Final Project

Benji Gold and Sam Alkalay

Introduction

Baseball, America's pastime, has a long and storied tradition that dates back well over 100 years. Since the 1850's, some form of statistics measuring how good a player is has been tracked. This began through the use of the box score, which tracked basic statistics, such as hits, runs, and errors, from which a player's batting average can be constructed. Over one hundred years later, a pioneering statistician by the name of Bill James introduced new statistical concepts, such as on-base percentage and runs created, in his annual Baseball Abstract (Lee 2018). As technology has improved, the statistics being tracked became more and more sophisticated. Then, in 2015 analytics in baseball took a giant leap. With the introduction of Statcast, teams were able to track novel metrics, such as a batter's exit velocity (the speed of the baseball as it comes off the bat, immediately after a batter makes contact) and barrel percentage (the percentage of baseballs hit off of the player's barrel) ("Statcast Search"). Around the league, teams adopted these new statistics to try and gain a competitive advantage, through which they would be able to better predict a player's potential. However, is this actually the case? While these new statistics are widely used, it is unclear whether they actually provide any useful information for predicting a player's potential. This research project intends to explore that idea through the use of a logistic regression model to predict whether a player is an all-star. The research question of interest is:

Do old or new wave statistics do a better job at predicting whether a player is selected as an all-star?

The response variables of interest are: All.Star: Whether a player is selected as an all-star.

For our analysis, we have selected two datasets. The first is from Baseball Reference, which consists of standard statistics that offer a broad view of a player's performance in a particular season. The second is from Statcast, which consists of each player's primary position. The final data file we have was compiled from baseballsavant.com with a mix of more traditional stats and statcast stats. This complete file can be found in the stats.csv file. Once that was done, we entered a players position, salary, and team from baseball prospectus. we used wikipedia to find rosters for the 2019 all-star game and created a categorical variable column with that information. ## Methodology

Packages and Data

```r
library(tidyverse)
library(tidymodels)
library(glmnet)
library(caret)
library(MASS)
library(lme4)
stats <- read.csv("data/stats.csv")


stats <- replace(stats, stats =="", NA)
stats <- stats %>%
  drop_na() %>%
  mutate(AVG300 = case_when(batting_avg >= .3 ~ "Greater than 300", TRUE ~ "Less than 300"
         HR40 = case_when(b_home_run >= 40 ~ "Greater than 40", TRUE ~ "Less than 40"), pi
view(stats)
```

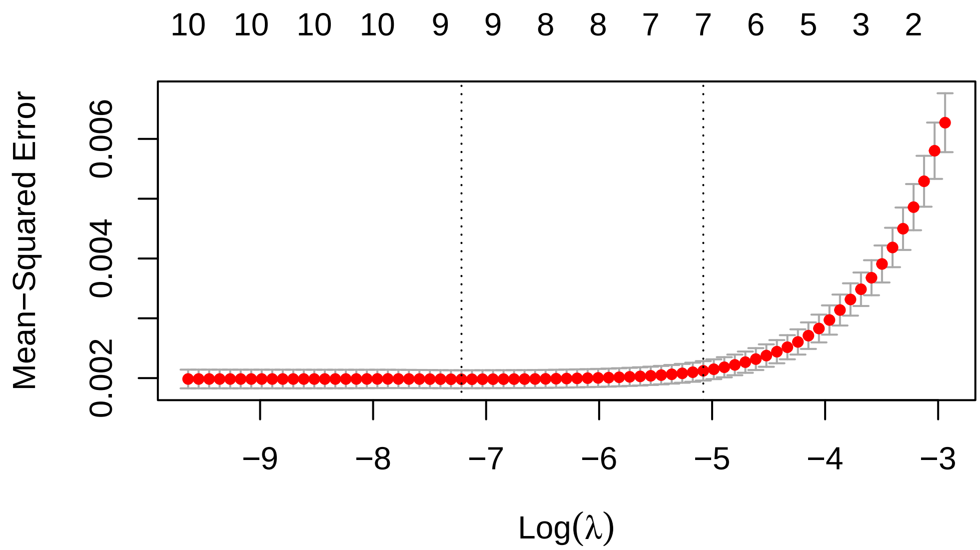
Results

```r
# Remove Nationals from Data
rol_stats <- stats |>
  filter(Team != "WAS")
```

```r
# obp percentage lasso for rol
set.seed(0)
y <- rol_stats$on_base_percent
x <- model.matrix(on_base_percent ~ launch_angle_avg + sweet_spot_percent +
                    barrel + solidcontact_percent + flareburner_percent +
                    hard_hit_percent + avg_hyper_speed + z_swing_percent +
                    oz_swing_percent + meatball_swing_percent, data = rol_stats)
m_lasso_cv <- cv.glmnet(x, y, alpha = 1)
best_lambda <- m_lasso_cv$lambda.min
best_lambda
```

```
[1] 0.0007332828
```

```r
plot(m_lasso_cv)
```




```r
m_best <- glmnet(x, y, alpha = 1, lambda = best_lambda)
m_best$beta
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                                  s0
(Intercept)                .
launch_angle_avg         -0.0001492831
sweet_spot_percent        0.0021040413
barrel                    0.0013140348
solidcontact_percent      0.0019525793
flareburner_percent       0.0036463362
hard_hit_percent          0.0009244833
avg_hyper_speed           .
z_swing_percent           0.0004967093
oz_swing_percent         -0.0028515915
meatball_swing_percent    0.0007020832
```

