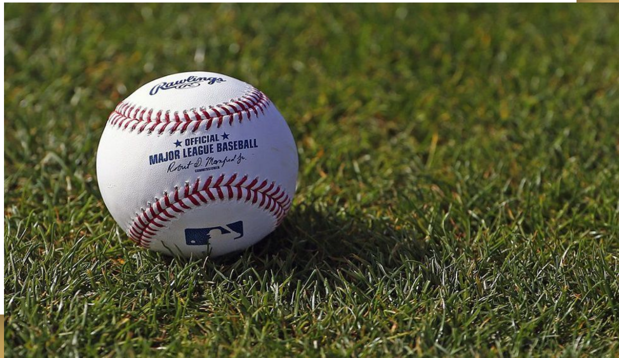
Final Project Presentation

MGSC 310

Ben Ziv, Brad Shimabuku, Lucas Sotto, Anne Schmidt



Introduction: Data Set and Motivation

- Data Set:
  - Lahman Baseball Data Set
  - Batting statistics from the 2019 MLB season

- Motivation:
  - Two of four group members play for the university
  - Interested in how models can impact MLB hitters

http://www.seanlahman.com/baseball-archive/statistics/

Introduction: Outcome and Methods

- Outcome Variable: Batting Average (AVG)
  - Hitting statistic used to measure average number of hits per 10 at bats
  - Gives a general idea of how well a player is doing
    - .300 AVG is universal for “strong” hitter (MLB average in 2019 = .252)
- Methods:

1. Linear Regression
   a. Predicting AVG using continuous variables
2. Decision Tree
   a. Predicting AVG using continuous variables
3. Lasso
   a. Minimizing/zeroing out variables that model finds to be insignificant

```
batting <- read.csv("datasets/Batting.csv")
batting_clean <- batting %>% filter(yearID == "2019") %>% mutate(AVG = H/AB) %>%
  filter(AB > 100)


dim(batting_clean)
```

```
[1] 444  23
```

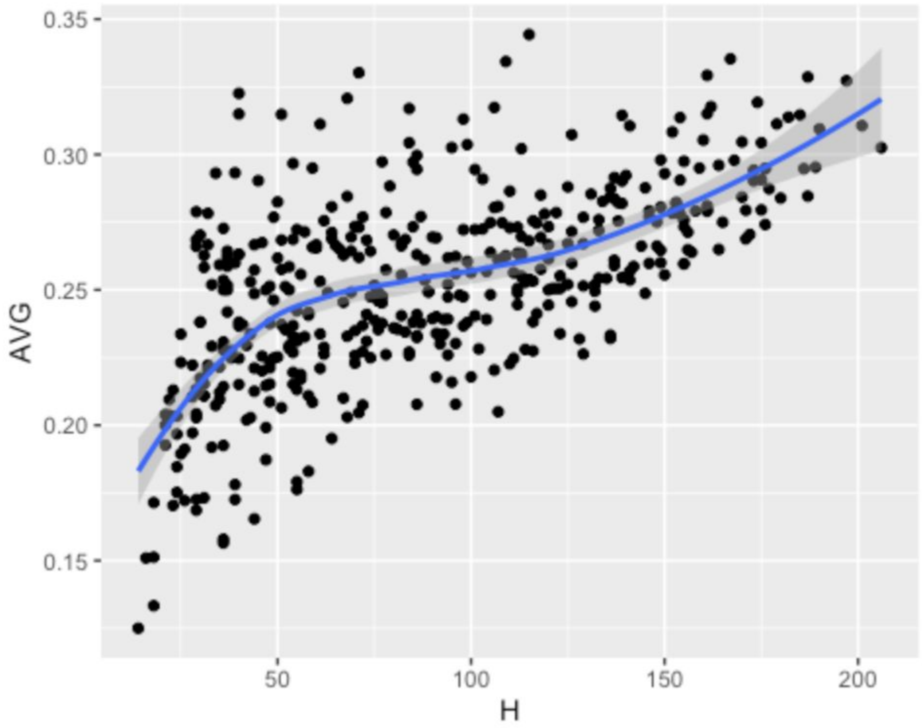
Introduction: Data Glimpse

```
batting_clean %>% glimpse()
```

```
Rows: 444
Columns: 23
$ playerID <chr> "abreujo02", "acunaro01", "adamewi01", "adamsma01", "adriaeh01", "a…
$ yearID   <int> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2…
$ stint    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ teamID   <chr> "CHA", "ATL", "TBA", "WAS", "MIN", "MIL", "ARI", "BAL", "ATL", "MIA…
$ lgID     <chr> "AL", "NL", "AL", "NL", "AL", "NL", "NL", "AL", "NL", "NL", "AL", "…
$ G        <int> 159, 156, 152, 111, 84, 94, 158, 139, 160, 130, 89, 130, 161, 67, 1…
$ AB       <int> 634, 626, 531, 310, 202, 222, 556, 524, 640, 431, 231, 339, 597, 21…
$ R        <int> 85, 127, 69, 42, 34, 26, 79, 62, 102, 44, 30, 41, 103, 23, 89, 58, …
$ H        <int> 180, 175, 135, 70, 55, 50, 141, 160, 189, 113, 53, 80, 155, 39, 149…
$ X2B      <int> 38, 22, 25, 14, 8, 9, 33, 21, 43, 14, 9, 11, 30, 6, 27, 26, 33, 32,…
$ X3B      <int> 1, 2, 1, 0, 3, 0, 6, 2, 8, 1, 3, 1, 2, 0, 3, 0, 1, 0, 4, 0, 1, 2, 1…
$ HR       <int> 33, 41, 20, 20, 5, 8, 19, 12, 24, 18, 4, 12, 53, 7, 31, 27, 20, 18,…
$ RBI      <int> 123, 101, 52, 56, 22, 34, 82, 51, 86, 57, 27, 32, 120, 27, 74, 78, …
$ SB       <int> 2, 37, 4, 0, 0, 0, 8, 4, 15, 4, 8, 2, 1, 0, 6, 0, 5, 17, 31, 7, 8, …
$ CS       <int> 2, 9, 2, 0, 2, 0, 2, 4, 4, 4, 2, 1, 0, 1, 5, 0, 1, 5, 8, 0, 5, 2, 2…
$ BB       <int> 36, 76, 46, 20, 20, 31, 52, 16, 54, 22, 11, 16, 72, 29, 41, 52, 44,…
$ SO       <int> 152, 188, 153, 115, 40, 59, 113, 50, 112, 154, 53, 62, 183, 53, 82,…
$ IBB      <int> 4, 4, 1, 1, 1, 0, 2, 1, 6, 1, 1, 4, 6, 2, 0, 4, 1, 0, 1, 2, 5, 11, …
$ HBP      <int> 13, 9, 3, 2, 6, 2, 4, 4, 4, 10, 9, 1, 21, 1, 3, 2, 14, 3, 4, 2, 1, …
$ SH       <int> 0, 0, 3, 0, 2, 0, 1, 3, 0, 0, 4, 5, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0…
$ SF       <int> 10, 1, 1, 1, 4, 4, 12, 3, 4, 2, 1, 2, 3, 2, 3, 2, 3, 2, 10, 2, 6, 8…
$ GIDP     <int> 24, 8, 9, 7, 2, 11, 15, 9, 2, 12, 3, 8, 13, 8, 19, 9, 15, 12, 16, 5…
$ AVG      <dbl> 0.2839117, 0.2795527, 0.2542373, 0.2258065, 0.2722772, 0.2252252, 0…
```

