STAT 345 Midterm Project

Due April 3

“NOBODY KNOWS ANYTHING. Not one person in the entire motion picture field knows for a certainty what’s going to work. Every time out it’s a guess—and, if you’re lucky, an educated one.” William Goldman, *Adventures in the Screen Trade*

Your data for the midterm project consists of the 1000 highest rated movies on the Internet Movie Database (IMDB). You can find the first 50 movies [here](https://www.imdb.com/search/title/?groups=top_1000&start=1), with navigational links to the other 950 movies.

Each IMDB page records a large amount of information about each movie. We are interested in the following:

* The average rating of the movie by IMDB users.
* The number of ratings of the movie.
* The year the movie was released.
* The gross revenue of the movie (US).
* The budget for the movie.
* The movie’s title.
* The movie’s genre(s).
* The four top-billed actors.
* The text of the 25 “most helpful” reviews, as well as their helpfulness (ratio of helpful votes out of total votes.)

Note that the first five (and last) variables are numeric, and the genre, title, and reviews are strings. In some cases, some of these variables may be missing for a particular movie.

In some cases, the business information page lists multiple gross revenues, depending on the country, or gross receipts by different dates. In case of ambiguity, we are interested in gross receipts for the US, and want to use the figure for the latest available date. If no gross revenue figure is available for the US, treat the gross revenue as missing.

**General advice:** Get started on this one early. If you wait to the last minute, it will not go well.

1. (30 pts) Write code to extract the variables described above from all 1000 movies, and store it in a data frame. For full credit, you should write a function which can extract this information from an arbitrary movie code (or url), and then further code which uses that function and applies it to all 1000 movies. For full credit, your code should avoid loops in favor of vectorized operations and apply (and sapply, lapply, etc., as convenient). Your code should handle missing values appropriately, and should not convert categorical variables into numbers, or numbers into strings, etc.

#appropriate packages  
library(rvest)  
library(tidyverse)  
library(dplyr)  
library(ggplot2)  
library(stringr)  
library(ggrepel)  
library(tidytext)  
library(textdata)

## a function that takes in the "tt" part of the url to access a specific title, ouputs new link  
titles\_fun <- function(key\_title) {  
 first\_t <- key\_title  
 second\_t <- "/?ref\_=adv\_li\_tt"  
 url\_title <- paste(first\_t, second\_t, sep = "")  
 return(url\_title)  
}  
  
##in order to pull 25 most helpful reviews, we need to use the tt in the url to access them by the specific title, ouitputs new link  
  
review\_fun <- function(key\_review) {  
 first\_r <- key\_review  
 second\_r <- "/reviews?ref\_=tt\_urv"  
 url\_review <- paste(first\_r, second\_r, sep = "")  
 return(url\_review)  
}  
  
##Read in full links to use for functions above  
imdb\_url <- read\_csv("imdb\_urls.csv")  
  
#Make vector that has links for each title  
title\_urls <- titles\_fun(imdb\_url$imdb\_urls)  
  
#Make vector that has links for each titles reviews  
review\_urls <- review\_fun(imdb\_url$imdb\_urls)

#Average Rating  
avg\_rating\_fun <- function(url) {  
 info <- read\_html(url) %>%   
 html\_nodes("strong span") %>%   
 html\_text() %>%  
 as.numeric()  
 return(info)  
}  
avg\_rating1 <- sapply(title\_urls[1:10], avg\_rating\_fun) %>% as.vector()

#Number of Ratings  
num\_ratings11 <- function(url) {  
 info <- read\_html(url) %>%   
 html\_nodes(".imdbRating a") %>%   
 html\_text() %>%  
 str\_replace\_all(",", "") %>%  
 as.numeric()  
 return(info)  
}  
num\_rating1 <- sapply(title\_urls[1:10], num\_ratings11) %>% as.vector()

#Release Year  
release\_year <- function(url) {  
 info <- read\_html(url) %>%   
 html\_nodes("#titleYear a") %>%   
 html\_text() %>%  
 as.numeric()  
 return(info)  
}  
year\_released1 <- sapply(title\_urls[1:10], release\_year) %>% as.vector()

