Statcast 2023 Cluster Analysis

Lance Brady

Introduction

Upon investigation of the Statcast 2023 Custom Leaderboards website, I previously performed Principal Components Analysis and found three main components for separating hitters: the "Get On Base" component, the "Quality of Contact" component, and the "Plate Approach" component. The next step in this process would be to determine which players have closely related statistics, and which players are different from each other. This is where cluster analysis comes in, which will allow us to group players based on their statistics and find out what "type" of a hitter they may be. I expect to see three main groups: power hitters, contact hitters, and hitters who focus on getting on base, simply based on general baseball knowledge. I will only use the best 20 hitters of 2023 (according to xwoba) for this analysis. Using Cluster Analysis and k-means clustering, I found 4 main clusters: The All-Around Hitter (Get on Base, Hit for Power), Solid Contact Hitters, Free Swingers (who often make contact), and Power Hitters.

Research Question

What distinguishes the best 20 hitters (according to xwoba) of 2023?

Variables Used

The variables originally used in this analysis are:

1. player_age: Player Age
2. pa: Plate Appearances
3. k_percent: Strikeout percentage
4. bb_percent: Walk percentage
5. babip: Batting Average on Balls in Play
6. b_intent_walk: Intentional Walks
7. xba: Expected Batting Average
8. xslg: Expected Slugging Percentage
9. xwoba: Expected Weighted On-Base Average
10. xobp: Expected On-Base Percentage
11. xiso: Expected Isolated Power
12. xbacon: Expected Batting Average on Contact
13. sweet_spot_percent: Sweet Spot Percentage
14. barrel_batted_rate: Barrel Batted Rate
15. hard_hit_percent: Hard Hit Percentage
16. avg_best_speed: EV50
17. avg_hyper_speed: Adjusted EV
18. whiff_percent: Whiff Percentage
19. swing_percent: Swing Percentage

Some of these variables are taken out in the final analysis, as they are not as relevant to the clustering.

Load in Libraries

```
par(ask=FALSE)
# Load relevant Cluster Analysis libraries
library(aplpack)
library(fpc)
library(cluster)
library(ape)
library(amap)

# We will also be using the tidyverse
library(tidyverse)
```

Load in Data

```
statcast2023Cluster_unclean = read.csv("statcastbatting2023.csv")
```

Check Dimensions and Variable Names

```
dim(statcast2023Cluster_unclean)
```

```
## [1] 134  22
```

```
names(statcast2023Cluster_unclean)
```

```
##  [1] "last_name..first_name" "player_id"             "year"
##  [4] "player_age"            "pa"                    "k_percent"
##  [7] "bb_percent"            "babip"                 "b_intent_walk"
## [10] "xba"                   "xslg"                  "xwoba"
## [13] "xobp"                  "xiso"                  "xbacon"
## [16] "sweet_spot_percent"    "barrel_batted_rate"    "hard_hit_percent"
## [19] "avg_best_speed"        "avg_hyper_speed"       "whiff_percent"
## [22] "swing_percent"
```

We have 134 players with 22 variable columns. We will not need `year`, as I only selected 2023 data. We will also not need `player_id` as we will focus only on the measured data (and already have a names column). We will also edit the names column so that it is cleaner for analysis. We will only use the players with the top 20 xwoba so that we can better understand our visualizations.

Let's first take out unnecessary columns, fix the player names variable.

```
## Change player names to be more readable
## Create last name and first name column using tidyverse
statcast2023Cluster_unclean = statcast2023Cluster_unclean %>%
  separate(last_name..first_name, c("last_name", "first_name"), sep = ", ",
           remove = FALSE) %>%
  ## Then use paste() function to concatenate the two columns
  mutate(player_name = paste(first_name, last_name, sep = " ")) %>%
  ## Relocate player_name to the first column
  relocate(player_name)

statcast2023Cluster_unclean = statcast2023Cluster_unclean[, -c(2:6)]
```

Select the top 20 xwoba players.

```
statcast2023Cluster = statcast2023Cluster_unclean %>%
  arrange(desc(xwoba)) %>%
  head(20)
```

Then, take out `xwoba` as a statistic.

```
statcast2023Cluster = statcast2023Cluster[ , -10]
```

MODIFICATION: After doing some Cluster Analysis (as seen below), I have decided to take out the `PA` and `player_age` columns, as they are not as relevant to the clustering. They do not really have to do with the "type" of hitter that someone is. I decided to do this modification here as to not re-download another csv file. We only included qualified hitters, which means hitters who had a minimum of 3.1 Plate Appearances per team game.

```
statcast2023Cluster = statcast2023Cluster[ , -c(2:3)]
```

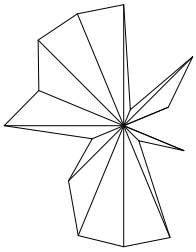
Finally, we will standardize the data, to make sure that all variables are on the same scale. This is important for clustering, as variables that are on different scales can have different impacts on the clustering algorithm.

```
statcast2023Cluster[ , -1] = scale(statcast2023Cluster[ , -1])
```

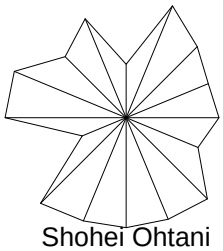
Cluster Analysis with Stars Method

One of the more whimsical methods of clustering is the use of the `stars` function from the `aplpack` package. This function creates a radar chart for each player, with each variable represented by a spoke. The length of the spoke represents the value of the variable.

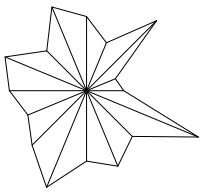
```
#Extend 'arms' to distance of each variable
stars(statcast2023Cluster[, -1], labels = statcast2023Cluster$player_name)
```



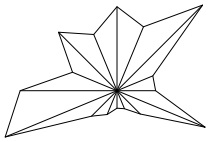
Ronald Acuña Jr.



