Predicting Total Wins Per Season in Major Leauge Baseball from Game Statistics

Daniel Brooks (daniel.brooks@spsmail.cuny.edu), Daniel Fanelli (daniel.fanelli@spsmail.cuny.edu), Christopher Fenton (christopher.fenton@spsmail.cuny.edu), James Hamski (james.hamski@spsmail.cuny.edu), Youqing Xiang (youqing.xiang@spsmail.cuny.edu)

6/19/2016

1 Introduction

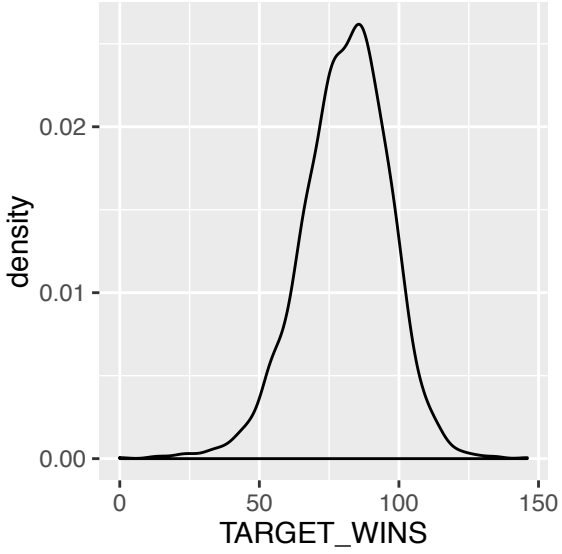
Baseballl is a sport that follows a sequence of pitches, at-bats, and innings where play is contained between discrete pitches. Unlike the more continuous play of soccer or basketball, this makes baseball conducive to gathering extensive data on individual and team performance.

In this report we attempt to model wins per season for Major League Baseball (MLB) teams (response variable). Our dataset includes 15 potential predictor variables, adjusted to reflect a standardized 162 game season, using MLB records from 1871 to 2006.

2 Data Exploration

2.1 Response Variable

Team Wins (TARGET_WINS) appears to be normally distributed with a slight left skew and a mean of 80.79, which is half of the total 162 game season.

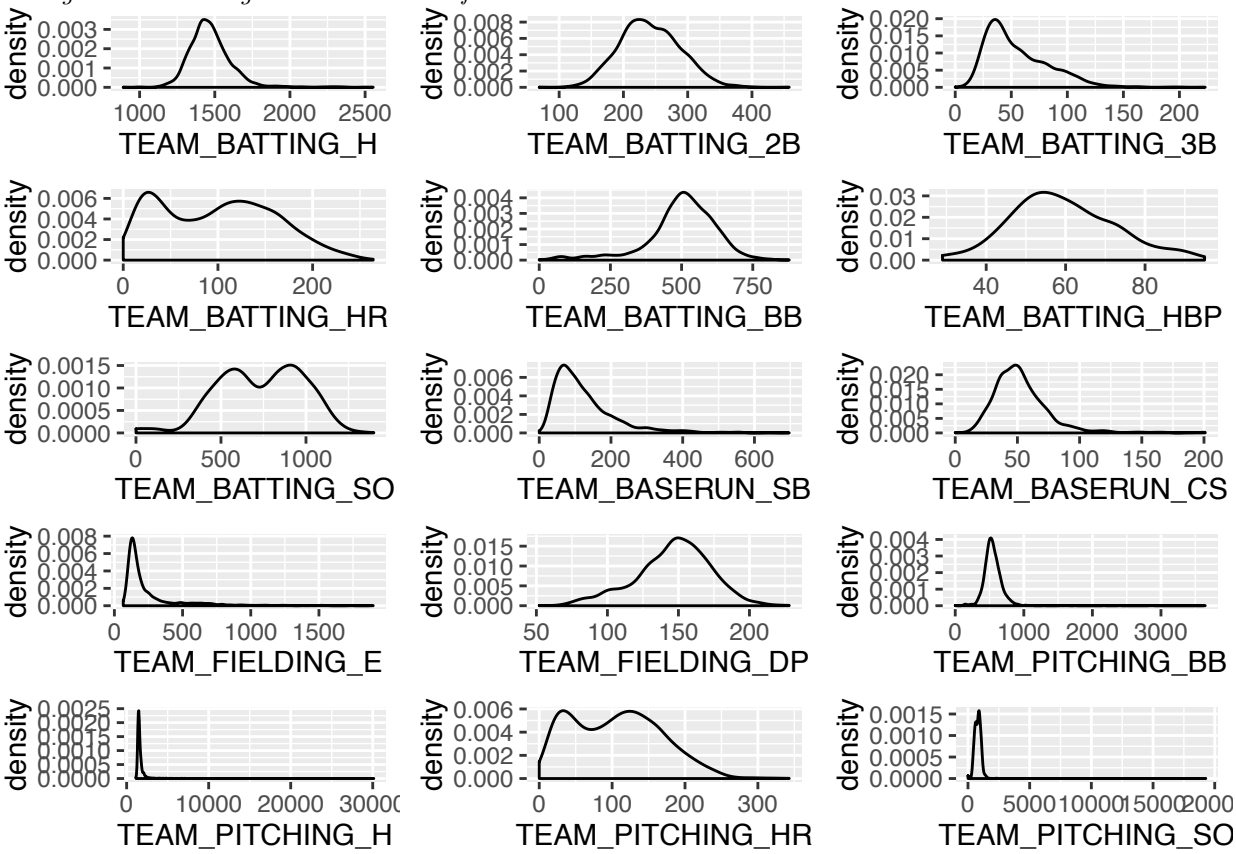


2.2 Predictor Variables

Most of the predictor variables appear to be approximately normally distributed. Interesesting results include:

