Rays Data Project

Kenny Miller

2/18/2022

System Comparison and Data Maintenance

Summary

For the start of this analysis I would like to compare the two systems in question for determining each batted ball instance. My analysis below will show that:

- System A appears to indeed be more accurate than System B,
- There is a statistically significant difference in the readings of speed and vertical angle between the two systems, and
- The best way to replace missing data for one system is to develop a model utilizing the other system

```
batting <- read.csv("battedBallData.csv", header = T)
```

Goal: Compare System A and B to see if one system appears to be more accurate and identify differences between each

Step 1: Compare Systems with Available Data

```
sum(complete.cases(batting))/nrow(batting) # ~88.5% of rows have info from both systems
```

```
## [1] 0.8850835
```

```
summary(batting) # more missing examples from A than B
```

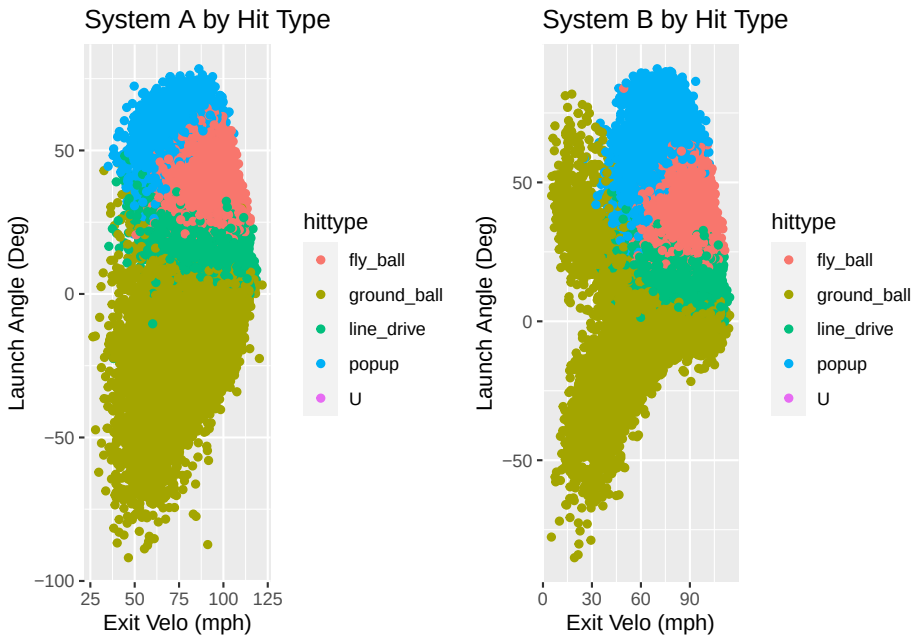
```
##      batter         pitcher      hittype            speed_A
##  Min.   :  1.0   Min.   :  1   Length:73375       Min.   : 26.46
##  1st Qu.:170.0   1st Qu.:121   Class :character   1st Qu.: 80.36
##  Median :341.0   Median :283   Mode  :character   Median : 90.64
##  Mean   :365.1   Mean   :290                      Mean   : 88.42
##  3rd Qu.:550.0   3rd Qu.:447                      3rd Qu.: 98.32
##  Max.   :816.0   Max.   :645                      Max.   :121.85
##                                                   NA's   :7572
##     vangle_A          speed_B           vangle_B
##  Min.   :-91.899   Min.   :  5.152   Min.   :-85.091
```

```
##  1st Qu.: -5.559   1st Qu.: 66.074   1st Qu.: -4.635
##  Median : 11.005   Median : 81.887   Median : 10.546
##  Mean   : 10.854   Mean   : 77.748   Mean   : 13.094
##  3rd Qu.: 27.405   3rd Qu.: 92.327   3rd Qu.: 28.881
##  Max.   : 78.461   Max.   :114.403   Max.   : 90.901
##  NA's   :7572      NA's   :1402      NA's   :1402
```

```
missing <- batting[which(is.na(batting$speed_A) & is.na(batting$speed_B)),]
summary(missing) # 542 rows without any data from the two systems
```

```
##      batter         pitcher        hittype             speed_A       vangle_A
##  Min.   :  4.0   Min.   :  1.0   Length:542         Min.   : NA   Min.   : NA
##  1st Qu.:173.0   1st Qu.:133.2   Class :character   1st Qu.: NA   1st Qu.: NA
##  Median :339.5   Median :291.0   Mode  :character   Median : NA   Median : NA
##  Mean   :364.5   Mean   :299.3                      Mean   :NaN   Mean   :NaN
##  3rd Qu.:533.8   3rd Qu.:480.0                      3rd Qu.: NA   3rd Qu.: NA
##  Max.   :815.0   Max.   :643.0                      Max.   : NA   Max.   : NA
##                                                     NA's   :542   NA's   :542
##     speed_B       vangle_B
##  Min.   : NA   Min.   : NA
##  1st Qu.: NA   1st Qu.: NA
##  Median : NA   Median : NA
##  Mean   :NaN   Mean   :NaN
##  3rd Qu.: NA   3rd Qu.: NA
##  Max.   : NA   Max.   : NA
##  NA's   :542   NA's   :542
```