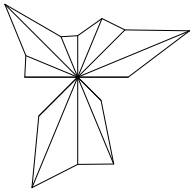
Shohei Ohtani



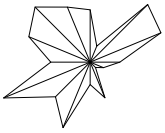
Corey Seager



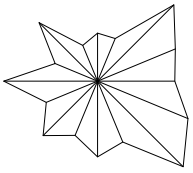
Freddie Freeman



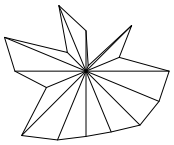
Juan Soto



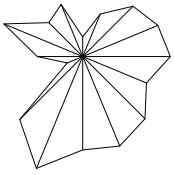
Mookie Betts



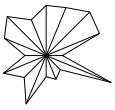
Bryce Harper



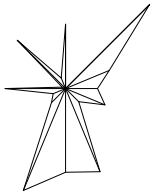
Marcell Ozuna



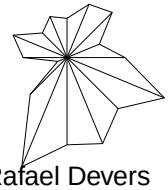
Matt Olson



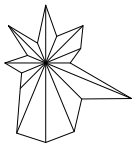
Kyle Tucker



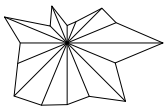
Yandy Díaz



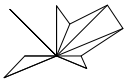
Rafael Devers



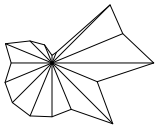
Vladimir Guerrero Jr.



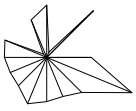
Jorge Soler



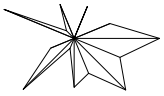
Adley Rutschman



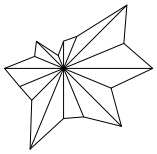
Triston Casas



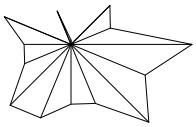
Bobby Witt Jr.



Pete Alonso



Paul Goldschmidt



Adolis García

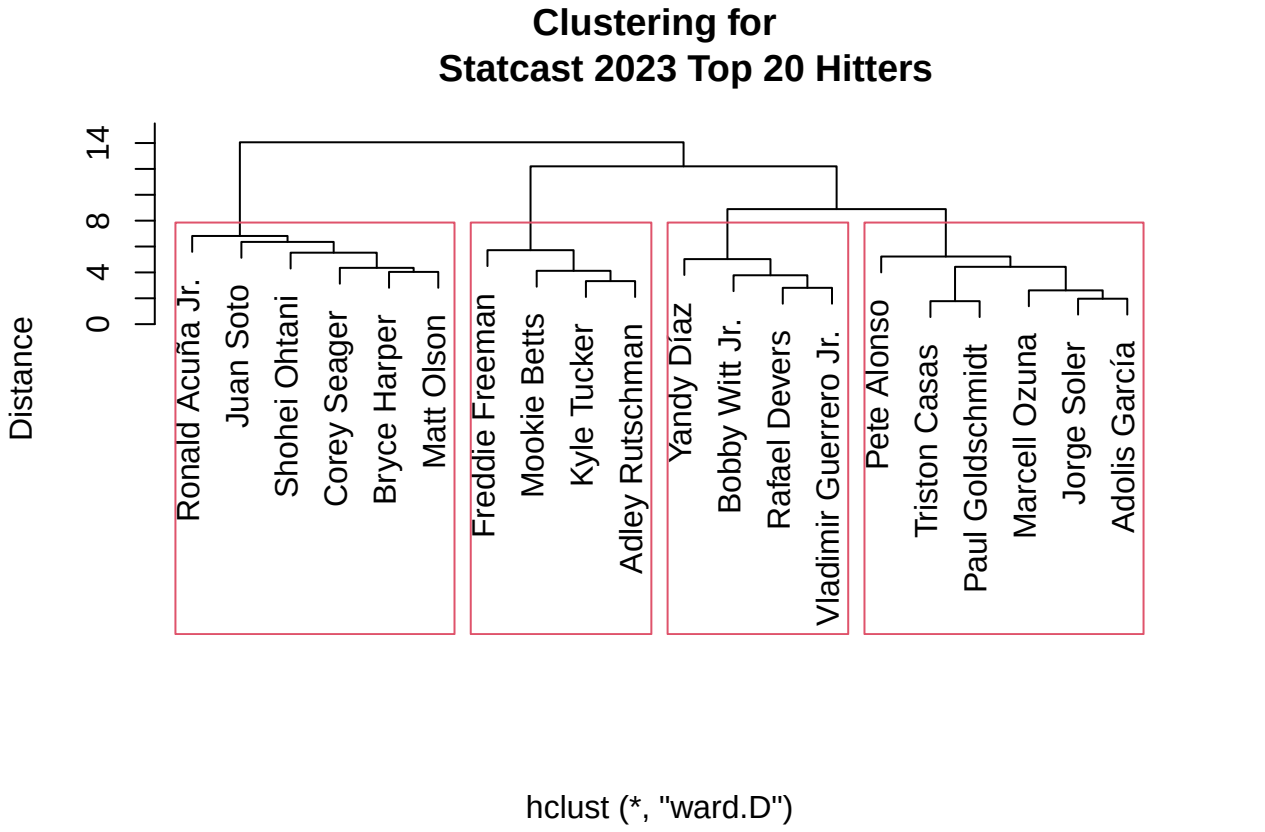
The Star Clustering Algorithm is, in general, much harder to read (which is why we will use clearer dendrograms later), but they do give a good sense of some particular differences between players. For example, we can see that players like Matt Olson, Paul Goldschmidt, and Pete Alonso have smaller left spokes and large right spokes, indicating that they may be similar types of hitters. This makes sense anecdotally, as they are all power-hitting first basemen.

However, it is more likely that Hierarchical Clustering will be of much more use to us.

