Analysis Exercise

Hailey Yabroudy

Task: Which shortstop converted the most outs above average?

Note: I interpreted "converted" as any play that resulted in an out that the shortstop was involved in.

Summary

For this exercise, I started by taking a look at the data and seeing which of the data provided will be able to help me answer the question. A lot of what I did to get the information that I was looking for involved filtering the data. After making a subset of data with the columns that I would be working with, the first thing I did was filter out any event that wouldn't have resulted in an out, so nothing would be in there like a single or a double. I then looked at how the play was tagged and made sure that these would also only reflect outs, so no errors or a fielder's choice that didn't result in an out. The third filter I made, made sure that the fielded play involved a 6 somewhere in the play, whether their fielding the ball or making the out at a base. I then tallied up the outs that each playerid had and found the average number for all the shortstops. The last filter I did removed any of the players who had a number of outs that was below the average.

I then found the number of opportunities that the player had to make a play on the ball (this now included errors, fielder's choice, etc.) by changing one of the filters that I had in place and tallied that up. To find the outs above average, I just subtracted the number of outs that the player converted by the average and put that into it's own column. I added these two new values to the leaderboard and was done!

The Analysis

```
library(readxl)
shortstopdefense <- read_excel("/Users/haileyemma/Documents/JObs/[redacted]/shortstopdefense.xlsx")
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in K12049 / R12049C11: got 'NA'
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in L12049 / R12049C12: got 'NA'
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in F13123 / R13123C6: got 'NA'
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in G13123 / R13123C7: got 'NA'
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in K19962 / R19962C11: got 'NA'
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in L19962 / R19962C12: got 'NA'
```

```
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
```

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

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

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

I started by taking out all of the events that didn't have the outcome of an out. Since we're only focusing on the outs the shortstops were involved in, I don't need to worry about the rest. I started doing this by reviewing what all of the unique outcomes were for the event type field. Using this, I knew what I needed to filter out.

```
unique(shortstopdefense$eventtype)
```

```
##  [1] "field_out"                "single"
##  [3] "force_out"                "grounded_into_double_play"
##  [5] "field_error"              "double"
##  [7] "sac_bunt"                 "fielders_choice"
##  [9] "fielders_choice_out"      "triple"
## [11] "double_play"              "batter_interference"
## [13] "triple_play"
```

I made a subset of data of all the columns I would be working with for this exercise so that I wouldn't be messing with the original data. The columns I chose to work with were id, playerid, eventtype, fieled_pos, fieldingplay, and fielded_scoring. With this subset, I filtered out all of the events that didn't result in an out. Since you can't see the full line of code, I included field out, force out, sac bunt, fielder's choice out, double play, triple play, grounded into double play, and fielder's choice.

```
shortdef <- data.frame(shortstopdefense$id, shortstopdefense$playerid, shortstopdefense$eventtype, shor
```

```
shortdef <- filter(shortdef, shortstopdefense.eventtype == "field_out" | shortstopdefense.eventtype ==
```

Next, I filtered the data based on how the play was tagged in the fielded_scoring column. I wanted to only focus on plays tagged as an assist or putout because those were the only tags that result in an out.

I decided to not use f_fielded_ball because a lot of them were fielder's choices where the player didn't throw the ball anywhere. I also got rid of any errors because they didn't result in outs.

```
unique(shortdef$shortstopdefense.fielded_scoring)
```

```
## [1] "f_assist"         "f_putout"         "f_throwing_error" "f_fielded_ball"
## [5] "f_fielding_error"
```

```
shortdef <- filter(shortdef, shortstopdefense.fielded_scoring == "f_assist" | shortstopdefense.fielded_s
```

I then used the fieldingplay column to filter out any play that didn't involve the shortstop (ie. 53, 143, etc.). If the shortstop is involved in the play, then they will be credited for the out in some way, as either an assist or a putout.

```
shortdef <- shortdef[shortdef$shortstopdefense.fieldingplay %like% "6", ]
```

I tallied up the number of outs each player made based on their ID and found the mean number of outs that the players made. Using the mean, I was able to apply a filter to the new dataset and only show the players who have a made a number of outs higher than the average.

```
playercount <- shortdef %>% group_by(shortstopdefense.playerid) %>% tally()
mean(playercount[["n"]])
```

```
## [1] 48.26214
```

```
leaders <- playercount %>% filter(n > mean(n))
```

The mean = 48.26214. For simplicity sake moving forward, I rounded the mean to 48 in order to just work with whole numbers.

