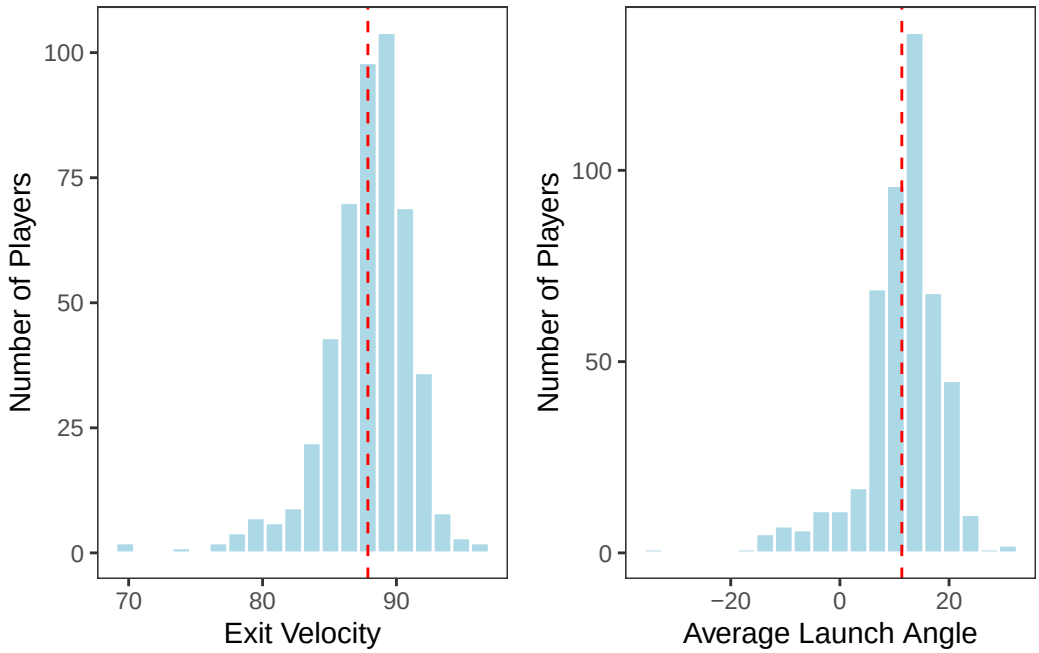
Exploring Advanced Offensive Performance Metrics in Baseball

Benji Gold and Sam Alkalay

Introduction:

Baseball, America's pastime, has a storied tradition that dates back well over 100 years. Since the 1850's, some form of statistic 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 improved, the statistics being tracked became more sophisticated.

In 2015 analytics in baseball took a giant leap. With the introduction of Statcast, teams were able to track novel metrics, such as a batted ball's exit velocity (the speed of the baseball as it comes off the bat, immediately after a batter makes contact) and launch angle (the angle at which a baseball is hit), as they provide more accurate and nuanced insights into a player's offensive abilities ("Statcast Search"). Below are histograms depicting the distribution of both statistics in the MLB.



As a result of these new metrics, teams across the league have incorporated these new metrics into their scouting and player development processes to gain a competitive edge in the game. Nevertheless, a question remains as to whether these metrics are truly effective in accurately predicting a player's potential. This research project intends to explore that idea through the use of a linear regression model to predict a player's on-base percentage, the percentage of at bats in which a player reaches a base, and slugging percentage, the total number of bases a player records per at-bat. To put these numbers into perspective, the average on-base and slugging percentages for a player in the dataset are 0.288 and 0.377 respectively (Appendix: 1.1). The research question of interest is:

How do advanced offensive performance metrics relate to a baseball player's on-base percentage and slugging percentage, and which metrics are the strongest predictors of these statistics?

To gather data for our study, we sourced a dataset from baseballsavant.com that included both traditional and Statcast statistics. The complete dataset can be accessed in the stats.csv file. Additionally, we supplemented the data with information on each player's position, salary, team, and all-star status, which was obtained from Baseball Prospectus, to provide a more comprehensive set of variables for our analysis. All packages used for data cleaning, processing, and analysis can be found in the appendix of our study (Appendix: 1.2).