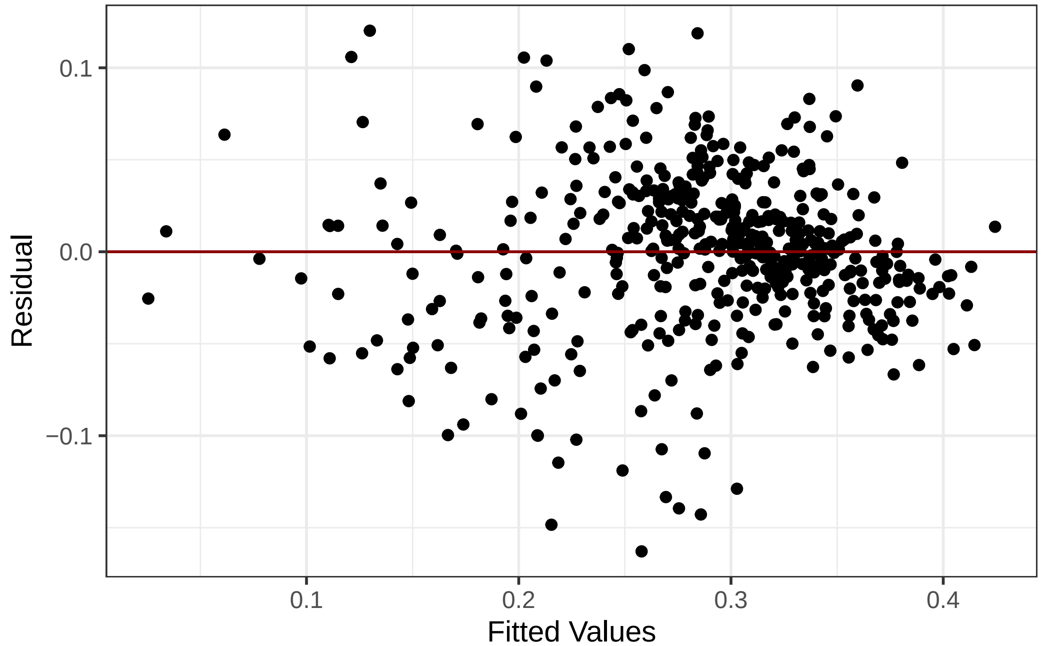
```r
# obp percentage prediction
m3 <- lm(on_base_percent ~ launch_angle_avg + sweet_spot_percent +
                    barrel + solidcontact_percent + flareburner_percent +
                    hard_hit_percent + z_swing_percent +
```

```r
                        oz_swing_percent + meatball_swing_percent,
   data = rol_stats)

summary(m3)
```

```

Call:
lm(formula = on_base_percent ~ launch_angle_avg + sweet_spot_percent +
    barrel + solidcontact_percent + flareburner_percent + hard_hit_percent +
    z_swing_percent + oz_swing_percent + meatball_swing_percent,
    data = rol_stats)

Residuals:
      Min        1Q    Median        3Q       Max
-0.162876 -0.022911  0.001195  0.026808  0.120152

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)             0.0660827  0.0198729   3.325 0.000955 ***
launch_angle_avg       -0.0004484  0.0003492  -1.284 0.199777
sweet_spot_percent      0.0021709  0.0004241   5.119 4.53e-07 ***
barrel                  0.0013401  0.0001692   7.923 1.80e-14 ***
solidcontact_percent    0.0022328  0.0008563   2.607 0.009421 **
flareburner_percent     0.0037954  0.0004981   7.619 1.49e-13 ***
hard_hit_percent        0.0008752  0.0003014   2.903 0.003871 **
z_swing_percent         0.0006523  0.0004437   1.470 0.142190
oz_swing_percent       -0.0030238  0.0003236  -9.343  < 2e-16 ***
meatball_swing_percent  0.0007344  0.0002548   2.883 0.004131 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04353 on 456 degrees of freedom
Multiple R-squared:  0.7052,    Adjusted R-squared:  0.6994
F-statistic: 121.2 on 9 and 456 DF,  p-value: < 2.2e-16
```

```r
m3_aug <- augment(m3)
m3_aug |>
ggplot(aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "darkred") +
  labs(x = "Fitted Values",
```