I organized the "leader" dataset I made into decreasing order and renamed the column names so it would be easier to work with. When I apply columns for the number of opportunities they had to make an out and the number of outs they had above the average, having the column names be the same makes it easier to join them together.

```
leaders <- leaders[order(-leaders$n),]
names(leaders)[names(leaders) == "shortstopdefense.playerid"] <- "playerid"
names(leaders)[names(leaders) == "n"] <- "Number_Outs"
```

To find the number of opportunities the player had, I went back to the original data and made a new dataset that included all eventtypes and filtered out any where the shortstop was not involved. I then tallied up how

many opportunites the player had based on their ID.

```
opps <- shortstopdefense[shortstopdefense$fieldingplay %like% "6", ]
opscount <- opps %>% group_by(playerid) %>% tally()
names(opscount)[names(opscount) == "n"] <- "Opportunities"
```

To find the outs above average, I subtracted 48 (the mean) from the number of out that the player made and added those numbers as a column to the leaders dataset.

```
OutsAboveAverage <- (leaders$Number_Outs - 48)
leaders <- cbind(leaders, OutsAboveAverage)
```

I made the final leaderboard by joining the leaders and the opscount dataset based on the playerid. I then needed to get rid of all of the NAs that were put into the dataset (because the two datasets had different number of rows) in order to get only the players who have made outs above the average again.

```
leaderboard <- leaders %>% full_join(opscount, by ="playerid")
leaderboard <- na.omit(leaderboard)
leaderboard
```

```
##    playerid Number_Outs OutsAboveAverage Opportunities
## 1    167960         195              147           224
## 2    162066         193              145           215
## 3    154448         186              138           209
## 4    161551         186              138           206
## 5    147431         171              123           192
## 6      5495         166              118           195
## 7    162393         161              113           180
## 8     11742         158              110           185
## 9    132551         158              110           175
## 10     5393         151              103           170
## 11   162648         149              101           163
## 12   206311         144               96           167
## 13    11869         138               90           162
## 14     5419         136               88           153
## 15     9425         134               86           153
## 16   160570         133               85           164
## 17     9074         129               81           144
## 18     2950         127               79           137
## 19   168314         127               79           154
## 20     9148         123               75           138
## 21   199814         117               69           135
## 22   171806         113               65           133
## 23     7580          93               45            98
## 24   113301          82               34            89
## 25   164881          76               28            87
## 26     9742          74               26            85
## 27   189664          70               22            85
## 28   197513          69               21            81
```

```
## 29   171885          64              16            80
## 30     3911          62              14            74
## 31     2087          61              13            71
## 32   208314          61              13            70
## 33   163714          59              11            66
## 34   159919          55               7            64
## 35     9424          52               4            56
```

Final Results

Based on my findings I found that the shortstop with the most outs above average is playerid "167960" with 195 converted outs (147 above the average) out of 224 opportunities.

To answer the other questions, I think that the original "shortstopdefense" dataset that was provided encompasses a lot in terms of data provided for a given play in a game. To answer the primary question of what variable or types of information do I think would be helpful in answering this question more effectivly, the data provided was plenty sufficient to answer the question as asked. If anything, there was a lot of extra data. The question of which shortstop converted the most outs is fairly simple to answer, but if you wanted to know more of the finer details like how they handle the balls in play, then the rest of the data begins to take purpose. For example, how good are shortstops at handling balls that are hit at higher velocities than ones that are hit at slower ones? What launch speed results in the most amount of outs made by the shortstops? Questions like these would help to utilize a lot of the data that is being collected. Another example is, where should the fielder be standing in order to maximize their out conversions? This would be able to use the location data that is provided. If there was data provided about the batters that were up to bat during those events, you could see patterns on where they tend to hit and how the shortstops perform against them based on the location of where they are during their at bats.

In terms of things that I found interesting, when I was working with the data, I started thinking about how errors can play into the short stops performance. I looked into how many errors each player made (I seperated fielding and throwing errors) and if they hadn't made those errors, how the leaderboard would change and who made the most errors. By looking at this data, we can start to think about what to work on with these players. Are they lacking more when it comes to throwing or fielding the ball? Out of how many opportunities they had, how many times did they make an error? I included a column to see a percentage of out of however many opportunities they had to make a play, what percent of them were errors.

The errors did not have any affect on the leaderboard. But, while I was investigating that I found that on average there are more fielding than throwing errors. In my experience, the shortstops that I have played with tend to have a lot more throwing errors than fielding ones, so I was always kept on my toes when I was playing first. To my surprise, the players in the dataset made more fielding errors. On average, the players made 2.13 fielding errors and 1.79 throwing errrors. I found these averages by using the code below.

mean(throwcount[["n"]])

mean(fieldcount[["n"]])