Methodology:

Our original research question involved predicting whether or not a player was an all-star in 2019 based on their full season statistics for that season. Using our domain knowledge we

realized that whether or not a palyer is designated an all-star is not simply a measure of skill (which typical baseball statistics are designed to measure), but of popularity. This is true because of how all-stars are determined. The starters for the all-star team in each league are determined by fan vote, so a bad player who is popular with fans might still be selected an all-star. Because of this, our first decision was to create two new categorical variables, one for if a player hit more than 40 Home Runs (HR) in 2019 and one for if a player had a batting average (AVG) higher than 0.300 (Appendix: 2.1). These variables are meant to measure fame based off a semi-arbitrary criteria based on how fans view baseball. Our domain knowledge tells us that a player is generally considered elite by fans if they hit more than 40 HRs or have an AVG higher than 0.300. Home runs and batting average are sort of heuristic statistics that fans use to measure player quality and thus these effects should be included in our model.

Part of our research question was comparing models using advanced statistics to those that use more typical baseball statistics. We divided the stats in our data set into three categories: General information, typical stats, and advanced stats (Appendix: Data Dictionary). The general information is basic player background statistics like age, and position. The typical stats were statistics we could find on the basic page of a player's baseballreference page. The advanced stats were things we could only find on baseballsavant. All analyses discussed were performed twice, once on the general variables combined with the typical statistics and once with the general variables combined with the advanced stats.

Our response variable is a categorical variable, with a 1 if a player was an all-star and a 0 if the player-was not an all-star. Thus, we decided to perform a logistic regression on our data. Prior to running our logistic regression we performed a lasso on both subsets of the data (Appendix: 2.2). We choose to use a lasso because we wanted to optimize the predictive power of the model. We also worried about correlation among our predictors, which would make step-wise selection methods work poorly.

We ran the lasso on both sets of models and then ran logistic regressions, measuring our success by the sensitivity of the model with a 32% probability threshold for categorization (Appendix: 2.3). The typical statistics model had a sensitivity of 55.5% and the advanced stats model had a sensitivity of about 44.5%. Neither of these results were very encouraging. Because of this, we went back and thought about our model. We realized that many of our predictors were highly correlated, which would make the lasso work poorly for variable selection (which is obvious if you look at the lasso in our appendix and see that variables that our domain knowledge said were likely to be relevant such as AVG were selected out of the model). We also think that we had a problem with scarce data. Only 54 players in our data set were all-stars in 2019, which makes it difficult for our model to differentiate between all-stars and non-all stars, even with our low probability threshold for categorization as an all-star because it didn't have a large sample of all-stars to train on. Due to the low sensitivity and methodological problems of these models we decided to try something new instead.

After reevaluating our research question, we decided to instead attempt to predict slugging percentage and on-base percentage from the advanced statistics (we couldn't predict using the typical statistics because these response variables are highly correlated, and in fact directly

calculated from the typical statistics we measured). This is a valuable project because it can help us compare the outcomes we get from measuring performance through advanced metrics to the ones we get using typical statistics. We also used this to evaluate tenancies of where a player should be placed in a batting order, comparing the popular knowledge about batting order determinations, which are based on the typical box-score statistics, to what advanced Statcast knowledge says you should do.

Our methods for this were very similar to our methods for the all-star regressions, except we started by removing players who were on the Washington Nationals, so we could use them as a test data set for our model (Appendix: 2.4). We then used a lasso for the same reasons described above (fears of correlation among the predictors as well as our ability to optimize for predictive performance). We then ran a linear regression to predict slugging percentage and on-base percentage separately. we ran lassos to separately select variables for these two different regressions. Batting average and slugging percentage are continuous variables thus making linear regression work quiet well. We chose not to include any interactions because each of the statistics are calculated to represent different aspects of player performance and are thus not meant to be taken together. Our final models were these linear models trained on the advanced statistics, as defined in the data dictionary.

