MLB Defensive Alignment Project Report

Members: Sam Bornstein and Paul Sirena

Project Objective:

In Major League Baseball (MLB), teams have recently discovered that shifting their defenders in accordance with a hitter's trends can help limit the number of hits allowed in a game. Many teams within the league have deployed limited shifts against polarizing types of hitters, but there is still more to be gained. In this project, we set out to create a custom defensive alignment for each hitter, while still adhering to realistic and traditional baseball philosophies.

Data Collection & Preparation:

The sport of baseball has evolved so rapidly over the last decade with the pitch-tracking devices used to measure pitch-flight, batted-ball, and much more data. In 2020, MLB introduced a new pitch-tracking device called Hawk-Eye, which caused some issues during the beginning of the already shortened season. For those reasons, we thought it would be appropriate to stick to the 2017-2019 season. Prior to the Hawk-Eye unit, MLB used Trackman (radar system) and cameras (Chyron Hego) to collect pitch-by-pitch data. All together these systems are called "Statcast," which is the name given by MLB to comprise their state-of-the-art technology and web application, Baseball Savant.

Because we decided to complete this analysis using three full seasons, we spent a considerable amount of time ensuring the data was collected precisely and appropriately. Statcast data can typically be pulled from https://baseballsavant.mlb.com/statcast_search by specifying the filters and executing the query. Since the website only allows up to 40,000 pitches to be pulled per query, it proved to be a complicated process. For reference, a typical MLB season sees

approximately 750,000 pitches. To combat this inconvenience, an R script was written to pull data by week from the 2017, 2018, 2019, and 2020 MLB seasons and combine them into two CSV files: one to be used as a training set (2017 & 2018) and one to be used as a test set (2019 & 2020). We will explain more on this later.

While each individual pitch tracked by Statcast results in 90 variables, only a handful were used in our analysis, which are listed and described below.

- launch_speed: velocity of the ball as it leaves the bat
- launch_angle: vertical angle of the ball as it leaves the bat
- hc_x: Statcast's x-coordinate of where the batted ball lands
- hc_y: Statcast's y-coordinate of where the batted ball lands
- bb_type: classification of hit type (groundball, line drive, fly ball, or pop up)
- events: outcome of the batted ball (out, single, double, etc.)

Each of these six variables play a role in building a model to construct optimal defensive alignments for a hitter. Both launch speed and launch angle are ingredients to estimate the exact location of a batted ball. The two hit coordinates variables (hc_x and hc_y) are commonly used on a scatter plot with a baseball field underlay to represent a "spray chart." This report includes several of these plots. Finally, the batted ball type (bb_type) and result of the play (events) play an important role in the model, both as further context and as a means of training the hit probability model (more on this later too).

There were two approaches used to find the best spot for a fielder, with bisecting k-means clustering giving an exact location for each fielder and a random forest model that shows where the hitter tends to hit balls with the best hit probability.

Before we dove into these two models, we loaded the necessary libraries in our Azure Databricks notebook. We used SparkR for modeling and data manipulation, ggplot2 for data visualization, sportyR for the baseball diamond visualization in the spray charts, and pROC for the ROC curve. Additionally, we used Paul's database, "psirena," to store the data. This folder was referenced at the beginning of the script.

```
library(SparkR)
library(ggplot2)
install.packages("sportyR")
library(sportyR)
library(pROC)
sql("use psirena")
```

There was also some filtering involved to make sure we had the best possible data to use.

```
train_raw <- sql("select * from savantdata1718_csv")
print(nrow(train_raw))
```

▸ (2) Spark Jobs

```
[1] 1428835
```

```
raw <- sql("select * from savantdata1920_csv")
print(nrow(raw))
```

▸ (2) Spark Jobs

```
[1] 994540
```

Basic SQL commands are all we had to do to get our training and testing data. With the data collected, we only had a few more steps.

```
train_inplay <- where(train_raw, train_raw$description=="hit_into_play")
print(nrow(train_inplay))
```

▸ (2) Spark Jobs

```
[1] 251535
```

To train the random forest model, we only need balls put in play from 2017 and 2018.

```
inplay <- where(raw, raw$description=="hit_into_play" & raw$events!="home_run" & raw$bb_type!="popup" & raw$hc_y>=140 & raw$hc_y <= 180)
print(nrow(inplay))
```