```
A <- ggplot(batting) +
  aes(x = speed_A, y = vangle_A, color = hittype) +
  geom_point() +
  ggtitle("System A by Hit Type") +
  xlab("Exit Velo (mph)") + ylab("Launch Angle (Deg)")
B <- ggplot(batting) +
  aes(x = speed_B, y = vangle_B, color = hittype) +
  geom_point() +
  ggtitle("System B by Hit Type") +
  xlab("Exit Velo (mph)") + ylab("Launch Angle (Deg)")
grid.arrange(A, B, ncol = 2)
```




```
# System A mean and SD values
mean(batting$speed_A, na.rm = T); sd(batting$speed_A, na.rm = T)
```

```
## [1] 88.42543
```

```
## [1] 13.19294
```

```
mean(batting$vangle_A, na.rm = T); sd(batting$vangle_A, na.rm = T)
```

```
## [1] 10.85356
```

```
## [1] 24.02406
```

```
# System B mean and SD values
mean(batting$speed_B, na.rm = T); sd(batting$speed_B, na.rm = T)
```

```
## [1] 77.74824
```

```
## [1] 18.62154
```

```
mean(batting$vangle_B, na.rm = T); sd(batting$vangle_B, na.rm = T)
```

```
## [1] 13.09431
```

```
## [1] 24.42954
```

```r
paired <- batting[complete.cases(batting),]
var.test(paired$speed_A, paired$speed_B)
```

```
##
##  F test to compare two variances
##
## data:  paired$speed_A and paired$speed_B
## F = 0.59663, num df = 64942, denom df = 64942, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.5875251 0.6058809
## sample estimates:
## ratio of variances
##          0.5966325
```

```r
t.test(paired$speed_A, paired$speed_B, paired = T)
```

```
##
##  Paired t-test
##
## data:  paired$speed_A and paired$speed_B
## t = 224.1, df = 64942, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  8.575846 8.727179
## sample estimates:
## mean of the differences
##                8.651512
```

```r
var.test(paired$vangle_A, paired$vangle_B)
```

```
##
##  F test to compare two variances
##
## data:  paired$vangle_A and paired$vangle_B
## F = 1.2266, num df = 64942, denom df = 64942, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  1.207913 1.245651
## sample estimates:
## ratio of variances
##           1.226637
```

```r
t.test(paired$vangle_A, paired$vangle_B, paired = T)
```

```
##
##  Paired t-test
##
## data:  paired$vangle_A and paired$vangle_B
## t = -65.985, df = 64942, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
##  -1.677977 -1.581168
## sample estimates:
## mean of the differences
##               -1.629573
```