Following the linear regression we ran diagnostics such as making a QQ-Plot and Residual plot (which can be seen in the code chunk labelled statcast-obp,in the results section). Neither regression shows a clear pattern in the residual plots, but here is some worry about constant variance in the On-base percentage regression around the fitted value of 0.2. In general, however, we judged that the assumptions of linearity and constant variance seem to be valid for both plots. The QQ-plots are meant to test normality. The on-base percentage QQ-Plot deviates from normality, but this deviation is largely in the tails, thus we think this assumption is satisfied. The QQ-plot doesn't' deviate much for the slugging percentage regression and thus we think normality is fine for the slugging percentage regression. Independence is also satisfied because we are looking at the player level. If we were looking at typical statistics their might be some violation of independence based on the team the player played for but advanced statistics are designed such that they are independent. This is why exit-velocity is used rather than say, hit distance. Exit velocity cannot be altered by environmental conditions (and thus what team a player plays for), but distance a ball travels is altered by the elevation of a stadium. These linear regressions satisfy our assumptions and can thus be used for analysis.

Results:

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. The LASSO regression (included below) 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.

```
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

Best Lambda: 0.0007332828
```

After using these variables in the linear regression (included below), 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, aside from the ends, 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).

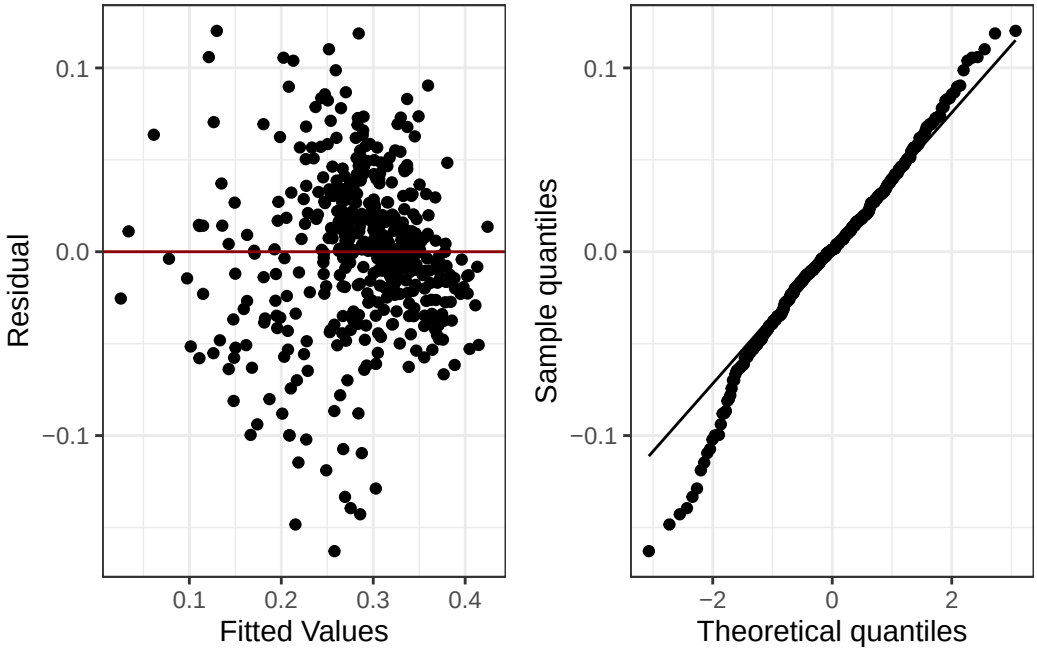
```
Call:
lm(formula = on_base_percent ~ launch_angle_avg + sweet_spot_percent +
    barrel + solidcontact_percent + flareburner_percent + hard_hit_percent +
    z_swing_percent + oz_swing_percent + meatball_swing_percent,
    data = rol_stats)

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
```



Among the significant variables in this linear regression, one of the most notable positive predictors was flareburner_percent. Holding all other predictors constant, 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. Holding all other predictors constant, 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 the LASSO regression for 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. The code chunks and outputs are not visualized here because they are the same as the above code chunks, just with a LASSO regression, linear model, and assumption graphs for slugging percentage instead. All of the code and associated outputs for these models can be found in appendix section 3.1.

Among the significant variables in slugging percentage model, the most notable predictors were sweet_spot_percent and barrel. Holding all other predictors constant, 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.

