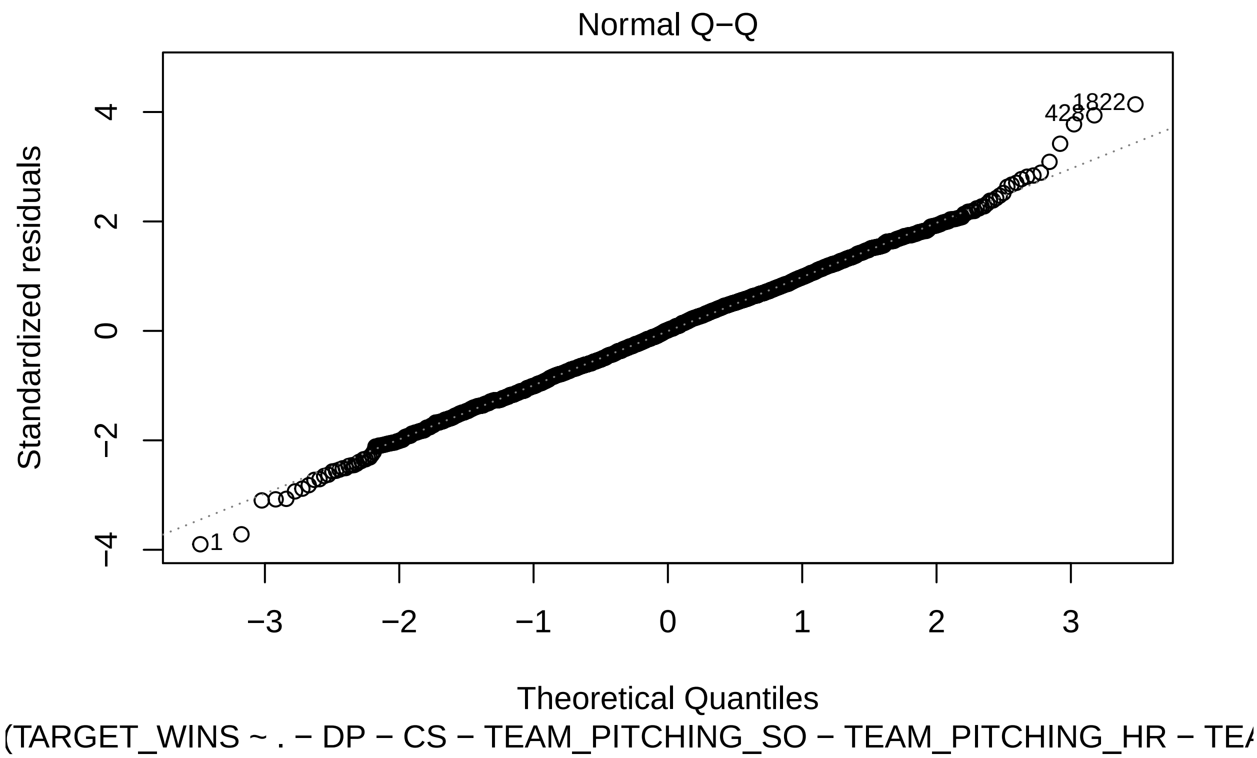
The mean of the square of the difference between real and predicted values.