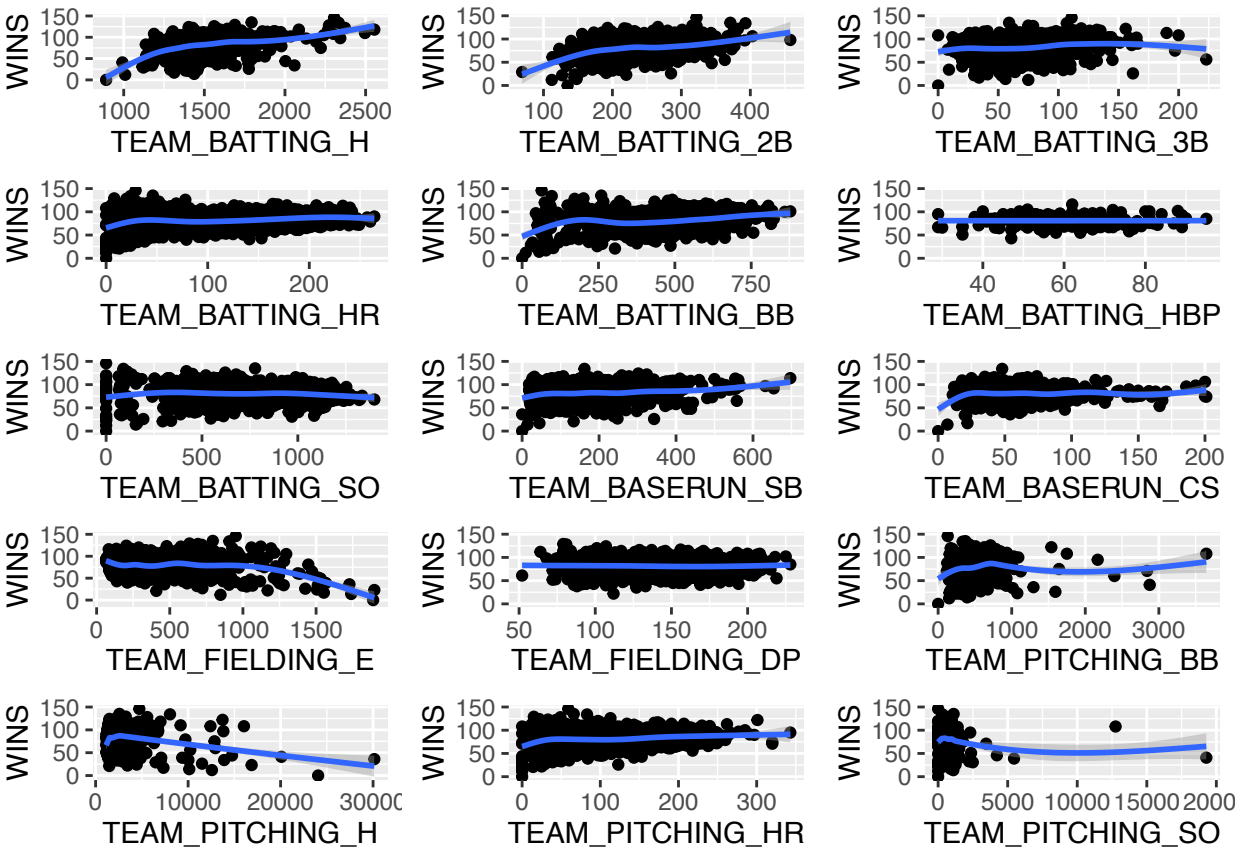
Homeruns (TEAM_BATTING_HR) appears to be multinomial. Because the dataset contains game results from 1871 to 2006, it includes time periods which are known to have influenced the occurance of homeruns, including "The Steriod Era" and the introduction of the designated hitter in the American League Greenberg, N. 2016. Batting Strikeouts (TEAM_BATTING_STRIKEOUTS) also appears multinomial.

Histograms indicating the distribution of each variable



The only variables which appear to be positively correlated with Team Wins over their entire range are Hits by Batters (TEAM_BATTING_H) and Doubles by Batters (TEAM_BATTING_2B). Errors (TEAM_FIELDING_E) is negatively correlated with wins at it's larger values. For the rest of the predictor variables, a smoothed conditional mean indicates a trend when plotted against Team Wins only at extreme high or low values where data points are sparse, or no trend at all.

Predictor variables plotted versus Team Wins, including a smoothed conditional mean



3 Data Preparation

3.1 Outliers and Non-sensical Values

The histograms and scatterplots above indicate a few variables which have outliers or non-sensical values.

3.1.0.1 Base Hits by Batters, Doubles, Triples, Homeruns

No changes made.

3.1.0.2 Strikeouts by Batters

Several 0 values were replaced with NAs, as it is virtually impossible to make it through an entire season with zero strike outs. The next low value (66) indicates 0 is missing data in this variable.

3.1.0.3 Hit by Pitch

See **Missing Data** section below.

3.1.0.4 Strikeout by Batters, Stolen bases, Caught Stealing, Errors, Double Plays

No changes made.

3.1.0.5 Errors, Walks Allowed, Strikeouts by Pitchers

Two outlier values for TEAM_PITCHING_SO are higher than the total number of out during a season excluding extra innings (4,374). In addition, these series tended to have unreasonably long right skewed tails. Therefore, high outliers, as defnited by three standard deviations from the mean, were replaced with NAs.

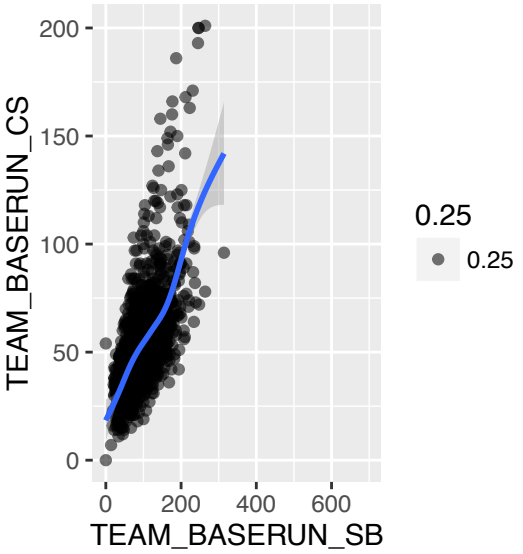
3.1.0.6 Hits Allowed

We would expect the maximum Hits Allowed (TEAM_PITCHING_H) to be on par with the maximum Hits by Batters (TEAM_BATTING_H). However, Hits allowed has many values that are thousands higher than Hits by Batters. Therefore, Hits Allowed greater than the maximum Hits by Batters were replaced with NAs.

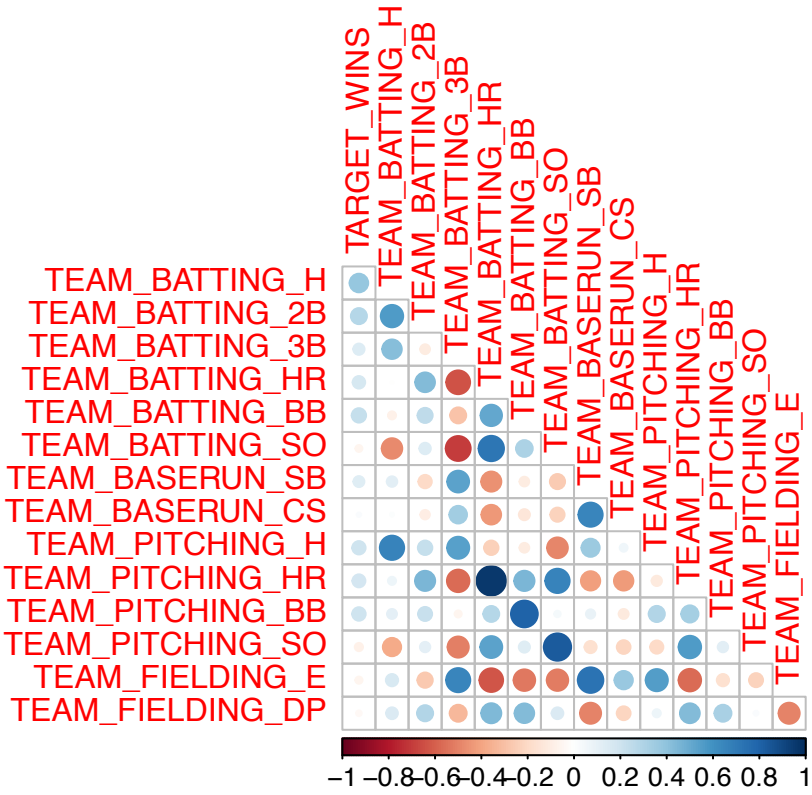
3.2 Multicollinearity

One of the challenges of this dataset is the existence of variables that are by-definition correlated. For a complete dataset for the variables here, for all teams in MLB, several variables will have common sums. Dy definition: for every one Hit by Batters there will be one Hit Allowed. This is the case for: Hits (singles through homeruns), strikeouts and walks. Teams tend to play within their leauge (American Leauge / National Leauge) and within their division frequently. This is perhaps an explanation for the existence of collinearity in the dataset.