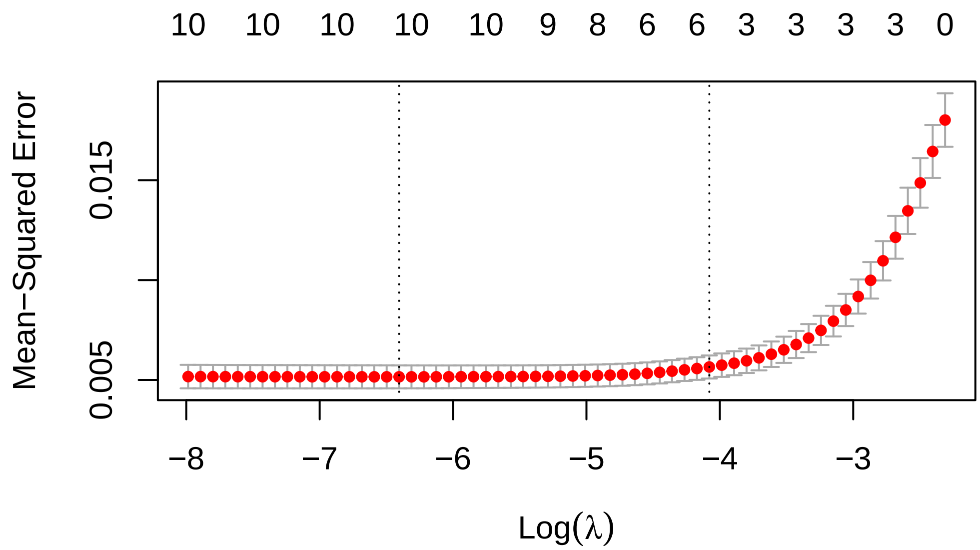
```
         y = "Residual") +
    theme_bw()
```




```
# slugging percentage lasso
set.seed(0)
y <- rol_stats$slg_percent
x <- model.matrix(slg_percent ~ launch_angle_avg + sweet_spot_percent +
                    barrel + solidcontact_percent + flareburner_percent +
                    hard_hit_percent + avg_hyper_speed + z_swing_percent +
                    oz_swing_percent + meatball_swing_percent, data = rol_stats)
m_lasso_cv <- cv.glmnet(x, y, alpha = 1)
best_lambda <- m_lasso_cv$lambda.min
best_lambda
```

```
[1] 0.001653873
```

```
plot(m_lasso_cv)
```




```
m_best <- glmnet(x, y, alpha = 1, lambda = best_lambda)
m_best$beta
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                                 s0
(Intercept)              .
launch_angle_avg         0.0004048935
sweet_spot_percent       0.0039725436
barrel                   0.0035324802
solidcontact_percent     0.0016992743
flareburner_percent      0.0027488668
hard_hit_percent         0.0014540632
avg_hyper_speed          0.0046269135
z_swing_percent          0.0010559425
oz_swing_percent        -0.0005609647
meatball_swing_percent   0.0006883043
```