After determining that the models fit the data well, we utilized them to forecast the on-base and slugging percentages for all Washington Nationals players. Using these projections, we employed a Sabermetrics batting order approach to create an optimal batting order for the team ("How Sabermetrics Influence Baseball"). The output of the batting order, including a pinch hitter at 10, is below, and the code used is included in the appendix under section 3.2.

```
# A tibble: 10 x 5
# Groups:   Position [10]
   last_name    first_name  Position Predicted_OBP Predicted_SLG
   <chr>        <noquote>   <chr>            <dbl>         <dbl>
 1 Soto          Juan       LF               0.406         0.577
 2 Rendon        Anthony    3B               0.416         0.600
 3 Turner        Trea       SS               0.321         0.441
 4 Kendrick III  Howie      1B               0.350         0.502
 5 Suzuki        Kurt       C                0.334         0.439
 6 Dozier        Brian      2B               0.338         0.417
 7 Eaton         Adam       RF               0.313         0.405
 8 Taylor        Michael A. CF               0.308         0.373
 9 Strasburg     Stephen    SP               0.249         0.269
10 Stevenson     Andrew     PH               0.264         0.326
```

Discussion:

In this study, we aimed to develop predictive models for on base percentage and slugging percentage using advanced statistics in the MLB. Our results showed that the best models for this were linear regressions, and the selected variables were significant predictors of on base and slugging percentage in the 2019 MLB season.

Our findings suggest that advanced statistics can be used to predict player performance and improve team strategies. For example, our models highlighted the importance of flare-burner_percent and oz_swing_percent in predicting on base percentage. These findings suggest that players who can hit a variety of pitches are more likely to perform better, while those who swing at bad pitches are less likely to reach base. These insights can be useful for teams when selecting players or developing strategies to optimize their lineups. Moreover, our models revealed that sweet_spot_percent and barrel were the most notable predictors of slugging percentage. These findings suggest that players who can consistently hit the ball at certain angles and on a specific part of their bat are more likely to generate more extra base hits. Therefore, teams can use this information to identify players who are more likely to produce high slugging percentages, which is a key factor in offensive success in baseball.

Despite the promising results of our study, there are some limitations to our analysis. First, our data only covers the 2019 season, and it is possible that our models may not be generalizable to other seasons or contexts. Furthermore, this is a generalized model that does not take into account unique player and stadium advantages. For example, certain players hit far better against right handed pitchers than left handed pitchers, and a team's optimal lineup should change based on the type of pitcher they are facing. Also, stadiums at higher altitudes, like the one in Colorado, favor more powerful hitters because its altitude is higher, which makes the baseball go approximately 10% further (Welling). This would also warrant a game specific change in the lineup.

However, 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.

Works Cited

"How Sabermetrics Influence Baseball Batting Order Strategy." Sports Betting Dime, https://www.sportsbettingdime.com/guides/strategy/batting-order-sabermetrics/.

LEE, Choonghwan. "History of Baseball Statistics." Medium, Medium, 7 Mar. 2018, https://medium.com/@190654/history-of-baseball-statistics-6f2b13f5de20.

"Statcast Search." Baseballsavant.com, https://baseballsavant.mlb.com/statcast_search.

Welling, Craig. "Did Humidifying the Baseball Decrease the Number of Homers at Coors Field"https://ww2.amstat.org/mam/2010/essays/PennBaseball.pdf.

Appendix

1.1:

```
# Average on-base and slugging percentage calculations
mean_obp <- mean(stats$on_base_percent)
mean_slg <- mean(stats$slg_percent)

round(mean_obp, 3)
```

```
[1] 0.288
```

```
round(mean_slg, 3)
```

```
[1] 0.377
```

1.2:

```
# Packages and data
library(tidyverse)
library(tidymodels)
library(glmnet)
library(caret)
library(MASS)
library(lme4)
library(gridExtra)
stats <- read.csv("data/stats.csv")
stats <- replace(stats, stats =="", NA)
```

2.1:

```
# Mutate batting average and home runs
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"),
                pitcher = case_when(Position == "SP" ~ "Yes", TRUE ~ "No"))
```