In addition, some variables are indicators of frequency attempted. Caught stealing and stolen bases are highly correlated by team. They're also correlated by individual player - Ricky Henderson holds the MLB record for stolen bases at 1,406 - but he also holds the record for most times caught stealing, at 335.



Correlation plot for the indicator and all predictor variables



3.3 Missing Data

In statistical analysis, it is important remain mindful of context and not ignore the mechanics of the system being studied. Hit-by-Pitch is missing 2085 records - 90% of the dataset. This variable was dropped completely from the dataset and ignored by future analysis for two reasons (1) the vast majority of records were missing and (2) hit-by-pitch is a random event that happens to a team (a team cannot be 'good at being hit by pitches'), therefore it is not expected to be an indicator of total wins.

The next highest missing value is Caught Stealing, with 33% of the records missing. We decided to test out two different approaches for handling missing data. One was to impute missing values based on the existing values, while the other approach ignored all incomplete records by keeping NAs intact.

3.3.1 Dealing with NAs - Imputing from Probability Distributions

Several variables have missing values (NAs), either from the original dataset or from the elimination of outliers.

	na.count
TARGET_WINS	0
TEAM_BATTING_H	0
TEAM_BATTING_2B	0
TEAM_BATTING_3B	0
TEAM_BATTING_HR	0
TEAM_BATTING_BB	0
TEAM_BATTING_SO	122
TEAM_BASERUN_SB	131
TEAM_BASERUN_CS	772

	na.count
TEAM_PITCHING_H	115
TEAM_PITCHING_HR	0
TEAM_PITCHING_BB	13
TEAM_PITCHING_SO	108
TEAM_FIELDING_E	57
TEAM_FIELDING_DP	286

For predictor variables with missing data we imputed values by sampling from a normal distribution parameterized by the present data for that variable.

3.3.1.1 Dealing with NAs - Eliminating non-complete cases (ignoring them)

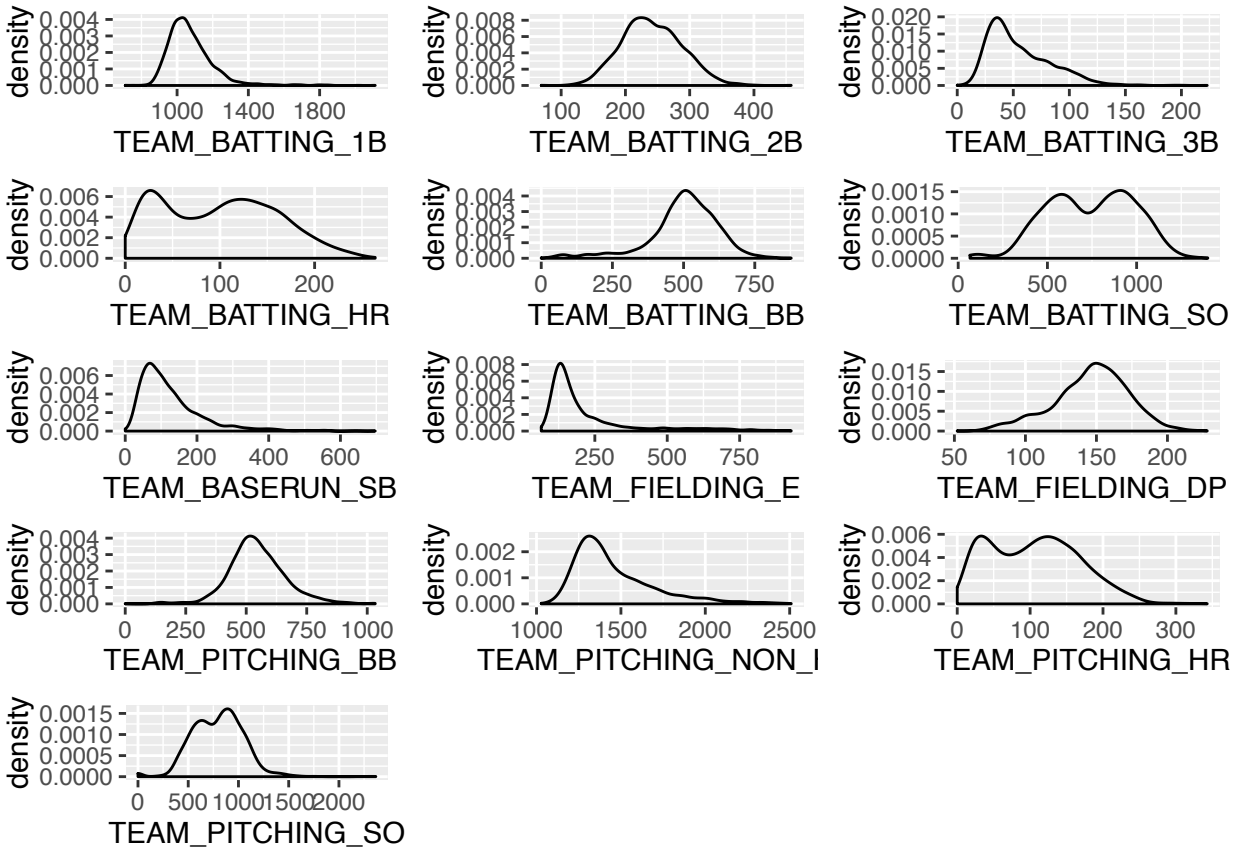
In addition to imputing data, we built one model with both imputed missing data and only complete data. The rationale for ignoring all but complete records was that we wanted to create derived values and thought it better to avoid the complexities introduced by imputation (any incorrect imputation assumptions would be compounded by their use in a derived value). In the models used in the *Modeling* section below, using complete cases resulted in more reasonable models than the imputed dataset.

3.4 Calculating Base Hits

The column recording Hits by Batters (TEAM_BATTING_H) was flagged as being a potential source of unidentifiablility, because it is composed of the sum of three additional columns: Doubles by Batters (TEAM_BATTING_2B), Triples by Batters (TEAM_BATTING_3B), and Homeruns by Batters(TEAM_BATTING_HR). While Hits by Batters may be have utility in modeling wins on its own, we determined it should not be combined in a model with doubles, triples, and homeruns. Therefore, we subtracted doubles, triples, and home runs from Hits by Batters to create Singles by Batters (TEAM_BATTING_1B).

Likewise, Hits Allowed was broken into Singles, Doubles and Triples as one variable "TEAM_PITCHING_NON_HR" by subtracting Homeruns Allowed.

Histograms indicating the distribution of each variable after data cleaning



4 Build Models

4.1 Model 1: Backwards Selection - NAs in dataset

For our first model, we used backward selection. In this method we calculate the linear model starting with all predictor variables, then remove the value with the highest P value until we have no signficance values greater than 0.05.