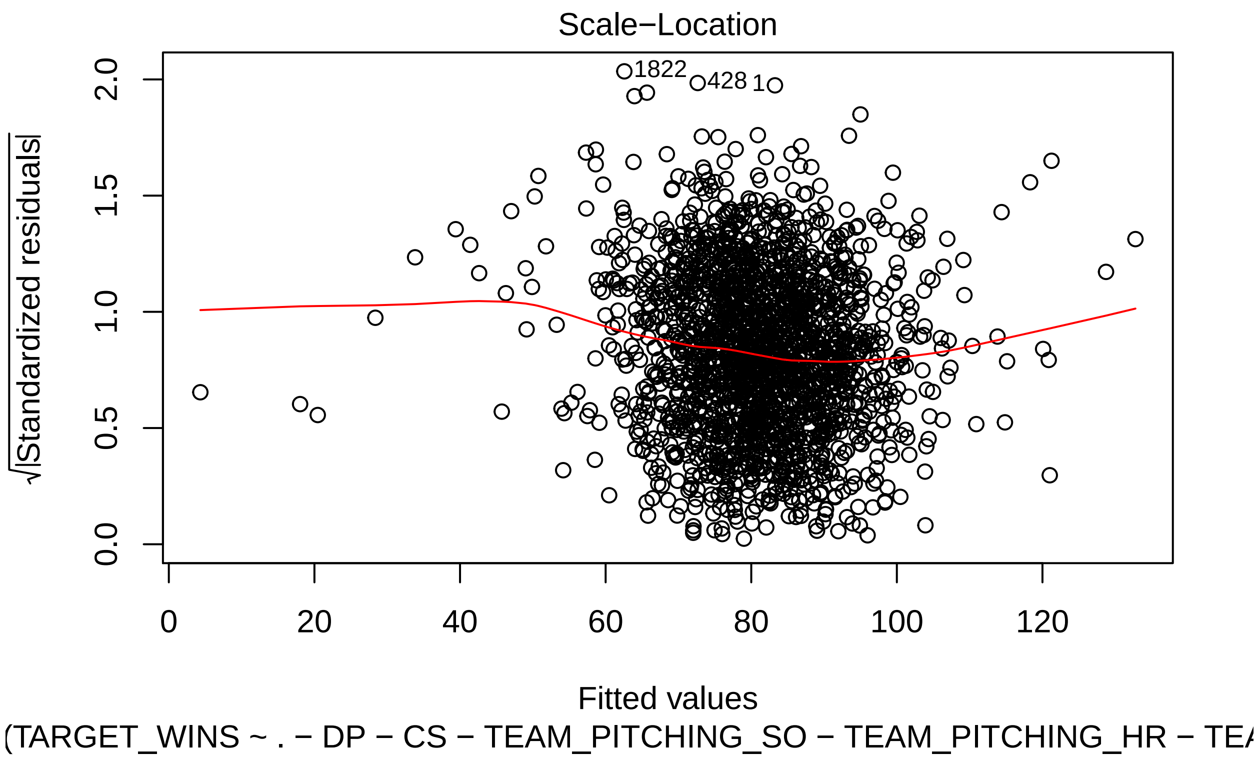
It looks like the trimmed model with added features based on correlations has a slightly lower mean-squared error than the all-in and trimmed models.