2.2:

```
# LASSO Variable Selection Basic Stats
y <- stats$All.Star
x <- model.matrix(All.Star ~ player_age + b_ab + b_total_pa + b_total_hits +
                    b_home_run + AVG300 * batting_avg +b_double + b_triple +
                    b_home_run * HR40 + b_strikeout + b_walk + batting_avg +
                    slg_percent + on_base_percent, data = stats)
m_lasso_cv <- cv.glmnet(x, y, alpha = 1)
best_lambda <- m_lasso_cv$lambda.min
best_lambda
```

```
[1] 0.0005440887
```

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

```
17 x 1 sparse Matrix of class "dgCMatrix"
                                           s0
(Intercept)                 .
player_age                 -0.0030009968
b_ab                       -0.0020576289
b_total_pa                  .
b_total_hits                0.0054583214
b_home_run                  0.0111409939
AVG300Less than 300        -0.1141651293
batting_avg                 0.5833023902
b_double                    0.0017330190
b_triple                    0.0055612043
HR40Less than 40           -0.1206692068
b_strikeout                -0.0005747544
b_walk                      0.0040288116
slg_percent                 .
on_base_percent            -1.1106763995
AVG300Less than 300:batting_avg  .
b_home_run:HR40Less than 40      0.0006324729
```

```
# 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
```

2.3:

```
# Basic all-star 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
```

```
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
```

```
#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
```

2.4:

```
# Seperate Nationals and rest of league
nationals_stats <- stats |>
  filter(Team == "WAS")

rol_stats <- stats |>
  filter(Team != "WAS")
```

3.1:

```
# 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
```

```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        0.0004048935
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
```

```r
# Rol slugging percentage linear model
m4 <- lm(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)
tidy(m4)
```

```
# A tibble: 11 x 5
   term                   estimate std.error statistic  p.value
   <chr>                     <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)            -0.0791    0.0327      -2.41  1.61e- 2
 2 launch_angle_avg        0.000450  0.000561     0.803 4.22e- 1
 3 sweet_spot_percent      0.00386   0.000683     5.65  2.79e- 8
 4 barrel                  0.00358   0.000277    12.9   9.84e-33
 5 solidcontact_percent    0.00218   0.00139      1.57  1.16e- 1
 6 flareburner_percent     0.00318   0.000819     3.88  1.20e- 4
 7 hard_hit_percent        0.00100   0.00105      0.951 3.42e- 1
 8 avg_hyper_speed         0.00673   0.00605      1.11  2.67e- 1
 9 z_swing_percent         0.00133   0.000713     1.86  6.35e- 2
10 oz_swing_percent       -0.000882  0.000524    -1.68  9.31e- 2
11 meatball_swing_percent  0.000728  0.000410     1.78  7.62e- 2
```

```r
summary(m4)$adj.r.squared
```

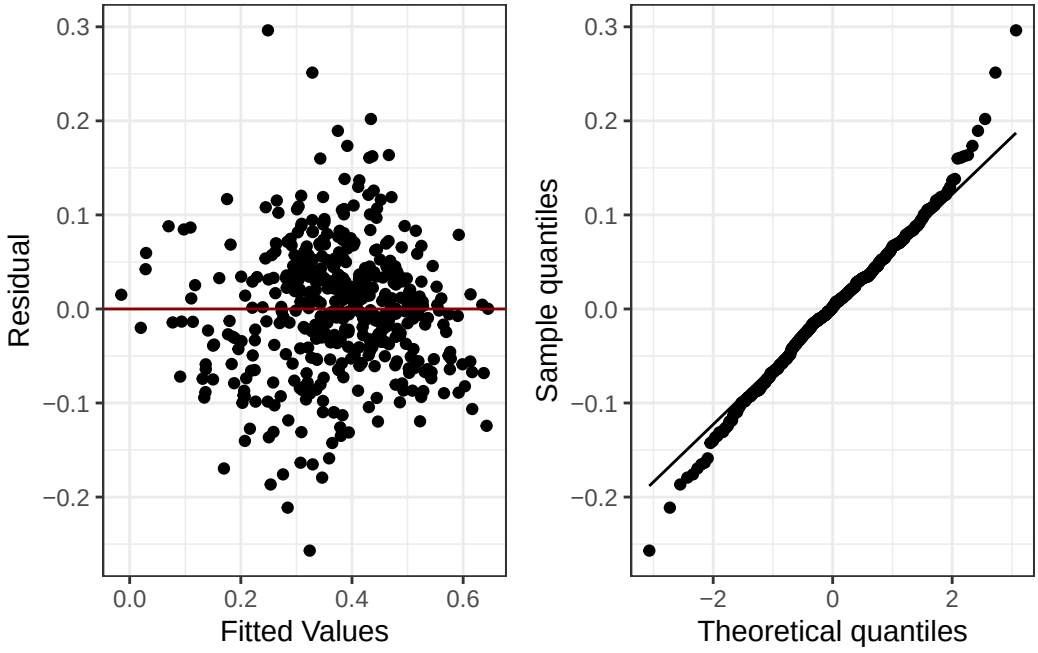
```
[1] 0.7286873
```

```r
m4_aug <- augment(m4)

m4_1 <- ggplot(m4_aug, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "darkred") +
  labs(x = "Fitted Values",
       y = "Residual") +
  theme_bw()

m4_2 <- ggplot(m4_aug, aes(sample = .resid)) +
  stat_qq() +
  stat_qq_line() +
  theme_bw() +
  labs(x = "Theoretical quantiles",
       y = "Sample quantiles")

grid.arrange(m4_1, m4_2, ncol = 2)
```



