# Data Project

Hailey Yabroudy

**Task: Develop plan to predict hitter performance in the next season. Project "true" average speed-off-bat for each batter in the following season.**

*Summary*

I started by reviewing the data and getting a feel for it. I quickly realized that I would need to clean the data, so I started by looking for outliers within the data. I used the quantile method to measure for extreme outliers (I chose to only look for extreme ones because there were too many on the mild outliers). I then went on to measure what the "true" exit velocity was for each player. I decided to find a weighted average for each hit of the system A and B (system A having more weight) and then aggregated all of my findings based on the batter. Using those averages, I came up with a point system to help project the players performances for the coming season. The scores are based on hit types and their exit velocities. Essentially, the harder the hit, the higher the score and the hit type that would most likely yield extra bases would get more points. At the end, I summarized my findings, and printed the head of the new data frame I created, displaying the top 5 hitters that are likely to perform the best.

*Data and Analysis*

```
library(readr)
battedBallData <- read_csv("/Users/haileyemma/Documents/JObs/[redacted]/battedBallData.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   batter = col_double(),
##   pitcher = col_double(),
##   hittype = col_character(),
##   speed_A = col_double(),
##   vangle_A = col_double(),
##   speed_B = col_double(),
##   vangle_B = col_double()
## )
```

```
library("dplyr", lib.loc="/Library/Frameworks/R.framework/Versions/4.0/Resources/library")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
##     intersect, setdiff, setequal, union
```

I started working on this project by exploring the data a little bit. I wanted to find the average v angle for the A and B method in order to know what hit type can be associated with what angle.

*Average v angle by hit type for A and B:*

```
aggregate(vangle_A ~ hittype, battedBallData, mean, na.rm=TRUE)
```

```
##       hittype  vangle_A
## 1    fly_ball  35.40010
## 2 ground_ball -10.76874
## 3  line_drive  15.82258
## 4       popup  56.11939
## 5           U  24.40359
```

```
aggregate(vangle_B ~ hittype, battedBallData, mean, na.rm=TRUE)
```

```
##       hittype   vangle_B
## 1    fly_ball 35.7038943
## 2 ground_ball -7.1193269
## 3  line_drive 15.3549462
## 4       popup 63.1098015
## 5           U  0.6103425
```

I then wanted to look at the ranges of this to see where the hit type angles started and ended. This is where I quickly found out that there were outliers in the data and possibly some incorrectly labeled hit types (the -41 angle for a flyball doesn't really make sense). I decided to look a little deeper into the outliers.

*Range of these angles:*

```
aggregate(vangle_A ~ hittype, battedBallData, range, na.rm=TRUE)
```

```
##       hittype vangle_A.1 vangle_A.2
## 1    fly_ball  -41.24443   65.63399
## 2 ground_ball  -91.89863   45.63272
## 3  line_drive  -22.50977   53.45513
## 4       popup   18.78228   78.46098
## 5           U   24.40359   24.40359
```

```
aggregate(vangle_B ~ hittype, battedBallData, range, na.rm=TRUE)
```

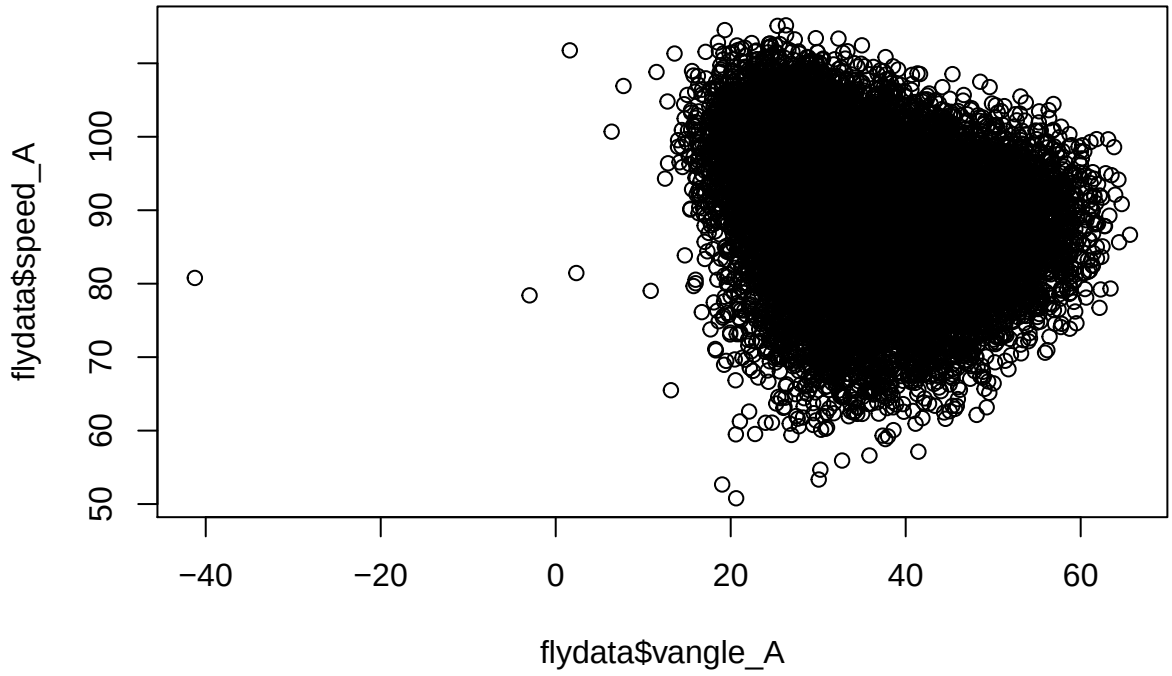
```
##       hittype  vangle_B.1  vangle_B.2
## 1    fly_ball   0.5774136  83.8713949
## 2 ground_ball -85.0909289  81.7766641
## 3  line_drive  -8.4645133  54.6402998
## 4       popup  17.5313325  90.9008191
## 5           U -22.8622889  24.0829738
```