The model that results from backward selection on complete data only is:

$$TeamWins = 60.5 - 0.031singles - 0.080doubles + 0.152triples + 0.129homeruns + 0.151strikeouts.batting +$$

$$0.070stolen.bases + 0.057pitching - non - homeruns - 0.111pitching.walks -$$

$$0.022pitching.strikeouts - 0.119errors - 0.113double.plays$$

Several of these coefficients indicate a relationship with wins that does not make sense. For instance, the coefficients for single and doubles are both negative. Strikeouts while batting is positive. We decided to keep this model despite these results because, while it may be assumed that more singles and doubles should indicate a better team and more wins, it also says something about the pitching they faced. It may be that the model is influenced by seasons where weak pitching resulted in many hits that didn't correspond to more

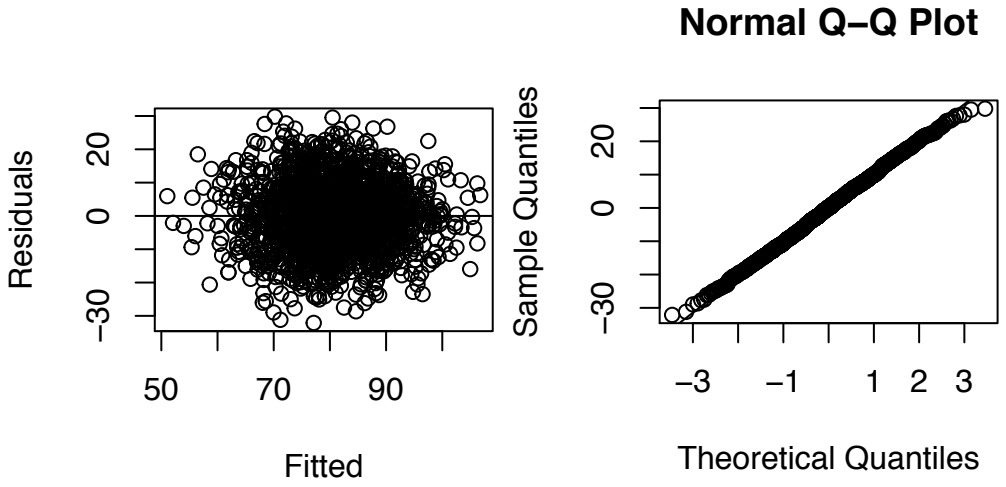
wins - just higher scoring games. Without the ability to adjust these metrics by 'at bats' (which is used for batter averages) we cannot improve on this aspect of the model.

```
## Warning in model.matrix.default(mt, mf, contrasts): the response appeared
## on the right-hand side and was dropped
```

```
## Warning in model.matrix.default(mt, mf, contrasts): problem with term 1 in
## model.matrix: no columns are assigned
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ TARGET_WINS + TEAM_BATTING_1B + TEAM_BATTING_2B +
##     TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BASERUN_SB +
##     TEAM_PITCHING_NON_HR + TEAM_PITCHING_BB + TEAM_PITCHING_SO +
##     TEAM_FIELDING_E + TEAM_FIELDING_DP, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.121  -7.181   0.137   6.900  29.800
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          60.493336   5.982599  10.112  < 2e-16 ***
## TEAM_BATTING_1B      -0.031955   0.014924  -2.141   0.0324 *
## TEAM_BATTING_2B      -0.080514   0.015485  -5.199 2.22e-07 ***
## TEAM_BATTING_3B       0.152310   0.022157   6.874 8.55e-12 ***
## TEAM_BATTING_HR       0.128822   0.007967  16.169  < 2e-16 ***
## TEAM_BATTING_BB       0.150695   0.035443   4.252 2.23e-05 ***
## TEAM_BASERUN_SB       0.070017   0.005508  12.713  < 2e-16 ***
## TEAM_PITCHING_NON_HR  0.057361   0.013417   4.275 2.01e-05 ***
## TEAM_PITCHING_BB     -0.111511   0.033533  -3.325   0.0009 ***
## TEAM_PITCHING_SO     -0.022159   0.002169 -10.217  < 2e-16 ***
## TEAM_FIELDING_E      -0.119393   0.007140 -16.722  < 2e-16 ***
## TEAM_FIELDING_DP     -0.112805   0.012255  -9.205  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.17 on 1822 degrees of freedom
##   (442 observations deleted due to missingness)
## Multiple R-squared:  0.4055, Adjusted R-squared:  0.4019
## F-statistic:   113 on 11 and 1822 DF,  p-value: < 2.2e-16
```

Residual plots from Model 1 indicate a random distribution of residuals, with some deviations at extreme values.



4.2 Model 1: Backwards Selection - Using imputed values dataset (no NAs)

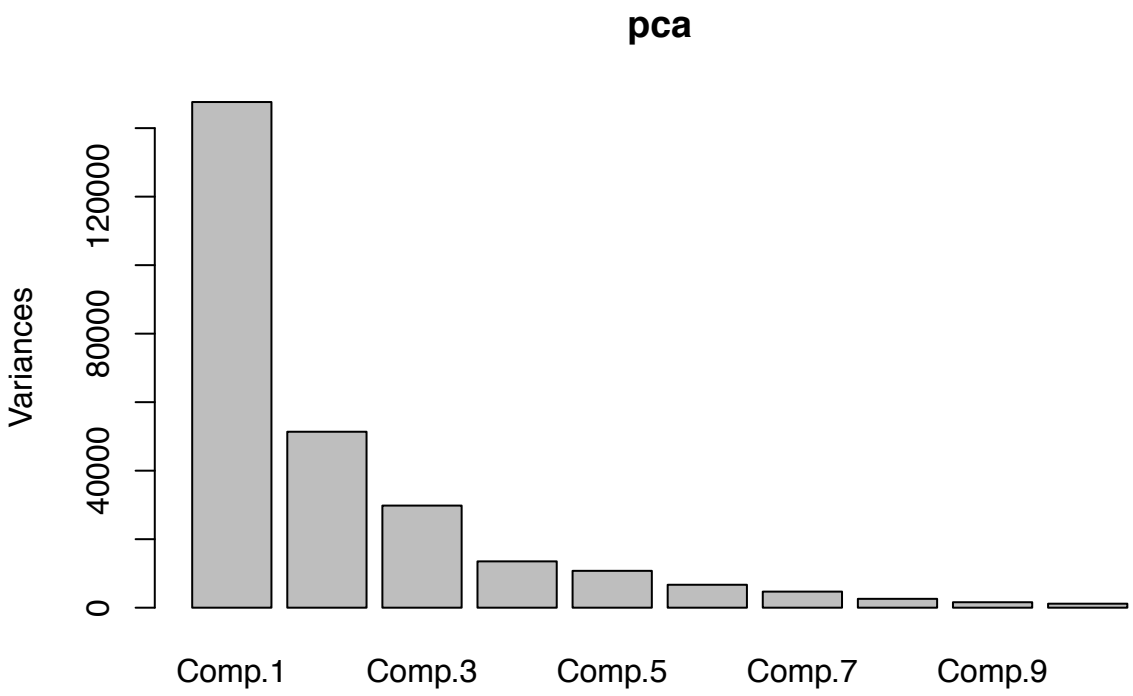
When using the dataset with imputed NA values, all variables are signficant.