#Gross Revenue (USA)  
gross\_rev <- function(url){  
 info <- read\_html(url) %>%   
 html\_nodes(".txt-block:nth-child(14)") %>%  
 html\_text() %>%  
 str\_subset("Gross USA")   
 if (length(info) == 0) {  
 info <- NA  
 }  
 else {  
 info <- info %>%  
 str\_replace\_all("\n", "") %>%  
 str\_replace\_all("Gross USA:", "") %>%  
 str\_replace\_all(",", "") %>%  
 str\_trim() %>%  
 str\_replace\_all("[^[:alnum:]]", "") %>%  
 as.numeric()  
 }  
 return(info)  
}  
reve <- sapply(title\_urls[1:10], gross\_rev) %>% as.vector()

#Budget  
budget\_fun <- function(url){  
 info <- read\_html(url) %>%   
 html\_nodes(".txt-block:nth-child(12)") %>%  
 html\_text() %>%  
 str\_subset("Budget")   
 if (length(info) == 0) {  
 info <- NA  
 }  
 else {  
 info <- info %>%  
 str\_replace\_all("Budget", "") %>%  
 str\_replace\_all("estimated", "") %>%  
 str\_trim() %>%  
 str\_replace\_all("[^[:alnum:]]", "") %>%  
 as.numeric()  
 }  
 return(info)  
}  
budget1 <- sapply(title\_urls[1:10], budget\_fun) %>% as.vector()

#Titles  
titles <- function(url) {  
 info <- read\_html(url) %>%   
 html\_nodes(".parent a") %>%   
 html\_text()  
 return(info)  
}  
titless <- sapply(review\_urls[1:10], titles) %>% as.vector()

#Genre  
genre\_fun <- function(url) {  
 info <- read\_html(url) %>%  
 html\_nodes(".txt-block~ .canwrap a") %>%  
 html\_text() %>%  
 str\_c(collapse = ", ")  
 return(info)  
}  
genre1 <- sapply(title\_urls[1:10], genre\_fun) %>% as.vector()

#Top 4 Actors  
actors <- function(url) {  
 info <- read\_html(url) %>%  
 html\_nodes(".even:nth-child(3) .primary\_photo+ td a , .odd:nth-child(4) .primary\_photo+ td a , .even:nth-child(5) .primary\_photo+ td a , .odd:nth-child(2) .primary\_photo+ td a") %>%  
 html\_text() %>%  
 str\_c(collapse = ", ") %>%  
 str\_replace\_all(".\n", "") %>%  
 str\_replace(" ", "")  
   
 return(info)  
}  
top4actor <- sapply(title\_urls[1:10], actors) %>% as.vector()

#25 Reviews  
reviews <- function(url) {  
 info <- read\_html(url) %>%  
 html\_nodes(".show-more\_\_control") %>%  
 html\_text() %>%  
 str\_replace\_all("\n", "")   
   
 info[info == ""] <- NA  
 info[info == " "] <- NA  
   
 return(info)  
}  
review <- sapply(review\_urls[1:10], reviews) %>% as.vector()   
  
#vectors for final dataframe for reviews  
review1 <- c()  
review2 <- c()  
review3 <- c()  
review4 <- c()  
review5 <- c()  
review6 <- c()  
review7 <- c()  
review8 <- c()  
review9 <- c()  
review10 <- c()  
review11 <- c()  
review12 <- c()  
review13 <- c()  
review14 <- c()  
review15 <- c()  
review16 <- c()  
review17 <- c()  
review18 <- c()  
review19 <- c()  
review20 <- c()  
review21 <- c()  
review22 <- c()  
review23 <- c()  
review24 <- c()  
review25 <- c()  
  