*Finding outliers for launch angles*

I decided to seperate the data into subsets of data based on the hit type. Each hit type will have a different norm when it comes to launch angle, so it would only make sense to seperate them. I found the outliers for each subset of data using the quantile method for both A and B. Once I found the outliers for both, I looked

to see if either list had the same event listed and if it did, I removed it from the data. If I had more time to work on this project, I would have found a way to do this using if/else statements and for loops so the code wouldn't be as repetitive as it is and it could just sweep through the data. I also decided to use extreme uppers and lowers for the quantile method because when it chose to only multiple the IQR by 1.5, it came back with too many results. I used the plots below to visualize if there were any glaring outliers in the data and then did calculations to confirm if there were true outliers or not.

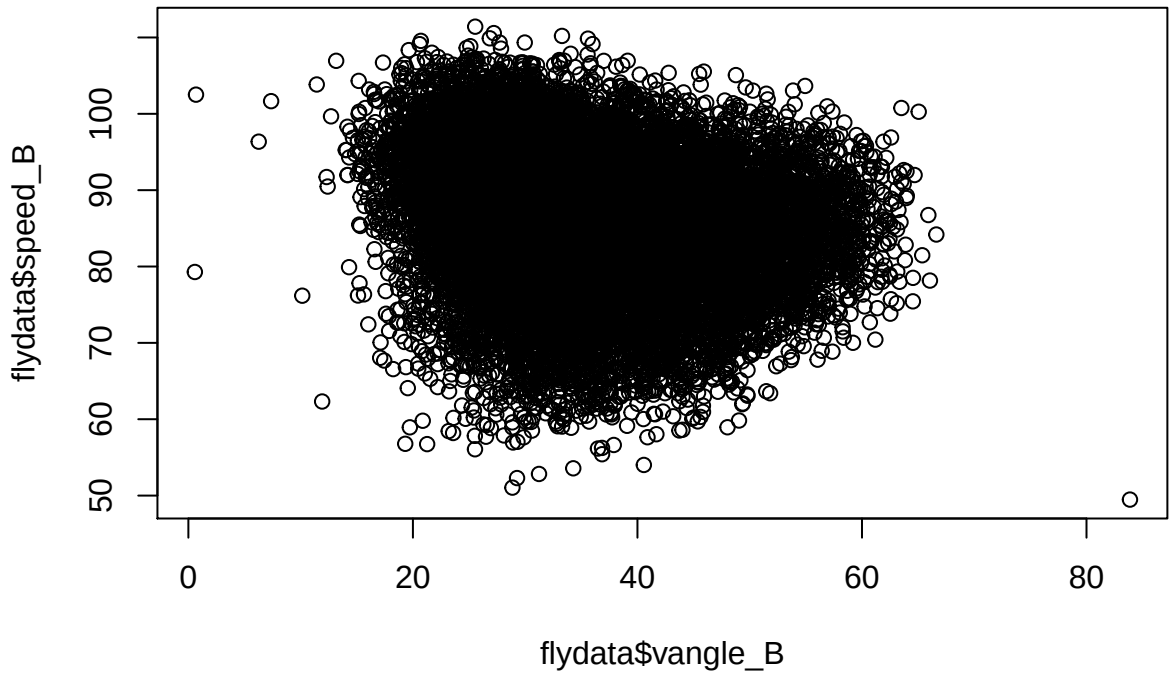
```
flydata <- battedBallData[which(battedBallData$hittype=='fly_ball'),]
plot(flydata$vangle_A,flydata$speed_A)
```




```
lowerq = quantile(flydata$vangle_A, na.rm=TRUE)[2]
upperq = quantile(flydata$vangle_A, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(flydata$vangle_A > extreme_upper | flydata$vangle_A < extreme_lower)
results
```

```
## [1] 6761
```