Hits vs. Batting Average

```
ggplot(batting_clean, aes(x = H, y = AVG)) + geom_point() + geom_smooth()
```



Model 1: Linear Regression

```
mod1 <- lm(AVG ~ G + AB + R + X2B + X3B + HR + RBI + SB + BB + SO + HBP,
           data = batting_train)
summary(mod1)
```

```

Call:
lm(formula = AVG ~ G + AB + R + X2B + X3B + HR + RBI + SB + BB +
    SO + HBP, data = batting_train)

Residuals:
      Min        1Q    Median        3Q       Max
-0.072396 -0.014272  0.000102  0.016021  0.060166

Coefficients:
               Estimate  Std. Error t value             Pr(>|t|)
(Intercept)  0.24075850  0.00458752  52.481 < 0.0000000000000002 ***
G           -0.00008784  0.00011070  -0.794              0.42807
AB          -0.00005772  0.00004341  -1.330              0.18468
R            0.00110886  0.00024171   4.588         0.0000066029 ***
X2B          0.00099519  0.00035919   2.771              0.00594 **
X3B          0.00120806  0.00098395   1.228              0.22050
HR          -0.00057894  0.00048139  -1.203              0.23007
RBI          0.00065210  0.00019790   3.295              0.00110 **
SB          -0.00006540  0.00027217  -0.240              0.81028
BB          -0.00053854  0.00013173  -4.088         0.0000559336 ***
SO          -0.00041577  0.00007128  -5.833         0.0000000141 ***
HBP         -0.00072349  0.00044027  -1.643              0.10137
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.02492 on 299 degrees of freedom
Multiple R-squared:  0.4994,	Adjusted R-squared:  0.481
F-statistic: 27.12 on 11 and 299 DF,  p-value: < 0.00000000000000022
```

Prediction

```
batting_preds_train <- predict(mod1)
head(batting_preds_train)
```

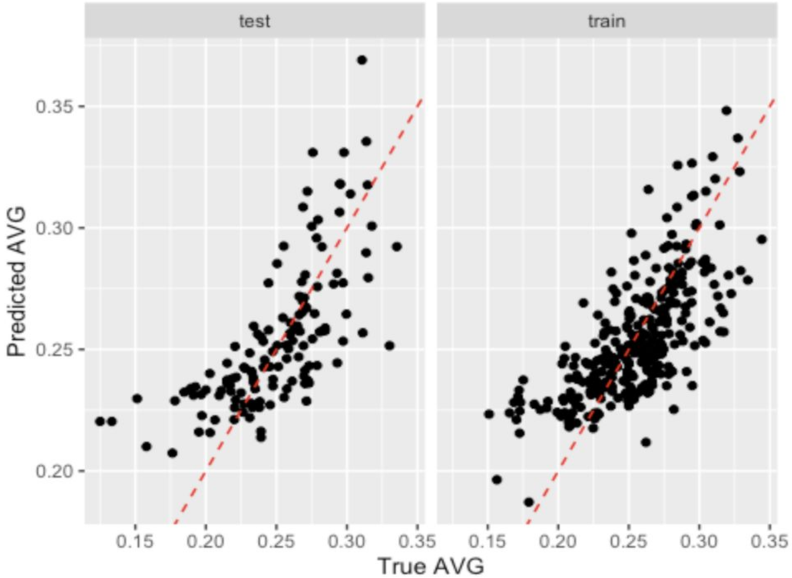
```
        1         2         3         4         5         6
0.2924616 0.2701623 0.2308719 0.2385288 0.2507172 0.2323434
```

```
batting_preds_test <- predict(mod1, newdata = batting_test)
head(batting_preds_test)
```

```
        9        13        14        18        22        24
0.3179914 0.2560936 0.2288856 0.2922932 0.3176424 0.2314401
```