#a loop that assigns the review vectors the respective element in the review list  
for (i in 1:10) {  
 review1[i] <- review %>% .[[i]] %>% .[1]  
 review2[i] <- review %>% .[[i]] %>% .[2]  
 review3[i] <- review %>% .[[i]] %>% .[3]  
 review4[i] <- review %>% .[[i]] %>% .[4]  
 review5[i] <- review %>% .[[i]] %>% .[5]  
 review6[i] <- review %>% .[[i]] %>% .[6]  
 review7[i] <- review %>% .[[i]] %>% .[7]  
 review8[i] <- review %>% .[[i]] %>% .[8]  
 review9[i] <- review %>% .[[i]] %>% .[9]  
 review10[i] <- review %>% .[[i]] %>% .[10]  
 review11[i] <- review %>% .[[i]] %>% .[11]  
 review12[i] <- review %>% .[[i]] %>% .[12]  
 review13[i] <- review %>% .[[i]] %>% .[13]  
 review14[i] <- review %>% .[[i]] %>% .[14]  
 review15[i] <- review %>% .[[i]] %>% .[15]  
 review16[i] <- review %>% .[[i]] %>% .[16]  
 review17[i] <- review %>% .[[i]] %>% .[17]  
 review18[i] <- review %>% .[[i]] %>% .[18]  
 review19[i] <- review %>% .[[i]] %>% .[19]  
 review20[i] <- review %>% .[[i]] %>% .[20]  
 review21[i] <- review %>% .[[i]] %>% .[21]  
 review22[i] <- review %>% .[[i]] %>% .[22]  
 review23[i] <- review %>% .[[i]] %>% .[23]  
 review24[i] <- review %>% .[[i]] %>% .[24]  
 review25[i] <- review %>% .[[i]] %>% .[25]  
}

#Helpfulness rating  
helpfulness <- function(url) {  
 info <- read\_html(url) %>%  
 html\_nodes(".text-muted") %>%  
 html\_text() %>%  
 str\_replace\_all("\n", "") %>%  
 str\_replace\_all(" out of ", "/") %>%  
 str\_trim() %>%  
 str\_replace\_all(" ", "") %>%  
 str\_replace\_all(". ", "") %>%  
 str\_replace\_all("found this helpful.Was this review helpful", "") %>%  
 str\_replace\_all("Sign in to votePermalink", "") %>%  
 str\_replace\_all(",", "") %>%  
 str\_trim() %>%  
 str\_replace\_all(" \\?", "") %>%  
 str\_split("/")   
 infonumerator <- info %>%  
 sapply("[[", 1) %>%  
 as.numeric()  
 infodenom <- info %>%  
 sapply("[[", 2) %>%  
 as.numeric()  
 info <- infonumerator / infodenom  
   
 return(info)  
}  
  
#Put first 10 links in df  
help <- sapply(review\_urls[1:10], helpfulness) %>% as.vector()  
  
help1 <- help[seq(1, length(help), 25)]  
help2 <- help[seq(2, length(help), 25)]  
help3 <- help[seq(3, length(help), 25)]  
help4 <- help[seq(4, length(help), 25)]  
help5 <- help[seq(5, length(help), 25)]  
help6 <- help[seq(6, length(help), 25)]  
help7 <- help[seq(7, length(help), 25)]  
help8 <- help[seq(8, length(help), 25)]  
help9 <- help[seq(9, length(help), 25)]  
help10 <- help[seq(10, length(help), 25)]  
help11 <- help[seq(11, length(help), 25)]  
help12 <- help[seq(12, length(help), 25)]  
help13 <- help[seq(13, length(help), 25)]  
help14 <- help[seq(14, length(help), 25)]  
help15 <- help[seq(15, length(help), 25)]  
help16 <- help[seq(16, length(help), 25)]  
help17 <- help[seq(17, length(help), 25)]  
help18 <- help[seq(18, length(help), 25)]  
help19 <- help[seq(19, length(help), 25)]  
help20 <- help[seq(20, length(help), 25)]  
help21 <- help[seq(21, length(help), 25)]  
help22 <- help[seq(22, length(help), 25)]  
help23 <- help[seq(23, length(help), 25)]  
help24 <- help[seq(24, length(help), 25)]  
help25 <- help[seq(25, length(help), 25)]

#FINAL DATAFRAME   
#ONLY first 10 movies (saving time)  
  