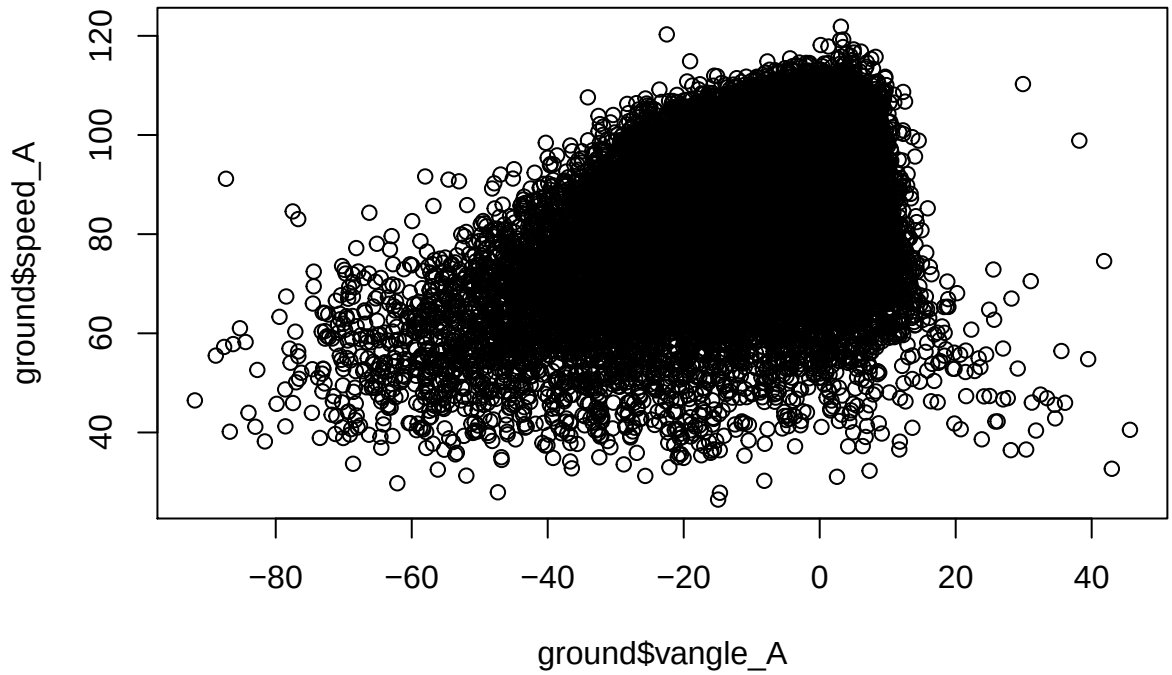
```
plot(flydata$vangle_B,flydata$speed_B)
```




```
lowerq = quantile(flydata$vangle_B, na.rm=TRUE)[2]
upperq = quantile(flydata$vangle_B, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(flydata$vangle_B > extreme_upper | flydata$vangle_B < extreme_lower)
results
```

```
## integer(0)
```

