Final Relection

Lucas Smielewski

4/15/2022

Link to Final Project - <https://github.com/Lucassmielewski/STA518_Project>

Objectives: Import, manage, and clean data. Create graphical displays and numerical summaries of data for exploratory analysis and presentations. Write R programs for simulations from probability models and randomization-based experiments. Use source documentation and other resources to troubleshoot and extend R programs. Write clear, efficient, and well-documented R programs.

For my final project I decided analyse run expectancy in the MLB. The goal of the project was to build an interactive Shiny App where the user would input a combination of game state variables including, inning number, home or away batting, number of outs and the location of runs on base. After gather user inputs the application would show the distribution of runs that a team is expected to score in the inning based on these selections. To accomplish this goal I downloaded play-by-play data from MLB Retrosheets - <https://www.retrosheet.org/>. The embedded code snippet below shows a small example of the raw text data that I first loaded into R.

ANA = read\_tsv(here::here("data", "2021ANA.EVA"), col\_names=FALSE)

## Rows: 13156 Columns: 1  
## -- Column specification --------------------------------------------------------  
## Delimiter: "\t"  
## chr (1): X1  
##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

head(ANA)

## # A tibble: 6 x 1  
## X1   
## <chr>   
## 1 id,ANA202104010   
## 2 version,2   
## 3 info,visteam,CHA   
## 4 info,hometeam,ANA   
## 5 info,site,ANA01   
## 6 info,date,2021/04/01

After reviewing the coded data, I knew this project was going to be a long process and I was right, it was. But while working on this project in the past few months as well as completing the in-class activities earlier in the semester, I have gained a lot of experience using R, a software I plan to use extensively in my job following graduation. In this document, I will go through a few of the steps I took to complete my project and use these to exhibit my compression of the following course objectives: Import, manage, and clean data. Create graphical displays and numerical summaries of data for exploratory analysis and presentations. Write R programs for simulations from probability models and randomization-based experiments. Use source documentation and other resources to troubleshoot and extend R programs. Write clear, efficient, and well-documented R programs.

This project included a large amount of data managing and cleaning, seeing that the cleaning process entailed converting encoded text files into useable tabular data sets. The embedded code below was my finalized function that took the list text data files and formatting them into a list of tabular data sets, one for each game a team played. The end result was multiple lists holding each game within a list storing each team.

clean\_teams <- function(team){  
   
 tsv = read\_tsv(here::here("data", team), col\_names=FALSE)  
 #Creating tibble of length 81 that store row number of each game ID  
 row\_index <- tsv %>%  
 mutate(rownumber = row\_number()) %>%  
 filter(str\_detect(tsv$X1, 'id,')==TRUE)  
   
 #Create Vector of Original Data  
 team\_vec = as.vector(tsv$X1)  
   
 #Create Vector of row numbers of ID's (position in team\_vec)  
 row\_vec=as.vector(row\_index$rownumber)  
   
 #Creating List to each Game's data  
 team\_list=list()  
   
 #Loop through row\_vec (length 81)  
 #Split team\_vec into each game, decided by ID position held in row\_vec  
 #Add each game vector into team\_list  
 for(row in seq(row\_vec+1)){  
 if(row!=length(row\_vec)){  
 new <- team\_vec[c(row\_vec[row]:(row\_vec[(row+1)]-1))]  
 team\_list[[row]]<- new  
 }  
 }  
   
 return(team\_list)  
}  
  
get\_results <- function(game\_codes){  
   
 plays = str\_subset(game\_codes, 'play')  
  
 #Works on GVSU R Server not on Laptop?  
 # game\_df = read\_csv(plays, col\_names = c('Play',   
 # 'Inning',   
 # 'Home Team',   
 # 'Retro ID',   
 # 'Count on Play',   
 # 'Pitch Sequence',  
 # 'Outcome'))  
   
 #Works on Laptop  
 game\_df <- as.tibble(plays) %>%   
 separate(value, into=c('Play',   
 'Inning',   
 'Home Team',   
 'Retro ID',   
 'Count on Play',   
 'Pitch Sequence',  
 'Outcome'),   
 sep=',')  
   
 game\_df <- game\_df %>%   
 #Creating Column to hold runner advancement codes  
 separate(Outcome, into = c('Outcome', 'Runner Advancements'),  
 "[.]", extra = "merge") %>%   
 filter(Outcome != 'NP') #Remove plays where nothing occurred  
   
   
 batter\_codes = c('S' = 1, 'D' = 2, 'T' = 3,   
 'H' = 4, 'HR' = 4, 'DGR' = 2,  
 'E' = 1, 'HP' = 1, 'I' = 1,   
 'IW' = 1, 'W' = 1, 'C' = 1,  
 'BK' = 0 , 'DI' = 0, 'SB' = 0,  
 'PB' = 0, 'WP' = 0, 'OA'=0)  
#These outcomes should still result in an out so they will not fill batter bases  
 #'CS' = 0 , 'PO' = 0, 'POCS' = 0, 'OA' = 0)   
  