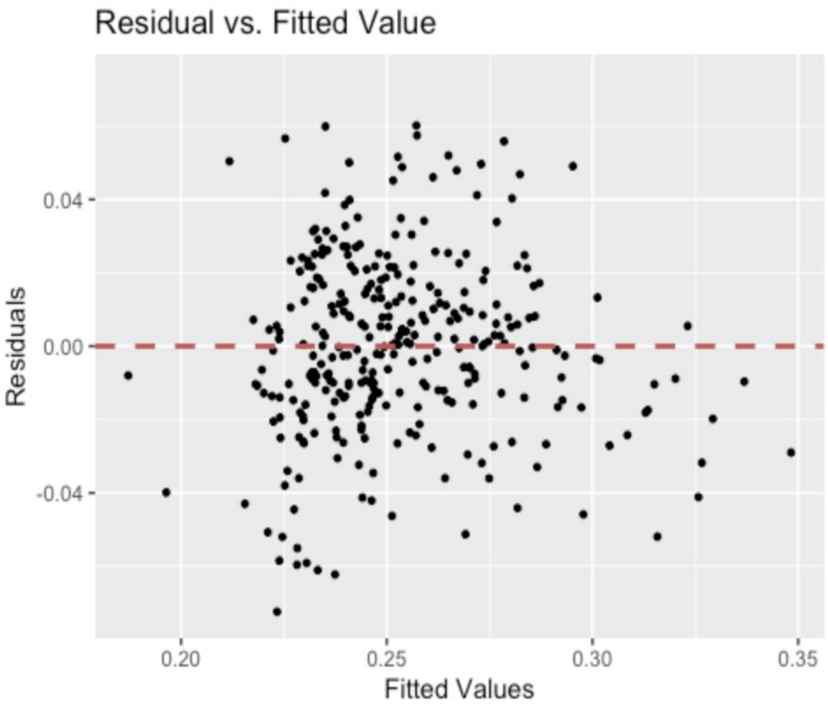
Predicted vs. True

```
preds_df <- data.frame(
  AVG = c(batting_train$AVG,batting_test$AVG),
  preds = c(batting_preds_train, batting_preds_test),
  type = c(rep("train", length(batting_preds_train)),
           rep("test", length(batting_preds_test)))
)

ggplot(preds_df, aes(x = AVG, y = preds)) + geom_point(alpha = 1) +
    geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
    facet_wrap(~type) +
    labs(x = "True AVG", y = "Predicted AVG")
```