Based on the above checks, it is fair to say that System A and System B have statistically significant differences in how they measure speed and vertical angle for a batted ball. With System A, on average, reading exit velocities (speed) at higher levels than System B and launch angles (vertical angles) at lower levels than System B.

The two plots above also appear to confirm that System A is more accurate than System B. I am comfortable saying this because of the high collection of "ground ball" observations System B measures for batted balls with Launch Angles above 20 degrees.

My next step is to determine the best method for ensuring that players/instances that are captured by only one device are properly accounted for in the data.

Step 2: Data Cleaning/Preparation

```
# Let's use models to predict missing values for the other system
# I will use the paired data to build the models prior to estimating NAs in the original data
speedA <- lm(speed_A ~ (hittype + speed_B + vangle_B)^2, data = paired)
summary(speedA)
```

```
##
## Call:
## lm(formula = speed_A ~ (hittype + speed_B + vangle_B)^2, data = paired)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -63.281  -1.235   0.055   1.177  62.363
##
## Coefficients: (2 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -2.228e+01  6.619e-01 -33.659  < 2e-16 ***
## hittypeground_ball           5.620e+01  6.964e-01  80.700  < 2e-16 ***
## hittypeline_drive            8.284e+00  6.638e-01  12.480  < 2e-16 ***
## hittypepopup                -5.425e+00  8.816e-01  -6.154 7.60e-10 ***
## hittypeU                    -1.237e+00  5.554e+00  -0.223    0.824
## speed_B                      1.314e+00  7.105e-03 184.871  < 2e-16 ***
## vangle_B                     7.233e-01  1.364e-02  53.024  < 2e-16 ***
## hittypeground_ball:speed_B  -6.235e-01  7.652e-03 -81.483  < 2e-16 ***
## hittypeline_drive:speed_B   -1.089e-01  6.803e-03 -16.013  < 2e-16 ***
## hittypepopup:speed_B         1.672e-01  1.287e-02  12.989  < 2e-16 ***
## hittypeU:speed_B                    NA         NA      NA       NA
## hittypeground_ball:vangle_B -7.069e-01  8.287e-03 -85.301  < 2e-16 ***
## hittypeline_drive:vangle_B   3.579e-02  7.931e-03   4.512 6.43e-06 ***
## hittypepopup:vangle_B       -1.554e-01  1.434e-02 -10.833  < 2e-16 ***
## hittypeU:vangle_B                   NA         NA      NA       NA
## speed_B:vangle_B            -8.768e-03  1.486e-04 -59.014  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.553 on 64929 degrees of freedom
```

```
## Multiple R-squared:  0.8212, Adjusted R-squared:  0.8212
## F-statistic: 2.294e+04 on 13 and 64929 DF,  p-value: < 2.2e-16
```

```
vangleA <- lm(vangle_A ~ (hittype + speed_B + vangle_B)^2, data = paired)
summary(vangleA)
```

```
##
## Call:
## lm(formula = vangle_A ~ (hittype + speed_B + vangle_B)^2, data = paired)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -85.296  -0.542  -0.025   0.673  76.405
##
## Coefficients: (2 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.694e+01  5.553e-01  30.499  < 2e-16 ***
## hittypeground_ball          -4.890e+01  5.843e-01 -83.689  < 2e-16 ***
## hittypeline_drive           -5.690e+00  5.570e-01 -10.216  < 2e-16 ***
## hittypepopup                 3.200e+00  7.397e-01   4.327 1.52e-05 ***
## hittypeU                     5.026e-01  4.660e+00   0.108  0.91411
## speed_B                     -1.873e-01  5.962e-03 -31.415  < 2e-16 ***
## vangle_B                     5.708e-01  1.145e-02  49.870  < 2e-16 ***
## hittypeground_ball:speed_B   5.534e-01  6.421e-03  86.194  < 2e-16 ***
## hittypeline_drive:speed_B    7.438e-02  5.708e-03  13.030  < 2e-16 ***
## hittypepopup:speed_B        -9.864e-02  1.080e-02  -9.134  < 2e-16 ***
## hittypeU:speed_B                    NA         NA      NA       NA
## hittypeground_ball:vangle_B -1.146e-01  6.954e-03 -16.481  < 2e-16 ***
## hittypeline_drive:vangle_B  -2.459e-02  6.655e-03  -3.695  0.00022 ***
## hittypepopup:vangle_B        9.283e-02  1.203e-02   7.715 1.23e-14 ***
## hittypeU:vangle_B                   NA         NA      NA       NA
## speed_B:vangle_B             4.638e-03  1.247e-04  37.201  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.659 on 64929 degrees of freedom
## Multiple R-squared:  0.9618, Adjusted R-squared:  0.9618
## F-statistic: 1.257e+05 on 13 and 64929 DF,  p-value: < 2.2e-16
```

```
speedB <- lm(speed_B ~ (hittype + speed_A + vangle_A)^2, data = paired)
summary(speedB)
```

```
##
## Call:
## lm(formula = speed_B ~ (hittype + speed_A + vangle_A)^2, data = paired)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -106.715   -1.485   -0.186    1.364   57.491
##
## Coefficients: (2 not defined because of singularities)
##                               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                  4.592e+01  5.104e-01   89.959   <2e-16 ***
```

```
