Matthew L. Miller mmiller319 — title: “Project 1: Explore and Prepare Data” subtitle: “CSE6242 - Data and Visual Analytics - Summer 2018: Sunday, June 17, 2018 at 11:59 PM UTC-12:00 on T-Square” output: html_notebook —

Note: This project involves getting data ready for analysis and doing some preliminary investigations. Project 2 will involve modeling and predictions on the same dataset, and will be released at a later date. Both projects will have equal weightage towards your grade. You may reuse some of the preprocessing/analysis steps from Project 1 in Project 2.

Data

In this project, you will explore a dataset that contains information about movies, including ratings, budget, gross revenue and other attributes. It was prepared by Dr. Guy Lebanon, and here is his description of the dataset:

The file movies_merged contains a dataframe with the same name that has 40K rows and 39 columns. Each row represents a movie title and each column represents a descriptor such as `Title`, `Actors`, and `Budget`. I collected the data by querying IMDbâ??s API (see www.omdbapi.com) and joining it with a separate dataset of movie budgets and gross earnings (unknown to you). The join key was the movie title. This data is available for personal use, but IMDbâ??s terms of service do not allow it to be used for commercial purposes or for creating a competing repository.

Objective

Your goal is to investigate the relationship between the movie descriptors and the box office success of movies, as represented by the variable `Gross`. This task is extremely important as it can help a studio decide which titles to fund for production, how much to bid on produced movies, when to release a title, how much to invest in marketing and PR, etc. This information is most useful before a title is released, but it is still very valuable after the movie is already released to the public (for example it can affect additional marketing spend or how much a studio should negotiate with on-demand streaming companies for â??second windowâ?? streaming rights).

Instructions

This is an R Markdown Notebook. Open this file in RStudio to get started.

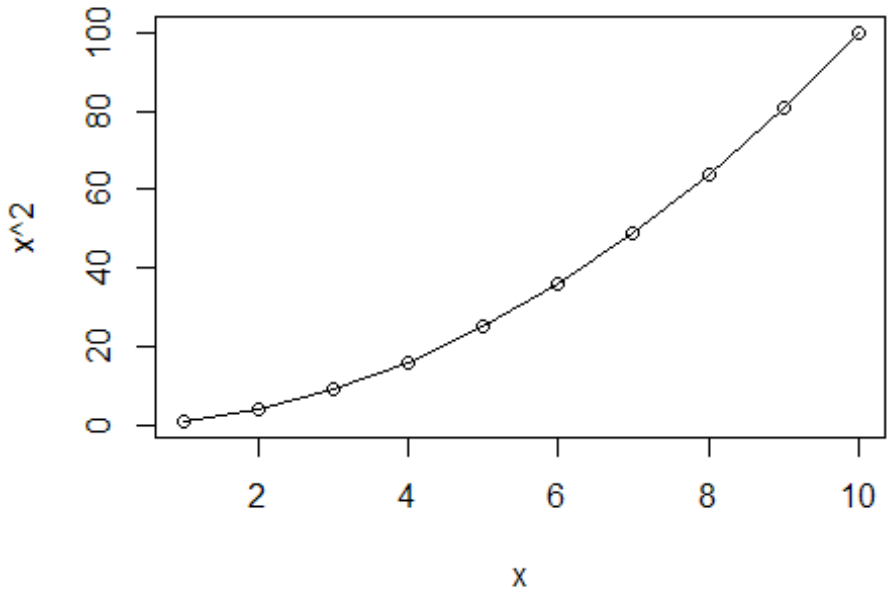
When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```
x = 1:10
print(x^2)
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

Plots appear inline too:

```
plot(x, x^2, 'o')
```



Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*. Enter some R code and run it.

```
#names(movies_merged)
```

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

Please complete all the tasks below by implementing code chunks that have a `TODO` comment in them, running all code chunks so that output and plots are displayed, and typing in answers to each question (**Q:** ...) next to/below the corresponding answer prompt (**A:**). Feel free to add code chunks/show additional output to support any of the answers.

When you are done, you will need to submit the final R markdown file (as **pr1.Rmd**) with all code chunks implemented and executed, and all text responses written in. You also need to submit a PDF export of the markdown file (as **pr1.pdf**), which should show your code, output, plots and written responses–this will be your project report. Compress these two files into a single .zip archive and upload it on T-Square.

Setup

Load data

Make sure you've downloaded the movies_merged file and it is in the current working directory. Now load it into memory:

```
load('movies_merged')
cat("Dataset has", dim(movies_merged)[1], "rows and", dim(movies_merged)[2],
"columns", end="\n", file="")

## Dataset has 40789 rows and 39 columns
```

This creates an object of the same name (`movies_merged`). For convenience, you can copy it to `df` and start using it:

```
df = movies_merged
cat("Column names:", end="\n", file="")

## Column names:

colnames(df)

##  [1] "Title"             "Year"              "Rated"
##  [4] "Released"          "Runtime"           "Genre"
##  [7] "Director"          "Writer"            "Actors"
## [10] "Plot"              "Language"          "Country"
## [13] "Awards"            "Poster"            "Metascore"
## [16] "imdbRating"        "imdbVotes"         "imdbID"
## [19] "Type"              "tomatoMeter"       "tomatoImage"
## [22] "tomatoRating"      "tomatoReviews"     "tomatoFresh"
## [25] "tomatoRotten"      "tomatoConsensus"   "tomatoUserMeter"
## [28] "tomatoUserRating"  "tomatoUserReviews" "tomatoURL"
## [31] "DVD"               "BoxOffice"         "Production"
## [34] "Website"           "Response"          "Budget"
## [37] "Domestic_Gross"    "Gross"             "Date"
```

Load R packages

Load any R packages that you will need to use. You can come back to this chunk, edit it and re-run to load any additional packages later.

```
library(ggplot2)
library(GGally)
```

If you are loading any non-standard packages (ones that have not been discussed in class or explicitly allowed for this project), please mention them below. Include any special instructions if they cannot be installed using the regular `install.packages('<pkg name>')` command.

Non-standard packages used: None

Tasks

Each task below is worth **10** points, and is meant to be performed sequentially, i.e. do step 2 after you have processed the data as described in step 1. Total points: **100**

Complete each task by implementing code chunks as described by `TODO` comments, and by responding to questions ("**Q**:") with written answers ("**A**:"). If you are unable to find a meaningful or strong relationship in any of the cases when requested, explain why not by referring to appropriate plots/statistics.