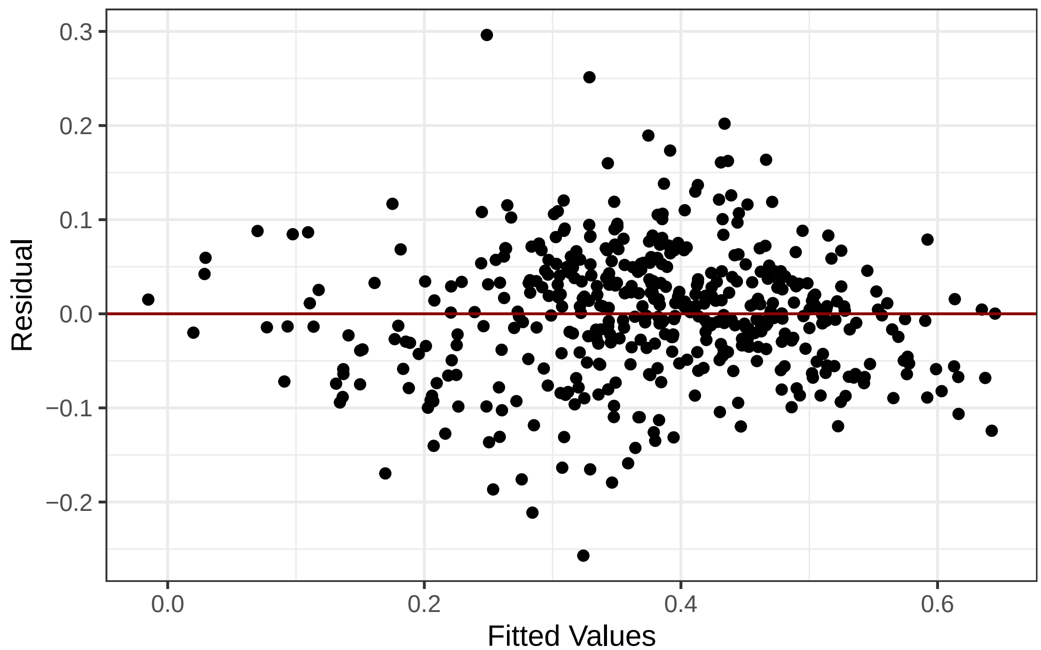
```
# slugging percentage prediction
m4 <- lm(slg_percent ~ launch_angle_avg + sweet_spot_percent +
                    barrel + solidcontact_percent + flareburner_percent +
                    hard_hit_percent + avg_hyper_speed + z_swing_percent +
```

```r
                      oz_swing_percent + meatball_swing_percent,
    data = rol_stats)

summary(m4)
```

```

Call:
lm(formula = slg_percent ~ launch_angle_avg + sweet_spot_percent +
    barrel + solidcontact_percent + flareburner_percent + hard_hit_percent +
    avg_hyper_speed + z_swing_percent + oz_swing_percent + meatball_swing_percent,
    data = rol_stats)

Residuals:
      Min        1Q    Median        3Q       Max
-0.256940 -0.041724  0.001751  0.040773  0.296249

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)            -0.0790676  0.0327432  -2.415  0.01614 *
launch_angle_avg        0.0004505  0.0005610   0.803  0.42241
sweet_spot_percent      0.0038602  0.0006829   5.653 2.79e-08 ***
barrel                  0.0035814  0.0002773  12.916  < 2e-16 ***
solidcontact_percent    0.0021829  0.0013866   1.574  0.11612
flareburner_percent     0.0031794  0.0008195   3.880  0.00012 ***
hard_hit_percent        0.0010022  0.0010543   0.951  0.34233
avg_hyper_speed         0.0067277  0.0060541   1.111  0.26704
z_swing_percent         0.0013267  0.0007133   1.860  0.06351 .
oz_swing_percent       -0.0008821  0.0005243  -1.683  0.09314 .
meatball_swing_percent  0.0007281  0.0004096   1.777  0.07618 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06994 on 455 degrees of freedom
Multiple R-squared:  0.7345,    Adjusted R-squared:  0.7287
F-statistic: 125.9 on 10 and 455 DF,  p-value: < 2.2e-16
```

```r
m4_aug <- augment(m4)
m4_aug |>
ggplot(aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "darkred") +
```

```
  labs(x = "Fitted Values",
       y = "Residual") +
  theme_bw()
```