## hittypeground_ball          -4.382e+01  6.012e-01  -72.888   <2e-16 ***
## hittypeline_drive           -1.813e+01  5.331e-01  -34.016   <2e-16 ***
## hittypepopup                 1.042e+01  7.305e-01   14.266   <2e-16 ***
## hittypeU                     1.144e+00  4.372e+00    0.262    0.794
## speed_A                      4.392e-01  5.157e-03   85.168   <2e-16 ***
## vangle_A                    -1.389e+00  9.893e-03 -140.363   <2e-16 ***
## hittypeground_ball:speed_A   4.607e-01  6.289e-03   73.255   <2e-16 ***
## hittypeline_drive:speed_A    2.187e-01  5.127e-03   42.659   <2e-16 ***
## hittypepopup:speed_A        -3.389e-01  1.000e-02  -33.881   <2e-16 ***
## hittypeU:speed_A                    NA         NA       NA       NA
## hittypeground_ball:vangle_A  9.409e-01  4.533e-03  207.560   <2e-16 ***
## hittypeline_drive:vangle_A  -5.777e-02  6.487e-03   -8.905   <2e-16 ***
## hittypepopup:vangle_A        3.361e-01  1.154e-02   29.125   <2e-16 ***
## hittypeU:vangle_A                   NA         NA       NA       NA
## speed_A:vangle_A             1.580e-02  1.018e-04  155.246   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.372 on 64929 degrees of freedom
## Multiple R-squared:  0.9339, Adjusted R-squared:  0.9339
## F-statistic: 7.056e+04 on 13 and 64929 DF,  p-value: < 2.2e-16
```

```
vangleB <- lm(vangle_B ~ (hittype + speed_A + vangle_A)^2, data = paired)
summary(vangleB)
```

```
##
## Call:
## lm(formula = vangle_B ~ (hittype + speed_A + vangle_A)^2, data = paired)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.085  -0.836  -0.124   0.465  91.840
##
## Coefficients: (2 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.954e+01  4.799e-01  40.710  < 2e-16 ***
## hittypeground_ball          -2.292e+01  5.652e-01 -40.552  < 2e-16 ***
## hittypeline_drive           -1.102e+01  5.012e-01 -21.989  < 2e-16 ***
## hittypepopup                 4.711e+00  6.867e-01   6.860 6.94e-12 ***
## hittypeU                    -4.656e-01  4.110e+00  -0.113    0.910
## speed_A                     -2.202e-01  4.848e-03 -45.418  < 2e-16 ***
## vangle_A                     3.733e-01  9.301e-03  40.136  < 2e-16 ***
## hittypeground_ball:speed_A   2.545e-01  5.912e-03  43.040  < 2e-16 ***
## hittypeline_drive:speed_A    1.181e-01  4.820e-03  24.508  < 2e-16 ***
## hittypepopup:speed_A        -1.632e-01  9.404e-03 -17.354  < 2e-16 ***
## hittypeU:speed_A                    NA         NA       NA       NA
## hittypeground_ball:vangle_A -3.358e-01  4.262e-03 -78.797  < 2e-16 ***
## hittypeline_drive:vangle_A  -5.774e-04  6.099e-03  -0.095    0.925
## hittypepopup:vangle_A        1.638e-01  1.085e-02  15.096  < 2e-16 ***
## hittypeU:vangle_A                   NA         NA       NA       NA
## speed_A:vangle_A             7.214e-03  9.568e-05  75.402  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 4.11 on 64929 degrees of freedom
## Multiple R-squared:  0.9635, Adjusted R-squared:  0.9635
## F-statistic: 1.319e+05 on 13 and 64929 DF,  p-value: < 2.2e-16
```

```
# Let's replace the NA values
batting.original <- batting # save original data before adjustments
for (i in 1:nrow(batting)) {
  row <- batting[i,]
  if (complete.cases(row)) { next }
  if (which(is.na(row))[1] == 4) {
    s <- predict(speedA, row)
    a <- predict(vangleA, row)
    batting[i,c(4,5)] <- c(s,a)
  } else if (which(is.na(row))[1] == 6) {
    s <- predict(speedB, row)
    a <- predict(vangleB, row)
    batting[i,c(6,7)] <- c(s,a)
  }
}
sum(complete.cases(batting))/nrow(batting)
```

```
## [1] 0.9926133
```

```
summary(batting); nrow(missing)
```

```
##      batter         pitcher      hittype             speed_A
##  Min.   :  1.0   Min.   :  1   Length:73375       Min.   : 26.46
##  1st Qu.:170.0   1st Qu.:121   Class :character   1st Qu.: 78.27
##  Median :341.0   Median :283   Mode  :character   Median : 89.22
##  Mean   :365.1   Mean   :290                      Mean   : 86.97
##  3rd Qu.:550.0   3rd Qu.:447                      3rd Qu.: 97.63
##  Max.   :816.0   Max.   :645                      Max.   :121.85
##                                                   NA's   :542
##     vangle_A          speed_B           vangle_B
##  Min.   :-91.899   Min.   :  5.152   Min.   :-85.091
##  1st Qu.: -7.095   1st Qu.: 66.013   1st Qu.: -4.699
##  Median : 10.216   Median : 81.809   Median : 10.556
##  Mean   : 10.928   Mean   : 77.685   Mean   : 13.143
##  3rd Qu.: 28.194   3rd Qu.: 92.284   3rd Qu.: 29.016
##  Max.   : 90.096   Max.   :114.403   Max.   : 90.901
##  NA's   :542       NA's   :542       NA's   :542
```

```
## [1] 542
```

```
# from above we know that there are 542 instances with no readings from either system
# for further analysis I will replace the values in these rows with the median values based on hit type
# I will replace values for both systems using median imputation
types <- unique(batting$hittype)

batting <- impute(batting) # median imputation
# source code found online: https://rdrr.io/cran/imputeMissings/man/impute.html

sum(complete.cases(batting))/nrow(batting)
```

```
## [1] 1
```

```
summary(batting)
```

```
##      batter         pitcher      hittype             speed_A
##  Min.   :  1.0   Min.   :  1   Length:73375       Min.   : 26.46
##  1st Qu.:170.0   1st Qu.:121   Class :character   1st Qu.: 78.37
##  Median :341.0   Median :283   Mode  :character   Median : 89.22
##  Mean   :365.1   Mean   :290                      Mean   : 86.99
##  3rd Qu.:550.0   3rd Qu.:447                      3rd Qu.: 97.57
##  Max.   :816.0   Max.   :645                      Max.   :121.85
##     vangle_A         speed_B           vangle_B
##  Min.   :-91.90   Min.   :  5.152   Min.   :-85.091
##  1st Qu.: -6.96   1st Qu.: 66.171   1st Qu.: -4.598
##  Median : 10.22   Median : 81.809   Median : 10.556
##  Mean   : 10.92   Mean   : 77.715   Mean   : 13.124
##  3rd Qu.: 28.02   3rd Qu.: 92.207   3rd Qu.: 28.857
##  Max.   : 90.10   Max.   :114.403   Max.   : 90.901
```