It is okay to handle missing values below by omission, but please omit as little as possible. It is worthwhile to invest in reusable and clear code as you may need to use it or modify it in project 2.

1. Remove non-movie rows

The variable `Type` captures whether the row is a movie, a TV series, or a game. Remove all rows from `df` that do not correspond to movies.

```
# TODO: Remove all rows from df that do not correspond to movies
df2 <- df[df$Type == "movie",]
dim(df2)

## [1] 40000    39
```

Q: How many rows are left after removal? *Enter your response below.*

A: 40000

2. Process `Runtime` column

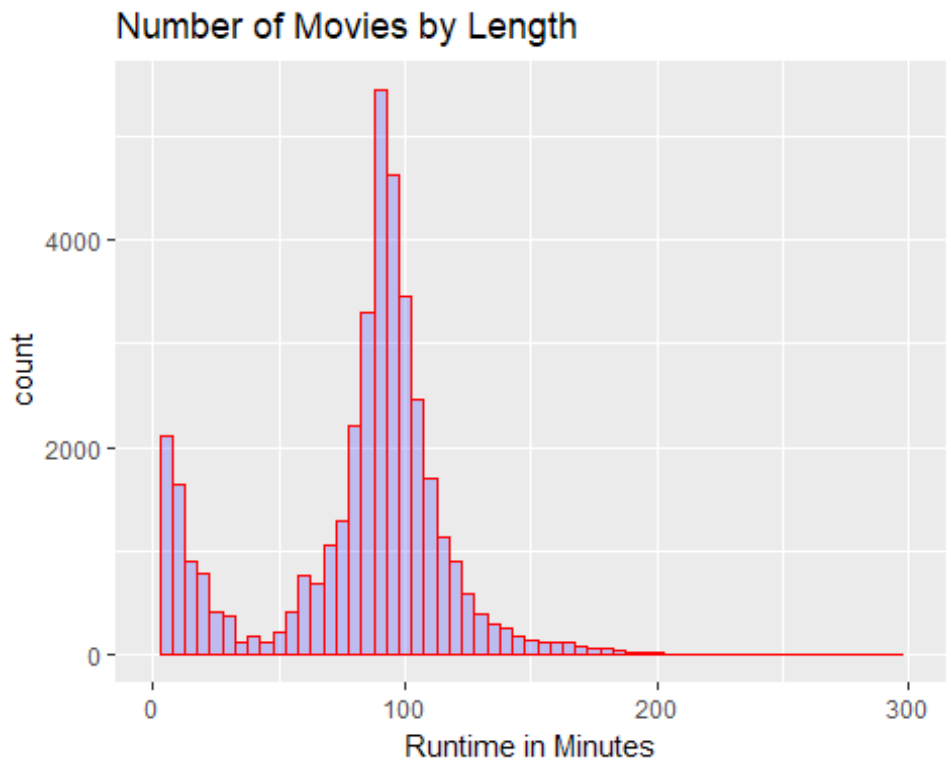
The variable `Runtime` represents the length of the title as a string. Write R code to convert it to a numeric value (in minutes) and replace `df$Runtime` with the new numeric column.

```
# TODO: Replace df$Runtime with a numeric column containing the runtime in
minutes
for(row in 1:nrow(df2)){
  rt = df2[row, "Runtime"]
  rt_parse = strsplit(rt, " ")[[1]]
  rt_numeric = 0

  if(rt_parse[1] == 'N/A'){
    rt_numeric = NA
  } else if (rt_parse[2] == 'min'){
    rt_numeric = as.numeric(as.character(rt_parse[1]))
  } else if (rt_parse[2] == 'h' & length(rt_parse) == 2){
    rt_numeric = 60 * as.numeric(as.character(rt_parse[1]))
  } else if (rt_parse[2] == 'h' & !is.null(rt_parse[3])){
    rt_numeric = 60 * as.numeric(as.character(rt_parse[1])) +
as.numeric(as.character(rt_parse[3]))
```

```
  }

  df2[row, "runtime_numeric"] = rt_numeric
}

df2$Runtime = df2$runtime_numeric
df2 = df2[-40]
```

Now investigate the distribution of `Runtime` values and how it changes over years (variable `Year`, which you can bucket into decades) and in relation to the budget (variable `Budget`). Include any plots that illustrate.

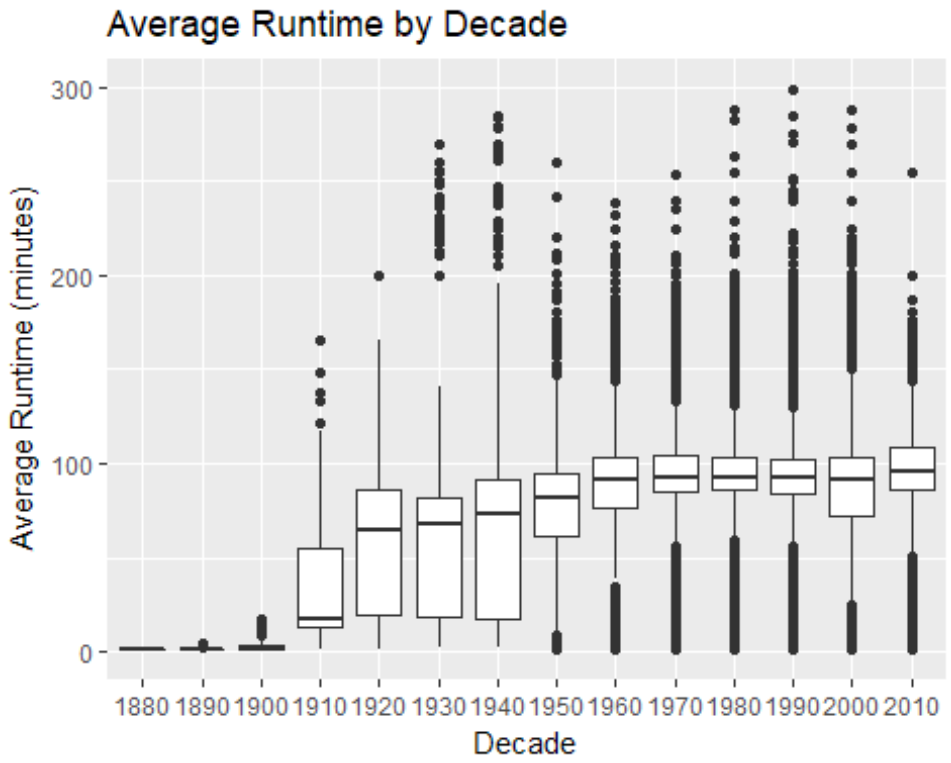
```
# TODO: Investigate the distribution of Runtime values and how it varies by Year and Budget
qplot(df2$Runtime, geom="histogram", binwidth=5, xlim=c(0,300), main="Number of Movies by Length", xlab="Runtime in Minutes", fill=I("blue"), col=I("red"), alpha=I(0.2), na.rm=TRUE)
```