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = train.fill.nas)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.296  -8.471   0.229   8.688  54.657
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.089e+01  5.399e+00   2.017  0.04378 *
## TEAM_BATTING_1B       4.134e-02  3.702e-03  11.167  < 2e-16 ***
## TEAM_BATTING_2B       3.171e-02  7.584e-03   4.181 3.01e-05 ***
## TEAM_BATTING_3B       1.064e-01  1.704e-02   6.242 5.13e-10 ***
## TEAM_BATTING_HR       8.188e-02  2.961e-02   2.765  0.00573 **
## TEAM_BATTING_BB       4.964e-02  6.209e-03   7.994 2.06e-15 ***
## TEAM_BATTING_SO       2.325e-05  2.630e-03   0.009  0.99295
## TEAM_BASERUN_SB       2.190e-02  4.256e-03   5.146 2.89e-07 ***
## TEAM_BASERUN_CS       4.432e-03  1.284e-02   0.345  0.73005
## TEAM_PITCHING_NON_HR  3.200e-03  1.889e-03   1.694  0.09034 .
## TEAM_PITCHING_HR      2.243e-02  2.652e-02   0.846  0.39771
## TEAM_PITCHING_BB     -2.585e-02  5.793e-03  -4.463 8.48e-06 ***
## TEAM_PITCHING_SO     -7.938e-04  2.086e-03  -0.381  0.70359
## TEAM_FIELDING_E      -4.463e-03  2.951e-03  -1.512  0.13057
## TEAM_FIELDING_DP     -1.050e-01  1.245e-02  -8.436  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.37 on 2261 degrees of freedom
## Multiple R-squared:  0.2839, Adjusted R-squared:  0.2795
## F-statistic: 64.02 on 14 and 2261 DF,  p-value: < 2.2e-16
```

4.3 Model 2:

For this model, we take advantage of principal component analysis method. First, we used an orthogonal transformation to convert our variables into a set of values of linearly uncorrelated variables, which called principal components. And then we chose the first five principal components that account for around 95% proportion of variance in the data. Finally, we used those chosen principal components to build a linear regression model.

```
# PCA analysis
Predictor <- train.fill.nas$TARGET_WINS
A <- as.matrix(select(train.fill.nas,-TARGET_WINS))
pca <- princomp(A,center=T,scale.=T)
plot(pca)
```




```
summary(pca)
```

```
## Importance of components:
##                             Comp.1      Comp.2      Comp.3       Comp.4
## Standard deviation     384.2192847 226.6345455 172.6607355 116.27354942
## Proportion of Variance   0.5445055   0.1894507   0.1099591   0.04986614
## Cumulative Proportion    0.5445055   0.7339561   0.8439152   0.89378138
##                             Comp.5      Comp.6     Comp.7       Comp.8
## Standard deviation     103.71302363 81.83448302 68.5373215 50.860622880
## Proportion of Variance   0.03967441  0.02470112  0.0173260  0.009541294
## Cumulative Proportion    0.93345579  0.95815691  0.9754829  0.985024199
##                             Comp.9      Comp.10      Comp.11      Comp.12
## Standard deviation     39.933740233 34.121932816 22.431916142 21.654141916
## Proportion of Variance  0.005881985  0.004294486  0.001855994  0.001729521
## Cumulative Proportion   0.990906184  0.995200670  0.997056664  0.998786186
##                            Comp.13      Comp.14
## Standard deviation     16.711557594 7.0575516547
```

```
## Proportion of Variance  0.001030096 0.0001837182
## Cumulative Proportion   0.999816282 1.0000000000
```

```
pca <- as.data.frame(pca$scores[,1:5])
train_pca <- cbind(TARGET_WINS=Predictor,pca)
head(train_pca)
```

```
##   TARGET_WINS    Comp.1      Comp.2     Comp.3     Comp.4     Comp.5
## 1          39 -98.71852 -139.043689   31.45711 -46.864102 97.1932047
## 2          70 583.58046   -2.050978  102.90234 153.928657 27.2148864
## 3          86 329.56343  -77.523496   39.83319  75.376295  9.8793404
## 4          70 266.43069  -35.978839 -129.80400 -63.649910 -0.7413798
## 5          82 334.73175 -120.281629 -129.60476  -3.396324 -1.7657030
## 6          75 397.45965  -82.204331 -170.81421 -12.825813 25.8195895
```

```
# Separate data into two parts, one for training models and the other for testing models
set.seed(45)
inTrain_pca <- createDataPartition(y=train_pca$TARGET_WINS, p=0.7,list=FALSE)
training_pca <- train_pca[inTrain_pca,]
testing_pca <- train_pca[-inTrain_pca,]