df <- data.frame(title = titless,  
 avg\_rating = avg\_rating1,  
 num\_rating = num\_rating1,  
 year\_released = year\_released1,  
 revenue\_usa = reve,  
 budget = budget1,  
 genre = genre1,  
 top4 = top4actor,  
 review1 = review1[1:10],  
 help1 = help1,  
 review2 = review2[1:10],  
 help2 = help2,  
 review3 = review3[1:10],  
 help3 = help3,  
 review4 = review4[1:10],  
 help4 = help4,  
 review5 = review5[1:10],  
 help5 = help5,  
 review6 = review6[1:10],  
 help6 = help6,  
 review7 = review7[1:10],  
 help7 = help7,  
 review8 = review8[1:10],  
 help8 = help8,  
 review9 = review9[1:10],  
 help9 = help9,  
 review10 = review10[1:10],  
 help10 = help10,  
 review11 = review11[1:10],  
 help11 = help11,  
 review12 = review12[1:10],  
 help12 = help12,  
 review13 = review13[1:10],  
 help13 = help13,  
 review14 = review14[1:10],  
 help14 = help14,  
 review15 = review15[1:10],  
 help15 = help15,  
 review16 = review16[1:10],  
 help16 = help16,  
 review17 = review17[1:10],  
 help17 = help17,  
 review18 = review18[1:10],  
 help18 = help18,  
 review19 = review19[1:10],  
 help19 = help19,  
 review20 = review20[1:10],  
 help20 = help20,  
 review21 = review21[1:10],  
 help21 = help21,  
 review22 = review22[1:10],  
 help22 = help22,  
 review23 = review23[1:10],  
 help23 = help23,  
 review24 = review24[1:10],  
 help24 = help24,  
 review25 = review25[1:10],  
 help25 = help25  
 )

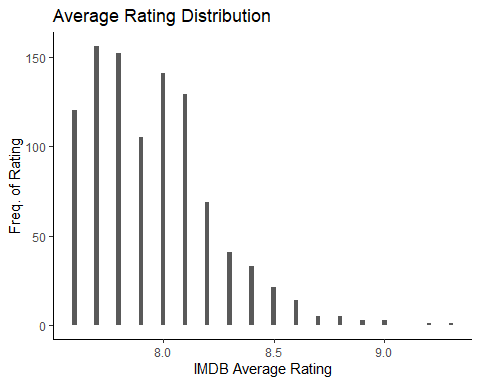
*Victory conditions:* You have a data frame with 1000 rows and columns that contain the first six variables, as well as each genre, review, and review helpfulness scores in appropriately formatted columns. Columns have short but clear names. Most rows have no missing values; the few rows where there are missing values have NA in the appropriate places.

*Mercy condition:* If you are struggling to get these data in a reasonable form, a compiled, reasonably clean and accurate version for either the URL list or movie data will be added to Canvas called imdb\_urls.csv and moviedata.Rdata respectively. Use them to answer the following parts, if necessary. Your work in this part, even if not complete, can be scored for partial credit.

#Data that was given  
data <- read\_csv("moviedata.csv")

1. (30 pts) Write code to plot the distributions of the first five variables listed above. Make sure missing values, if any, are handled gracefully. Your plots should be appropriately labeled, titled, colored, etc. Comment on the features each plot presents – what information is gained by seeing these graphics?

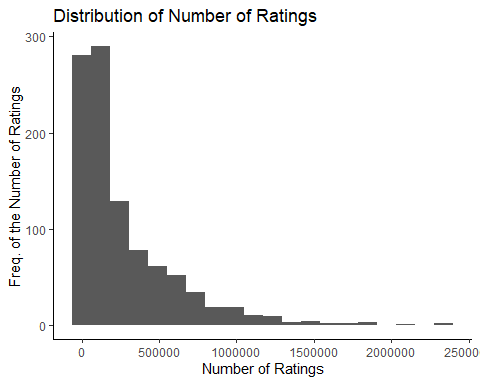
#Distribution of avg rating  
data %>%  
 ggplot(aes(rating)) +   
 geom\_histogram(binwidth = .020) + #can fit each different rating  
 xlab("IMDB Average Rating") +  
 ylab("Freq. of Rating") +  
 ggtitle("Average Rating Distribution") +  
 theme\_classic()



#returns max and min for average rating  
#data %>%  
 #summarise(range = range(rating))  
  
#returns highest rated film  
#data %>%   
 #filter(rating == max(rating))

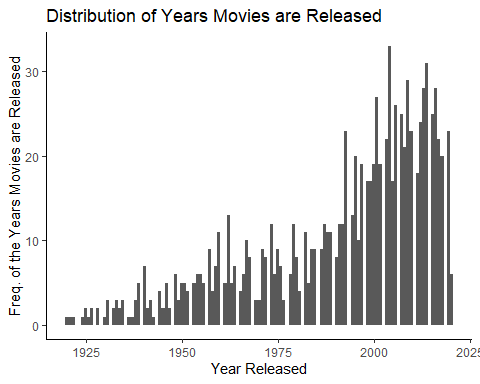
The average rating for the top 1000 films is distributed from 7.6 to 9.3. I would expect the ratings to be above 7 as these movies are considered the top 1000. The winner for the most highly rated film is The Shawshank Redemption. It looks like I need another movie to add to the watchlist.