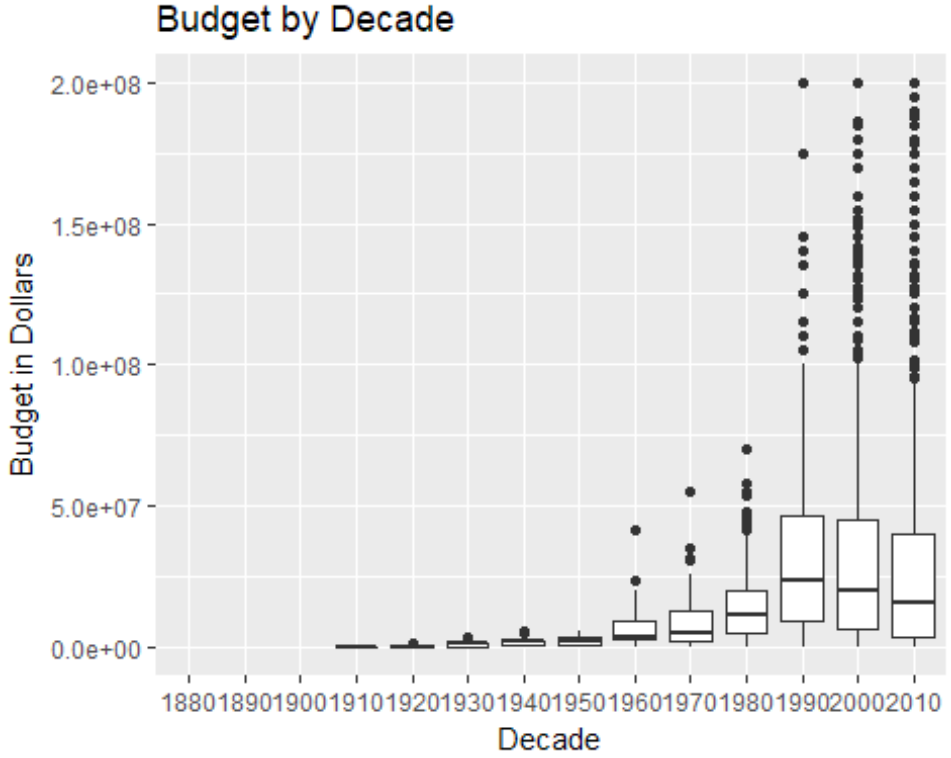
Feel free to insert additional code chunks as necessary.

```
for(row in 1:nrow(df2)){

 year = df2[row, "Year"]
 decade = 0

 if(year > 1880 & year < 1890){
   decade = 1880
```

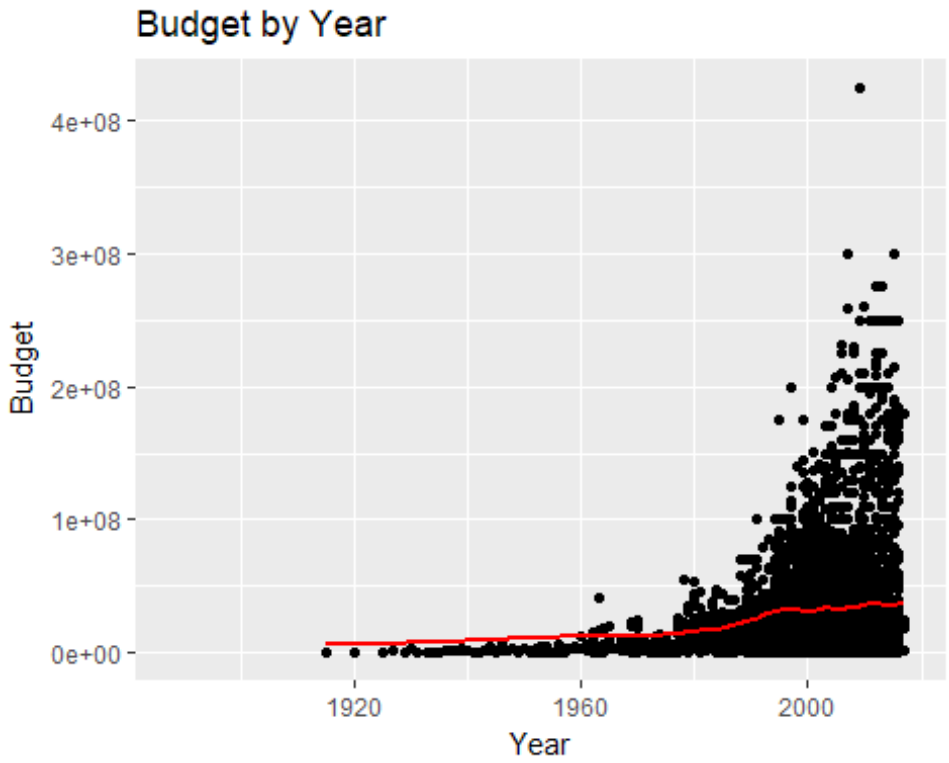
```
 } else if (year >= 1890 & year < 1900){
   decade = 1890
 } else if (year >= 1900 & year < 1910){
   decade = 1900
 } else if (year >= 1910 & year < 1920){
   decade = 1910
 } else if (year >= 1920 & year < 1930){
   decade = 1920
 } else if (year >= 1930 & year < 1940){
   decade = 1930
 } else if (year >= 1940 & year < 1950){
   decade = 1940
 } else if (year >= 1950 & year < 1960){
   decade = 1950
 } else if (year >= 1960 & year < 1970){
   decade = 1960
 } else if (year >= 1970 & year < 1980){
   decade = 1970
 } else if (year >= 1980 & year < 1990){
   decade = 1980
 } else if (year >= 1990 & year < 2000){
   decade = 1990
 } else if (year >= 2000 & year < 2010){
   decade = 2000
 } else if (year >= 2010){
   decade = 2010
 }

 df2[row, "Decade"] = as.character(as.numeric(decade))

}
ggplot(df2, aes(x=Decade, y=Runtime)) + geom_boxplot(na.rm = TRUE) +
xlab("Decade") + ylab("Average Runtime (minutes)") + ylim(0,300) +
ggtitle("Average Runtime by Decade")
```




```
ggplot(df2, aes(Decade, Budget)) + geom_boxplot(na.rm = TRUE) +
xlab("Decade") + ylab("Budget in Dollars") + ylim(0,2e+08) + ggtitle("Budget
by Decade")
```



```
qplot(Year, Budget, data=df2, na.rm=TRUE, main="Budget by Year") +
stat_smooth(method="loess", method.args = list(degree=0), span=0.2, se=FALSE,
color="red", size=1, na.rm=TRUE)
```