▸ (2) Spark Jobs

```
[1] 54874
```

Command took 9.16 seconds -- by psirena@uiowa.edu at 5/1/2021, 8:38:35 PM on s21_bais6110

```
inplay2 <- where(raw, raw$description=="hit_into_play" & raw$events!="home_run" & raw$bb_type!="popup" & raw$hc_y< 140)
print(nrow(inplay2))
```

▸ (2) Spark Jobs

```
[1] 84541
```

For k-means clustering, this gives us all the data we need to make clusters for the infielders and outfielders. Necessary filters have been added for hit location, which will be updated further to match what makes the spray chart accurate.

```
inplay_total <- where(raw, raw$description=="hit_into_play" & raw$events!="home_run" & raw$bb_type!="popup")
print(nrow(inplay_total))
```

▸ (2) Spark Jobs

```
[1] 148039
```

This is the training data for the random forest model. Home runs are not important since there is normally nothing fielders can do about them and pop ups are excluded since fielders have plenty of time to get to them.

Methodology:

As mentioned already, we used two approaches to find the best spot for a fielder: one which gave an exact location, represented by a black dot (as seen later on), and the other approach which showed an optimal zone, represented by individual clusters.

The first approach here involved bisecting k-means clustering. Upon initial exploration we found that the model had a mind of its own and did not understand baseball quite well, obviously. With that in mind, we had to add logical constraints to remain realistic in our