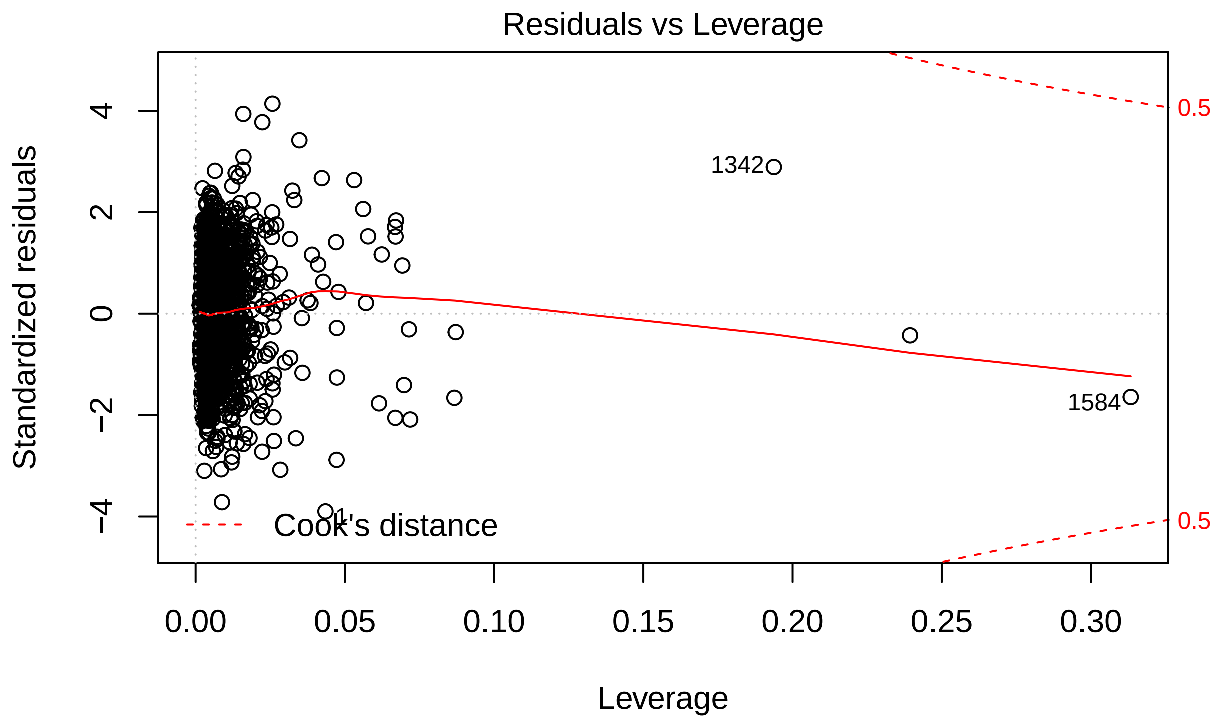
Residuals

```
plot(m3)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
(TARGET_WINS ~ . − DP − CS − TEAM_PITCHING_SO − TEAM_PITCHING_HR − TEA
1822
428
1
4
2
0
−2
−4
−3
−2
−1
0
1
2
3


Scale−Location
1822
428
1
2.0
1.5
1.0
0.5
0.0
√|Standardized residuals|
0
20
40
60
80
100
120
Fitted values
(TARGET_WINS ~ . − DP − CS − TEAM_PITCHING_SO − TEAM_PITCHING_HR − TEA



(TARGET_WINS ~ . − DP − CS − TEAM_PITCHING_SO − TEAM_PITCHING_HR − TE/

EXPORT PREDICTIONS

```
p <- predict.glm(m3, ev.imp)
write.csv(p, "predictions.csv")
```

CODE APPENDIX

```
library(tidyverse)
library(mice)
library(pROC)
library(corrplot)

set.seed(1337)

tr <- read.csv("moneyball-training-data.csv")
ev <- read.csv("moneyball-evaluation-data.csv")

imp <- c(7, 8, 9, 10, 14, 16)

as.binMat <- function(df) {
  m <- c()
  for (i in colnames(df)) {
    x <- sum(is.na(df[,i]))
    m <- append(m, x)
    m <- append(m, nrow(df) - x)
  }

  a <- matrix(m, nrow = 2)
  rownames(a) <- c("Missing", "Present")
  colnames(a) <- colnames(df)

  return(a)
}

tr.binmat <- as.binMat(tr)
ev.binmat <- as.binMat(ev)

barplot(tr.binmat[,imp+1], main = "Missing information, training set",
        xlab = "Frequency")
colSums(is.na(tr))
barplot(ev.binmat[,imp], main = "Missing information, evaluation set",
        xlab = "Frequency")
colSums(is.na(ev))

hist(tr$TEAM_BATTING_HBP)
hist(tr$TEAM_BASERUN_CS)

tr <- tr[,-1] %>%
  mutate(HBP = is.na(TEAM_BATTING_HBP)) %>%
  mutate(BSO = is.na(TEAM_BATTING_SO)) %>%
  mutate(PSO = is.na(TEAM_PITCHING_SO)) %>%
```