Hierarchical Clustering Analysis

For our Hierarchical Clustering, we will first use Euclidean multivariate distance (root sum-of-squares differences on each variable) to figure out the distance between all pairs of objects. We will then use the `hclust` function to perform the clustering, using Ward's Method, which minimizes internal sum of squares. We chose Ward's Method as several studies of clustering suggest that Ward's method works the "best". We will make a dendrogram to visualize the clusters, and put rectangles around 4 possible groups, as initial viewing of the dendrogram suggest 4 groups may be optimal.

```
## Get Euclidean distance matrix
statcast2023.dist = dist(statcast2023Cluster[ , -1], method = "euclidean")

## Perform cluster analysis
statcast2023.hclust = hclust(statcast2023.dist, method = "ward.D")

## Make dendrogram
statcast2023.hclust$labels <- as.character(statcast2023Cluster[ , 1])
plot(statcast2023.hclust, xlab = "",ylab = "Distance", main = "Clustering for
     Statcast 2023 Top 20 Hitters")
## identify three groups
rect.hclust(statcast2023.hclust, k = 4)
```



Grouping these hitters into four clusters seems to be a good fit for this data, as there are fairly large distances between the clusters. Based on this dendrogram, we can imagine four clusters:

Cluster #1: The All-Around Hitter (Get on Base, Hit for Power)

Players: Ronald Acuña Jr., Juan Soto, Shohei Ohtani, Corey Seager, Bryce Harper, Matt Olson

Notes on Defining Characteristics of Cluster #1 (after looking at data frame):

- Above Average xOBP
- Above Average Barrel Rate
- Above Average Hard Hit %
- Below Average Sweet Spot %
- Average K%
- Above Average BB%
- Above Average xSLG
- Above Average xISO
- High EV50
- High Adjusted EV

Overall Characterization of Cluster #1:

This cluster is made up of some all-around great hitters. They hit the ball hard and with power, as evidenced by their high barrel rates, hard hit percentages, expected isolated power, and expected slugging percentages. They also have high expected on-base percentages, and are able to walk at a high rate. Overall, these players get on base at a high rate, and have the power to hit for extra bases and home runs.

Cluster #2: Solid Contact Hitters

Players: Freddie Freeman, Mookie Betts, Kyle Tucker, Adley Rutschman

Notes on Defining Characteristics of Cluster #2 (after looking at data frame):

- Low K Percentage
- Above Average Walk Percentage
- High xBA
- High xOBP
- Low xISO
- High Sweet Spot %
- Low Barrel Rate
- Below Average Hard Hit Percentage
- Low EV50
- Low Adjusted EV
- Low Whiff Percentage

Overall Characterization of Cluster #2:

In term of their plate approach, these hitters do not whiff or strike out a lot compared to other top hitters, and walk at an average rate. With these characteristics and their high Sweet Spot percentage, they are likely to hit the ball with optimal launch angles and make solid contact (even with low exit velocities), allowing them to get on base at a fairly high clip. Overall, these hitters make solid contact and get on base at a high rate. They have the capabilities to hit homers, but are not necessarily power hitters, with a low xISO, EV50, Barrel Rate, Adjusted EV.

Cluster #3: Free Swingers (who often make contact)

Players: Yandy Díaz, Bobby Witt Jr., Rafael Devers, Vladimir Guerrero Jr.

Notes on Defining Characteristics of Cluster #3 (after looking at data frame):

- Low K%
- Low BB%
- Average xBA

- Below Average xSLG
- Below Average xISO
- Low xBACON
- Low Sweet Spot %
- Low Barrel Rate
- Above Average Hard Hit %
- Above Average EV50
- Above Average Adjusted EV
- Below Average Whiff %
- Above Average Swing %

Overall Defining Characteristics of Cluster #3:

These hitters have somewhat low power numbers (xSLG, xISO), but they still hit the ball hard, although not with optimal launch angles (low Sweet Spot % and Barrel Rate). They swing a lot, but do not miss often, leading to low K% and low BB%. Overall, these are free swingers who often make contact. They have high exit velocitieis but low launch angle.s

Cluster #4: Power Hitters

Players: Pete Alonso, Triston Casas, Paul Goldschmidt, Marcell Ozuna, Jorge Soler, Adolis García

Notes on Defining Characteristics of Cluster #4 (after looking at data frame):

- High K%
- Average BB%
- Low xBA
- Below Average xSLG
- Below Average xOBP
- Above Average xISO
- Below Average Hard Hit %
- Below Average EV50
- Above Average Barrel %
- Below Average Adjusted EV
- High Whiff %
- Average Swing %

Overall Defining Characteristics of Cluster #4:

These players do not swing an extreme amount, but when they do, they are swinging for the fences. Even with EV50 and Adjusted EV not being too high, they have high barrel rates and expected isolated power, meaning they are achieving optimal launch angles regardless of consistently hitting the ball incredibly hard. Knowing these players, they do hit the ball harder than most when they make excellent contact (High MAX EV), but they whiff enough and do not make solid contact enough for their EV50 and Adjusted EV to be above average. They have high strikeout rates and only average walk rates, leading to below average xOBP. They are power hitters who hit the ball hard when they make contact, but they do not make contact as often as other top hitters.