#Dist of number of ratings   
data %>%  
 ggplot(aes(num\_ratings)) +  
 geom\_histogram(bins = 20) + #bins are small, num rating has different values  
 xlab("Number of Ratings") +  
 ylab("Freq. of the Number of Ratings") +  
 ggtitle("Distribution of Number of Ratings") +  
 theme\_classic()



For the total number of ratings for each film, most fall under 1,000,000 ratings per film. For each film, the amount of people who decided to rate the film on IMDB after watching is what represents the frequency. I wonder what the ratio of total ratings to total views to see what percent of people rate the films after watching, this could be more meaningful in the analysis of the distribution of the total number of ratings for each film.

#Dist for year released  
data %>%  
 ggplot(aes(year)) +  
 geom\_histogram(bins = 125) + #bin each year on scale  
 xlab("Year Released") +  
 ylab("Freq. of the Years Movies are Released") +  
 ggtitle("Distribution of Years Movies are Released") +  
 theme\_classic()

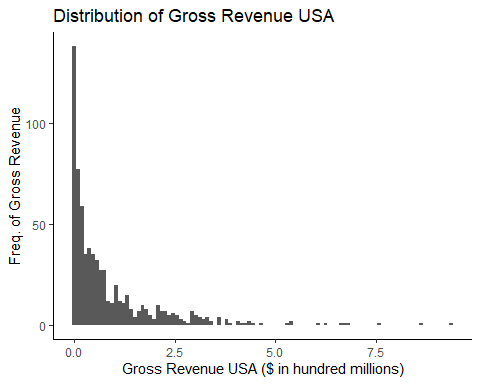


#filters all movies older than 1940  
#data %>%  
 #filter(year < 1940)

This distribution shows that the top 1000 movies were released in the last 25 years. With the rapid growth of technology, we are able to access information about different films and read reviews to decide if a movie is worth watching. In addition, the quality of films are only getting better. It makes sense why we see newer movies rated highly, but older films that are influential, such as Gone with the Wind or Wizard of Oz, I could see as being included in the top 1000.

#Dist for gross revenue of the movie  
data %>%  
 ggplot(aes(gross/10^8)) +  
 geom\_histogram(bins = 100) + #bins are large  
 xlab("Gross Revenue USA ($ in hundred millions)") +  
 ylab("Freq. of Gross Revenue") +  
 ggtitle("Distribution of Gross Revenue USA") +  
 theme\_classic()

## Warning: Removed 307 rows containing non-finite values (stat\_bin).

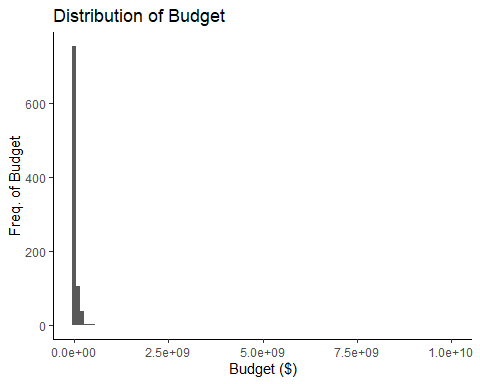


#data %>%  
 #filter(gross > 750000000)  
#data %>%  
 #summarise(median = median(gross, na.rm = TRUE))

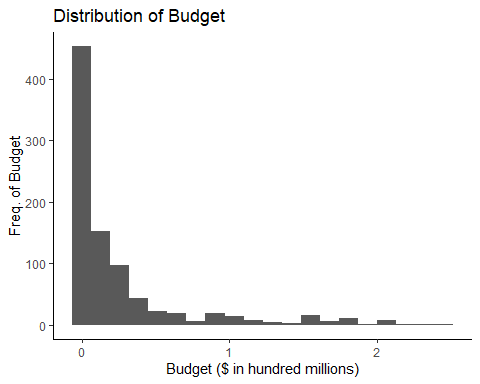
The distribution is a left skew, a majority of the films had gross revenues less than 250,000,000 dollars. The distribution is most dense less than $250,000,000. There are a few outliers in the distribution that generated lots of revenue in the USA. Avengers: Endgame, Avatar, and Star Wars VII all generated more than 750,000,000 dollars of revenue in the USA. The median of the distribution is 42,125,180 dollars. The films that generated more than 750,000,000 dollars in revenue make all the other movies look like failures in comparison, but thats why we have to look at the budgets distribution.