 #Creating a mapping function to map batter\_codes (basically replaces loop)  
 #quo saves this condition rather than evaluating it.   
 #The !! and !!! are how you actually evaluate this code  
 batter\_conditions <- map(names(batter\_codes),   
 ~quo(str\_starts(  
 Outcome,   
 regex(  
 paste(!!.x,"+([0-9]|/|$)|E|(W+)|(SBH;)",   
 sep="")))~batter\_codes[!!.x]))  
   
 #Applying the batter code condition to the game dataframe  
 game\_df <- game\_df %>%   
 mutate(`Batter Bases` = case\_when(!!!batter\_conditions))  
   
   
   
 #Creating vector to hold indices of inning (lag from Dplyr is really helpful)  
 new\_inning = c(1, which(game\_df$`Home Team` != lag(game\_df$`Home Team`)))   
   
 #Creating three binary variables, First Second and Third to hold runs  
   
 game\_df <- game\_df %>%   
 mutate(`First` = case\_when(row\_number() %in% new\_inning ~ 0,  
 lag(`Batter Bases`==1) ~ 1,  
 lag(`Batter Bases`==4) ~ 0),  
 `Second`= case\_when(row\_number() %in% new\_inning ~ 0,  
 lag(`Batter Bases`==2) &   
 (is.na(`Runner Advancements`)) ~ 1,  
 lag(`Batter Bases`==4) ~ 0),  
 `Third`= case\_when(row\_number() %in% new\_inning ~ 0,  
 lag(`Batter Bases`==3) &   
 (is.na(`Runner Advancements`)) ~ 1,  
 lag(`Batter Bases`==4) ~ 0))  
   
 game\_df <- game\_df %>%   
 separate(`Runner Advancements`, into = c('Runner Advancements',   
 'Runner Advancements1',   
 'Runner Advancements2',   
 'Runner Advancements3'),  
 "[;]", extra = "merge")  
   
 game\_df <- game\_df %>%   
 mutate(`First` = case\_when(   
 lag(str\_detect(`Runner Advancements3`, "-1")) ~ 1,  
 lag(str\_detect(`Runner Advancements3`, "1-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements2`, "-1")) ~ 1,  
 lag(str\_detect(`Runner Advancements2`, "1-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements1`, "-1")) ~ 1,  
 lag(str\_detect(`Runner Advancements1`, "1-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements`, "-1")) ~ 1,  
 lag(str\_detect(`Runner Advancements`, "1-")) ~ 0,  
  
 #Stolen bases & Pickoffs  
 lag(str\_detect(Outcome, "SB2")) ~ 0, #Leaves First to Steal Second  
 lag(str\_detect(Outcome, "CS2")) ~ 0, #Caught Stealing at 2nd   
 lag(str\_detect(Outcome, "PO1")) ~ 0, #Picked Off of First  
 TRUE ~ as.numeric(First)),  
   
 `Second` = case\_when(  
 lag(str\_detect(`Runner Advancements3`, "-2")) ~ 1,  
 lag(str\_detect(`Runner Advancements3`, "2-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements2`, "-2")) ~ 1,  
 lag(str\_detect(`Runner Advancements2`, "2-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements1`, "-2")) ~ 1,  
 lag(str\_detect(`Runner Advancements1`, "2-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements`, "-2")) ~ 1,  
 lag(str\_detect(`Runner Advancements`, "2-")) ~ 0,  
   
 #Stolen Bases & Pickoffs  
 lag(str\_detect(Outcome, "SB2")) ~ 1, #Steals Second  
 lag(str\_detect(Outcome, "SB3")) ~ 0, #Leaves Second to Steal Third  
 lag(str\_detect(Outcome, "CS3")) ~ 0, #Caught Stealing at Third  
 lag(str\_detect(Outcome, "PO2")) ~ 0, #Picked Off of Second  
 TRUE ~ as.numeric(Second)),  
   