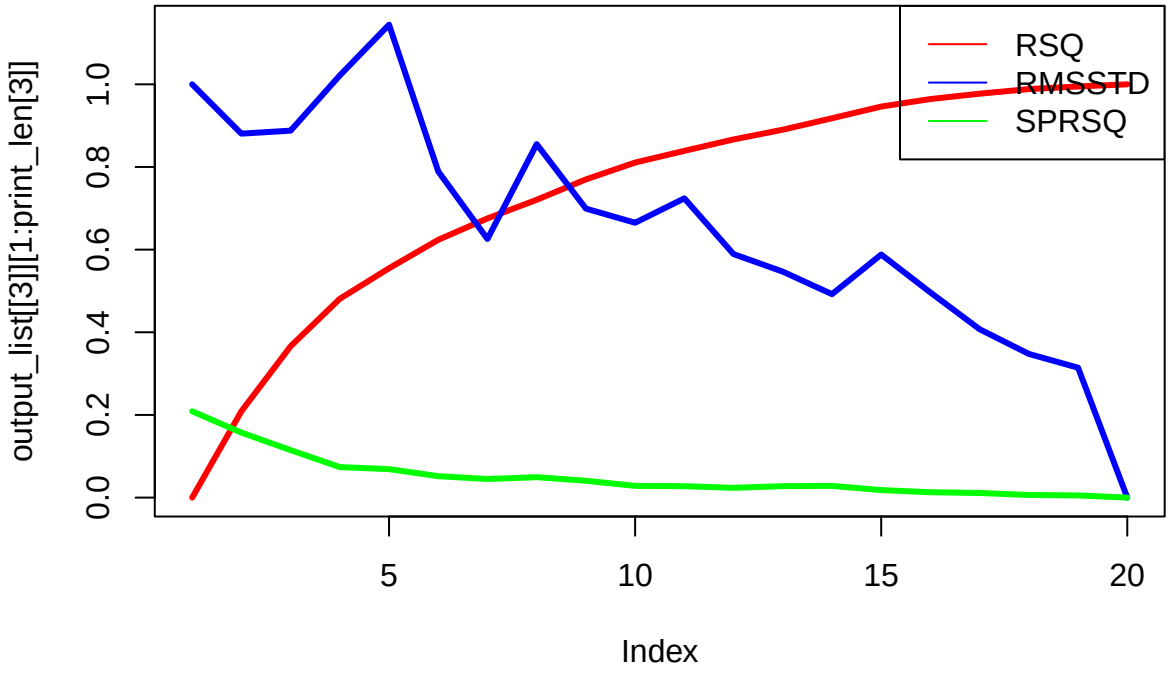
Getting an "Optimal" Number of Clusters

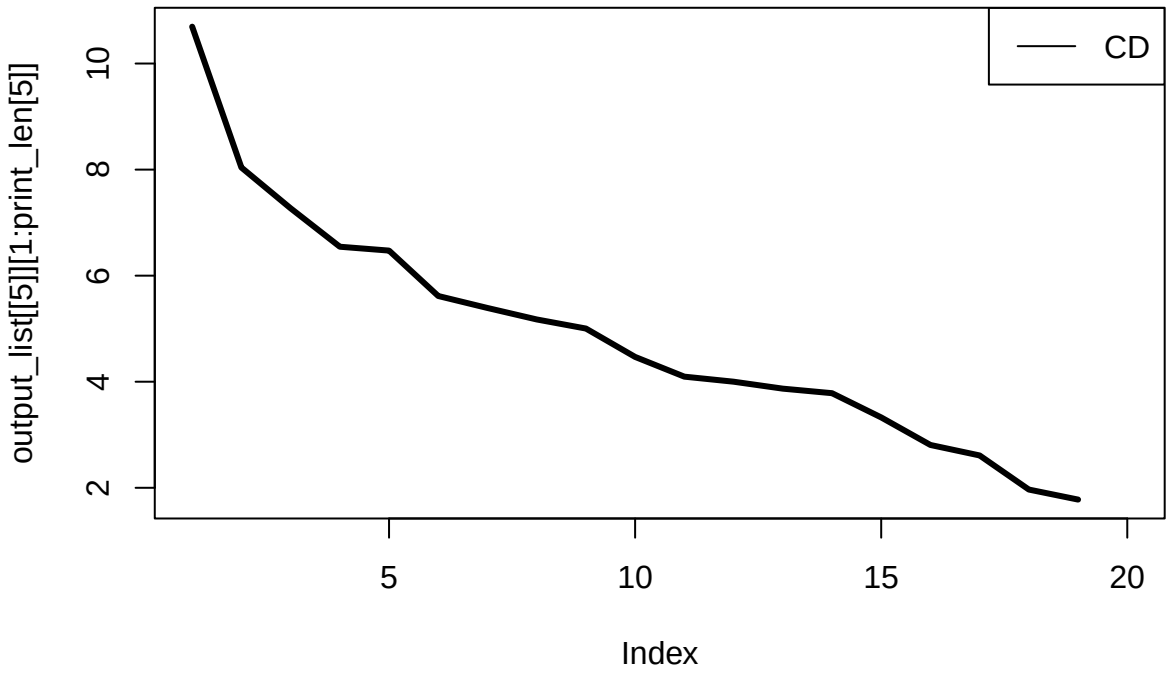
Now that we have a general idea of what the clusters look like, we can try to find an optimal number of clusters. This works because we already standardized our data. We will look up to 20 clusters.

```r
source("http://reuningscherer.net/multivariate/R/HClusEval3.R.txt")
rownames(statcast2023Cluster) = statcast2023Cluster$player_name
statcast2023Cluster = statcast2023Cluster[ , -1]
```

```r
#Call the function, which only works with complete linkage (Furthest neighbors)
hclus_eval(statcast2023Cluster, dist_m = 'euclidean', clus_m = 'complete', plot_op = T, print_num = 20)
```

```
## [1] "Creating Distance Matrix using euclidean"
## [1] "Clustering using complete"
## [1] "Clustering Complete. Access the Cluster object in first element of output"
## [1] "Calculating RMSSTD"
## [1] "RMSSTD Done. Access in Element 2"
## [1] "Calculating RSQ"
## [1] "RSQ Done. Access in Element 3"
## [1] "Calculating SPRSQ"
## [1] "SPRSQ Done. Access in Element 4"
## [1] "Calculating Cluster Dist. "
## [1] "CD Done. Access in Element 5"
```






```
## [[1]]
##
## Call:
## hclust(d = dist1, method = clus_m)
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 20
##
##
## [[2]]
##  [1] 1.0000000 0.8806267 0.8878898 1.0216028 1.1442377 0.7891524 0.6258130
##  [8] 0.8552561 0.6992538 0.6650270 0.7240489 0.5891550 0.5467645 0.4922434
## [15] 0.5881044 0.4962923 0.4073816 0.3476274 0.3142533 0.0000000
##
## [[3]]
##  [1] 0.0000000 0.2088559 0.3661590 0.4810688 0.5548837 0.6237932 0.6755701
##  [8] 0.7205751 0.7699793 0.8106456 0.8389959 0.8665878 0.8901616 0.9178217
## [15] 0.9461659 0.9643694 0.9773329 0.9884421 0.9948024 1.0000000
##
## [[4]]
##  [1] 0.208855939 0.157303097 0.114909755 0.073814899 0.068909470 0.051776977
##  [7] 0.045004991 0.049404168 0.040666343 0.028350267 0.027591937 0.023573751
## [13] 0.027660142 0.028344179 0.018203518 0.012963476 0.011109197 0.006360254
## [19] 0.005197639 0.000000000
##
## [[5]]
##  [1] 10.694932  8.039677  7.270336  6.545706  6.472786  5.616311  5.390659
##  [8]  5.174366  5.002941  4.468013  4.095839  4.000415  3.869591  3.783700
## [15]  3.326821  2.807453  2.609843  1.966478  1.777685
```

Observations:

Based on Root-Mean-square Standard Deviation, we would like to have about 2 clusters, as there is a local minimum there. Based on R Squared, we see diminishing returns around cluster 4, so this method suggests 4 clusters. Based on Semi-Partial R-Squared, we see an elbow around cluster 4, so this method suggests 4 clusters. Based on Cluster Distance, we see an elbow at cluster 4, so this method suggest 4 clusters. Based on these methods, we suggest either 2 or 4 clusters.