The final models for predicting on base and slugging percentage are both linear regressions. Their variables were chosen using LASSO for the advanced statistics, as defined by the data dictionary. These models were trained on data from the 2019 season for each team, aside from the Washington Nationals, which were used as a test set. We decided it was most appropriate to use a whole team as a test set because it provided a variety of positions and was most useful in terms of comparing expected lineups (lineups that would be expected to generate the most runs) to actual lineups. Now, the LASSO for the on base percentage model had a best lambda of approximately 0.000733 using 10-fold cross validation, and the variables included were launch_angle_avg, sweet_spot_percent, barrel, solidcontact_percent, flareburner_percent, hard_hit_percent, z_swing_percent, oz_swing_percent, and meatball_swing_percent. After using these variables in the linear regression, we were able to conclude that the assumptions for a linear regression were satisfied because in the residual plot we see mostly symmetrically distributed and evenly spaced observations, satisfying the criteria for linearity and constant variance. Furthermore, there is a slight correlation between the predictors and no clear deviation patterns in the Q-Q plot, satisfying the assumption for independence and normality. Moreover, 69.94% of the variation in OBP was explained by the predictors in the model (adjusted R-squared).

Among the significant variables in this linear regression, one of the most notable positive predictors was flareburner_percent. For each 1% increase in flareburner_percent, expected on base percentage increases by approximately 0.380%. On the other hand, one of the most notable negative predictors was oz_swing_percent. For each 1% increase in flareburner_percent, expected on base percentage decreases by approximately 0.302%. These results both make sense in the context of the game because being able to hit a variety of pitches would make a batter better, while swinging at bad pitches would make a batter worse. Moving on to slugging percentage, we observed a best lambda of approximately 0.00165, and the variables included were launch_angle_avg, sweet_spot_percent, barrel, solidcontact_percent, flareburner_percent, hard_hit_percent, avg_hyper_speed, z_swing_percent, oz_swing_percent, and meatball_swing_percent. Like the other model, all assumptions for a linear regression were met, and 72.87% of the variation was explained by the predictors in the model.

Among the significant variables in this model, the most notable predictors were sweet_spot_percent and barrel. For each 1% increase in each of these predictors, expected slugging percentage increases by approximately 0.386% and 0.358% respectively. oz_swing_percent still had a negative correlation in this model, but its impact was far less. These results make sense as slugging percentage is based on how good a player's hit is, and both of these predictors are strongly associated with doubles, triples, and home runs, while OBP is only based on whether a player reaches a base.