In Step 2, I replaced missing values for all of the batted ball observations that had data from only one of the two systems, if there wasn't an observation for the batted ball between the two systems the values for both systems were replaced by median imputation by hit type; the code to complete this was found online, a link to the code is included in the code chunk above. The models I generated in this step were all significant and showed promise as predictors for the other system's metrics. Specifically both systems are able to explain the variance of the measurements from their counterpart at no worse than 82%.

Predicting Next Season's Speed-Off-Bat

Goal: Predict next season's speed-off-bat for each batter in the original dataset

Summary

I was tasked with predicting the average speed-off-bat for next season for all of the players in our dataset. This prediction utilized the past season's batted ball data collected by two systems; due to the variability in some of the classifications for System B and the visualizations from the beginning of this project, I have chosen to use the measurements from system A for my scale. I found that a decision tree does well to estimate true speed off the bat.

When making final predictions for next season I utilized the Central Limit Theorem's guidelines, if a batter had more than 25 balls in play I used their true average predictions for what we can expect next year but if the batter had 25 balls in play or less I used a weighted average with the weights based on the proportion of the balls in play from the original data set. Predictions for all 816 batters can be found in the bat_speed data frame or the "Final_Batted_Ball_Predictions_Miller.csv" file included in my submission folder.

Cleaning

- Most of the necessary data cleaning was completed above during the system comparison and maintenance

- I verified that the data matched the original in terms of number of batters in the data set
- Removed the two "U" hit type observations, since both batters had many other observations in the data set
- Split the data into Training and Holdout samples using stratified random sampling, 80% proportion by hit type

Model Development

- Started with a vanilla linear regression and progressed to additional models in increasing complexity
- Found a Decision Tree does very well with this data
- My laptop struggled with models more advanced than a vanilla decision tree
- Using decision tree model, I generated predicted speed-off-bat measures for each instance. Once I had this I needed to generate predictions for each player, this was dependent on the number of balls in play from each batter:
  - If the batter has more than 25 balls in play, since the speed_A distribution is roughly normal we can use the Central Limit Theorem and assume that the average of all of their balls in play is representative of their "true" capabilities
  - If the batter has 25 or less balls in play, I used a weighted average, based on the proportion of hit types in the original data frame, to determine their "true" speed-off-bat. To ensure all hit types are included in a batter's average, I included the mean exit velocity from the data frame, by hit type, for hit types not observed from a batter. In making this decision I am assuming each batter has "normal" abilities compared to the entire population of batters in the data set

Future Recommendations

- More computing power and time to determine if more complex models would preform better
- Include pitch specific information to determine how batters exit velocity changes from specific pitch characteristics
- Include previous seasons data to see how players are trending year over year with exit velocity

---

Step 1: Data Check and any additional cleaning

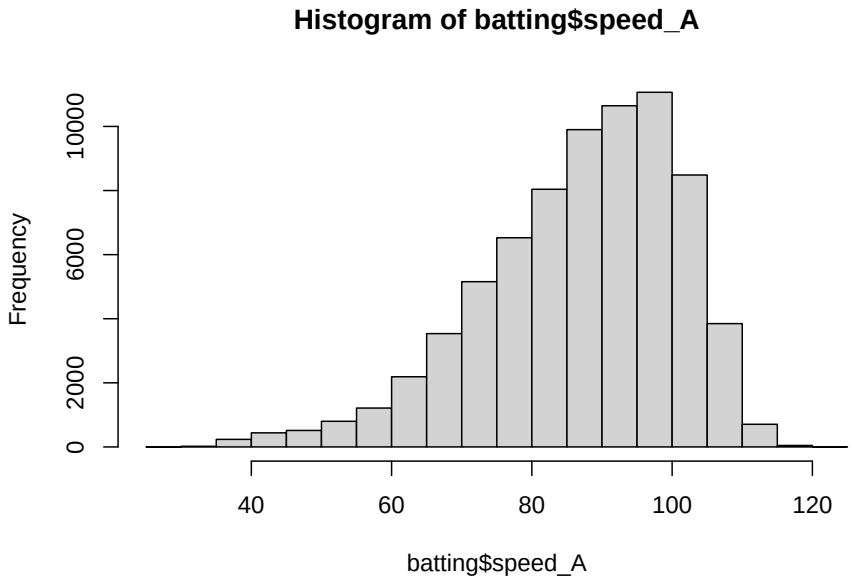
```
batters.original <- unique(batting.original$batter)
length(batters.original) # 816
```

```
## [1] 816
```

```
batters <- unique(batting$batter)
length(batters) # 816
```

```
## [1] 816
```

```
hist(batting$speed_A) # roughly normal with a slight left skew
```




```
BIP <- aggregate(pitcher~batter, data=batting, FUN = length) # balls in play
colnames(BIP) <- c("batter","abs")
length(which(BIP$abs <= 25))
```

```
## [1] 342
```

```
# number of batters where I will need to adjust the averages since Central Limit Theorem won't hold
# table(BIP$abs)

# table(batting$pitcher)

table(batting$hittype) # what is U and why are there only 2 instances? Will investigate
```

```
##
##    fly_ball ground_ball  line_drive       popup           U
##       16722       33239       18166        5246           2
```

```
batting[which(batting$hittype == "U"),]
```

```
##       batter pitcher hittype  speed_A  vangle_A  speed_B  vangle_B
## 25541    493     138       U 92.80331  24.40359 89.71565  24.08297
## 48327    479     228       U 51.57663 -13.36131 60.52011 -22.86229
```

```
BIP[which(BIP$batter %in% c(493,479)),]
```

```
##     batter abs
## 479    479 195
## 493    493  91
```

```
# I will drop the U observations since batters have 195 and 91 ABs
batting <- batting[which(batting$hittype != "U"),]
batting <- droplevels(batting)


# checking for near zero and zero variance columns
infodensity <- nearZeroVar(batting, saveMetrics= TRUE)
infodensity[infodensity$nzv,] # there are no columns to worry about for near-zero variance
```

```
## [1] freqRatio     percentUnique zeroVar       nzv
## <0 rows> (or 0-length row.names)
```

```
# checking for highly correlated values
highlycorrelated <- findCorrelation(cor_matrix(batting), cutoff = 0.95)
colnames(batting)[highlycorrelated]
```

```
## [1] "speed_A"
```

```
# speed_A is possibly highly correlated but I will leave it alone
```