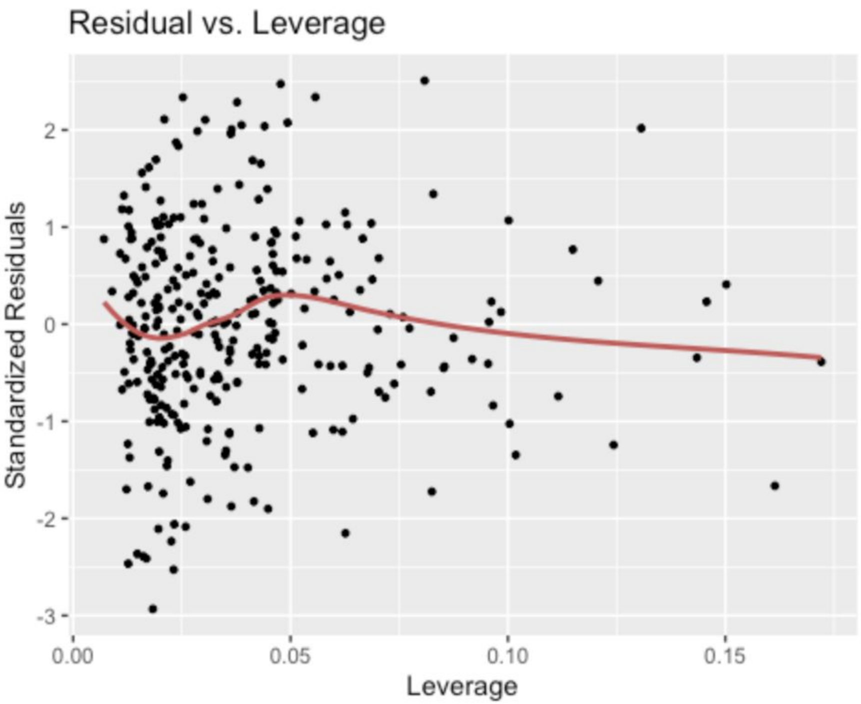


Residuals

```
gg_resfitted(mod1)
```

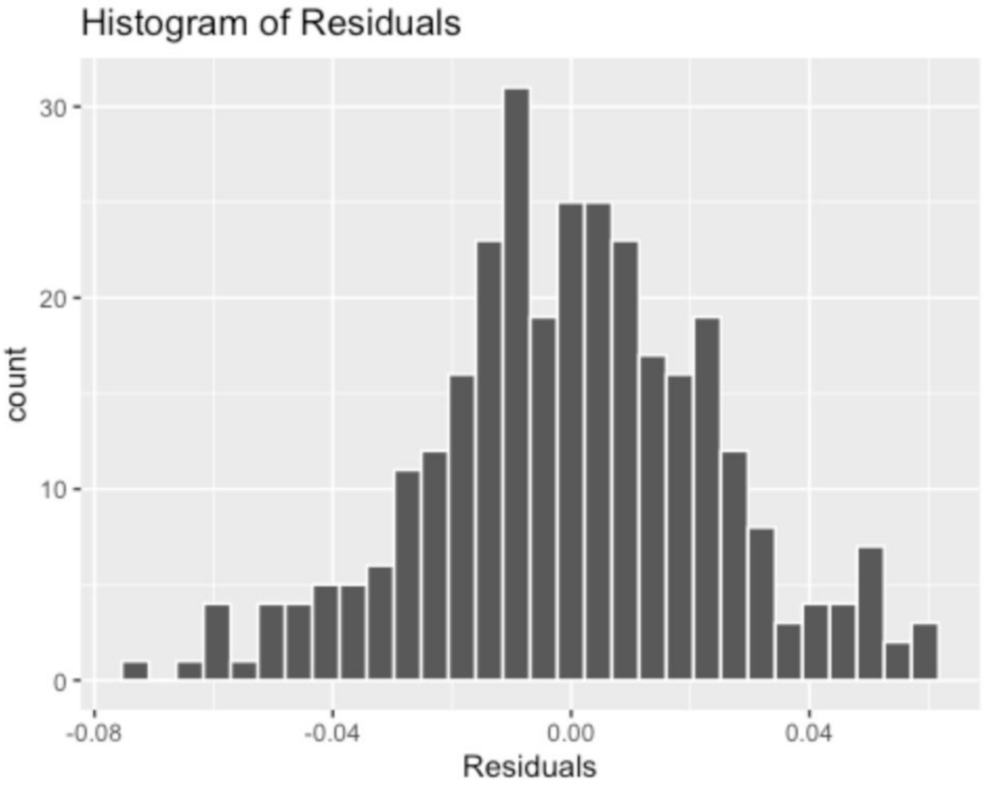



```
gg_resleverage(mod1)
```



Histogram of Residuals

```
gg_reshist(mod1)
```



RMSE

```
#RMSE for the train
RMSE(pred = batting_preds_train, obs = batting_train$AVG)
```

```
[1] 0.02443209
```

```
#RMSE for the test
RMSE(pred = batting_preds_test, obs = batting_test$AVG)
```

```
[1] 0.02817837
```

R^2

```
R2(pred = batting_preds_train, obs = batting_train$AVG)
```

```
[1] 0.4994023
```

```
R2(pred = batting_preds_test, obs = batting_test$AVG)
```

```
[1] 0.4856554
```

Model 1 Conclusions

- Because of low RMSE scores and decent R2 scores, we can say that our model was fairly accurate with minimal overfitting.
- Heteroskedasticity in our model showed that some error may have been present but, there wasn't error all over the place.
- Runs (R), doubles (X2B), runs batted in (RBI), walks (BB), and strikeouts (SO), were the most significant variables in predicting AVG based on the p-value that was associated with them.

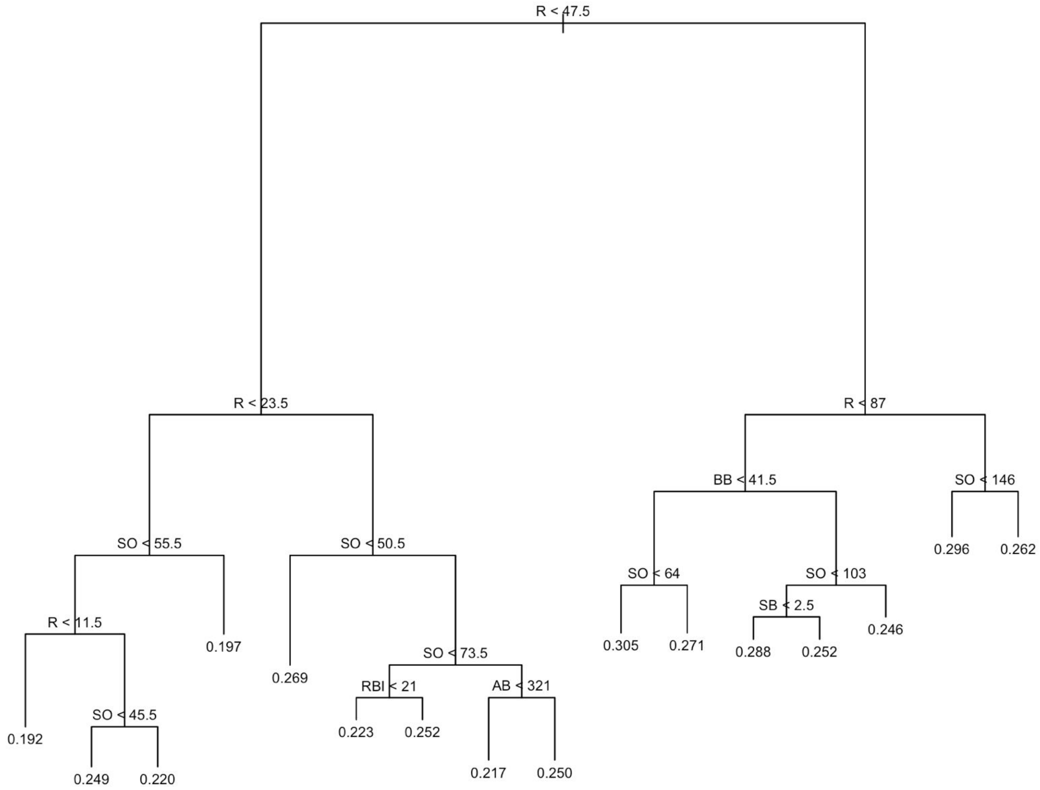
Model 2: Decision Tree

```r
# Train the tree
# outcome: AVG
library(tree)
tree_batting <- tree(AVG ~ G + AB + R + RBI + SB + BB + SO + HBP,
                     data = batting_train)

# Report the results
summary(tree_batting)
```

```
Regression tree:
tree(formula = AVG ~ G + AB + R + RBI + SB + BB + SO + HBP, data = batting_train)
Variables actually used in tree construction:
[1] "R"   "SO"  "RBI" "AB"  "BB"  "SB"
Number of terminal nodes:  16
Residual mean deviance:  0.0005169 = 0.1525 / 295
Distribution of residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.0514300 -0.0146800  0.0007775  0.0000000  0.0136700  0.0657000
```