 `Third` = case\_when(  
 lag(str\_detect(`Runner Advancements3`, "-3")) ~ 1,  
 lag(str\_detect(`Runner Advancements3`, "3-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements2`, "-3")) ~ 1,  
 lag(str\_detect(`Runner Advancements2`, "3-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements1`, "-3")) ~ 1,  
 lag(str\_detect(`Runner Advancements1`, "3-")) ~ 0,  
   
 lag(str\_detect(`Runner Advancements`, "-3")) ~ 1,  
 lag(str\_detect(`Runner Advancements`, "3-")) ~ 0,  
   
 #Stolen Bases & Pickoffs  
 lag(str\_detect(Outcome, "SB3")) ~ 1, #Steals Third  
 lag(str\_detect(Outcome, "SBH")) ~ 0, #Leaves Third to Steal H  
 lag(str\_detect(Outcome, "CSH")) ~ 0, #Caught Stealing at Third  
 lag(str\_detect(Outcome, "PO3")) ~ 0, #Picked Off of Third  
 TRUE ~ as.numeric(Third))) %>%   
   
 fill(First, Second, Third, .direction="down")  
   
 #OUTS ERROR TESTING  
 game\_df <- game\_df %>%   
 mutate(`Batter Bases` = case\_when(  
 (str\_detect(`Runner Advancements`, "B")) |   
 (str\_detect(`Runner Advancements1`, "B")) |  
 (str\_detect(`Runner Advancements2`, "B")) |  
 (str\_detect(`Runner Advancements3`, "B")) ~ 1,   
 TRUE ~ as.numeric(`Batter Bases`)))  
   
 game\_df <- game\_df %>%   
 group\_by(Inning, `Home Team`) %>%   
 mutate(Outs = cumsum(is.na(`Batter Bases`))-1,  
 `Inning Runs` = sum(as.numeric(  
 str\_detect(`Runner Advancements`,"(-H)")),  
 as.numeric(  
 str\_detect(`Runner Advancements1`,"(-H)")),  
 as.numeric(  
 str\_detect(`Runner Advancements2`,"(-H)")),  
 as.numeric(  
 str\_detect(`Runner Advancements3`,"(-H)")),  
 #Steals of Home and Home Runs  
 str\_detect(Outcome, "SBH"), (`Batter Bases`==4),   
 na.rm=TRUE))  
   
 game\_df$Outs = case\_when(game\_df$Outs<0 ~ 0, TRUE ~ as.numeric(game\_df$Outs))  
  
 #OUTS ERROR TESTING - NOTICED ERRORS WERE NOT WORKING  
 ind = ((game\_df$Outs >= 3) | (game\_df$Outs<0))  
 if(nrow(game\_df[ind,])>0){  
 print(game\_df %>%   
 select(c(Outcome,   
 `Runner Advancements`,   
 `Runner Advancements1`,   
 `Batter Bases`,  
 First, Second, Third, `Inning Runs`, Outs)))  
 }  
   
 agg\_df <- expand(game\_df, nesting(First, Second, Third, Outs, `Inning Runs`))  
   
   
 return(agg\_df)  
}

Finally after attaining the long list of play-by-play game data sets, they were condensed into one final data set holding every combination of innings, outs, and base runners. An sample of this final data set can be seen below.

## Rows: 150292 Columns: 7  
## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## dbl (7): Inning, Home Team, First, Second, Third, Outs, Inning Runs  
##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

head(agg\_total)

## # A tibble: 6 x 7  
## Inning `Home Team` First Second Third Outs `Inning Runs`  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 0 0 0 0 0 0  
## 2 1 0 0 0 0 1 0  
## 3 1 0 1 0 0 2 0  
## 4 1 1 0 0 0 0 0  
## 5 1 1 0 0 0 1 0  
## 6 1 1 0 0 0 2 0

These two code chunks not only show the strides I have taken over the course of the semester in my ability to import, manage, and clean data in R, but also my ability to write user-create functions that can be understood and reproduced by the reader. I felt that I was behind in various points throughout the semester, but after working through a larger-scale comprehensive project I feel much more comfortable with the program. Iterating through the large amount of text file and being able to condense them together also show the ability to run simulations and functions within R. After attaining this final data set. I first wrote to into my files as a csv in order to not have to run the entire program again and then started my analysis. I started my analysis by creating exploratory visualizations to decide the route I wanted to take. The code chunks below show the steps I took to create a few of these graphics.

df <- read\_csv(here::here('Results','Seaon Total Innings.csv'))