Step 2: Create Training and Holdout Sample

```
write.csv(batting,"preprocessing_batting.csv", row.names = F)
set.seed(23); TRAIN <- stratified(batting, c("hittype"), .8, bothSets = T)$SAMP1
set.seed(23); HOLDOUT <- stratified(batting, c("hittype"), .8, bothSets = T)$SAMP2
# code found for stratified sampling via StackOverflow
  # https://stackoverflow.com/questions/23479512/stratified-random-sampling-from-data-frame
# remove batter identifier from both Training and Holdout samples
TRAIN <- TRAIN[,-c("batter","pitcher")]; HOLDOUT <- HOLDOUT[,-c("batter","pitcher")]
```

I decided to do a stratified random sample for this data by hit type to ensure that all hit types are seen in the training and holdout samples. I removed batter and pitcher though after because they are IDs and should not be used in the model.

Step 3: Model Development

```
# using system A for all model creation
# y = speed_A
# Set up how generalization error is to be estimated (5-fold crossvalidation shown here)
fitControl <- trainControl(method="cv",number=5, allowParallel = TRUE)

# Vanilla Linear Regression
set.seed(23); GLM <- train(speed_A~.,data=TRAIN,method='glm',
                           trControl=fitControl,preProc=c("center", "scale") )
GLM$results
```

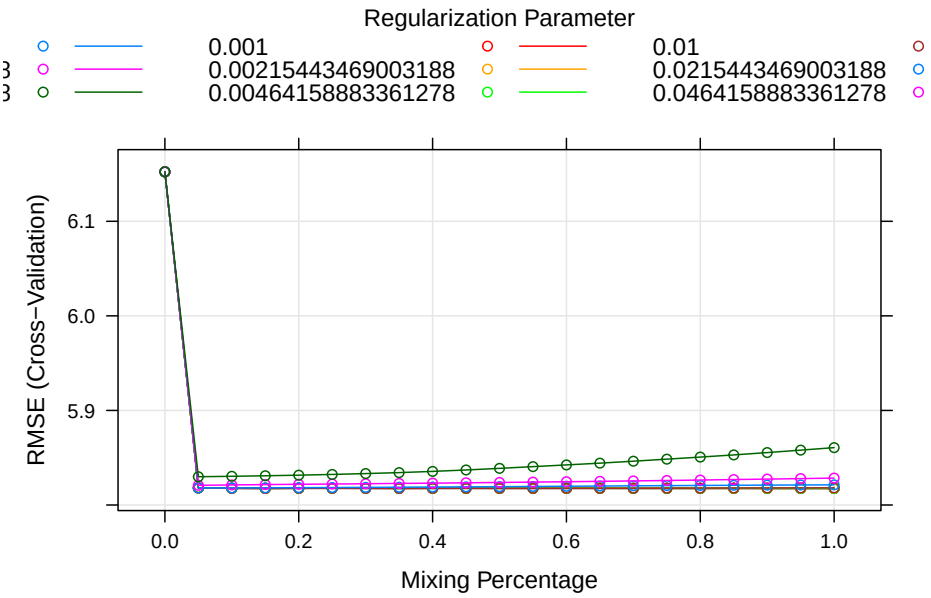
```
##   parameter     RMSE  Rsquared      MAE    RMSESD  RsquaredSD      MAESD
## 1      none 5.816894 0.8274847 4.146754 0.1209579 0.006789247 0.04130927
```

```
postResample(predict(GLM,newdata=HOLDOUT),HOLDOUT$speed_A) # RMSE: 5.8330801
```

```
##      RMSE  Rsquared       MAE
## 5.8330801 0.8258977 4.1783654
```

```
# Regularized logistic regression
glmnetGrid <- expand.grid(alpha = seq(0,1,.05),lambda = 10^seq(-4,-1,length=10))

set.seed(23); GLMnet <- train(speed_A~.,data=TRAIN,method='glmnet', tuneGrid=glmnetGrid,
                              trControl=fitControl, preProc = c("center", "scale"))
# GLMnet
plot(GLMnet)
```




```
GLMnet$bestTune
```

```
##     alpha      lambda
## 186   0.9 0.004641589
```

```
# GLMnet$results
GLMnet$results[rownames(GLMnet$bestTune),]
```

```
##     alpha      lambda    RMSE Rsquared      MAE    RMSESD  RsquaredSD
## 186   0.9 0.004641589 5.81755 0.827447 4.153517 0.1208694 0.006809032
##          MAESD
## 186 0.04187438
```

```
postResample(predict(GLMnet,newdata=HOLDOUT),HOLDOUT$speed_A) # RMSE: 5.8336888
```