Q: Comment on the distribution as well as relationships. Are there any patterns or trends that you can observe?

A: Based on the histogram that shows the number of movies by length, it is clear there are two distinct groups: movies that have lengths around 30 minutes or less, and a larger group of movies that tend to cluster around the 90-minute mark. One explanation is that the movies dataset contains a large number of "Short" films, as described in the Genre category. Also, it is clear from the box plot showing Average Runtime by Decade, that the typical length of a film has increased since the late 1800's and early 1900's, when most movies in the dataset tend to be shorter than those after around 1950. Most movies made between 1880-1910 have lengths of only a few minutes, while the median and Interquartile Range for movie lengths is less than 90 minutes until the 1950s. By the 1960's, the median movie length stabilizes at approximately 90 minutes and remains unchanged until the 2010's. The Interquartile Range also becomes much smaller, indicating that more movies are clustered around the 90-minute mark after the 1960's.

When analyzing movie budgets over time, it is clear from the box plot that budgets have increased as well throughout the 20th century. There is a steady increase in median and IQR for budgets in the 2nd half of the century. Interestingly, the median and IQR for movie budgets have declined from a high in the 1990s through the 2010's. However, there appears to be a larger number of outliers in the 2000s and 2010s that have substantially

higher than average budgets. This likely indicates that there is a growth in very high budget blockbuster films during this time, while also a substantially growth in lower budget films as well. The scatter plot of budget vs. time also shows a general upward trend in budget over the decades, but it also shows a large number of very high budget films costing $100 million or more.

3. Encode `Genre` column

The column `Genre` represents a list of genres associated with the movie in a string format. Write code to parse each text string into a binary vector with 1s representing the presence of a genre and 0s the absence, and add it to the dataframe as additional columns. Then remove the original `Genre` column.

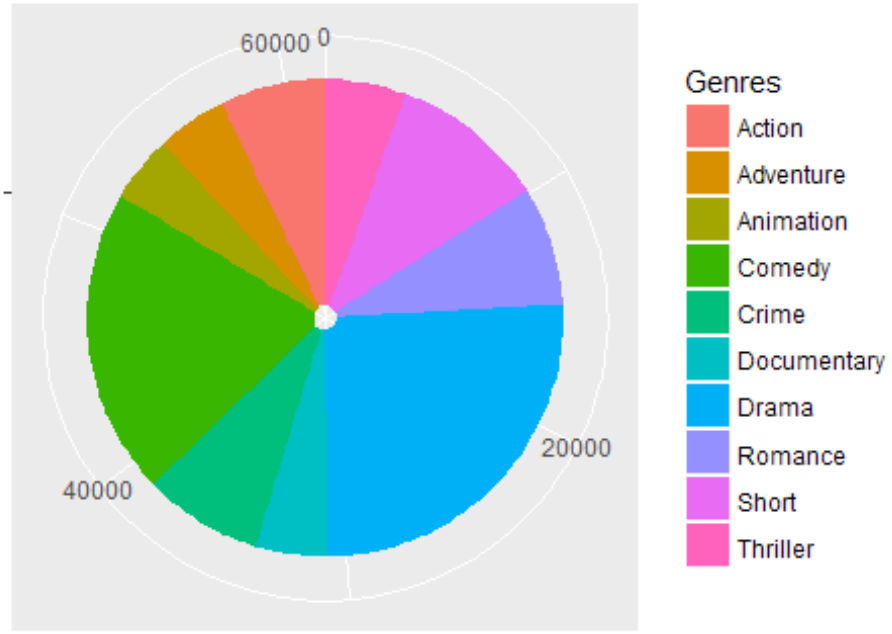
For example, if there are a total of 3 genres: Drama, Comedy, and Action, a movie that is both Action and Comedy should be represented by a binary vector <0, 1, 1>. Note that you need to first compile a dictionary of all possible genres and then figure out which movie has which genres (you can use the R `tm` package to create the dictionary).

```
# TODO: Replace Genre with a collection of binary columns
dict = c()

# Create dictionary of all genres
for(row in 1:nrow(df2)){

  genre = df2[row, "Genre"]
  genre_parse = strsplit(genre, ", ")[[1]]

  for(i in 1:length(genre_parse)){
    word = genre_parse[i]

    if(!word %in% dict & word != 'N/A'){
      dict = c(dict, word)
    }
  }
}

dict = sort(dict)
#**********************************


# Create new columns with 1's and 0's
for(row in 1:nrow(df2)){

  genre = df2[row, "Genre"]
  genre_parse = strsplit(genre, ", ")[[1]]

  for(i in 1:length(dict)){
```

```
    if(dict[i] %in% genre_parse){
      df2[row, dict[i]] = 1
    } else {
      df2[row, dict[i]] = 0
    }
  }
}

df2 <- df2[-c(6)]
```

Plot the relative proportions of movies having the top 10 most common genres.