#Dist for the budget  
data %>%  
 ggplot(aes(budget)) +  
 geom\_histogram(bins = 100) +  
 xlab("Budget ($)") +  
 ylab("Freq. of Budget") +  
 ggtitle("Distribution of Budget") +  
 theme\_classic()

## Warning: Removed 92 rows containing non-finite values (stat\_bin).



#Distribution is skewed from two large budgets, want to see where majority of data falls  
data %>%  
 filter(budget < 250000000) %>%  
 ggplot(aes(budget/10^8)) +  
 geom\_histogram(bins = 20) +  
 xlab("Budget ($ in hundred millions)") +  
 ylab("Freq. of Budget") +  
 ggtitle("Distribution of Budget") +  
 theme\_classic()



#Outrageous budgets   
#data %>%  
 #filter(budget >= 2500000000)

Looking at the first distribution for the budget, the budget is heavily distributed towards films that were less than 250,000,000 dollars. I wanted to filter the data to get a better look at distribution of budgets from the different films. I filtered the data and found that most of the movies had budgeted less than 50,000,000 dollars. The two movies that were outliers that skewed our original distribution were The Handmaiden and Baahubali 2: The Conclusion. Both movies had insanley large budgets of 10,000,000,000 dollars and 2,500,000,000 dollars, respectively! By a quick glance, both budgets exceed the gross revenue in the USA so lets hope the movies were films intended for an international audience!

1. (20 pts) Complete a sentiment analyses on the 25 most helpful reviews for each movie. The choice of lexicon is up to you, but explain your reasons why your choice is the most reasonable/appropriate option. Add a summary of this information to your original data frame.

afinn <- get\_sentiments("afinn")   
  
#This function will intake the review and output the total score of sentiments using the afinn lexicon  
sentiment\_fun <- function(text\_box) {   
 input <- text\_box  
 score\_df <- input %>%   
 tibble(review = input[1:999], text = input) %>%  
 unnest\_tokens(word, text) %>%   
 filter(!word %in% stop\_words$word) %>%  
 inner\_join(afinn) %>%  
 group\_by(review) %>%  
 summarise(sent\_score = sum(value))  
 return(score\_df[["sent\_score"]])  
  
}  
#assigning the total scores for review 1 through 25  
review1\_score <- sentiment\_fun(data$Review\_1)   
review2\_score <- sentiment\_fun(data$Review\_2)  
review3\_score <- sentiment\_fun(data$Review\_3)  
review4\_score <- sentiment\_fun(data$Review\_4)  
review5\_score <- sentiment\_fun(data$Review\_5)  
review6\_score <- sentiment\_fun(data$Review\_6)  
review7\_score <- sentiment\_fun(data$Review\_7)  
review8\_score <- sentiment\_fun(data$Review\_8)  
review9\_score <- sentiment\_fun(data$Review\_9)  
review10\_score <- sentiment\_fun(data$Review\_10)  
review11\_score <- sentiment\_fun(data$Review\_11)  
review12\_score <- sentiment\_fun(data$Review\_12)  
review13\_score <- sentiment\_fun(data$Review\_13)  
review14\_score <- sentiment\_fun(data$Review\_14)  
review15\_score <- sentiment\_fun(data$Review\_15)  
review16\_score <- sentiment\_fun(data$Review\_16)  
review17\_score <- sentiment\_fun(data$Review\_17)  
review18\_score <- sentiment\_fun(data$Review\_18)  
review19\_score <- sentiment\_fun(data$Review\_19)  
review20\_score <- sentiment\_fun(data$Review\_20)  
review21\_score <- sentiment\_fun(data$Review\_21)  
review22\_score <- sentiment\_fun(data$Review\_22)  
review23\_score <- sentiment\_fun(data$Review\_23)  
review24\_score <- sentiment\_fun(data$Review\_24)  
review25\_score <- sentiment\_fun(data$Review\_25)  
  