observations. One example of this is for there to be a maximum distance allowed between first base and its nearest cluster. This is due to the fact that there needs to be a fielder close enough to first base to cover the bag and record an out for a ground ball in the infield. Similarly, there is no need to place one of the seven clusters near the pitcher's mound as there will always be a pitcher there to field any batted ball close to the mound. Another example is restricting the number of clusters populating in the outfield. Traditionally, baseball teams deploy three outfielders. In extremely rare cases we have seen teams send an infielder to the outfield to expand that to a four-man outfield. This is considered a radical idea by its own standards, so we have decided to follow along with these philosophies to keep this analysis realistic. With that being said, we restricted the number of clusters in the outfield to three and the number of clusters in the infield to four.

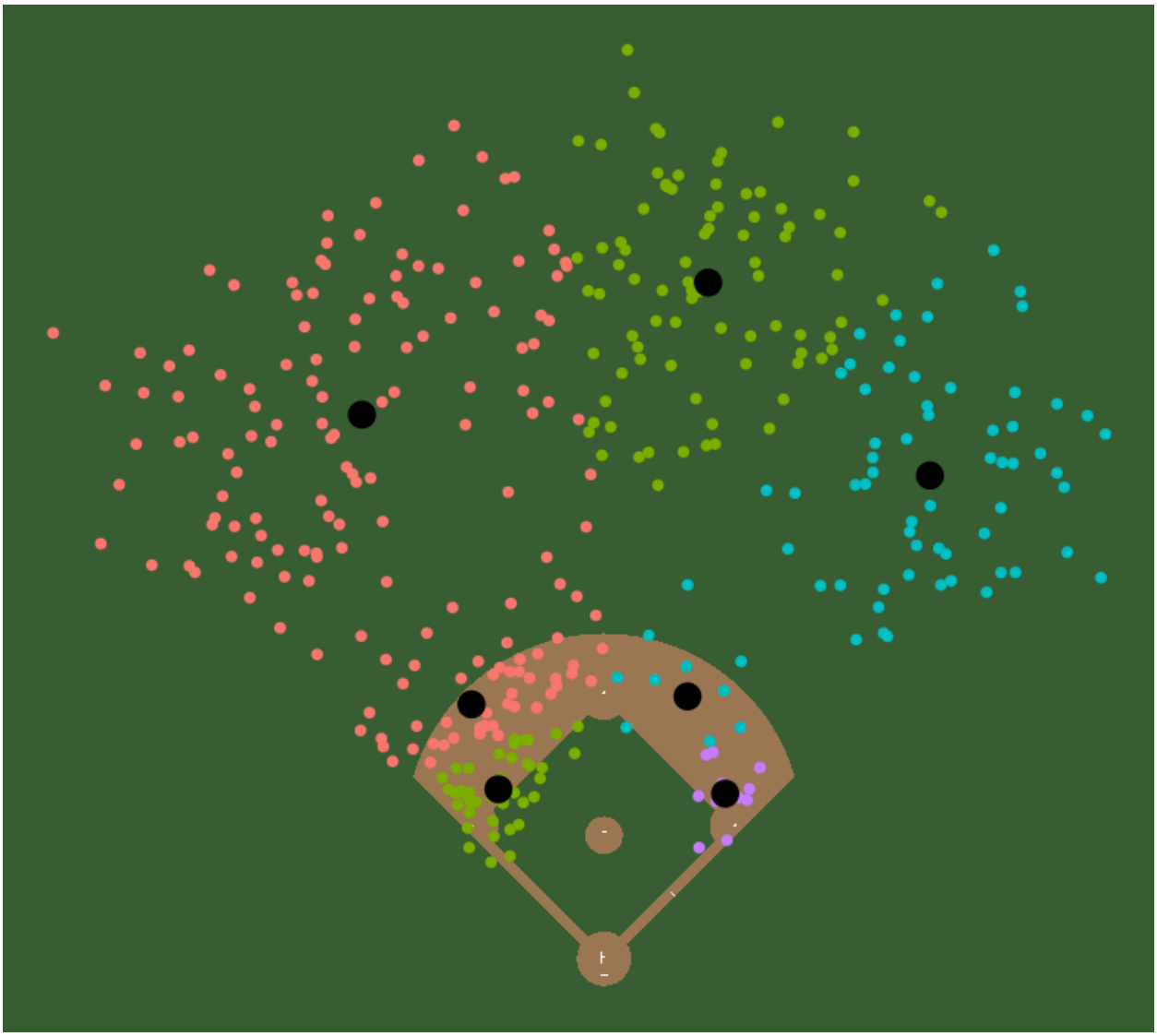
These seven clusters were created using batted ball data from the 2019 and 2020 seasons. To capture exactly where the ball was hit, only hc_x and hc_y were used to create the clusters. Exit velocity and launch angle were also considered, but these tended to lead to poor fielder placement due to their inability to factor in the direction of the batted ball. The result of this approach was two separate clustering models with one for balls hit on the infield and the other