Plot

```
plot(tree_batting)
text(tree_batting, digits = 3, pretty = 0, cex = 0.7)
```



Prediction

```
preds_tree_train <- predict(tree_batting)
preds_tree_test <- predict(tree_batting, newdata = batting_test)

head(preds_tree_train)
```

```
        1         2         3         4         5         6 
0.2708530 0.2619508 0.2456463 0.2168399 0.2691128 0.2521637 
```

```
head(preds_tree_test)
```

```
        9        13        14        18        22        24 
0.2961613 0.2619508 0.2198139 0.2708530 0.2961613 0.1919604 
```

RMSE

```
# Measure RMSE test and train
library(caret)
RMSE(preds_tree_train, batting_train$AVG)
```

```
[1] 0.02214345
```

```
RMSE(preds_tree_test, batting_test$AVG)
```

```
[1] 0.0301597
```

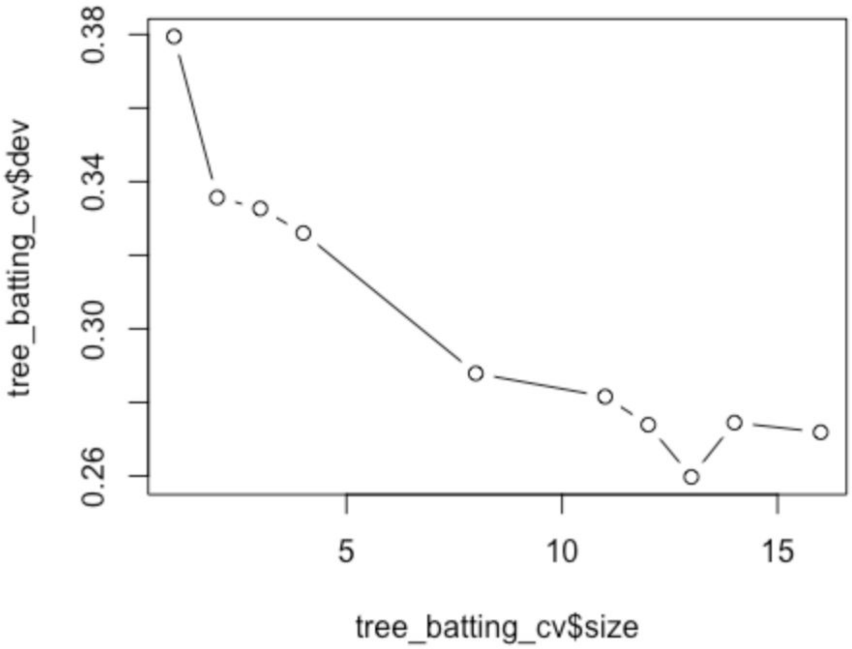
Cross Validation

```
# cross-validation to find best tree size
tree_batting_cv <- cv.tree(tree_batting)

# report the results
print(tree_batting_cv)
```

```
$size
 [1] 16 14 13 12 11  8  4  3  2  1

$dev
 [1] 0.2718961 0.2745011 0.2597444 0.2739180 0.2815733 0.2878846 0.3260313 0.3326844 0.3356904 0.3794370

$k
 [1]        -Inf 0.003720590 0.005359121 0.006776363 0.007660552 0.008055702 0.014467667 0.018585907 0.023868008
0.066415457

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

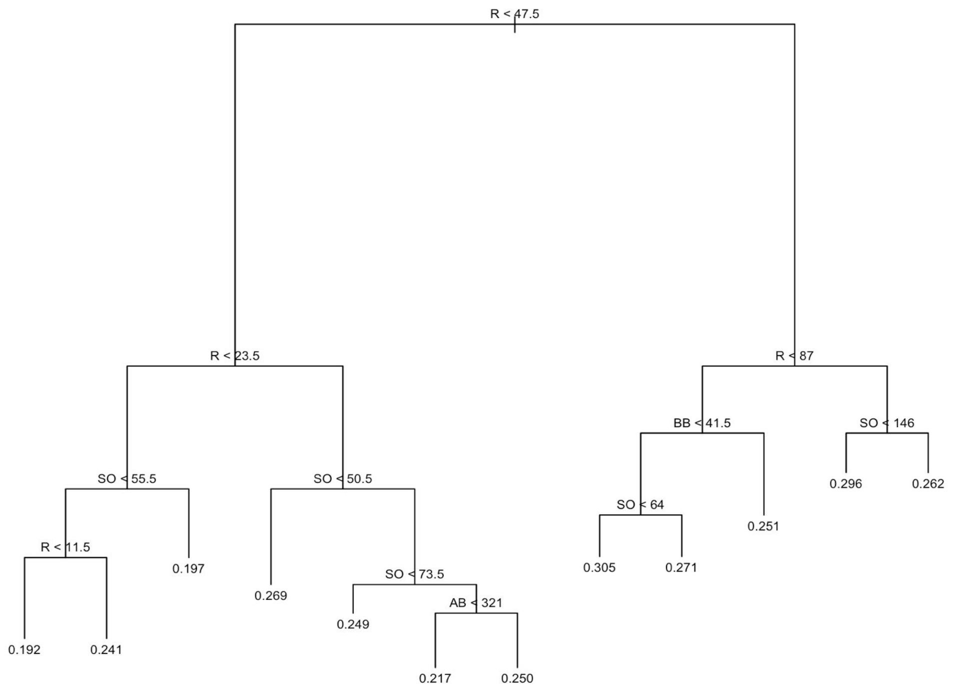
Optimal Number of Splits