```
  mutate(DP = is.na(TEAM_FIELDING_DP)) %>%
  mutate(SB = is.na(TEAM_BASERUN_SB)) %>%
  mutate(CS = is.na(TEAM_BASERUN_CS)) %>%
  as.matrix()

x <- mice(tr, maxit = 20)
y <- complete(x, 1)
tr.imp <- cbind(tr[,-imp], y[,imp])

ev <- ev[,-1] %>%
  mutate(HBP = is.na(TEAM_BATTING_HBP)) %>%
  mutate(BSO = is.na(TEAM_BATTING_SO)) %>%
  mutate(PSO = is.na(TEAM_PITCHING_SO)) %>%
  mutate(DP = is.na(TEAM_FIELDING_DP)) %>%
  mutate(SB = is.na(TEAM_BASERUN_SB)) %>%
  mutate(CS = is.na(TEAM_BASERUN_CS)) %>%
  as.matrix()


f <- mice(ev, maxit = 20, method = "norm.predict")
g <- complete(f, 1)
ev.imp <- cbind(ev[,-imp], g[,imp])

colSums(is.na(ev.imp))

ev.imp$TEAM_BATTING_SO[is.na(ev.imp$TEAM_BATTING_SO)] <- mean(tr.imp$TEAM_BATTING_SO)
ev.imp$TEAM_PITCHING_SO[is.na(ev.imp$TEAM_PITCHING_SO)] <- mean(tr.imp$TEAM_PITCHING_SO)
ev.imp$TEAM_FIELDING_DP[is.na(ev.imp$TEAM_FIELDING_DP)] <- mean(tr.imp$TEAM_FIELDING_DP)

anyNA(ev.imp)

corrplot(cor(tr),
         type = 'upper',
         tl.col = 'black',
         tl.cex = 0.5)

corrplot(cor(tr.imp),
         type = 'upper',
         tl.col = 'black',
         tl.cex = 0.5)

ggplot(tr.imp, aes(x = tr[, 1], y = TEAM_BATTING_H)) +
  geom_point() + ggtitle(label = "Wins vs Hits at bat")

  ggplot(tr.imp, aes(x = TEAM_BATTING_SO, y = TEAM_BATTING_3B)) +
  geom_point() + ggtitle("Batting strikeouts vs batting triples")

  ggplot(tr.imp, aes(x = TEAM_FIELDING_E, y = TEAM_FIELDING_DP)) +
  geom_point() + ggtitle("Fielding errors vs. fielding double plays")

  ggplot(tr.imp, aes(x = TEAM_BATTING_BB, y = TEAM_FIELDING_E)) +
  geom_point() + ggtitle("Walks by batters vs. Fielding error")

  tr.tr <- tr.imp[1:2000,]
```

```
tr.ev <- tr.imp[2001:2535,]

summary(m1 <- lm(TARGET_WINS ~ . -BSO,
                 tr.tr))

                 summary(m2 <- lm(TARGET_WINS ~ . -DP -CS -TEAM_PITCHING_SO
                 -TEAM_PITCHING_HR -TEAM_PITCHING_BB -BSO
                 -TEAM_BASERUN_CS,
                 tr.tr))

summary(m2 <- lm(TARGET_WINS ~ . -DP -CS -TEAM_PITCHING_SO
                 -TEAM_PITCHING_HR -TEAM_PITCHING_BB -BSO
                 -TEAM_BASERUN_CS,
                 tr.tr))

summary(m3 <- lm(TARGET_WINS ~ . -DP -CS -TEAM_PITCHING_SO
                 -TEAM_PITCHING_HR -TEAM_PITCHING_BB -BSO -TEAM_BASERUN_CS
                 +TEAM_BATTING_BB*TEAM_FIELDING_E,
                 tr.tr))

p1 <- predict(m1, tr.ev)
p2 <- predict(m2, tr.ev)
p3 <- predict(m3, tr.ev)

(mse1 <- mean(m1$residuals^2))
(mse2 <- mean(m2$residuals^2))
(mse3 <- mean(m3$residuals^2))

plot(m3)

p <- predict.glm(m3, ev.imp)
write.csv(p, "predictions.csv")
```