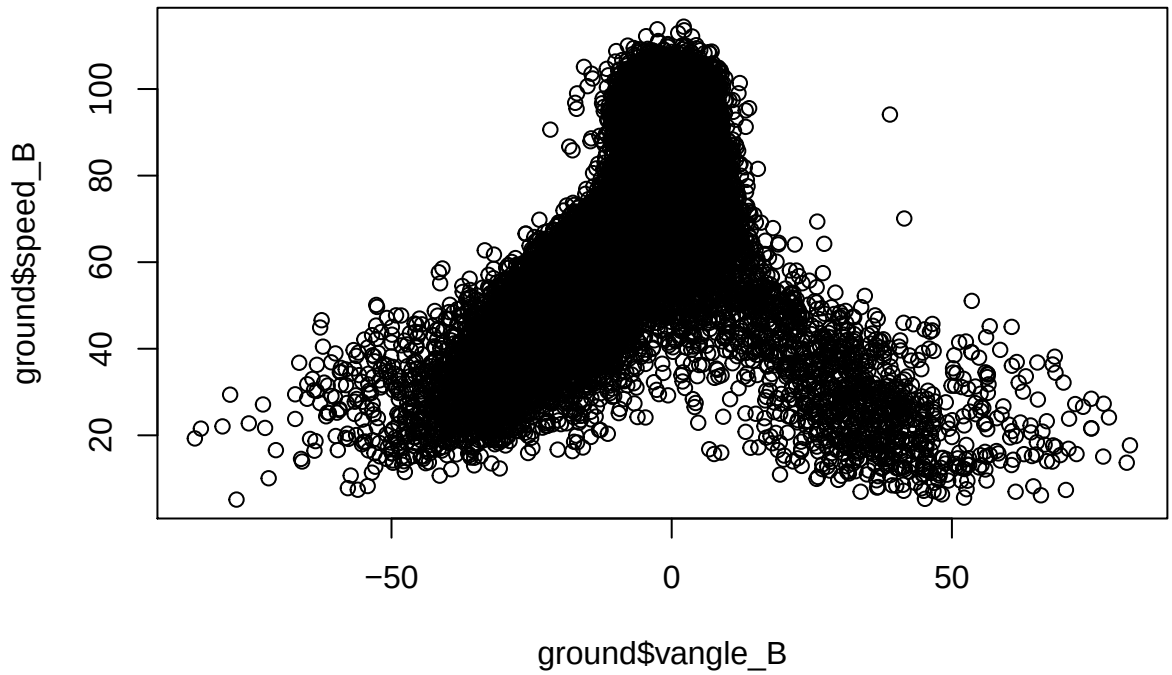
```
ground <- battedBallData[which(battedBallData$hittype=='ground_ball'),]
plot(ground$vangle_A,ground$speed_A)
```




```
lowerq = quantile(ground$vangle_A, na.rm=TRUE)[2]
upperq = quantile(ground$vangle_A, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(ground$vangle_A > extreme_upper | ground$vangle_A < extreme_lower)
results
```

```
##  [1]   785  2856  3090  3204  4262  4668  5281  6903  8221  9305  9412 10384
## [13] 10464 10515 10854 10875 11655 12027 12216 12878 13729 14806 15067 15077
## [25] 15620 16503 17699 18395 20896 21127 21381 21680 22170 22811 22960 24235
## [37] 24778 25193 25539 26294 26497 26834 28973 29266 29683 30558 31633 32074
```