# Build a model
lm2_a <- lm(TARGET_WINS ~ ., data=training_pca)
summary(lm2_a)
```

```
##
## Call:
## lm(formula = TARGET_WINS ~ ., data = training_pca)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.671  -9.384   0.770   9.790  63.983
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 80.8683822  0.3732178 216.679  < 2e-16 ***
## Comp.1      -0.0043681  0.0009728  -4.490 7.62e-06 ***
## Comp.2       0.0022652  0.0016569   1.367  0.17178
## Comp.3       0.0242682  0.0021596  11.237  < 2e-16 ***
## Comp.4      -0.0086712  0.0032026  -2.708  0.00685 **
## Comp.5       0.0003899  0.0036242   0.108  0.91433
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.9 on 1589 degrees of freedom
## Multiple R-squared:  0.08909,    Adjusted R-squared:  0.08622
## F-statistic: 31.08 on 5 and 1589 DF,  p-value: < 2.2e-16
```

4.4 Model 3: Using Variable Ratios

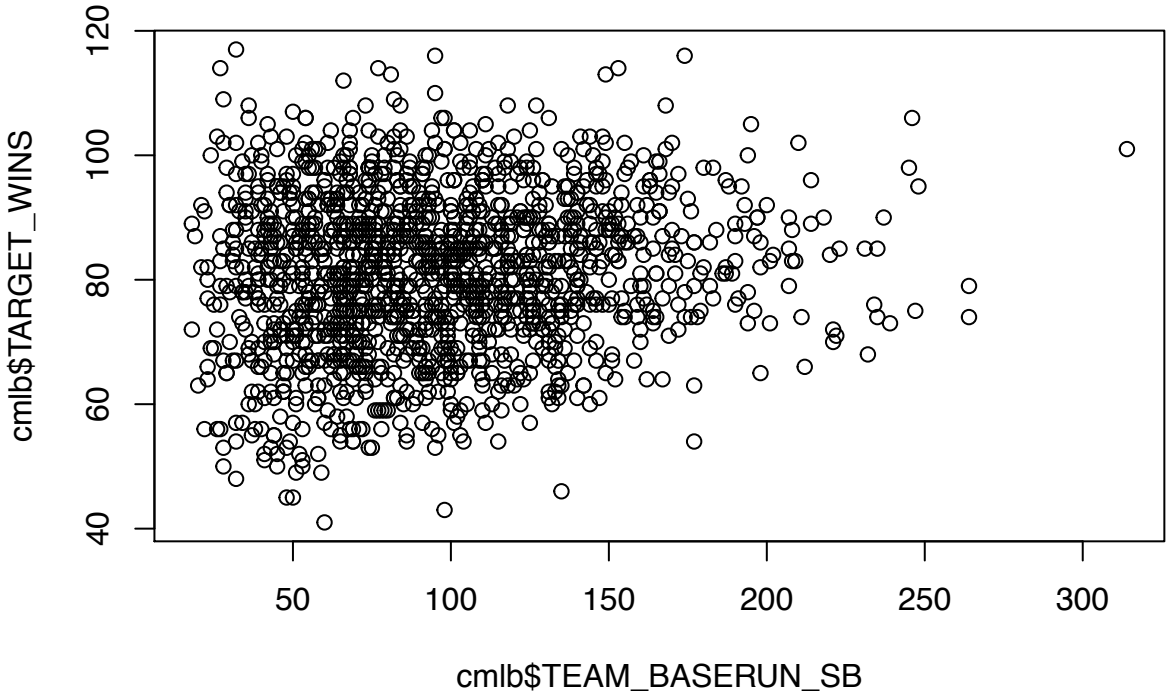
As discussed above, several variables display collinearity. Some variables by definition have the same sum (for every one batting walk, another team has a pitching walk) and some variables indicate a hidden variable

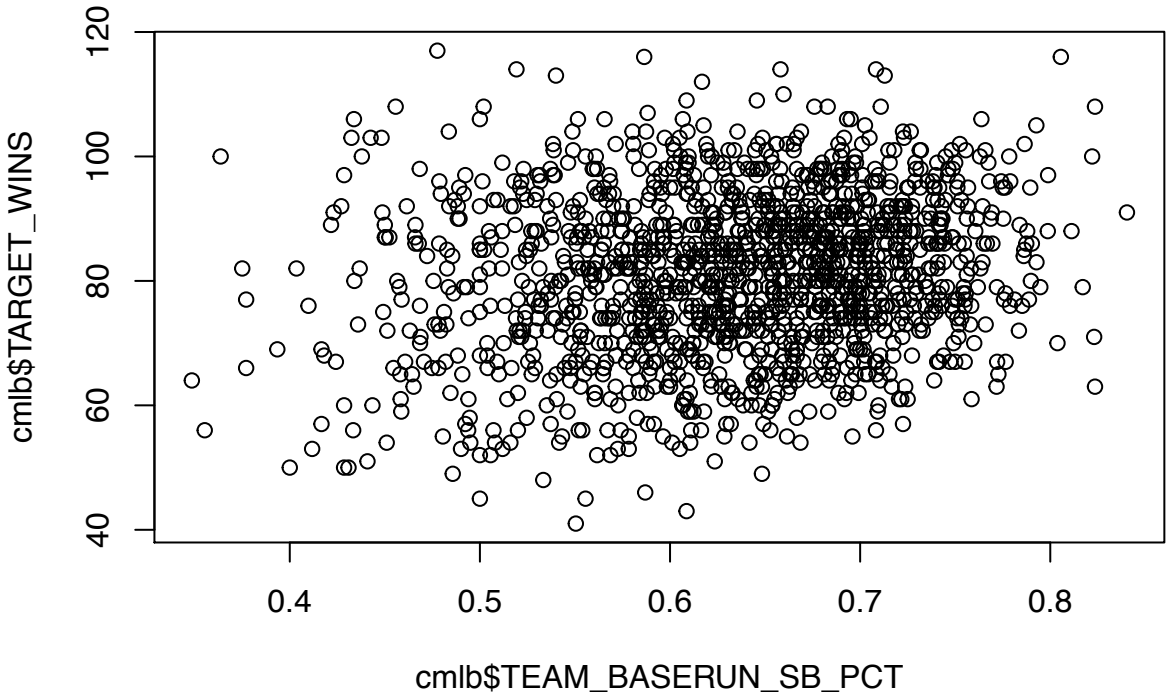
(stolen bases and caught stealing appear to indicate stolen base attempts). Therefore, we used several ratios among predictor variables.

4.5 Deriving Ratio Variables

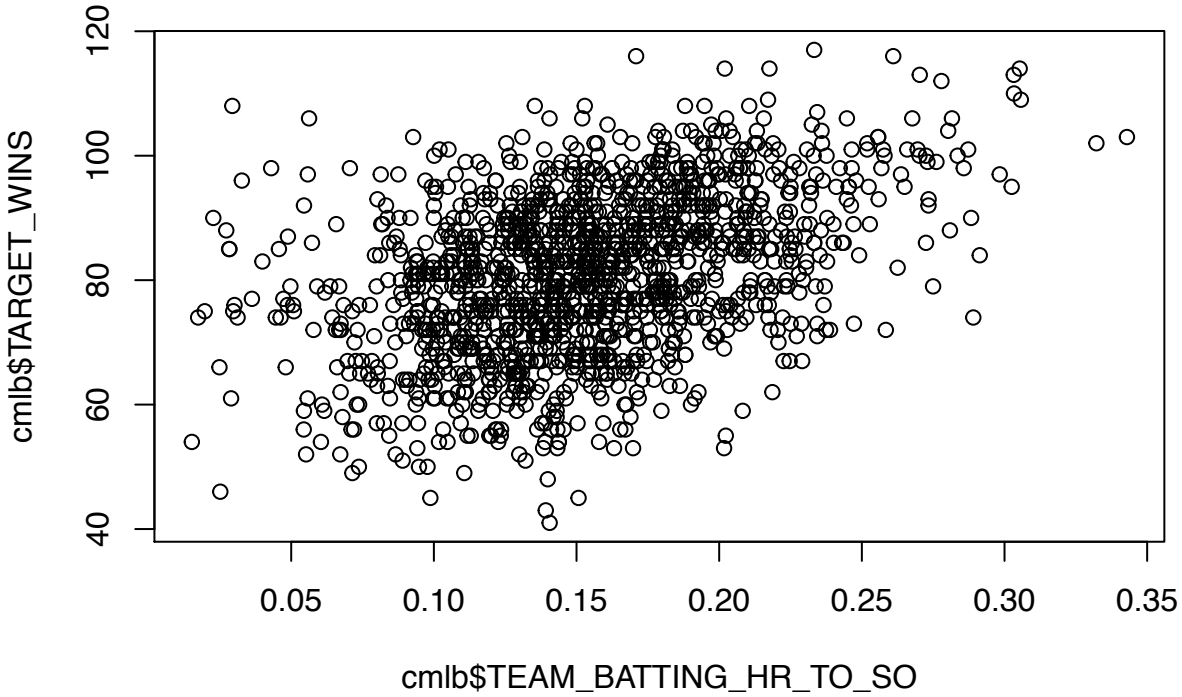
One area of interest was whether or not variables relative to another relevant variable would prove to be a better predictor than the raw data on its own. To test this out, we created 3 ratio variables: Stolen base percentage, HR to Strikout (batting) percentage, and Strikeout to Walk (pitching, otherwise known as K/BB) percentage. These ratios were calculated from the dataset that ignored records with missing values (discussed later).