#This loop will create the vector for the overall sent scores from all 25 reviews summed up for each movie  
overall\_movie\_sent\_score <- c()  
for (i in 1:999) {  
 overall\_movie\_sent\_score[i] <-   
 review1\_score[i] +  
 review2\_score[i] +  
 review3\_score[i] +  
 review4\_score[i] +  
 review5\_score[i] +  
 review6\_score[i] +  
 review7\_score[i] +  
 review8\_score[i] +  
 review9\_score[i] +  
 review10\_score[i] +  
 review11\_score[i] +  
 review12\_score[i] +  
 review13\_score[i] +  
 review14\_score[i] +  
 review15\_score[i] +  
 review16\_score[i] +  
 review17\_score[i] +  
 review18\_score[i] +  
 review19\_score[i] +  
 review20\_score[i] +  
 review21\_score[i] +  
 review22\_score[i] +  
 review23\_score[i] +  
 review24\_score[i] +  
 review25\_score[i]   
   
}

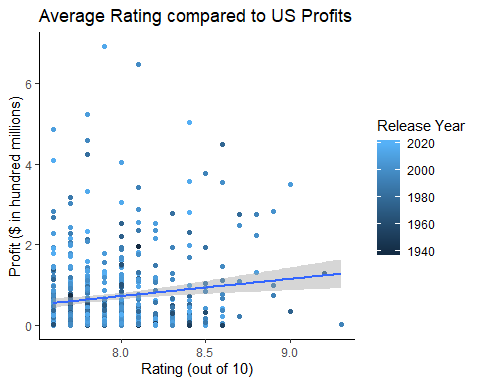
I decided to use the afinn lexicon when doing the sentiment analysis. This lexicon works for this specific example because we are using the sentiments to determine how positive/how negative the sum of the sentiments using a -5/5 scale with -5 being negative and 5 being positive. Some words hold more of a weight because of meaning the english language. I wanted to measure the polarity of the sentiments for each review, meaning how positive the sentiments are (sum of the values the words hold). For example, if a film has a large sum from all 25 reviews, then we could assume that the movie has good reviews because there are lots of positive sentiments found in the review. The length of the review has an effect on the final score, so we would need to look farther into that to determine if we can make that conclusion that a large sentiment score means the film is worth watching.

#adding data to data given to us  
data <- data %>%  
 cbind(overall\_movie\_sent\_score)  
  
#adding data to my dataframe I created  
df <- df %>%  
 cbind(overall\_movie\_sent\_score[1:10])

1. (20 pts) Variable Relationships. Create one plot that displays the relationship (or lack thereof) between any of the movie variables. Your plot should be appropriately labeled, titled, colored, etc. Your plot should display at minimum 3 variables. A plot with more variables included will be scored more favorably (as long as the plot is reasonable and legible). Reviews and helpfulness/sentiment scores are considered aggregate variables (e.g., 25 helpfulness scores will be considered a single variable).

A relationship I wanted to visualize was if rating had an effect on what profits the film generated in the United States. In order to do this, I needed to mutate the original data frame to subtract the budget from the gross revenue. Then I filtered the data to only look at movies that were profitable in the US. I grouped the films by the year they were released, so I could analyze if the year the movie was released had an effect on the profit and rating of the movie. I would expect to see more highly rated movies correlate with higher profits.

#Variable Relationships(budget, gross, profit(us), rating, year)  
  
data %>%  
 mutate(profit = gross - budget) %>%  
 filter(profit > 0) %>%  
 ggplot(aes(x=rating, y= profit/10^8)) +  
 geom\_point(aes(color=year)) +  
 geom\_smooth(method = "lm") +  
 xlab("Rating (out of 10)") +  
 ylab("Profit ($ in hundred millions)") +  
 ggtitle("Average Rating compared to US Profits") +  
 labs(color = 'Release Year') +  
 theme\_classic()



From the trendline presented in the figure, it looks like profit does increased as rating increases. It would be challenging to generalize this for all films because the data has a lot of variance. The release year seems to not have an effect on the rating or the profit. The legend helps us to understand what time period the films came from. The graph shows not pattern of different colors so the release year is independent of how much the film generated in profit as well as the rating of the film.