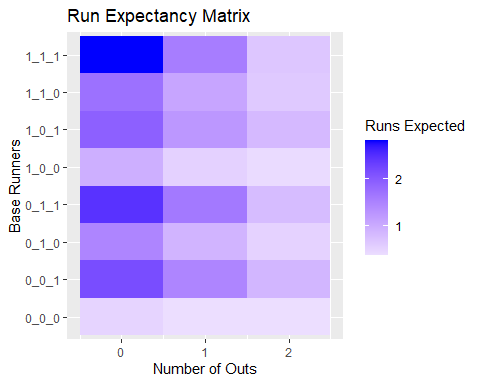
## Rows: 150292 Columns: 7  
## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## dbl (7): Inning, Home Team, First, Second, Third, Outs, Inning Runs  
##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

df <- df %>%   
 mutate(`Scored` = if\_else(`Inning Runs` >= 1,TRUE,FALSE))  
  
  
df\_group <- df %>%   
 group\_by(First, Second, Third, Outs) %>%   
 summarise(Runs = sum(`Inning Runs`),  
 Occurences = n(),  
 `Probabilty of Scoring` = mean(Scored))

## `summarise()` has grouped output by 'First', 'Second', 'Third'. You can  
## override using the `.groups` argument.

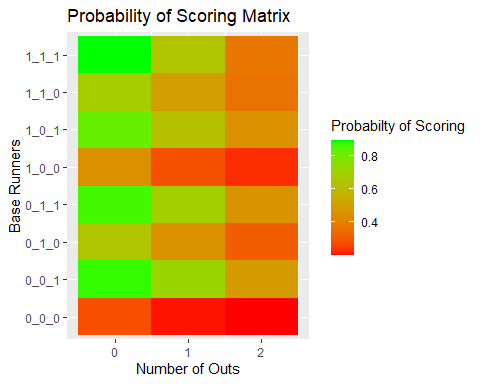
df\_group <- df\_group %>%   
 mutate(`Runs Expected` = Runs/Occurences) %>%   
 unite(Onbase, First:Third)   
  
runs\_expected\_matrix <- df\_group %>%   
 pivot\_wider(id\_cols=Onbase, names\_from=Outs,   
 values\_from=`Runs Expected`,   
 names\_prefix='Outs: ', names\_sep=' ')  
  
   
  
run\_prob\_matrix <- df\_group %>%   
 pivot\_wider(id\_cols=Onbase, names\_from=Outs,   
 values\_from=`Probabilty of Scoring`,   
 names\_prefix='Outs: ', names\_sep=' ')

# p <-   
df\_group %>%   
 ggplot(aes(y=Onbase, fill=`Runs Expected`, x=Outs))+  
 geom\_tile()+  
 scale\_fill\_gradient2(low="red", high="blue", mid="white") +  
 ggtitle("Run Expectancy Matrix") +  
 xlab("Number of Outs") + ylab("Base Runners")



# ggplotly(p)

# p <-   
df\_group %>%   
 ggplot(aes(y=Onbase, fill=`Probabilty of Scoring`, x=Outs))+  
 geom\_tile() +  
 scale\_fill\_gradient(low="red", high="green") +  
 ggtitle("Probability of Scoring Matrix") +  
 xlab("Number of Outs") + ylab("Base Runners")



# ggplotly(p)

These graphs were a part of my main goal of this project and I am very glad I was able to accomplish this. These show how the expected number of runs in an inning for a MLB game in the 2021 varying based on the combination between the locations of runners and the number of outs in said inning. I also found the probability of scoring at least one run in the inning for this same matrix. While working on these visualizations I found documentation for the package, Plotly, a software I had used previously in Python. I especially enjoyed incorporated this into my project because I find interactive plots much more useful than static plots when visualizing complex data situations. These graphs along with the work I did to build an interactive shiny app, I will get into this part next, are the reasons that I feel I have accomplished the objectives of creating graphical displays and numerical summaries of data for exploratory analysis and presentations as well as using source documentation and other resources to troubleshoot and extend R programs.

Finally I get to the most intriguing part of my project, building an interactive Shiny app. Even though I was able to produce a simple app producing all of the same concepts that I had set out to do at the start of my project, it gives me error when opening the application from the shinyapps.io server, so this is something I will have to investigate further. Nevertheless my Shiny app will run and be produced correctly when launched from my Rstudio session a screenshot of this can be seen below.

Overall, I believe the examples I have provided throughout this reflection exemplify the vast improvements I have made in R programming throughout this semester. Because of these improvements and my ability to show examples demonstrating each of Course Objectives, I feel that I have earned an ‘A’ for my work in this class. I have really enjoyed this class and the work I have put in has felt rewarding. Even though there were times of struggle, I feel that this project and the activities have helped me tremendously. The experience I’ve gained will come in very handy at my new job come graduation.