Throwing Error Code

```
throwing <- data.frame(shortstopdefense$id, shortstopdefense$playerid, shortstopdefense$eventtype, shor
throwing <- filter(throwing, shortstopdefense$fielded_scoring == "f_throwing_error")
```

```
throwing <- throwing[throwing$shortstopdefense.fieldingplay %like% "6", ]
throwcount <- throwing %>% group_by(shortstopdefense.playerid) %>% tally()
throwcount <- throwcount[order(-throwcount$n),]
names(throwcount)[names(throwcount) == "shortstopdefense.playerid"] <- "playerid"
names(throwcount)[names(throwcount) == "n"] <- "throwing_errors"
throwcount <- throwcount %>% full_join(opscount, by ="playerid")
throwcount <- na.omit(throwcount)
error_percentage <- (throwcount$throwing_errors / throwcount$Opportunities)
error_percentage <- percent(error_percentage)
throwcount <- cbind(throwcount, error_percentage)
throwcount
```

```
##    playerid throwing_errors Opportunities error_percentage
## 1    171806               5           133          3.7594%
## 2    199814               4           135          2.9630%
## 3      9210               3            50          6.0000%
## 4     11742               3           185          1.6216%
## 5    147431               3           192          1.5625%
## 6    167960               3           224          1.3393%
## 7    171885               3            80          3.7500%
## 8    206311               3           167          1.7964%
## 9      5419               2           153          1.3072%
## 10     9148               2           138          1.4493%
## 11    11500               2            21          9.5238%
## 12    11869               2           162          1.2346%
## 13   154448               2           209          0.9569%
## 14   162393               2           180          1.1111%
## 15   162648               2           163          1.2270%
## 16   189664               2            85          2.3529%
## 17     2087               1            71          1.4085%
## 18     3911               1            74          1.3514%
## 19     4379               1            14          7.1429%
## 20     5495               1           195          0.5128%
## 21     6619               1            48          2.0833%
## 22     7580               1            98          1.0204%
## 23     9074               1           144          0.6944%
## 24   158358               1            29          3.4483%
## 25   159919               1            64          1.5625%
## 26   160570               1           164          0.6098%
## 27   161551               1           206          0.4854%
## 28   162066               1           215          0.4651%
## 29   164908               1             4         25.0000%
## 30   169390               1            13          7.6923%
## 31   200265               1            45          2.2222%
## 32   206262               1            22          4.5455%
## 33   207428               1            54          1.8519%
## 34   208314               1            70          1.4286%
```

Fielding Error Code

```
fielding <- data.frame(shortstopdefense$id, shortstopdefense$playerid, shortstopdefense$eventtype, shor
fielding <- filter(fielding, shortstopdefense$fielded_scoring == "f_fielding_error")
fielding <- fielding[fielding$shortstopdefense.fieldingplay %like% "6", ]
```

```
fieldcount <- fielding %>% group_by(shortstopdefense.playerid) %>% tally()
fieldcount <- fieldcount[order(-fieldcount$n),]
names(fieldcount)[names(fieldcount) == "shortstopdefense.playerid"] <- "playerid"
names(fieldcount)[names(fieldcount) == "n"] <- "fielding_errors"
fieldcount <- fieldcount %>% full_join(opscount, by ="playerid")
fieldcount <- na.omit(fieldcount)
error_percentage <- (fieldcount$fielding_errors / fieldcount$Opportunities)
error_percentage <- percent(error_percentage)
fieldcount <- cbind(fieldcount, error_percentage)
fieldcount
```

```
##    playerid fielding_errors Opportunities error_percentage
## 1    160570               6           164           3.659%
## 2    167960               6           224           2.679%
## 3      5419               5           153           3.268%
## 4    168314               5           154           3.247%
## 5      5495               4           195           2.051%
## 6      9425               4           153           2.614%
## 7    147431               4           192           2.083%
## 8      5393               3           170           1.765%
## 9     11742               3           185           1.622%
## 10   113301               3            89           3.371%
## 11   159919               3            64           4.688%
## 12   171806               3           133           2.256%
## 13   189664               3            85           3.529%
## 14   197513               3            81           3.704%
## 15     2087               2            71           2.817%
## 16     2950               2           137           1.460%
## 17     6619               2            48           4.167%
## 18     9148               2           138           1.449%
## 19     9210               2            50           4.000%
## 20   154448               2           209           0.957%
## 21   158255               2            16          12.500%
## 22   162393               2           180           1.111%
## 23   200265               2            45           4.444%
## 24   205278               2            13          15.385%
## 25     3911               1            74           1.351%
## 26     4331               1             9          11.111%
## 27     9074               1           144           0.694%
## 28     9424               1            56           1.786%
## 29     9742               1            85           1.176%
## 30    10596               1             7          14.286%
## 31   132551               1           175           0.571%
## 32   158358               1            29           3.448%
## 33   159128               1            37           2.703%
## 34   161551               1           206           0.485%
## 35   162066               1           215           0.465%
## 36   162648               1           163           0.613%
## 37   164881               1            87           1.149%
## 38   169374               1            10          10.000%
## 39   171819               1            13           7.692%
## 40   171885               1            80           1.250%
## 41   199814               1           135           0.741%
## 42   201896               1             5          20.000%
```

```
## 43   203336              1              7          14.286%
## 44   206311              1            167           0.599%
## 45   207428              1             54           1.852%
```