for balls hit to the outfield. The function for this approach gives a plot for a specified hitter with centroid marked to represent the optimal fielding position.

```
SprayChartOptimizer <- function(batter) {
  test <- where(inplay, inplay$player_name==batter)
  test <- na.omit(select(test,test$hc_x,test$hc_y))
  test_bkm <- spark.bisectingKmeans(data = test, ~ hc_x + hc_y, k=4, maxIter=10, minDivisibleClusterSize=10)

  test2 <- where(inplay2, inplay2$player_name==batter)
  test2 <- na.omit(select(test2,test2$hc_x,test2$hc_y))
  test2_bkm <- spark.bisectingKmeans(data = test2, ~ hc_x + hc_y, k=3, maxIter=10, minDivisibleClusterSize=10)

  centers <- as.DataFrame(as.data.frame(cbind(summary(test_bkm)$coefficients[,1],summary(test_bkm)$coefficients[,2])))
  names(centers) <- c("hc_x","hc_y")

  centers2 <- as.DataFrame(as.data.frame(cbind(summary(test2_bkm)$coefficients[,1],summary(test2_bkm)$coefficients[,2])))
  names(centers2) <- c("hc_x","hc_y")

  output_bkm <- predict(test_bkm,test)
  output2_bkm <- predict(test2_bkm,test2)

  output_bkm_r <- collect(output_bkm)
  output2_bkm_r <- collect(output2_bkm)
  centers_r <- collect(centers)
  centers2_r <- collect(centers2)

  plot <- geom_baseball(league="MLB") +
  geom_point(data=output_bkm_r, aes(x= 2.5 * (hc_x - 125.42), y=2.5 * (198.27 - hc_y), color=factor(prediction))) +
  geom_point(data=centers_r,aes(2.5*(hc_x-125.42),2.5*(198.27-hc_y),fill="black",size=18)) +
  geom_point(data=output2_bkm_r, aes(x= 2.5 * (hc_x - 125.42), y=2.5 * (198.27 - hc_y), color=factor(prediction))) +
  geom_point(data=centers2_r,aes(2.5*(hc_x-125.42),2.5*(198.27-hc_y),fill="black",size=18))

  return(plot)
}
```