K Means Clustering

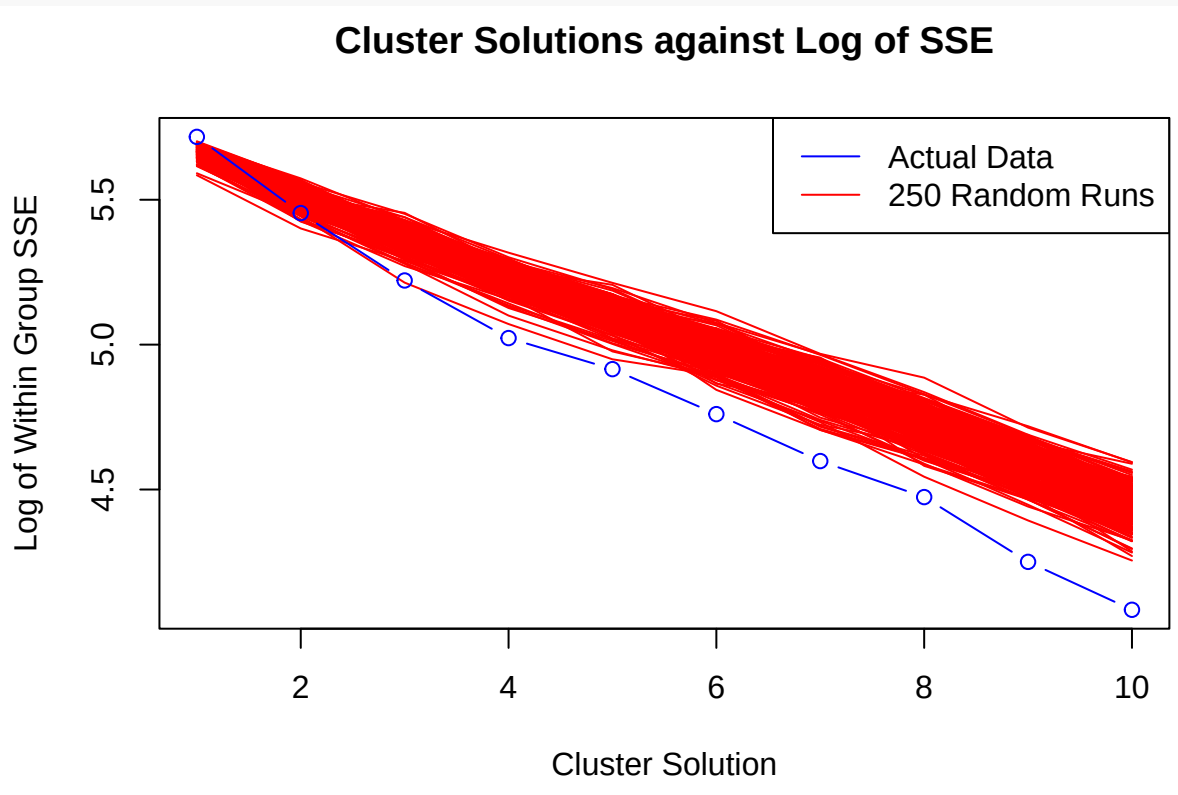
Let's run k-means clustering on the data. We will try it with 4 clusters.

```
# Create k-means clusters
km1 <- kmeans(statcast2023Cluster, centers = 4)
```

Now, let's make some graphs to test our k-means clustering. Use the following code to make the important graphs:

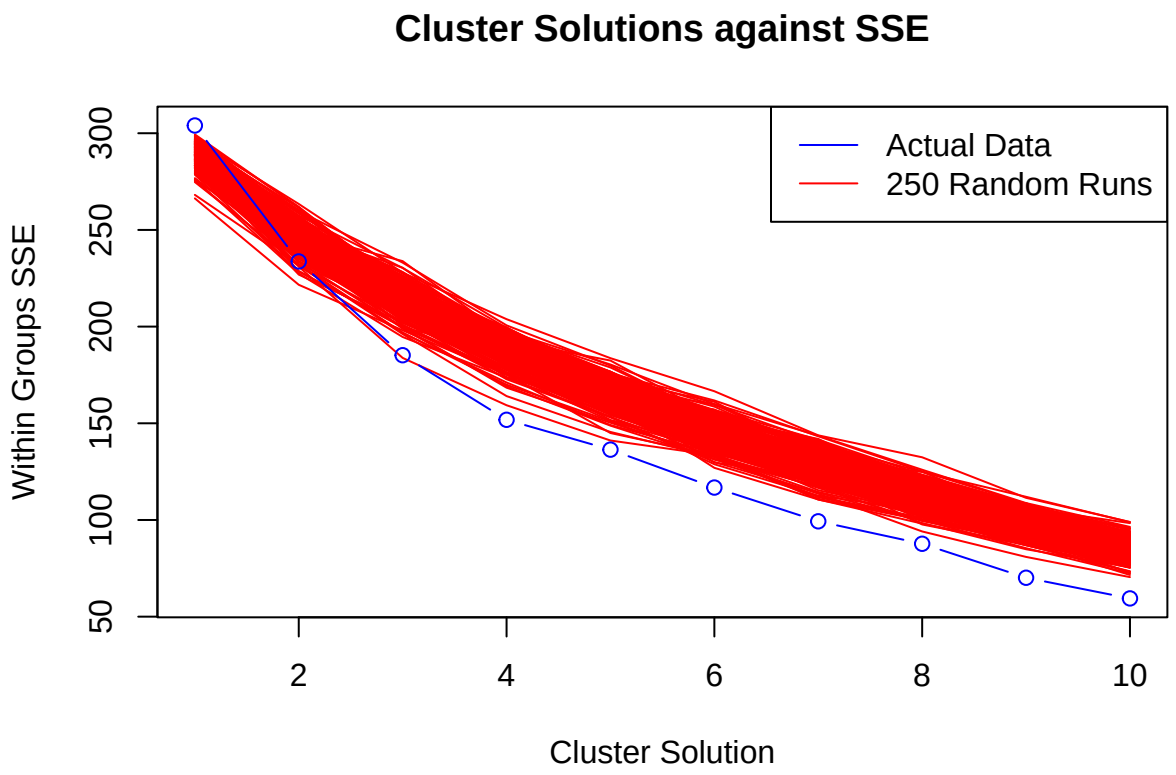
```
kdata <- data.matrix(statcast2023Cluster)

n.lev <- 10  #set max value for number of clusters k

## Calculate the within groups sum of squared error (SSE)
## for the number of cluster solutions selected by the user
wss <- rnorm(10)
while (prod(wss==sort(wss,decreasing=T))==0) {
  wss <- (nrow(kdata)-1)*sum(apply(kdata,2,var))
  for (i in 2:n.lev) wss[i] <- sum(kmeans(kdata, centers=i)$withinss)}

## Calculate the within groups SSE for 250 randomized data
## sets (based on the original input data)
k.rand <- function(x){
  km.rand <- matrix(sample(x),dim(x)[1],dim(x)[2])
  rand.wss <- as.matrix(dim(x)[1]-1)*sum(apply(km.rand,2,var))
  for (i in 2:n.lev) rand.wss[i] <- sum(kmeans(km.rand, centers=i)$withinss)
  rand.wss <- as.matrix(rand.wss)
  return(rand.wss)
}

rand.mat <- matrix(0,n.lev,250)

k.1 <- function(x) {
  for (i in 1:250) {
    r.mat <- as.matrix(suppressWarnings(k.rand(kdata)))
    rand.mat[,i] <- r.mat}
  return(rand.mat)
}

# Same function as above for data with < 3 column variables
k.2.rand <- function(x){
  rand.mat <- matrix(0,n.lev,250)
  km.rand <- matrix(sample(x),dim(x)[1],dim(x)[2])
  rand.wss <- as.matrix(dim(x)[1]-1)*sum(apply(km.rand,2,var))
  for (i in 2:n.lev) rand.wss[i] <- sum(kmeans(km.rand, centers=i)$withinss)
  rand.wss <- as.matrix(rand.wss)
```

```
  return(rand.wss)
}

k.2 <- function(x){
  for (i in 1:250) {
    r.1 <- k.2.rand(kdata)
    rand.mat[,i] <- r.1}
  return(rand.mat)
}

## Determine if the data data table has >
## or < 3 variables and call appropriate function above
if (dim(kdata)[2] == 2) { rand.mat <- k.2(kdata) } else { rand.mat <- k.1(kdata) }

## Plot within groups SSE against all tested
## cluster solutions for actual and randomized data -
## 1st: Log scale, 2nd: Normal scale

xrange <- range(1:n.lev)
yrange <- range(log(rand.mat),log(wss))
plot(xrange,yrange, type='n', xlab='Cluster Solution',
     ylab='Log of Within Group SSE',
     main='Cluster Solutions against Log of SSE')
for (i in 1:250) lines(log(rand.mat[,i]),type='l',col='red')
lines(log(wss), type="b", col='blue')
legend('topright',c('Actual Data', '250 Random Runs'), col=c('blue', 'red'), lty=1)
```



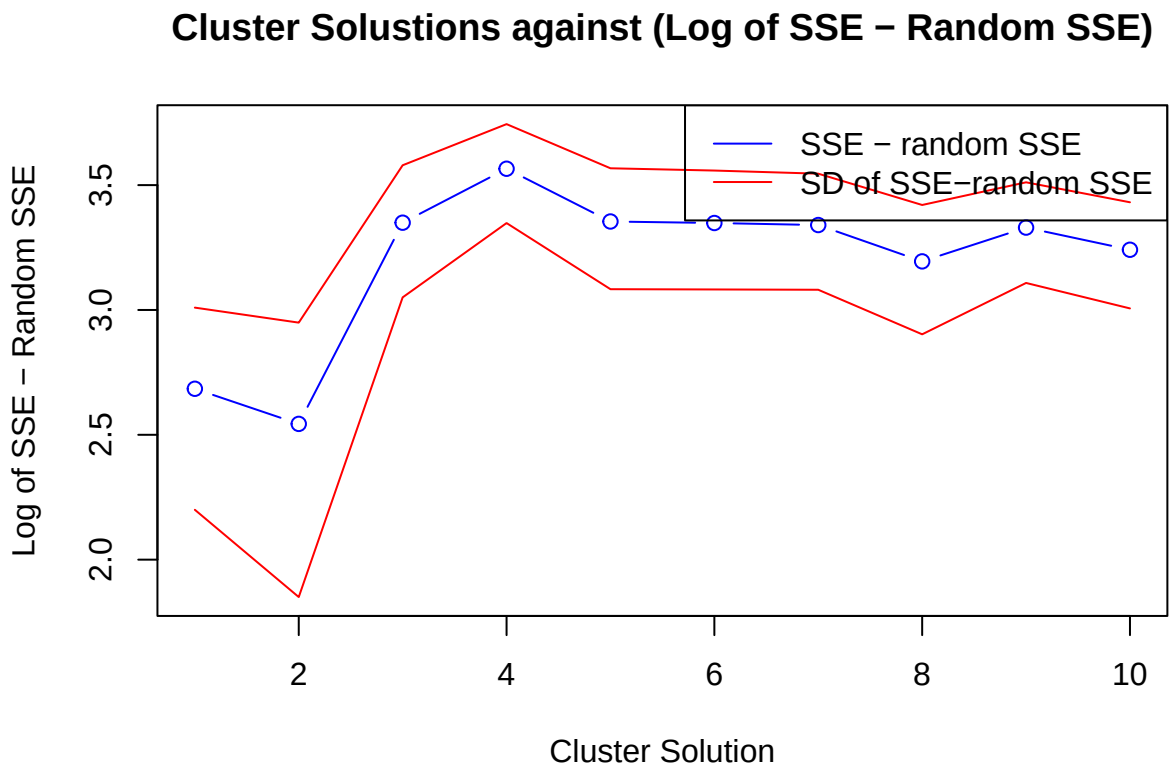

```
yrange <- range(rand.mat,wss)
plot(xrange,yrange, type='n', xlab="Cluster Solution",
```

```
     ylab="Within Groups SSE", main="Cluster Solutions against SSE")
for (i in 1:250) lines(rand.mat[,i],type='l',col='red')
lines(1:n.lev, wss, type="b", col='blue')
legend('topright',c('Actual Data', '250 Random Runs'), col=c('blue', 'red'), lty=1)
```