Discussion

The biggest limitation of our model is that it only predicts on base and slugging percentage for players who are already in the major leagues. While this is helpful for decisions for future seasons, the model is not able to shine any light on how a player may perform before they actually play in the major leagues. Fortunately, major league baseball franchises have a well developed farm system where players spend years in the minor leagues. This is the highest level of baseball, aside from the major league, in the United States, and they track all of the same stats that the MLB does. Therefore, a natural progression for future work would be to try and predict a player's on base and slugging percentage based on their advanced statistics in the minor leagues. This would give teams a better gauge on how players would perform in the major leagues, rather than how players would perform in future seasons after they have already made it to the majors. With this type of model, teams would have a better idea of how to strategically bring up players and put the team that gives them the best chance of winning on the field.

```r
# Subset for nationals
nationals_stats <- stats |>
  filter(Team == "WAS")

# Predict
```

```r
pred_obp <- predict(m3, nationals_stats)
pred_slg <- predict(m4, nationals_stats)

# Add to DF
nationals_stats <- nationals_stats |>
  mutate(Predicted_OBP = pred_obp,
         Predicted_SLG = pred_slg)

# Display
nationals_pred_stats <- nationals_stats[ , c("last_name", "first_name", "Position",
                                             "Predicted_OBP", "Predicted_SLG")]
print(nationals_pred_stats)
```

```
      last_name  first_name Position Predicted_OBP Predicted_SLG
1       Sanchez      Anibal       SP    0.07961863     0.1275467
2  Kendrick III       Howie       1B    0.35010631     0.5018412
3        Suzuki        Kurt        C    0.33371079     0.4386308
4      Scherzer         Max       SP    0.20971532     0.2277935
5     Zimmerman        Ryan       1B    0.28173250     0.3623809
6     Hellickson      Jeremy       SP    0.18646363     0.1018744
7         Gomes         Yan        C    0.27653808     0.3802595
8        Rendon     Anthony       3B    0.41590403     0.6003806
9     Strasburg     Stephen       SP    0.24879639     0.2691283
10        Adams        Matt       1B    0.31008350     0.4743130
11       Corbin     Patrick       SP    0.14354136     0.1522589
12       Taylor  Michael A.       CF    0.30846931     0.3732054
13       Dozier       Brian       2B    0.33754471     0.4167834
14         Difo      Wilmer       SS    0.29585135     0.3388092
15        Eaton        Adam       RF    0.31271661     0.4048301
16       Turner        Trea       SS    0.32119304     0.4405392
17       Robles      Victor       CF    0.28957201     0.3891161
18    Stevenson      Andrew       PH    0.26422635     0.3259004
19         Soto        Juan       LF    0.40647774     0.5769956
20         Noll        Jake       PH    0.15891038     0.1942342
```