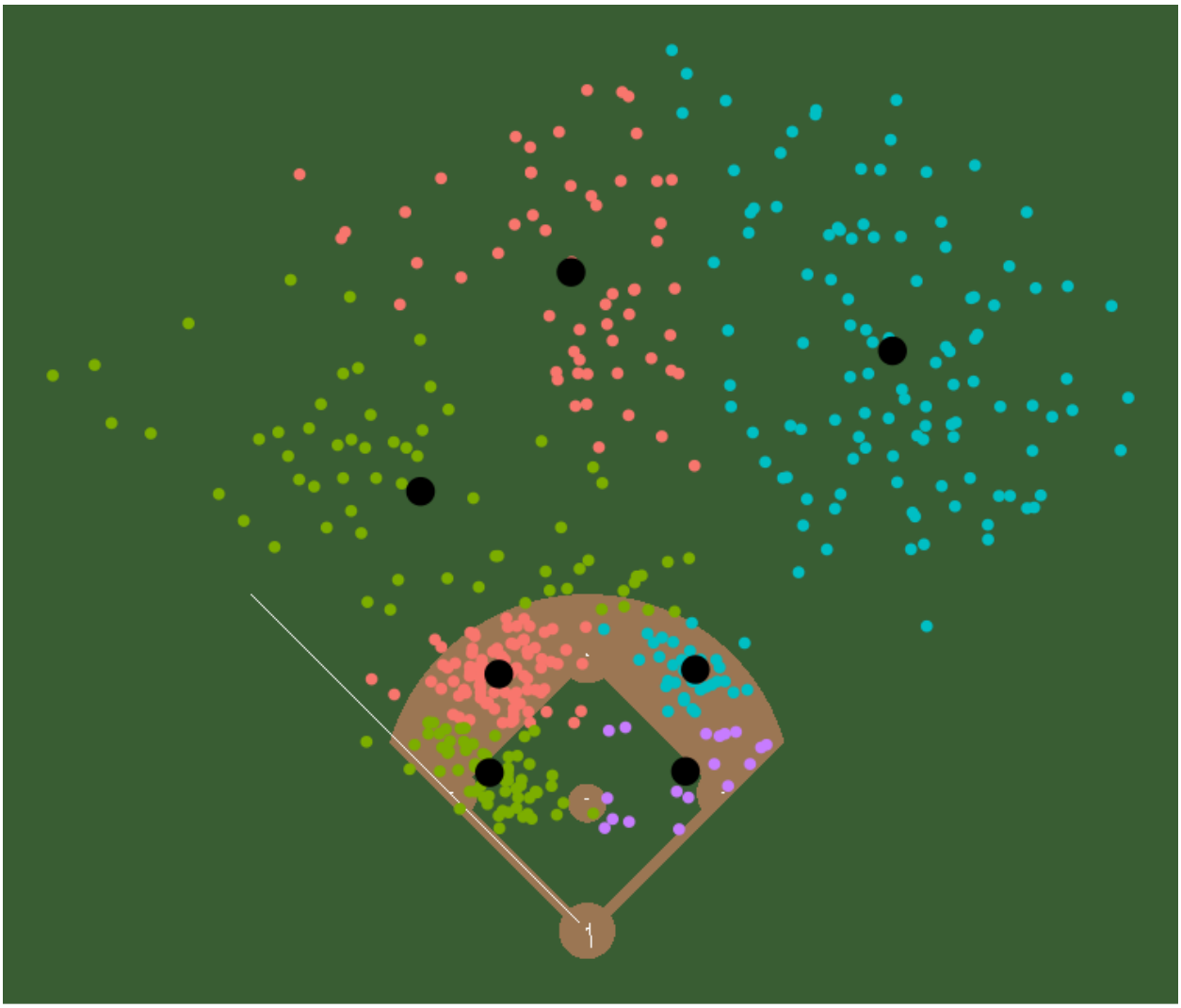
This will now be tested on two example hitters, Mike Trout and Wilson Ramos.

```
SprayChartOptimizer("Trout, Mike")
```



This example plot comes from batted ball data for Mike Trout with home runs removed. We can see that on balls hit on the infield, a slight shift towards third base should be applied for the shortstop. To the outfield, we see that the outfielders should be shifted slightly to the right with a lot of damage going to right-center field.

```
SprayChartOptimizer("Ramos, Wilson")
```



We also thought it would be interesting to look at Wilson Ramos, who had the lowest average launch angle in the 2019 season. The infielders should be playing a standard alignment. If we were able to account for speed, we could move everyone back a little since Ramos is one of the slowest players in the league. The outfield alignment is more interesting and quite different from Trout. The left fielder can play very shallow, while the center fielder and right fielder can play straight up.