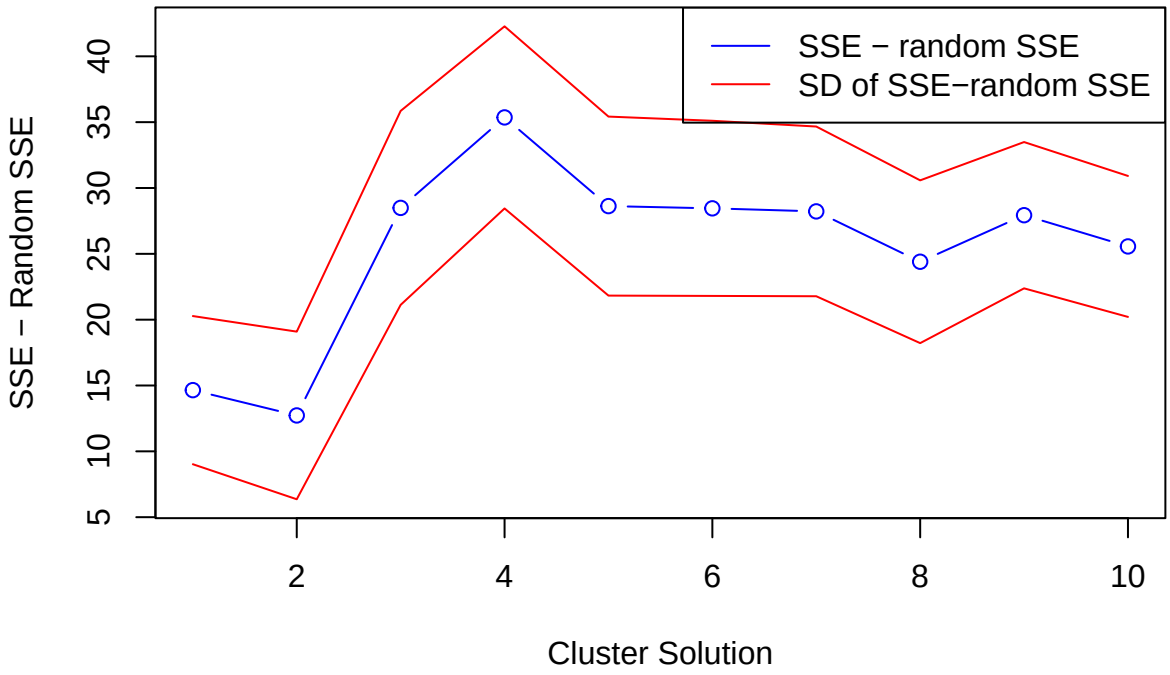



```
## Calculate the mean and standard deviation of difference
## between SSE of actual data and SSE of 250 randomized datasets
r.sse <- matrix(0,dim(rand.mat)[1],dim(rand.mat)[2])
wss.1 <- as.matrix(wss)
for (i in 1:dim(r.sse)[2]) {
  r.temp <- abs(rand.mat[,i]-wss.1[,1])
  r.sse[,i] <- r.temp}
r.sse.m <- apply(r.sse,1,mean)
r.sse.sd <- apply(r.sse,1,sd)
r.sse.plus <- r.sse.m + r.sse.sd
r.sse.min <- r.sse.m - r.sse.sd

## Plot difference between actual SSE mean SSE
## from 250 randomized datasets - 1st: Log scale, 2nd: Normal scale

xrange <- range(1:n.lev)
if (min(r.sse.min) < 0){
   yrange <- range(log(r.sse.plus - min(r.sse.min)*1.05),
                   log(r.sse.min - min(r.sse.min)*1.05))
} else {
   yrange <- range(log(r.sse.plus), log(r.sse.min))
}

plot(xrange,yrange, type='n',xlab='Cluster Solution',
```

```
     ylab='Log of SSE - Random SSE',
     main='Cluster Solustions against (Log of SSE - Random SSE)')
lines(log(r.sse.m), type="b", col='blue')
lines(log(r.sse.plus), type='l', col='red')
lines(log(r.sse.min), type='l', col='red')
legend('topright',c('SSE - random SSE', 'SD of SSE-random SSE'),
       col=c('blue', 'red'), lty=1)
```




```
xrange <- range(1:n.lev)
yrange <- range(r.sse.plus,r.sse.min)
plot(xrange,yrange, type='n',xlab='Cluster Solution',
     ylab='SSE - Random SSE',
     main='Cluster Solutions against (SSE - Random SSE)')
lines(r.sse.m, type="b", col='blue')
lines(r.sse.plus, type='l', col='red')
lines(r.sse.min, type='l', col='red')
legend('topright',c('SSE - random SSE', 'SD of SSE-random SSE'),
       col = c('blue', 'red'), lty = 1)
```




```
clust.level <- 4

# Apply K-means cluster solutions - append clusters to CSV file
fit <- kmeans(kdata, 4)
aggregate(kdata, by=list(fit$cluster), FUN=mean)
```

```
##   Group.1   k_percent  bb_percent      babip b_intent_walk         xba
## 1       1 -0.47548958 -0.01814642  0.8209918     0.9375011  1.30520471
## 2       2  0.85704630 -0.57424635 -0.8288481    -0.7730272 -0.86383113
## 3       3 -0.94697427  0.17209829  0.2841391     0.2467108  0.48235826
## 4       4  0.01029829  0.40164631  0.1962225     0.1198310 -0.09458005
##         xslg       xobp       xiso      xbacon sweet_spot_percent
## 1  1.8830411  1.0450761  1.3947275  1.55715259         0.13125737
## 2 -0.2224118 -1.1349490  0.2818278 -0.52230001        -0.03630523
## 3 -0.2512304  0.5137329 -0.5968825 -0.33141276         1.24235695
## 4 -0.4728188  0.2313620 -0.4982313 -0.03028667        -0.73505264
##   barrel_batted_rate hard_hit_percent avg_best_speed avg_hyper_speed
## 1          1.1447216        0.9824480      1.2006531       1.2703470
## 2          0.3745725       -0.5229947     -0.2240521      -0.3507202
## 3         -1.0607053       -1.0508772     -1.5042101      -1.3628716
## 4         -0.2055398        0.6277333      0.5370277       0.5349667
##   whiff_percent swing_percent
## 1     0.1851088    0.68581164
## 2     0.5319907    0.00373277
## 3    -1.2402604   -0.44589634
## 4     0.1733959   -0.04232089
```