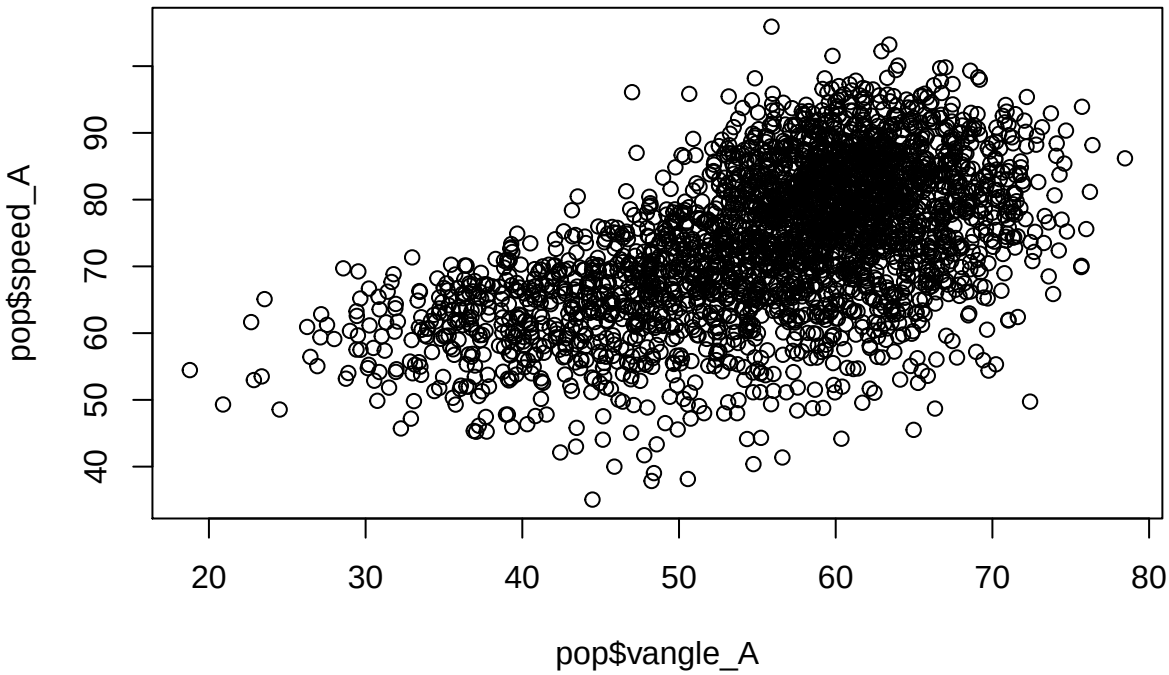
```
plot(ground$vangle_B,ground$speed_B)
```




```
lowerq = quantile(ground$vangle_B, na.rm=TRUE)[2]
upperq = quantile(ground$vangle_B, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(ground$vangle_B > extreme_upper | ground$vangle_B < extreme_lower)
results
```

```
##   [1]   119   433   463   767   776   817   825   902  1184  1198  1247  1331
##  [13]  1350  1379  1412  1431  1468  1620  1764  2217  2285  2736  2859  3030
##  [25]  3043  3440  3505  3556  3601  3681  3889  3918  4012  4148  4171  4316
##  [37]  4397  4404  4409  4448  4549  4555  4816  5000  5172  5340  5530  5580
##  [49]  5729  5798  5804  6017  6342  6397  6445  6481  6777  6833  7079  7092
##  [61]  7277  7473  7478  7711  7872  7883  7954  7965  8034  8221  8670  8673
##  [73]  9410  9458  9464  9718  9726  9731  9809  9902 10322 10383 10675 10688
##  [85] 10988 10992 10999 11022 11239 11241 11536 11729 11876 11936 12453 12485
##  [97] 12574 12613 12695 12698 12755 12976 13037 13054 13159 13217 13236 13456
## [109] 13847 14079 14169 14222 14256 14292 14396 14542 14678 15014 15073 15259
## [121] 15488 15504 15542 15636 15637 15711 15746 15768 15842 15864 16198 16377
## [133] 16520 16557 16581 17117 17390 17408 17562 17587 17618 17721 17751 18154
## [145] 18332 18889 19087 19223 19487 19813 20014 20061 20234 20457 20736 20976
## [157] 21090 21187 21305 21565 21572 21798 21806 21834 22024 22160 22273 22276
## [169] 22394 22405 22579 22721 22788 22815 22819 22930 23153 23292 23511 23637
## [181] 23740 23950 24199 24235 24300 24317 25288 25381 25437 25652 25655 25765
## [193] 25833 26057 26095 26611 26650 26840 27472 28062 28181 28237 28240 28453
## [205] 28615 28639 28723 28758 28761 29049 29247 29320 29323 29518 29575 29750
## [217] 29781 29823 29879 29984 30150 30558 30583 30668 30763 30770 30864 30904
## [229] 30928 31206 31368 31430 31605 31750 31762 32516 32592 32800 33195
```

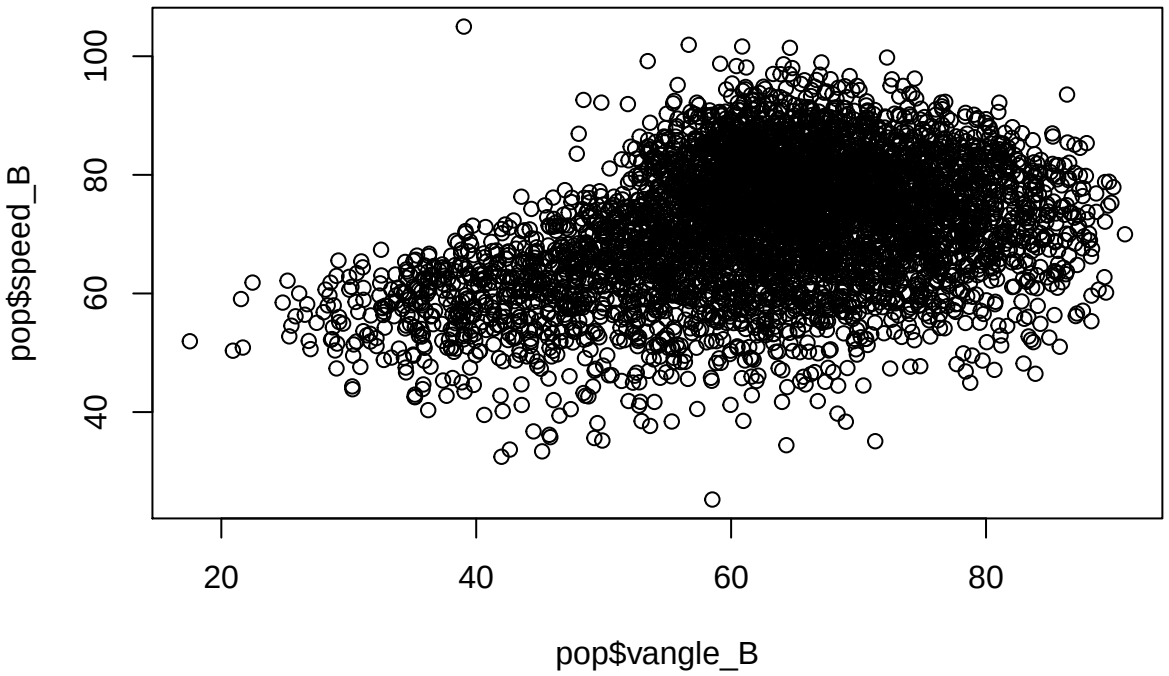
```
pop <- battedBallData[which(battedBallData$hittype=='popup'),]
plot(pop$vangle_A,pop$speed_A)
```




```
lowerq = quantile(pop$vangle_A, na.rm=TRUE)[2]
upperq = quantile(pop$vangle_A, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(pop$vangle_A > extreme_upper | pop$vangle_A < extreme_lower)
results
```

```
## integer(0)
```