Variable Selections and Regressions we tried

```r
# LASSO Variable Selection Basic Stats
y <- stats$All.Star
x <- model.matrix(All.Star ~ player_age + b_ab + b_total_pa + b_total_hits + b_home_run +
                    b_double + b_triple + b_home_run * HR40 + b_strikeout + b_walk +
                    batting_avg + slg_percent + on_base_percent + Position, data = stats)
m_lasso_cv <- cv.glmnet(x, y, alpha = 1)
best_lambda <- m_lasso_cv$lambda.min
best_lambda
```

```
[1] 0.0005971366
```

```r
m_best <- glmnet(x, y, alpha = 1, lambda = best_lambda)
m_best$beta
```

```
29 x 1 sparse Matrix of class "dgCMatrix"
                                         s0
(Intercept)                       .
player_age                       -0.0029767462
b_ab                             -0.0016283645
b_total_pa                        .
b_total_hits                      0.0043184447
b_home_run                        0.0117187722
AVG300Less than 300              -0.1242223268
batting_avg                       .
b_double                          0.0025726213
b_triple                          0.0043507975
HR40Less than 40                 -0.1082317461
b_strikeout                      -0.0007458214
b_walk                            0.0033086510
slg_percent                       0.0627793959
on_base_percent                  -0.3042934878
Position2B                        0.0395393326
Position3B                       -0.0346355227
PositionC                         0.0650214403
PositionCF                        0.0537743645
PositionCH                        0.0079932117
PositionDH                       -0.0385415609
PositionDNP                      -0.0066508435
```

```
PositionLF                       -0.0511356574
PositionPH                        0.0034339028
PositionRF                        0.0018072204
PositionSP                        0.1709299345
PositionSS                        0.0457111491
AVG300Less than 300:batting_avg   .
b_home_run:HR40Less than 40       0.0002396631
```

```
# LASSO Variable Selection Advanced Stats
y <- stats$All.Star
x <- model.matrix(All.Star ~ player_age + launch_angle_avg + sweet_spot_percent +
                    barrel + solidcontact_percent + flareburner_percent +
                    hard_hit_percent + avg_hyper_speed + z_swing_percent +
                    oz_swing_percent + meatball_swing_percent, data = stats)
m_lasso_cv <- cv.glmnet(x, y, alpha = 1)
best_lambda <- m_lasso_cv$lambda.min
best_lambda
```

```
[1] 0.005820234
```

```
m_best <- glmnet(x, y, alpha = 1, lambda = best_lambda)
m_best$beta
```

```
12 x 1 sparse Matrix of class "dgCMatrix"
                                  s0
(Intercept)             .
player_age             -0.0018096273
launch_angle_avg       -0.0001599061
sweet_spot_percent      .
barrel                  0.0080466625
solidcontact_percent   -0.0030825651
flareburner_percent    -0.0018259472
hard_hit_percent       -0.0018130937
avg_hyper_speed         .
z_swing_percent         .
oz_swing_percent        .
meatball_swing_percent -0.0021930005
```

```r
#Basic model
m1 <- glm(All.Star ~ player_age + b_ab +  b_total_hits +
                    b_double + b_triple + b_home_run + b_strikeout +
                    b_bb_percent + AVG300 + slg_percent +
                    on_base_percent + Position,
  data = stats,
  family = "binomial"
)
tidy(m1)
```

```
# A tibble: 24 x 5
   term                estimate std.error statistic p.value
   <chr>                  <dbl>     <dbl>     <dbl>   <dbl>
 1 (Intercept)         -5.96      2.78        -2.15   0.0319
 2 player_age          -0.0369    0.0570      -0.647  0.518
 3 b_ab                -0.00925   0.00925     -1.00   0.317
 4 b_total_hits         0.0450    0.0300       1.50   0.133
 5 b_double             0.0182    0.0411       0.443  0.657
 6 b_triple            -0.0469    0.116       -0.404  0.686
 7 b_home_run           0.0719    0.0524       1.37   0.170
 8 b_strikeout         -0.00188   0.00930     -0.203  0.839
 9 b_bb_percent         0.155     0.105        1.48   0.140
10 AVG300Less than 300 -0.617     0.768       -0.804  0.421
# ... with 14 more rows
```

```r
m1_aug <- augment(m1) %>%
  mutate(prob = exp(.fitted)/(1 + exp(.fitted)),
         pred_leg = ifelse(prob > 0.32, "All-Star", "Not All-Star"))
table(m1_aug$pred_leg, m1_aug$All.Star)
```

```
                0   1
  All-Star     22  30
  Not All-Star 410  24
```

```r
#Advanced model
m2 <- glm(All.Star ~ player_age + launch_angle_avg +
                    barrel + solidcontact_percent + flareburner_percent +
                    hard_hit_percent + meatball_swing_percent,
  data = stats,
```

```
  family = "binomial"
)
tidy(m2)
```

```
# A tibble: 8 x 5
  term                    estimate std.error statistic  p.value
  <chr>                      <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)               1.38      1.76       0.785 4.32e- 1
2 player_age              -0.0361    0.0468    -0.772 4.40e- 1
3 launch_angle_avg        -0.00881   0.0283    -0.312 7.55e- 1
4 barrel                   0.0852    0.0136     6.27  3.56e-10
5 solidcontact_percent    -0.0805    0.0943    -0.854 3.93e- 1
6 flareburner_percent     -0.0129    0.0379    -0.341 7.33e- 1
7 hard_hit_percent        -0.0256    0.0263    -0.974 3.30e- 1
8 meatball_swing_percent -0.0358    0.0162    -2.21  2.72e- 2
```

```
m2_aug <- augment(m2) %>%
  mutate(prob = exp(.fitted)/(1 + exp(.fitted)),
         pred_leg = ifelse(prob > 0.32, "All-Star", "Not All-Star"))
table(m2_aug$pred_leg, m2_aug$All.Star)
```

```
                
                 0   1
  All-Star      22  20
  Not All-Star 410  34
```

ARTICLE ABOUT BATTING ORDER STRATEGY: https://www.sportsbettingdime.com/guides/strategy/ba
order-sabermetrics/