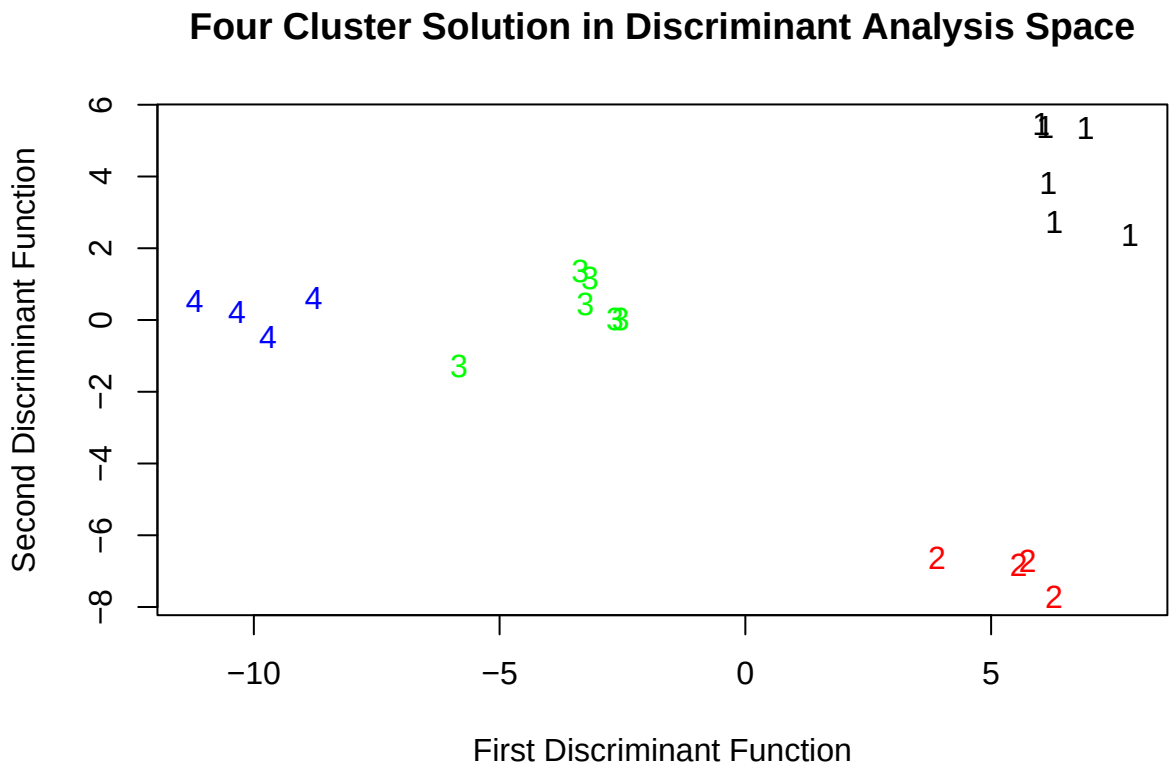
```
clust.out <- fit$cluster
kclust <- as.matrix(clust.out)
kclust.out <- cbind(kclust, statcast2023Cluster)
```

```
write.table(kclust.out, file="kmeans_out.csv", sep=",")
# end of script
```

Based on the Cluster Number against Log of Within-Group SSE graph, we can see that an optimal number of clusters is 4, as that is where our data begins to separate and level off from the 250 random runs. The same goes for the Cluster Number against Within-Group SSE graph. Also, in the Cluster Solutions against (Log of SSE - Random SSE) and the Cluster Solutions against (SSE - Random SSE), we see a spike at 4 clusters, although the SD's are large throughout. Even with somewhat unclear graphs, we would suggest four clusters.

Plotting Results in Discriminant Analysis Space

```
## Make plot of four cluster solution in space designated by
## first two discriminant functions
cuts <- cutree(statcast2023.hclust, k = 4)
plotcluster(kdata, cuts, main="Four Cluster Solution in Discriminant Analysis Space",
            xlab="First Discriminant Function",
            ylab="Second Discriminant Function")
```



In this plot of the four cluster solution in Discriminant Analysis space, we can see that the clusters are fairly well separated. This is a good sign that our clustering is working well.

Conclusion

In conclusion, we found 4 main clusters of hitters in the top 20 xwOBA hitters of 2023. These clusters are:

1. The All-Around Hitter (Get on Base, Hit for Power)
2. Solid Contact Hitters
3. Free Swingers (who often make contact)

4. Power Hitters

This is a bit different than my expected results, as I expected three main groups: power hitters, contact hitters, and hitters who focus on getting on base.

Based on distance measures like Root-Mean-square Standard Deviation and Cluster Distance, we confirmed that 4 clusters would be optimal in terms of maximizing the distance between clusters. Using k-means clustering, we confirmed that four clusters is the optimal number of clusters, as the difference between within-group SSE/log-SEE and random runs were maximized there. We then plotted the results in Discriminant Analysis space, and found that the clusters are fairly well separated. In conclusion, based on the 2023 Statcast data, we can see that there are four main types of hitters in the top 20 xwoba hitters of 2023. This is a good start to understanding the types of hitters in the MLB, and can be used to help teams understand what type of hitter they may need to add to their lineup.