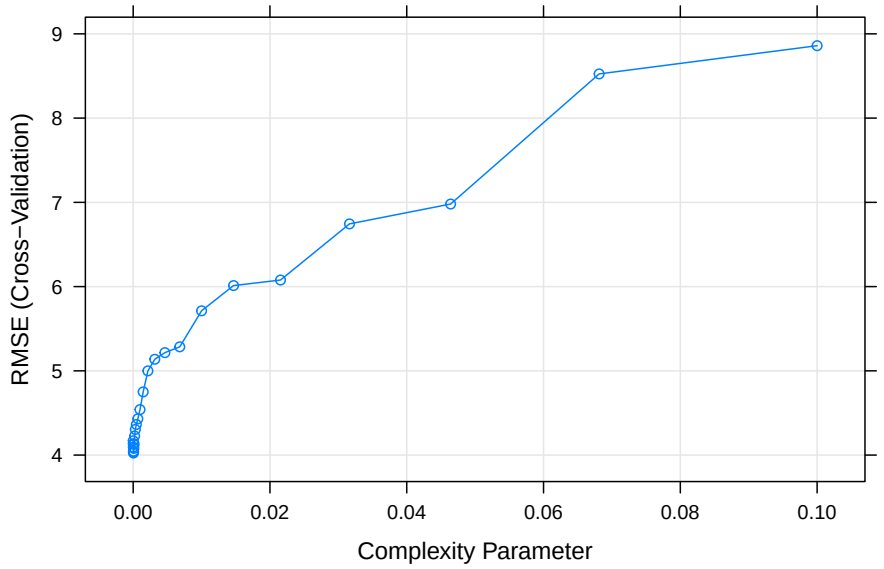
```
##      RMSE  Rsquared       MAE
## 5.8336888 0.8258599 4.1847472
```

```
# No improvement over the vanilla linear

# Vanilla Partition
treeGrid <- expand.grid(cp=10^seq(-5,-1,length=25))

set.seed(23); TREE <- train(speed_A~.,data=TRAIN,method='rpart', tuneGrid=treeGrid,
                            trControl=fitControl, preProc = c("center", "scale"))

# TREE
plot(TREE)
```




```
TREE$bestTune
```

```
##             cp
## 5 4.641589e-05
```

```
# TREE$results
TREE$results[rownames(TREE$bestTune),] # RMSESD: 0.1795576
```

```
##             cp     RMSE  Rsquared      MAE    RMSESD RsquaredSD      MAESD
## 5 4.641589e-05 4.024218 0.9174387 2.150389 0.1795576 0.00705765 0.04919672
```

```
postResample(predict(TREE,newdata=HOLDOUT),HOLDOUT$speed_A) # RMSE: 4.0457643
```

```
##      RMSE  Rsquared       MAE
## 4.0457643 0.9162791 2.1785301
```

```
# To visualize the tree if wanted
# TREE <- rpart(speed_A~.,data=TRAIN,cp=4.641589e-05)
```

```
# visualize_model(TREE)

# Improvement over the previous models

# random forest
# (significant run time and I am unable to get it to complete in a reasonable amount of time)
# forestGrid <- expand.grid(mtry=c(1,3,5,12))
# cluster <- makeCluster(detectCores() - 1) #parallelization
# registerDoParallel(cluster)
# FOREST <- train(speed_A~.,data=TRAIN,method='rf',tuneGrid=forestGrid,
#                                 trControl=fitControl, preProc = c("center", "scale"))
# stopCluster(cluster)
# registerDoSEQ()
# plot(FOREST)
# FOREST$bestTune
# FOREST$results[rownames(FOREST$bestTune),]
# postResample(predict(FOREST,newdata=HOLDOUT),HOLDOUT$speed_A) # RMSE:

# Boosted Tree
# Same as Random Forest
# gbmGrid <- expand.grid(n.trees=c(100,200,500),interaction.depth=1:4,
#                         shrinkage=c(.01,.001),n.minobsinnode=c(5,10))
# cluster <- makeCluster(detectCores() - 1) # parallelization
# registerDoParallel(cluster)
# set.seed(23); GBM <- train(speed_A~.,data=TRAIN, method='gbm',tuneGrid=gbmGrid,
#                             verbose=FALSE,trControl=fitControl, preProc = c("center", "scale"))
# stopCluster(cluster)
# registerDoSEQ()
# plot(GBM)
# GBM$bestTune
# GBM$results[rownames(GBM$bestTune),]
# postResample(predict(GBM,newdata=HOLDOUT,n.trees=500),HOLDOUT$speed_A)
# RMSE:
```