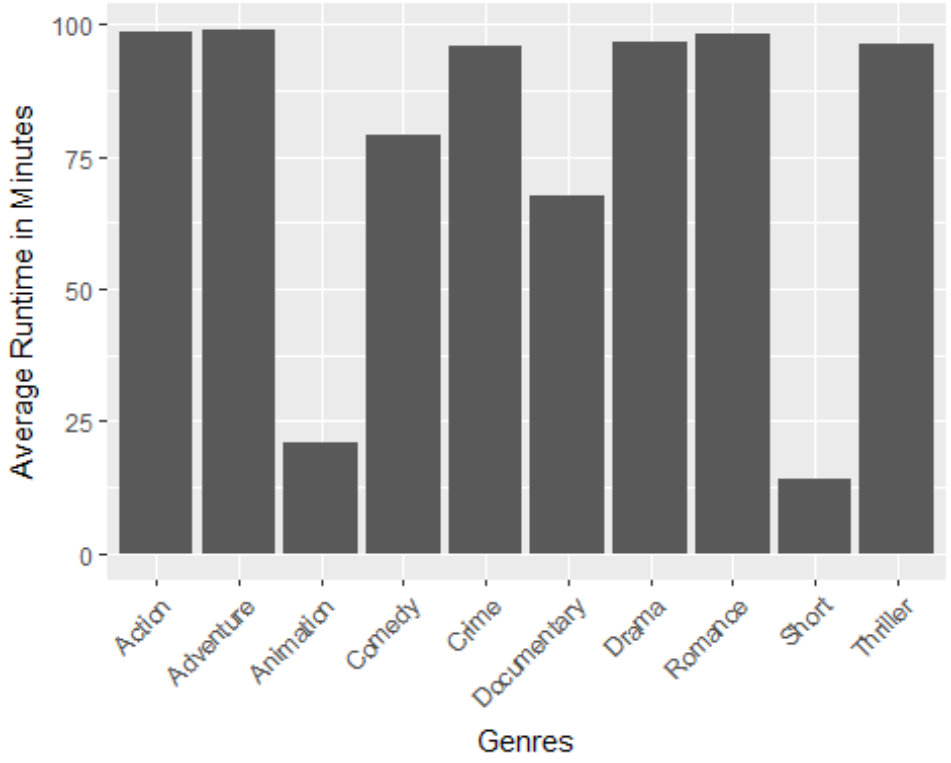
```
# TODO: Select movies from top 10 most common genres and plot their relative
proportions
Genres <-
c("Action","Adventure","Animation","Comedy","Crime","Documentary","Drama","Ro
mance","Short","Thriller")
Count <- c(4413, 2928, 2788, 12849, 4962, 3051, 15859, 4975, 6516, 3380)
Genre_Count <- data.frame(Genres, Count)
ggplot(Genre_Count, aes(x="", y=Count, fill=Genres)) +
geom_bar(stat="identity") + coord_polar("y", start=0) + xlab("") +
ylab("Proportion of Movies by Genre")
```



Proportion of Movies by Genre

Examine how the distribution of `Runtime` changes across genres for the top 10 most common genres.

```
# TODO: Plot Runtime distribution for top 10 most common genres
Genres <-
c("Action","Adventure","Animation","Comedy","Crime","Documentary","Drama","Ro
mance","Short","Thriller")
Runtimes <- c(98.7, 99.1, 20.89, 79.1, 95.9, 67.8, 96.9, 98.1, 13.9, 96.4)
Genres_Runtimes <- data.frame(Genres, Runtimes)
ggplot(Genres_Runtimes, aes(x=Genres, y=Runtimes)) +
geom_bar(stat="identity") + ylab("Average Runtime in Minutes") +
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Q: Describe the interesting relationship(s) you observe. Are there any expected or unexpected trends that are evident?

A: Overall, most of the genres, especially the more popular ones such as Action, Adventure, Romance, and Drama, all have average runtimes very close to the 90-minute mark. Not surprisingly, though, the Short category has the shortest average runtimes at less than 15 minutes. Animation is the second lowest average runtime, which is likely due to a large number of short cartoons being included in the database along with feature-length animated films. Documentaries are also somewhat lower than the others, mostly likely due to many documentaries lasting approximately one hour or so.

4. Eliminate mismatched rows

The dataframe was put together by merging two different sources of data and it is possible that the merging process was inaccurate in some cases (the merge was done based on movie title, but there are cases of different movies with the same title). There are 3

columns that contain date information: `Year` (numeric year), `Date` (numeric year), and `Released` (string representation of the release date).

Find and remove all rows where you suspect a merge error occurred based on a mismatch between these variables. To make sure subsequent analysis and modeling work well, avoid removing more than 10% of the rows that have a `Gross` value present.

Note: Do not remove the rows with `Gross == NA` at this point, just use this a guideline.

```
# TODO: Remove rows with Year/Date/Released mismatch
df2$row_to_keep = TRUE

for(row in 1:nrow(df2)){

  title = df2[row, "Title"]
  year = df2[row, "Year"]
  date = df2[row, "Date"]
  released = df2[row, "Released"]
  released = substring(released, 1, 4)
  released = as.numeric(as.character(released))



    if(!is.na(date) & !is.na(released) & !is.na(year)){

      if(abs(date-released)>10 | abs(released-year)>10 | abs(year-date)>10){
        df2[row, "row_to_keep"] = FALSE
      }
    }

    if(!is.na(date) & !is.na(released) & is.na(year)){

      if(abs(date - released) > 10){
        df2[row, "row_to_keep"] = FALSE
      }
    }

    if(!is.na(date) & is.na(released) & !is.na(year)){

      if(abs(date - year) > 10){
        df2[row, "row_to_keep"] = FALSE
      }
    }

    if(is.na(date) & !is.na(released) & !is.na(year)){

      if(abs(released - year) > 10){
        df2[row, "row_to_keep"] = FALSE
      }
```