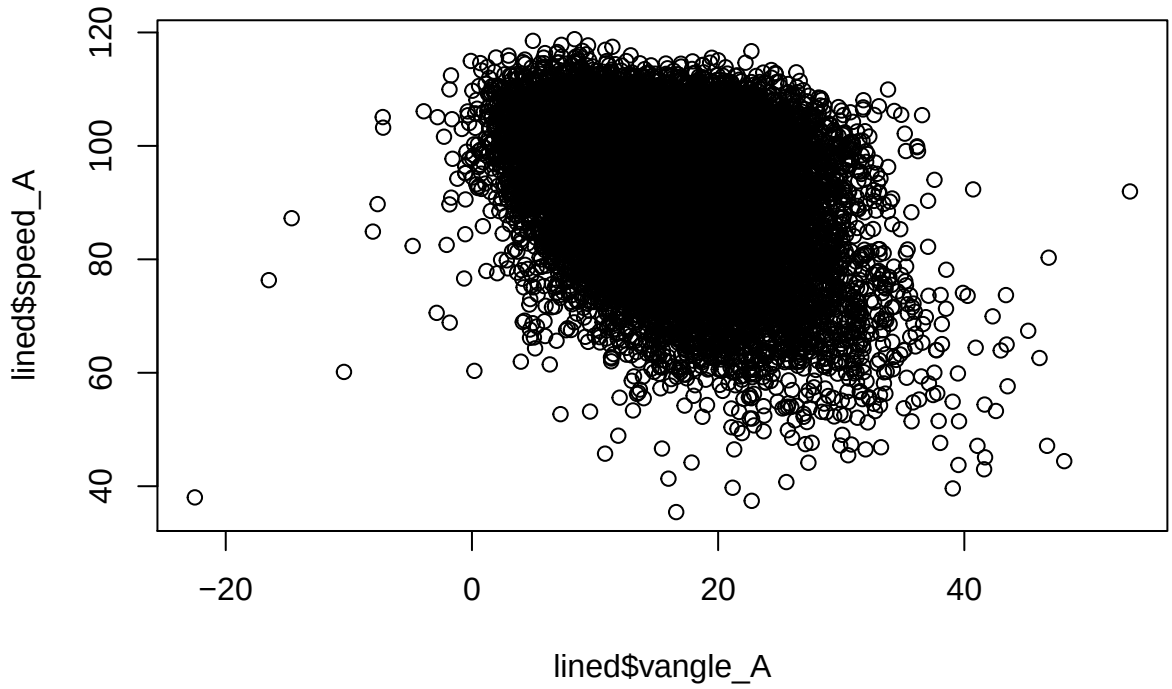
```
plot(pop$vangle_B,pop$speed_B)
```



```
lowerq = quantile(pop$vangle_B, na.rm=TRUE)[2]
upperq = quantile(pop$vangle_B, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(pop$vangle_B > extreme_upper | pop$vangle_B < extreme_lower)
results
```

```
## integer(0)
```