```
# plot the error vs tree
plot(tree_batting_cv$size, tree_batting_cv$dev,
     type = 'b')
```



Pruned Tree

```r
# prune the tree by the best size
pruned_tree <- prune.tree(tree_batting, best = 12)

# plot the pruned tree (plot for the best sub-tree)
plot(pruned_tree)
text(pruned_tree, digits = 3, pretty = 0, cex = 0.7)
```

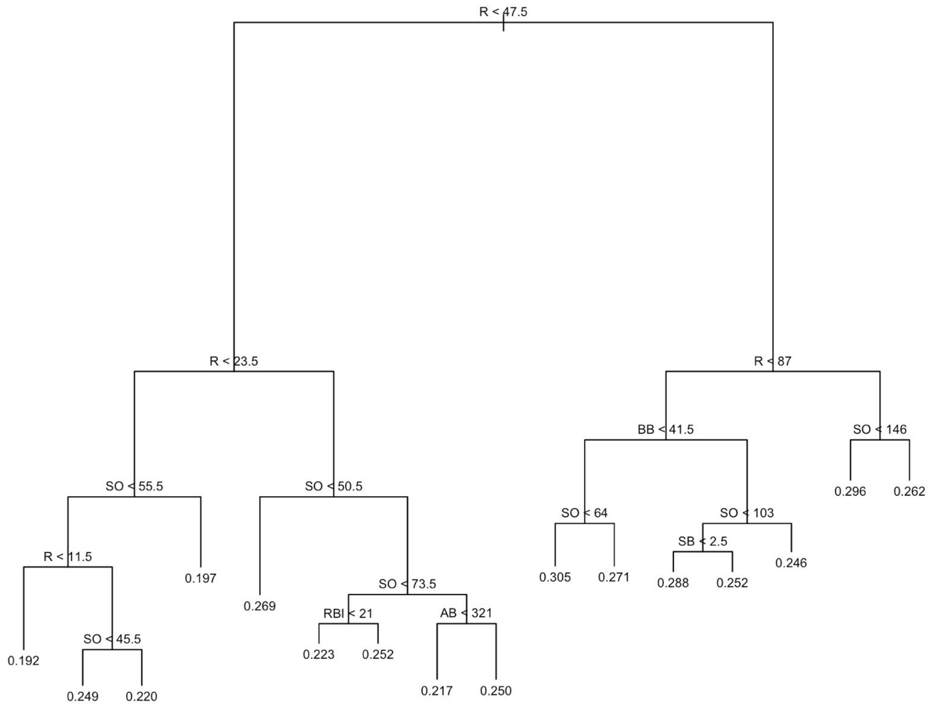



```r
# Predict the test
preds_tree_pruned <- predict(pruned_tree, newdata = batting_test)

RMSE(preds_tree_pruned, batting_test$AVG)
```

```
[1] 0.02872111
```

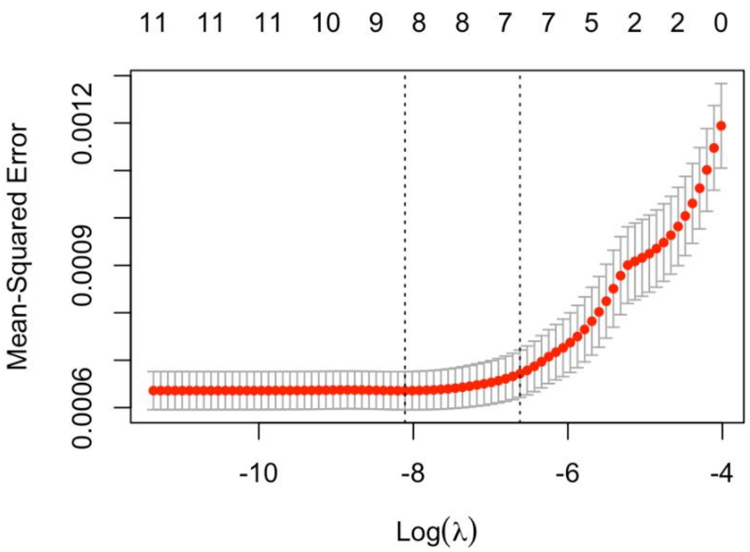
Model 2 Conclusions

- 12 splits gave lowest RMSE
- Both tree and pruned tree RMSE's were low and close to each other, indicating an accurate model with minimal overfitting.
- Based off pruned tree we can conclude:
  - Runs (R) is the most important split followed by strikeouts (SO) and walks (BB)
  - Players with:
    - R > 47.5, R > 87, SO < 146 == 0.296 AVG
    - R < 47.5, R < 23.5, SO < 55.5, R < 11.5 == 0.192 AVG



Model 3: Lasso Regression