```
    }
}

keep = as.vector(df2$row_to_keep)
df2 = df2[keep,]
df2 = df2[-c(67)]
```

Q: What is your precise removal logic, and how many rows remain in the resulting dataset?

A: I first experimented with removing all rows in which the Date, Year, and Released did not occur in the same year. However, I felt this removed too many movies because the Year column seems to be referring to the year in which the movie was filmed, while the Released column appears to refer to the date on which the movie was officially released in theaters. There were many films which had a Year value that was one less than the Release date. For example, there were a large number of films that had a Year value of 2014 and a Release or Date value of 2015. Therefore, I decided not to remove these films since I presume that Year could be referring to when filming took place or was started.

To ensure that movies were not mistakenly removed, I decided to remove films only when their Year, Date and/or Release values differed by more than 10 years. I think this is a reasonable assumption because it is unlikely that a movie would be filmed and then remain unreleased for a period of time of more than a decade. At the same time, it keeps films that have a one year discrepency between Year, Date, and Release, and also allows for some films that may take especially long time to produce due to factors such as special effects.

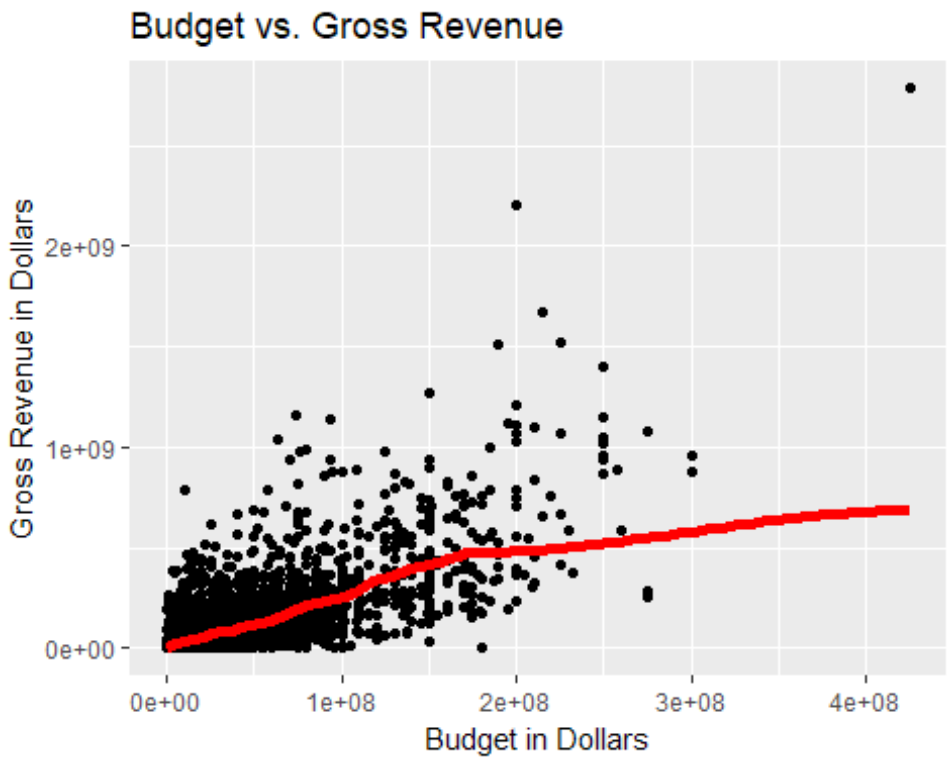
The final number of rows remaining in the dataset was 39,671, with 329 removed. This means that only about 1% of films were removed, but I felt that reducing the difference between Year, Date, and Release values would result in a higher number of films being incorrectly removed. Therefore, I felt it was best to be cautious of removing too many films, unless it is clear that there is a significant discrepancy in the Year, Date, and Release values.

5. Explore Gross revenue

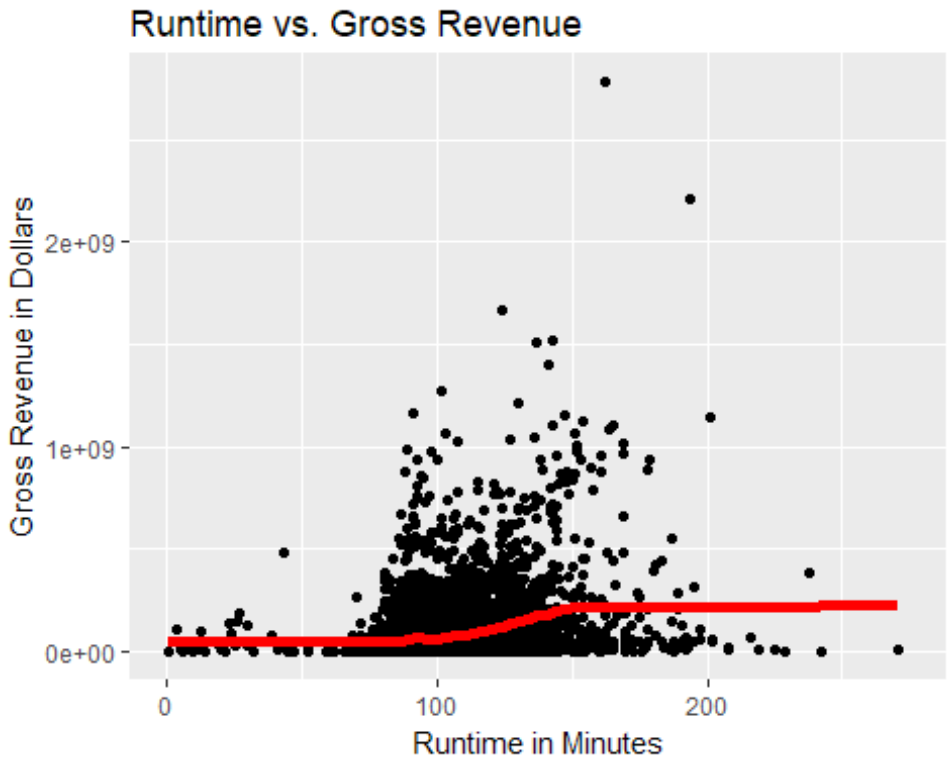
For the commercial success of a movie, production houses want to maximize Gross revenue. Investigate if Gross revenue is related to Budget, Runtime or Genre in any way.

Note: To get a meaningful relationship, you may have to partition the movies into subsets such as short vs. long duration, or by genre, etc.

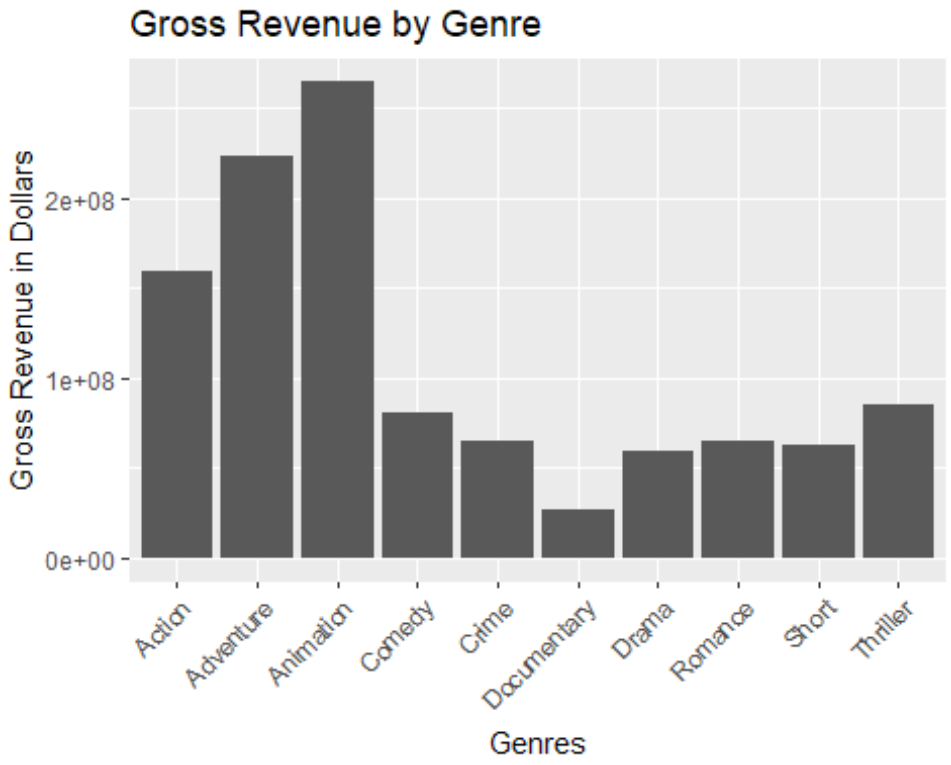
```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
qplot(Budget, Gross, data=df2, main="Budget vs. Gross Revenue", na.rm=TRUE) +
stat_smooth(method="loess", method.args=list(degree=0), span=0.1, se=FALSE,
color="red", size=2, na.rm=TRUE) + xlab("Budget in Dollars") + ylab("Gross
Revenue in Dollars")
```