3.2:

```
# Predict Nationals stats
pred_obp <- predict(m3, nationals_stats)
pred_slg <- predict(m4, nationals_stats)

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

# Create optimal lineup using one of each position
nationals_pred_stats <- nationals_stats |>
  group_by(Position) |>
  slice(which.max(Predicted_OBP))

# Group important columns for output
nationals_pred_stats <- nationals_pred_stats[ ,c("last_name","first_name","Position",
                                                 "Predicted_OBP", "Predicted_SLG")]

# Manually change batting order based on Sabermetrics
last_name_order <- c("Soto", "Rendon", "Turner", "Kendrick III", "Suzuki", "Dozier", "Eato
                     "Taylor", "Strasburg", "Stevenson")

nationals_pred_stats <- nationals_pred_stats[match(last_name_order,
                                                   nationals_pred_stats$last_name),]

# Remove quotation marks around the first name column
nationals_pred_stats$first_name <- noquote(nationals_pred_stats$first_name)

# Display results
print(nationals_pred_stats)
```

Data Dictionary:

General Information:

last_name: The last name of the player.

first_name: The first name of the player.

player_id: The identifying number of the player.

year: The year of the season. player_age: The age of the player.

position: The position of the player.

All.Star: Whether the player is an all-star.

Typical Statistics:

b_ab: The number of at bats for the player.

b_total_pa: The number of total plate appearances by the player.

b_total_hits: The number of total hits by the player.

b_double: The number of doubles hit by the player.

b_triple: The number of triples hit by the player.

b_home_run: The number of home runs hit by the player.

b_strikeout: The number of times a player strikes out.

b_walk: The number of times a player gets walked.

b_k_percent: The player's strikeout percentage.

b_bb_percent: The player's walk percentage.

batting_avg: The player's batting average.

slg_percent: The player's slugging percentage.

on_base_percent: The player's on base percentage.

Advanced Statistics:

xba: The player's expected batting average.

xslg: The player's expected slugging percentage.

woba: The player's weighted on base average.

xwoba: The player's expected weighted on base average.

xobp: The player's expected on base percentage.

xiso: The player's expected isolated power.

xslgdiff: The player's expected slugging differential.

exit_velocity_avg: The player's average exit velocity.

launch_angle_avg: The player's average launch angle.

sweet_spot_percent: The player's sweet spot percentage.

barrel: The player's barrel percentage.

solidcontact_percent: The player's percentage of solid contact.

flareburner_percent: The player's second-best batted ball types for base hits percentage.

hard_hit_percent: The player's hard hit percentage.

avg_hyper_speed: The player's average bat speed.

z_swing_percent: The player's swing percentage on pitches inside the zone.

oz_swing_percent: The player's swing percentage on pitches outside the zone.

meatball_swing_percent: The player's swing percentage on meatball pitches.