Step 4: Choosing the best model from those tested

Since the Decision Tree had a RMSE of 4.045 and a SD of 0.1796, which is more than one standard deviation better than the other models that completed, I will use this for my predictions.

```
model_data <- batting[,-c(1,2)] # ensuring batter and pitcher aren't in final model
set.seed(23); FINAL <- train(speed_A~., data=model_data,
                             method="rpart", tuneGrid=expand.grid(cp=4.641589e-05),
                             trControl=fitControl, preProc=c("center","scale"))

predictions <- predict(FINAL, newdata = batting)
```

Step 5: Final Player Averages

```
batting_final <- batting
batting_final$prediction <- predictions

nrow(BIP) # 816 batters in final predictions
```

```
## [1] 816
```

```
length(unique(batting_final$batter))
```

```
## [1] 816
```

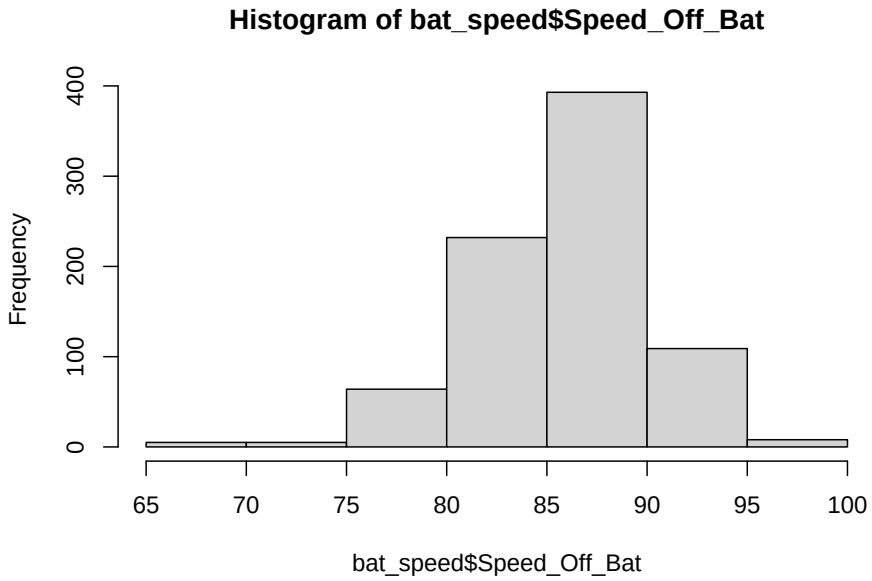
```
length(which(BIP$abs <= 25)) # 342 batters will not meet CLT guidelines and will need
```

```
## [1] 342
```

```
weights <- table(batting_final$hittype)/nrow(batting_final) # original weight table

ht_averages <- batting %>% group_by(hittype) %>% summarise(prediction=mean(speed_A))
# ^ average speed-off-bat by hit type for all batters in the data set
# average is named "prediction" in this to help with row binding within the loop
bat_speed <- setNames(data.frame(matrix(ncol=2,nrow=816)), c("Batter","Speed_Off_Bat"))
for (i in 1:nrow(BIP)) {
  b <- BIP[i,]
  sub <- subset(batting_final, batter == b$batter)
  if (b$abs > 25) {
    sob <- mean(sub$prediction) # sob = speed of bat
  } else {
    x <- aggregate(prediction~hittype, data=sub, FUN=mean)
    missing_hts <- setdiff(ht_averages$hittype, x$hittype)
    x <- rbind(x, ht_averages[which(ht_averages$hittype %in% missing_hts),])
    x <- x[order(x$hittype),]
    sob <- weighted.mean(x$prediction, weights)
  }
  bat_speed$Batter[i] <- b$batter
  bat_speed$Speed_Off_Bat[i] <- sob
}
head(bat_speed); hist(bat_speed$Speed_Off_Bat); summary(bat_speed$Speed_Off_Bat)
```

```
##   Batter Speed_Off_Bat
## 1      1      87.01846
## 2      2      79.59827
## 3      3      91.00431
## 4      4      87.67384
## 5      5      88.49754
## 6      6      82.27692
```




```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   66.55   83.30   86.30   85.83   88.65   98.45
```

```
nrow(bat_speed)
```

```
## [1] 816
```

```
write.csv(bat_speed, "Final_Batted_Ball_Predictions_Miller.csv", row.names = F)
```