```
qplot(Runtime, Gross, data=df2, main="Runtime vs. Gross Revenue", na.rm=TRUE)
+ stat_smooth(method="loess", method.args=list(degree=0), span=0.2, se=FALSE,
color="red", size=2, na.rm=TRUE) + xlab("Runtime in Minutes") + ylab("Gross
Revenue in Dollars") + xlim(0,275)
```




```
Genres <-
c("Action","Adventure","Animation","Comedy","Crime","Documentary","Drama","Ro
mance","Short","Thriller")
Gross <-
c(158765080,222708565,264267548,80673030,64685216,26691040,59743636,64516566,
62504121,85345918)
Genres_Gross <- data.frame(Genres, Gross)
ggplot(Genres_Gross, aes(x=Genres, y=Gross)) + geom_bar(stat="identity") +
ylab("Gross Revenue in Dollars") + theme(axis.text.x = element_text(angle =
45, hjust = 1)) + ggtitle("Gross Revenue by Genre")
```



Q: Did you find any observable relationships or combinations of Budget/Runtime/Genre that result in high Gross revenue? If you divided the movies into different subsets, you may get different answers for them - point out interesting ones.

A: The strongest relationship for Gross Revenue is clearly with budget. As budget increases, revenue increases with a correlation of 0.742. However, the relationship becomes less strong as budgets increase beyond the 200 million mark. This indicates that larger budget movies generally do have higher revenue, but some very large budget movies may underperform and have a revenue that is less than the budget.

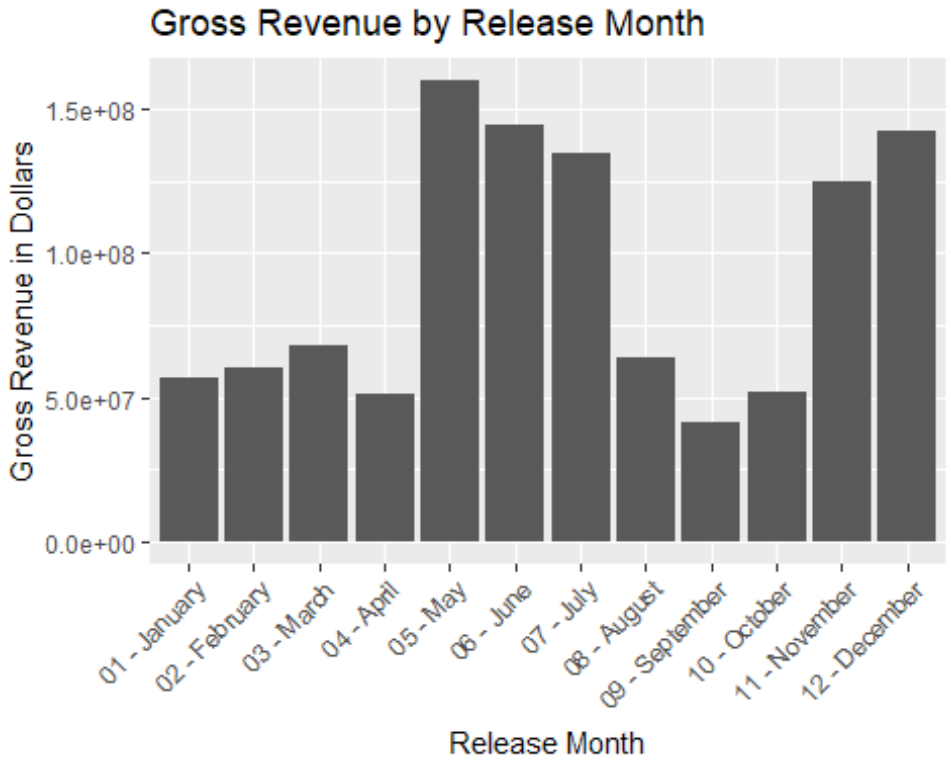
When comparing revenue to Runtime, there is a much smaller correlation of only 0.26. However, the plot does show a small upward trend where revenue increases along with runtime until the 150 minute mark. When runtime exceeds 150 minutes, revenue decreases. This suggests that movies that are too long may experience lower revenues than those that last a more typical 90 - 150 minutes or so. Also, many of the shortest movies have very low revenue, suggesting that films with a runtime of less than 60 minutes will earn lower revenues.

By genre, it is clear that Animation has the highest average revenue, with Adventure and Action having the second and third highest average revenues. The lowest average revenue is for Documentaries, while the other six most common genres have roughly comparable average revenues.

Based on the plots, the highest revenue for a film is likely to come from an Animation, Action, or Adventure film that lasts between 90-150 minutes and has a moderately high budget of 100-200 million dollars. Films of other genres clearly have lower average

revenue while films that exceed 150 minutes show a declining revenue. The correlation between budget and revenue shows that it is generally worthwhile to have a larger budget for a film, but a very large budet of over $200 million is less likely to substantially increase the revenue.