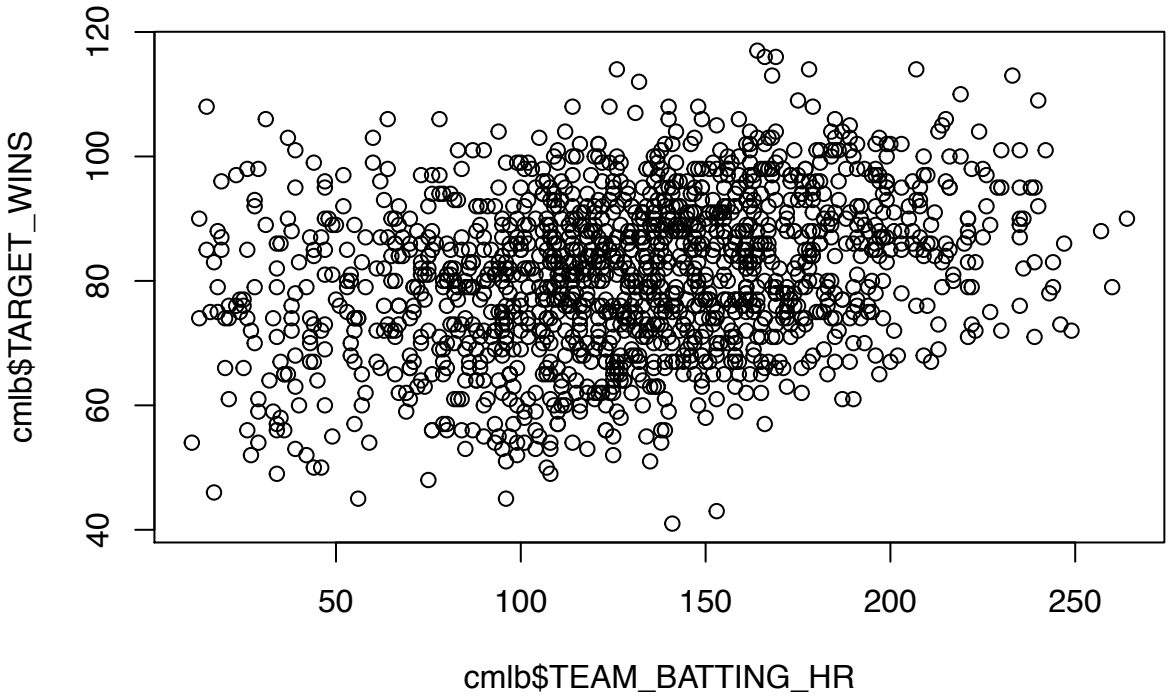
Stolen base percentage measured stolen bases over stolen bases plus caught stealing, which would constitue total stolen base attempts. Plotting the graphs of stolen bases against wins and comparing to stolen base percentage against wins showed that the percentage had a slightly positive linear relationship with wins. Stolen bases did have a positive correlation with wins (.1203), but the percentage had a stronger relationship (.172). Due to the stronger and more linear relationship, we will use the percentage over stolen bases.





HR to Strikout percentage was derived because batting HRs and Strikeouts had a correlation of .6402. Thus perhaps more valuable than knowing the gross amounts of either variable would be the ratio of one to the other. And it did appear graphically that there was a stronger linear relationship using the HR to Strikout ratio than simply HRs. The homerun to strikeout percentage was more highly correlated with wins (.419) than homeruns (.2834), so we will use Home Run to Strikeout percenatage in our model.

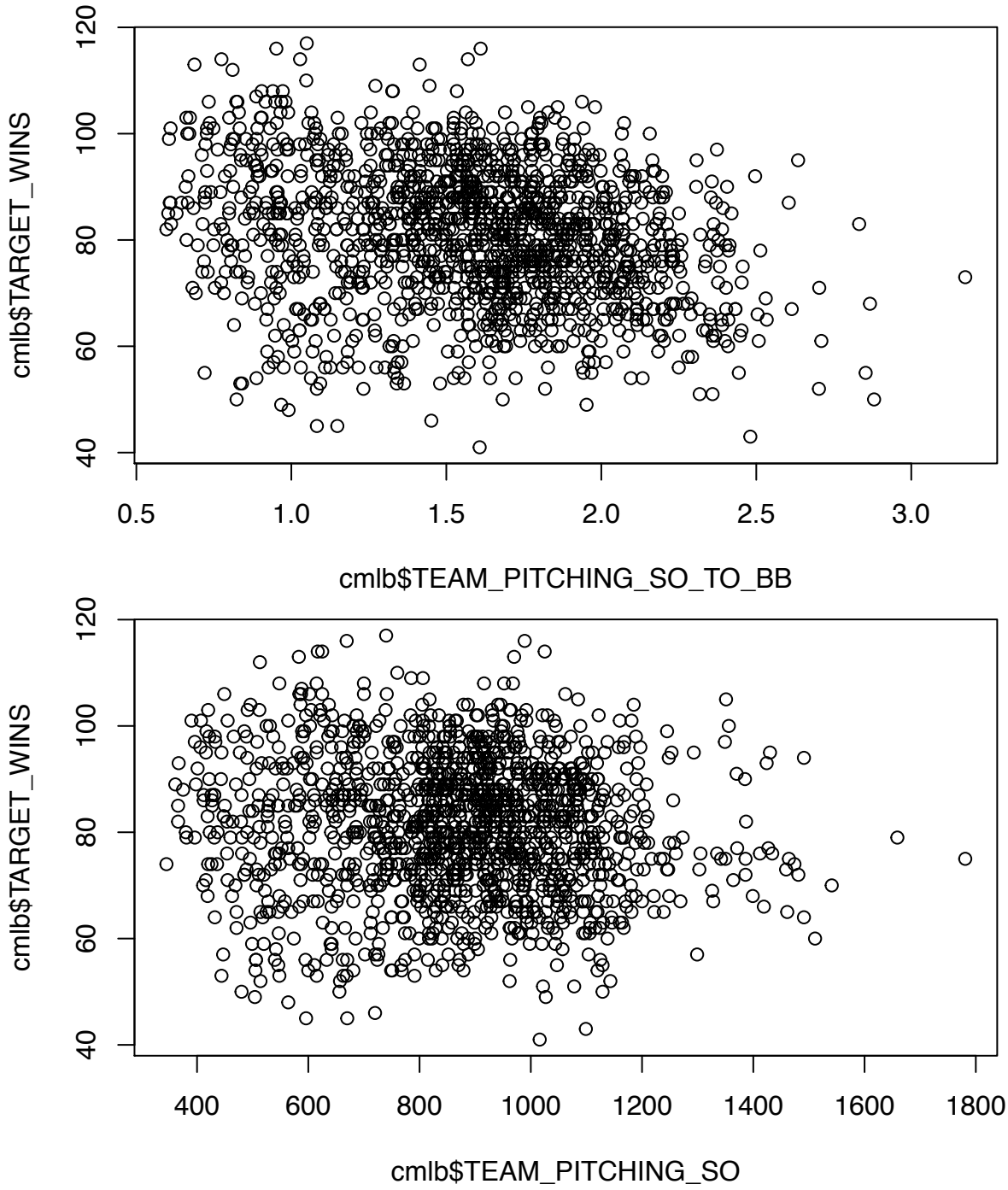




For pitching, Strikeout to Walk (K/BB) Ratio was calculated. This is a traditional baseball statistic that has currently come under scrutiny with the modernization of baseball analysis. With that in mind we thought it would be of interest to see what kind of impact including this variable would have on a model.

Counterintuitively, both Strikouts and K/BB showed negative correlations with wins (-.067 and -0.2312). Since strikouts are generally a good thing for the pitching team, and walks are generally bad, this is a surprising finding. One would think that maximizing the ratio of good events to bad events would ultimately lead to more wins, but this decidedly not the case. Further investigation into this matter would be of interest, but is beyond both the scope and the available data in this study.