The second approach is to calculate hit probability and show where each hitter tends to hit balls that have the best chance of being a hit. Unlike with k-means clustering, we will not have

specific points for each fielder and will use only one model since exit velocity and launch angle are important in addition to hit location. We will also include popups since their hit probabilities may show interesting results. The model was trained on data from 2017 and 2018 and tested on the same 2019-2020 data as the first model.

```
train_raw$hit <- ifelse(train_raw$events %in% c("single","double","triple","home_run"),1,0)
inplay_total$hit <- ifelse(inplay_total$events %in% c("single","double","triple","home_run"),1,0)
xBA_rf <- spark.randomForest(train_raw, hit ~ launch_speed + launch_angle + hc_x + hc_y,"classification",
                             numTrees = 500, maxDepth = 5, handleInvalid = "skip")
summary(xBA_rf)
```

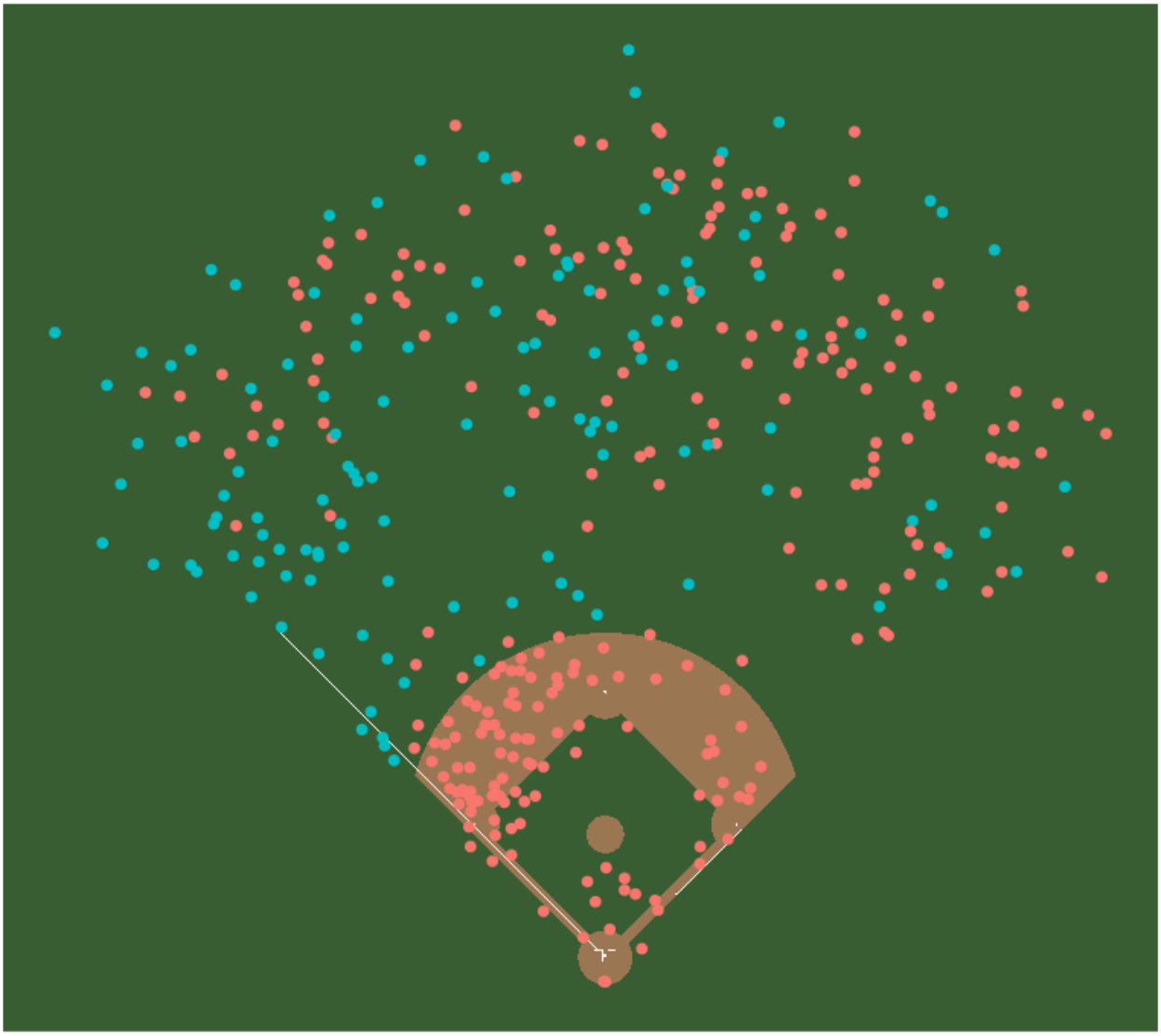
The “hit” variable has been created to easily show if the batted ball resulted in a hit. This is what we are trying to predict with the model. We will be able to predict if every batted ball is a hit and extract only those of the specified hitter.

```
output_rf <- predict(xBA_rf,inplay_total)

xBA <- function(batter) {
  batterdata <- where(output_rf,output_rf$player_name==batter)
  batterdata <- select(batterdata,batterdata$launch_speed,batterdata$launch_angle,
                       batterdata$hc_x,batterdata$hc_y,batterdata$hit,batterdata$prediction)

  output_rf_r <- collect(batterdata)
  accurate <- where(batterdata,batterdata$prediction == batterdata$hit)

  plot <- geom_baseball(league="MLB") +
  geom_point(data=output_rf_r, aes(x= 2.5 * (hc_x - 125.42), y=2.5 * (198.27 - hc_y), color=factor(prediction))) +
  ggtitle(paste("Accuracy: ",nrow(accurate)/nrow(batterdata)))
  return(plot)
}
```