```
lasso_bat <- cv.glmnet(AVG ~ G + AB + R + X2B + X3B + HR + RBI + SB + BB + SO + HBP,
data = batting_train,
alpha = 1)

plot(lasso_bat)
```



Suggested Lambda Values

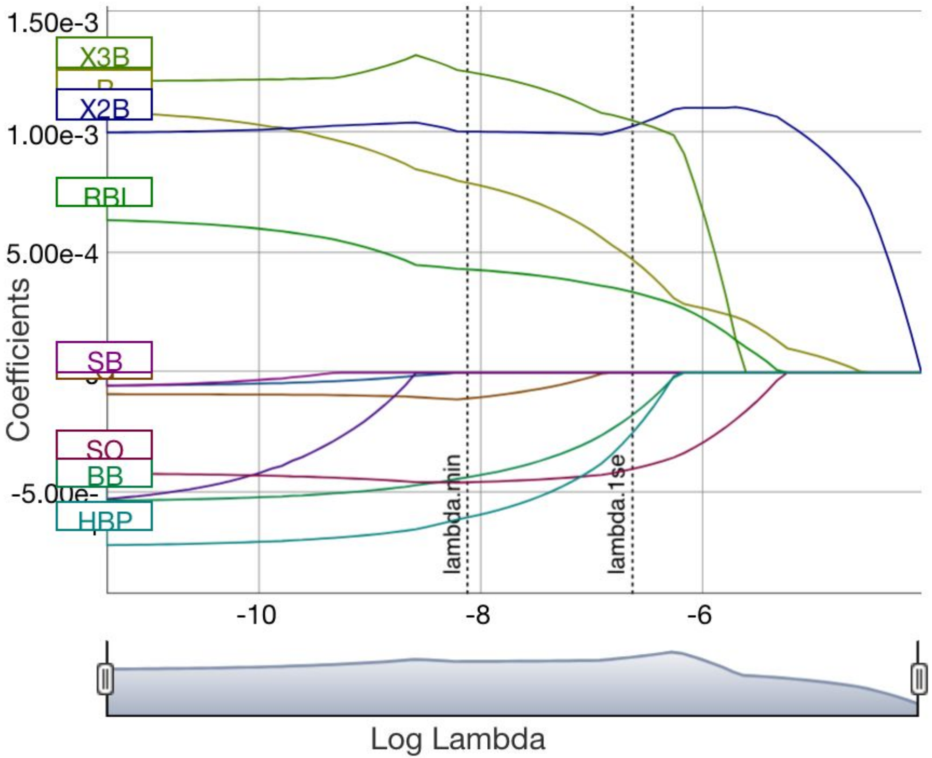
```r
#suggested values of lambda
print(lasso_bat$lambda.min)
## [1] 0.0003010214
print(lasso_bat$lambda.1se)
## [1] 0.001333712
```

```r
# to examine the coefficients we must say what value of
# lambda we want to use.
coef(lasso_bat, s = lasso_bat$lambda.min) %>%
round(6)
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept)  0.240010
## G           -0.000108
## AB           .
## R            0.000791
## X2B          0.001004
## X3B          0.001253
## HR           .
## RBI          0.000430
## SB           .
## BB          -0.000435
## SO          -0.000457
## HBP         -0.000603
# print coefficient using lambda.1se
coef(lasso_bat, s = lasso_bat$lambda.1se) %>%
round(6)
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept)  0.234622
## G            .
## AB           .
## R            0.000471
## X2B          0.001026
## X3B          0.001050
## HR           .
## RBI          0.000335
## SB           .
## BB          -0.000176
## SO          -0.000401
## HBP         -0.000251
```

```r
lasso_coef_bat <- data.frame(
lasso_min = coef(lasso_bat, s = lasso_bat$lambda.min) %>%
round(6) %>% as.matrix() ,
lasso_1se = coef(lasso_bat, s = lasso_bat$lambda.1se) %>%
round(6) %>% as.matrix()
) %>% rename(lasso_min = 1, lasso_1se = 2)

print(lasso_coef_bat)
##              lasso_min lasso_1se
## (Intercept)   0.240010  0.234622
## G            -0.000108  0.000000
## AB            0.000000  0.000000
## R             0.000791  0.000471
## X2B           0.001004  0.001026
## X3B           0.001253  0.001050
## HR            0.000000  0.000000
## RBI           0.000430  0.000335
## SB            0.000000  0.000000
## BB           -0.000435 -0.000176
## SO           -0.000457 -0.000401
## HBP          -0.000603 -0.000251
```

Coefficient Shrinkage Path