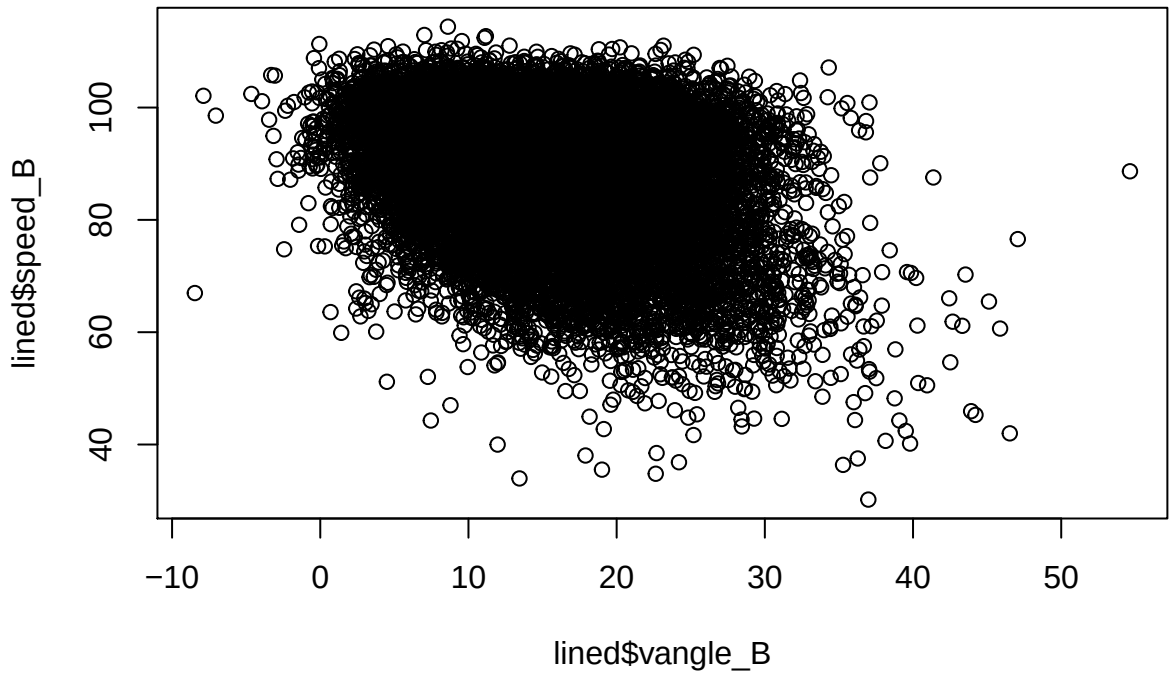
```
lined <- battedBallData[which(battedBallData$hittype=='line_drive'),]
plot(lined$vangle_A,lined$speed_A)
```




```
lowerq = quantile(lined$vangle_A, na.rm=TRUE)[2]
upperq = quantile(lined$vangle_A, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(lined$vangle_A > extreme_upper | lined$vangle_A < extreme_lower)
results
```

```
## [1]  4141 13568
```

```
plot(lined$vangle_B,lined$speed_B)
```




```
lowerq = quantile(lined$vangle_B, na.rm=TRUE)[2]
upperq = quantile(lined$vangle_B, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(lined$vangle_B > extreme_upper | lined$vangle_B < extreme_lower)
results
```

```
## [1] 4141
```

*Outliers for speed?*

I didn't find that there were any outliers for spped, so I didn't need to do much for it.

```
lowerq = quantile(battedBallData$speed_A, na.rm=TRUE)[2]
upperq = quantile(battedBallData$speed_A, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(battedBallData$speed_A > extreme_upper | battedBallData$speed_A < extreme_lower)
results
```

```
## [1] 4161
```

```
lowerq = quantile(battedBallData$speed_B, na.rm=TRUE)[2]
upperq = quantile(battedBallData$speed_B, na.rm=TRUE)[4]
iqr = upperq - lowerq
extreme_upper = (iqr*3) + upperq
extreme_lower = lowerq - (iqr*3)
results <- which(battedBallData$speed_B > extreme_upper | battedBallData$speed_B < extreme_lower)
results
```

```
## integer(0)
```

*Finding "true" average of speed-off-bat for EACH player.*

I started by making a new data frame that was grouped by the batters. I found the average of each batters data from systems A and B and then I found the weighted average of those two. I chose to do a weighted average because system A has been found to be much more accurate, but I think it is still important to take system B into account. I added the "true" averages I found to the data frame that I made.

```
batters <- cbind(aggregate(.~batter, battedBallData[,-c(2,3)], mean))
true_average <- (batters$speed_A*0.75) + (batters$speed_B*0.25)
batters$true_average <- true_average
```

*Find average speed and sd for each hit type*

```
trueav <- (battedBallData$speed_A*0.75) + (battedBallData$speed_B*0.25)

hitmean <- aggregate(trueav ~ hittype, battedBallData, mean, na.rm=TRUE)

hitsd <- aggregate(trueav ~ hittype, battedBallData, sd, na.rm=TRUE)
```