This function will be tested on the same two players, Mike Trout and Wilson Ramos.

```
xBA("Trout, Mike")
```

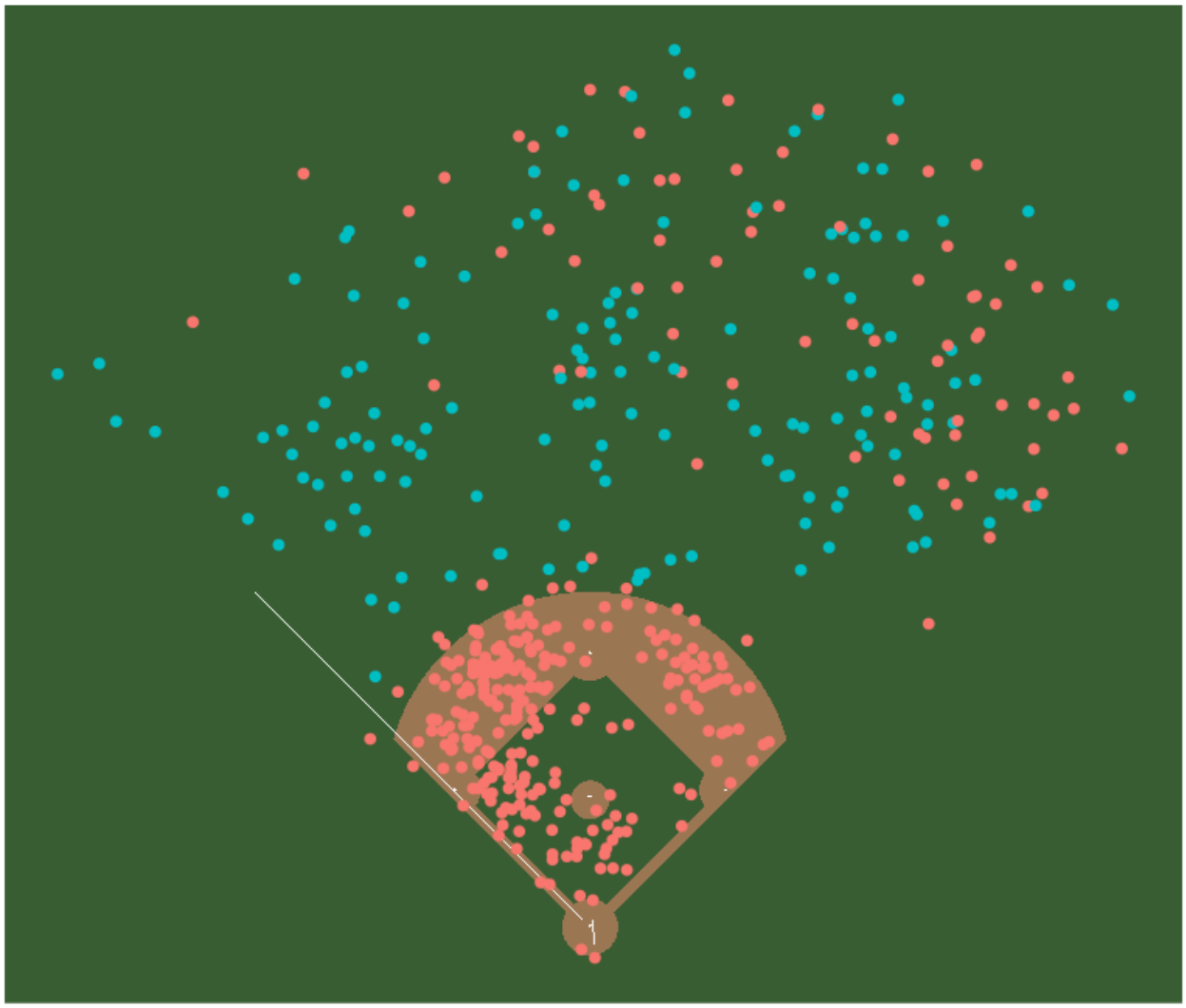
Accuracy: 0.861618798955614



This is Mike Trout's plot with red points being projected outs and blue points being projected hits based on hit probability from the random forest model. The accuracy for Trout's batted balls is about 86%. This spray chart suggests more of an infield shift than the first model since most projected hits are to the right side. The third baseman should be playing close to the left field line, the shortstop should be in shallow left field, and the second baseman should be up the middle. The first baseman will have to stay near first base to cover it and field balls hit down the right field line.

```
xBA("Ramos, Wilson")
```

```
Accuracy:  0.90187891440501
```



This is the plot for Wilson Ramos using the random forest model. Note that the accuracy for his batted balls is slightly higher than Trout's at about 90%. His ground ball frequency combined with his lack of speed really decreases his value. Teams can play him deep with a slight shift to the left on the infield. The outfield shows a similar pattern to the k-means model with shallow left field showing some well-hit balls, while poorly-hit balls to the outfield are mostly to the right side. We would recommend the exact same alignment with this approach.

Analysis:

Evaluating our bisecting k-means model is exceedingly difficult. One thing we notice with our infield and outfield clusters is that there is a good amount of space between each fielder's position. Earlier attempts had some instances where fielders were positioned shoulder to shoulder or with one outfielder directly in front of the other. Avoiding this in the model was critical to providing meaningful analysis and we accomplished this successfully.

The random forest model was easier to evaluate. We first measure the model's accuracy, precision, and recall. While the two plots shown above have the accuracy for the specific players, we will be using the overall results to get these measurements.

```
TP <- nrow(where(output_rf, output_rf$hit == 1 & output_rf$prediction == 1))
FP <- nrow(where(output_rf, output_rf$hit == 0 & output_rf$prediction == 1))
TN <- nrow(where(output_rf, output_rf$hit == 0 & output_rf$prediction == 0))
FN <- nrow(where(output_rf, output_rf$hit == 1 & output_rf$prediction == 0))

Accuracy <- (TP + TN) / nrow(output_rf)
Precision <- TP / (TP + FP)
Recall <- TP / (TP + FN)

print(paste("Accuracy: ", Accuracy))
print(paste("Precision: ", Precision))
print(paste("Recall: ", Recall))
```