```r
library(coefplot)
coefpath(lasso_bat)
```



Non-zero predictors

```
lasso_coef_bat %>% head()
```

	lasso_min <dbl>	lasso_1se <dbl>
(Intercept)	0.240010	0.234622
G	-0.000108	0.000000
AB	0.000000	0.000000
R	0.000791	0.000471
X2B	0.001004	0.001026
X3B	0.001253	0.001050

6 rows

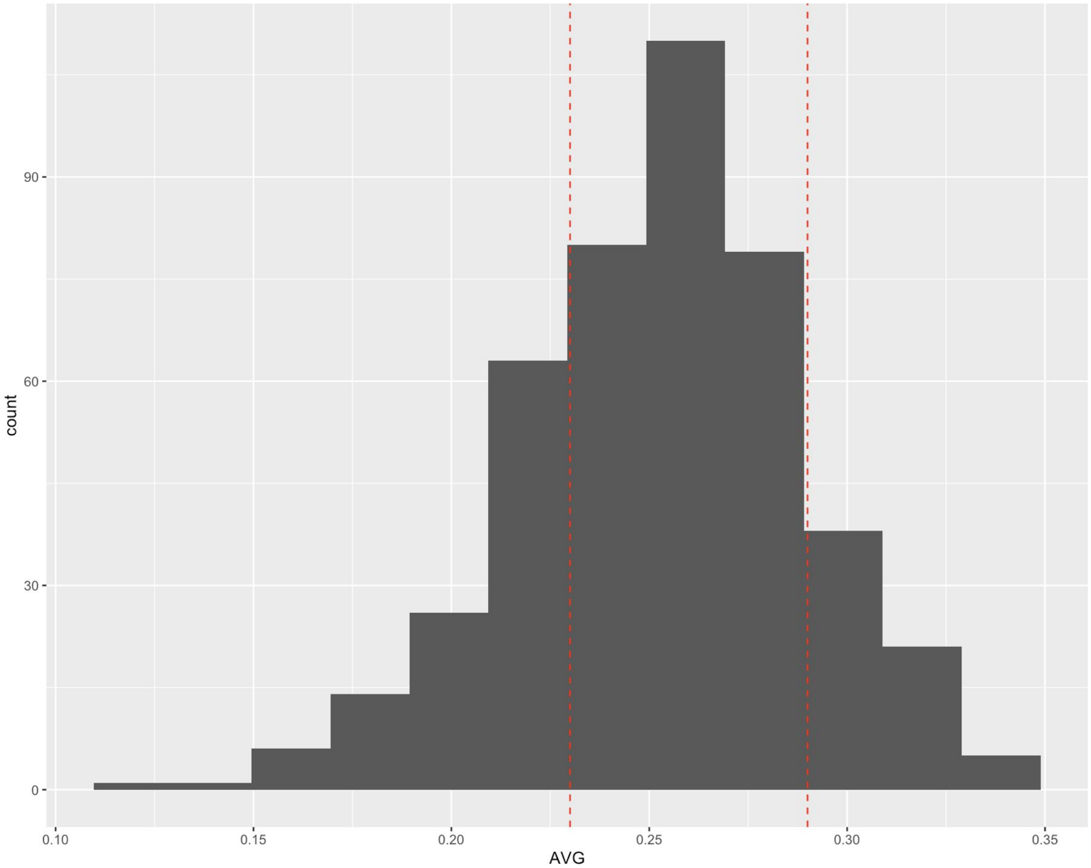
```
# for lambda.min
lasso_coef_bat %>%
select(lasso_min) %>%
filter(lasso_min!=0) %>%
nrow()
## [1] 9
# for lambda.1se
lasso_coef_bat %>%
select(lasso_1se) %>%
filter(lasso_1se!=0) %>%
nrow()
## [1] 8
```

RMSE

```r
# prediction
preds_train_bat <- predict(lasso_bat, s = lasso_bat$lambda.min, batting_train)
preds_test_bat <- predict(lasso_bat, s = lasso_bat$lambda.min, batting_test)
```

```r
# RMSE
# train
RMSE(preds_train_bat, batting_train$AVG)
## [1] 0.02457797
# test
RMSE(preds_test_bat, batting_test$AVG)
## [1] 0.02805086
```

```
ggplot(batting_clean, aes(x = AVG)) +
geom_histogram(bins=12) +
geom_vline(xintercept = 0.23, color='red', linetype='dashed')+
geom_vline(xintercept = 0.29, color='red', linetype='dashed')
```



Model 3: Conclusion

- Both the training and testing set RMSE values were low and close to each other indicating an accurate model with minimal overfitting.
  - RMSE of training set was 0.0246 off of the predicted batting average for the training set.
  - RMSE of testing set was 0.0281 off of the predicted batting average of the testing set.

- Lasso model zeroed out insignificant variables:
  - Games (G), At Bats (AB), Home Runs (HR), Stolen Bases (SB)

- Variable Selection
  - Runs (R), Doubles (X2B), Triples (X3B), Runs Batted In (RBI), Walks (BB), Strikeouts (SO), Hit by Pitch (HBP)

	lasso_min <dbl>	lasso_1se <dbl>
(Intercept)	0.240010	0.234622
G	-0.000108	0.000000
AB	0.000000	0.000000
R	0.000791	0.000471
X2B	0.001004	0.001026
X3B	0.001253	0.001050
HR	0.000000	0.000000
RBI	0.000430	0.000335
SB	0.000000	0.000000
BB	-0.000435	-0.000176

1-10 of 12 rows Previous 1 2 Next

Project Conclusion

- Model Implementation
  - Decision Tree Model
  - Best use case for MLB hitters

- Comparison
  - Lowest RMSE out of all models
  - Easiest to interpret for real MLB hitters to analyze