Created a data frame with the information that we will be working with.

```
df <- data.frame(battedBallData$batter, battedBallData$hittype, battedBallData$speed_A, battedBallData$
df <- na.omit(df)
df["point"] <- NA
```

I created a point system based on the type of hit and the exit velocity of each hit. I assigned more points for hit types based on the following order: line drives got the most, then fly balls, ground balls, and pop ups got the least amount of points. I decided to give line drives and fly balls more points because those types of hits usually lead to extra bases. Ground balls usually result in singles or outs, and pop ups are almost always outs. To put this scoring system into affect, I came up with a bunch of if/else statements to apply the points system based on a set of criterias. The scoring was based on how far the true averages were from the mean based on standard deviations. If it was greater than 1 standard devaition from the mean, then it would get a higher score because it was a more powerful hit. If it was higher than the mean, but not one full standard deviation higher, it was scored a little less. If it was a little less than the mean it would be a little less. Finally, if it was less than 1 standard deviation away from the mean than it would be scored the lowest because it was a weak hit.

```
linemean = 91.83464
linesd = 11.471882
lineabove = linemean + linesd
linebelow = linemean - linesd

groundmean = 82.56772
groundsd = 14.931219
groundabove = groundmean + groundsd
groundbelow = groundmean - groundsd

popmean = 72.99905
popsd = 11.323978
popabove = popmean + popsd
popbelow = popmean - popsd

flymean = 89.12745
```

```
flysd = 9.252213
flyabove = flymean + flysd
flybelow = flymean - flysd

for (row in 1:nrow(df)) {
  if(df[row, "battedBallData.hittype"] == "ground_ball"){
    if(df[row, "trueav"] == 'NA'){
      df[row, "point"] = 0
    } else if(df[row, "trueav"] > groundabove){
      df[row, "point"] = 6
    } else if(df[row, "trueav"] > groundmean){
      df[row, "point"] = 3
    } else if(df[row, "trueav"] < groundbelow){
      df[row, "point"] = 1
    } else if(df[row, "trueav"] < groundmean){
      df[row, "point"] = 0
    }
  } else if(df[row, "battedBallData.hittype"] == "line_drive"){
    if(df[row, "trueav"] == 'NA'){
      df[row, "point"] = 0
    } else if(df[row, "trueav"] > lineabove){
      df[row, "point"] = 8
    } else if(df[row, "trueav"] > linemean){
      df[row, "point"] = 6
    } else if(df[row, "trueav"] < linebelow){
      df[row, "point"] = 4
    } else if(df[row, "trueav"] < linemean){
      df[row, "point"] = 0
    }
  } else if(df[row, "battedBallData.hittype"] == "popup"){
    if(df[row, "trueav"] == 'NA'){
      df[row, "point"] = 0
    } else if(df[row, "trueav"] > popabove){
      df[row, "point"] = 4
    } else if(df[row, "trueav"] > popmean){
      df[row, "point"] = 2
    } else if(df[row, "trueav"] < popbelow){
      df[row, "point"] = 0
    } else if(df[row, "trueav"] < popmean){
      df[row, "point"] = 0
    }
  }else if(df[row, "battedBallData.hittype"] == "fly_ball"){
    if(df[row, "trueav"] == 'NA'){
      df[row, "point"] = 0
    } else if(df[row, "trueav"] > flyabove){
      df[row, "point"] = 7
    } else if(df[row, "trueav"] > flymean){
      df[row, "point"] = 5
    } else if(df[row, "trueav"] < flybelow){
      df[row, "point"] = 3
    } else if(df[row, "trueav"] < flymean){
      df[row, "point"] = 0
    }
```

```
  }
}
```

After assigning a point to each hit event, I found each individual batters average score and sorted those findings into descending order. The higher the point value, the harder the batter is hitting the ball and the more consistently they are hitting the ball at an angle that produces successful outcomes. Given more time, I could get more precise on the v angle and the more I could have cleaned up the data to make sure the hits were properly tagged with the correct hit type.

Below are the top 5 players that are projected to perform the best.

```
projection <- cbind(aggregate(.~battedBallData.batter, df[,-c(2, 3, 4)], mean))
projection <- projection[order(-projection$point),]
head(projection)
```

```
##     battedBallData.batter    trueav point
## 683                   697 110.70548     8
## 722                   737 104.25892     7
## 222                   225 101.44869     6
## 223                   226  94.38088     6
## 233                   236 104.03780     6
## 252                   256  98.46638     6
```