Accuracy	Precision	Recall
.859	.789	.763

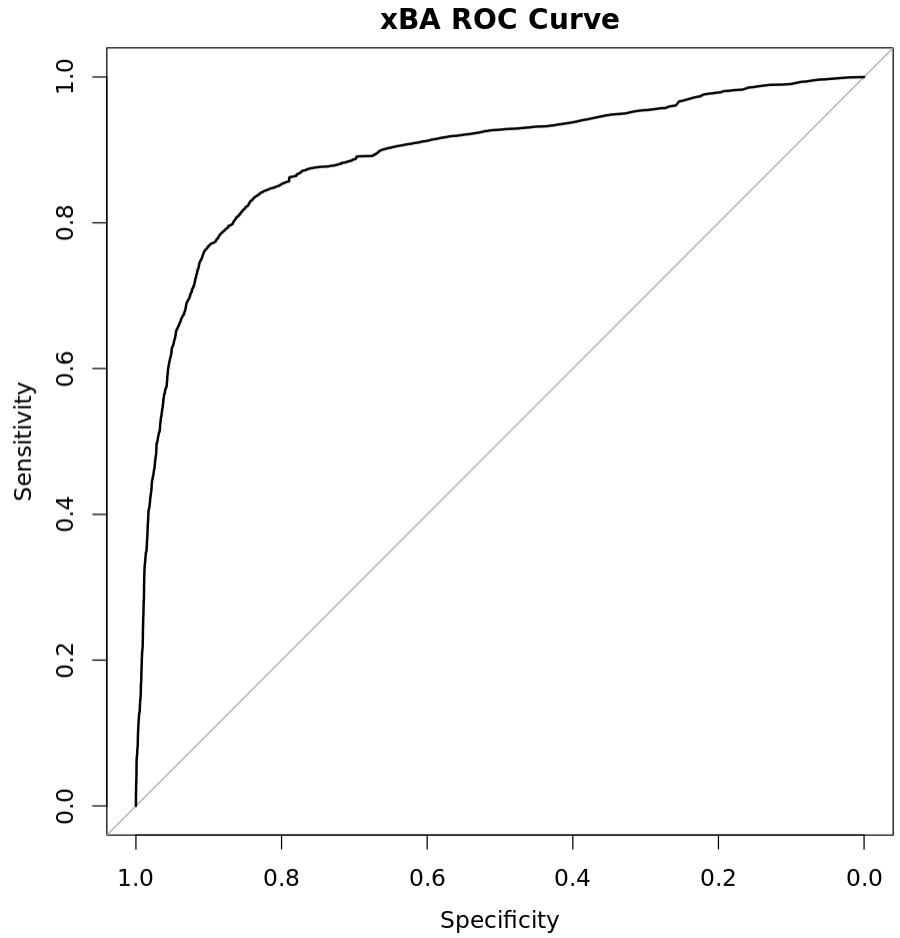
This model is much better than guessing. Overall, it is about 85.9% accurate. It is also about 78.9% accurate when predicting a hit (Precision) and correctly predicts about 76.3% of all hits. The last step is to find the area under the ROC curve for this model.

```
output_rf_r <- collect(output_rf)
output_rf_r$predPosProb = 0
for(i in 1:nrow(output_rf_r)){
  output_rf_r$predPosProb[i]=output_rf_r$probability[[i]][["values"]][[2]]
}
auc(output_rf_r$hit, output_rf_r$predPosProb)
```

This gives us the area under the ROC curve we will produce.

```
plot.roc(output_rf_r$hit,output_rf_r$predPosProb,main="xBA ROC Curve")
```

This gives us the ROC curve shown below.



The goal is to have the area under this curve be greater than the area under the line with area .5, which resembles what would happen if we just guessed. The area under this curve is .8892, which is an excellent result and shows that we have done an excellent job predicting hits.

Conclusion:

It is impossible to know before the pitch exactly where the ball is going to be hit, but baseball is a game of odds and is friendly to machine learning techniques such as the ones explored in this report. We have toyed the line with radical ideas to implement into fielder positioning, and while we don't have a means to deploy these optimal shifts against the best baseball players in the world, we enjoyed conducting this analysis. We believe that interpreting trends in hitters to find optimal placement for each fielder will be valuable to MLB teams. These organizations are already years into deploying significant shifts within their defensive alignments, but these are mostly determined based the frequency of all batted balls in the aggregate. Our method of setting clusters in the highest hit probability locations serves as a yet to be used tool and one we hope to be implemented in years to come to prevent batted balls from turning into base-hits.

Bibliography

Ross Drucker (2021). sportyR: Plot Scaled 'ggplot' Representations of Sports Playing Surfaces. R package version 1.0.1. https://CRAN.R-project.org/package=sportyR

Willman, Daren. “Baseball Savant: Trending MLB Players, Statcast and Visualizations.” *Baseball Savant*, MLB Advanced Media, 2 May 2021, baseballsavant.mlb.com/.