K/BB appeared to have a slightly more linear relationship visually, so this combined with it's higher correlation made in the factor that we chose for the ratio model.



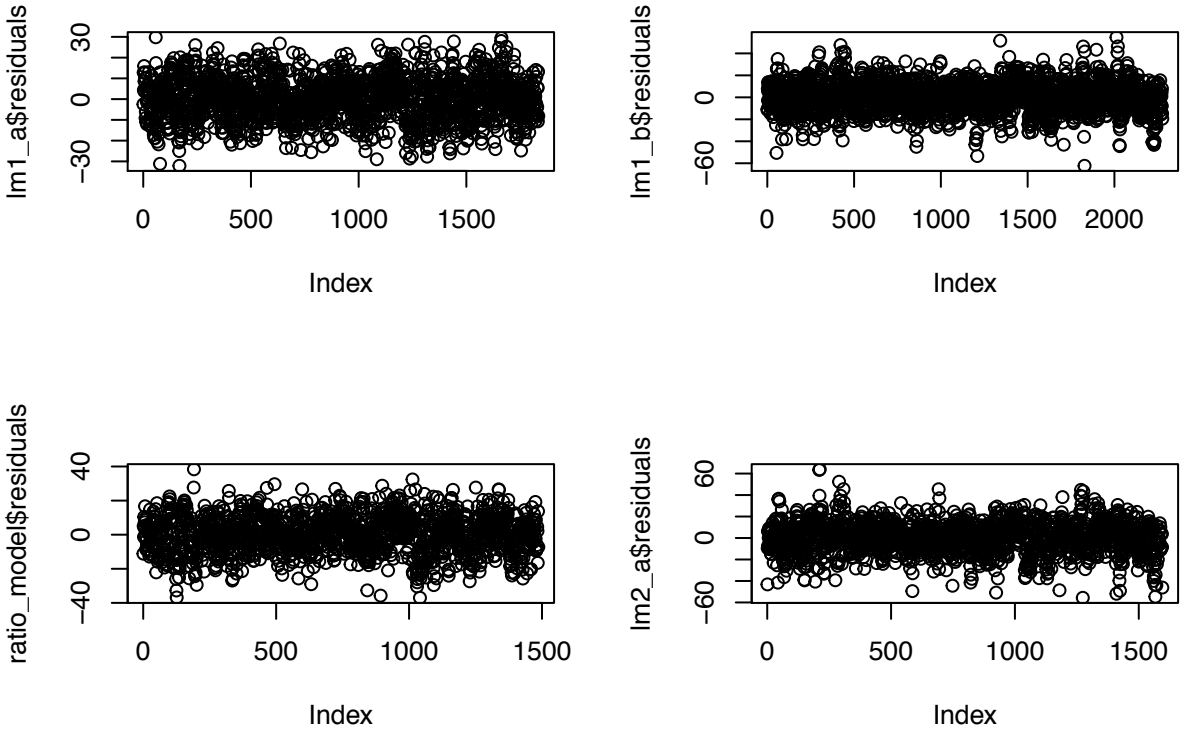
In the end, the ratio model wound up being calculated as follows:

Predicted_wins = 56.44 + 34.16(StolenBasePct) + 96.45(HR/SO) - 7.5(K/BB)

5 Select Model: Evaluating The Models

The below table shows the 3 models and their corresponding statistics for selecting an appropriate model:

Model Name	Mean Sq. Error	R Squared	F Stat
lm1_a	102.812661984364	0.40552401333798	78.1113972466726
lm1_b	177.610315532526	0.283890840739642	149.265969291314
ratio_model	120.359246926108	0.252517590606597	58.9419811545502
pca	221.140824912914	0.0890865092903272	20.3792678947301

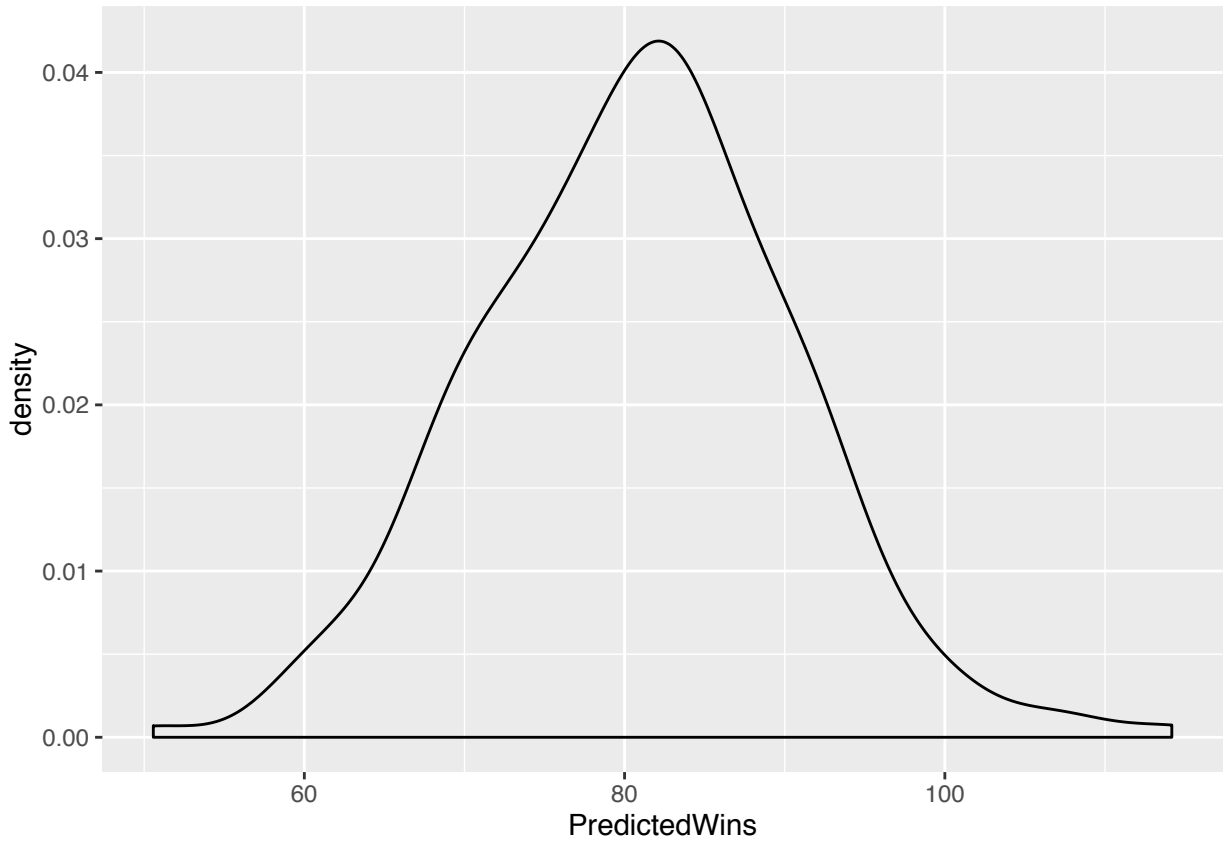


5.0.1 Predicted wins

The first model fit, lm1_a, used backwards stepwise selection and the dataset containing NAs. It has the lowest mean square error, highest coefficient of determination and second highest F-Statistic. It's residuals do not indicate any red flags. Therefore, we have chosen it as our best model for predicting team wins.

Using the predict.lm() function, we use the first linear model created, lm1_a, to predict team wins from the evaluation dataset. This density plot of the wins output resembles the training data plot above. This is a good indication our model is reasonable.

```
## Warning: Removed 54 rows containing non-finite values (stat_density).
```




```
write.csv(predicted.wins, file = "model_predicted_wins.csv")
```