```
# TODO: Investigate if Gross Revenue is related to Release Month
months <- c("01 - January", "02 - February", "03 - March", "04 - April", "05 - May", "06 - June", "07 - July", "08 - August", "09 - September", "10 - October", "11 - November", "12 - December")
Gross <- c(57087252, 60600000, 67683799, 50975920, 160101906, 144610894, 135000000, 63900000, 41407110, 51597922, 125138775, 142663428)
GrossByMonth <- data.frame(months, Gross)
ggplot(GrossByMonth, aes(x=months, y=Gross)) + geom_bar(stat="identity") + ylab("Gross Revenue in Dollars") + xlab("Release Month") + ggtitle("Gross Revenue by Release Month") + theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



6. Process `Awards` column

The variable `Awards` describes nominations and awards in text format. Convert it to 2 numeric columns, the first capturing the number of wins, and the second capturing nominations. Replace the `Awards` column with these new columns, and then study the relationship of `Gross` revenue with respect to them.

Note: The format of the `Awards` column is not standard; you may have to use regular expressions to find the relevant values. Try your best to process them, and you may leave the ones that don't have enough information as NAs or set them to 0s.

```
# TODO: Convert Awards to 2 numeric columns: wins and nominations
for(row in 1:nrow(df2)){

 awards = df2[row, "Awards"]
 awards_parse = strsplit(awards, " ")[[1]]
 wins_num = 0
 noms_num = 0

 if(awards == 'N/A'){
    df2[row, "Wins"] = 0
    df2[row, "Nominations"] = 0
 } else {

   for(i in 1:length(awards_parse)){
     word = awards_parse[i]

     if(grepl("win", word)){
       wins_num =  wins_num + as.numeric(as.character(awards_parse[i-1]))
     }

     if(grepl("nomination", word)){
       noms_num = noms_num + as.numeric(as.character(awards_parse[i-1]))
     }

     if(grepl("Won", word) | grepl("won", word)){
       wins_num = wins_num + as.numeric(as.character(awards_parse[i+1]))
     }

     if(grepl("Nominated", word) & grepl("for", awards_parse[i+1])){
       noms_num = noms_num + as.numeric(as.character(awards_parse[i+2]))
     }
   }

   df2[row, "Wins"] = wins_num
   df2[row, "Nominations"] = noms_num
 }
}
```

Q: How did you construct your conversion mechanism? How many rows had valid/non-zero wins or nominations?

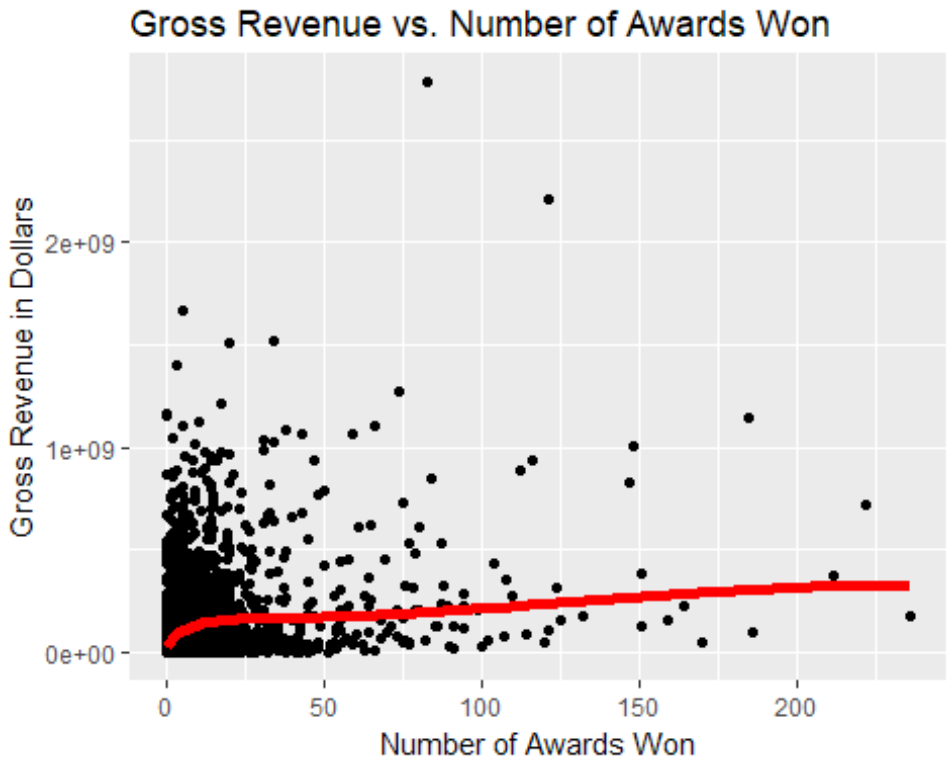
A: I began by using a for loop to cycle through the rows in df2 and accessing the "Awards" column for each row. The text of "Awards" was then split by spaces so that each word in the text was turned into an element of a vector called awards_parse. I then used the grepl function to search for words such as "win" or "nomination" because I noticed a common

pattern in which the awards were described as “X wins & Y nominations.” If “win” or “nomination” was found in the awards_parse vector, then the word just before it, at the i-1 index, was always the number of wins or nominations for the movie. I did not find any cases in which non-numeric text preceded “win” or “nomination”.

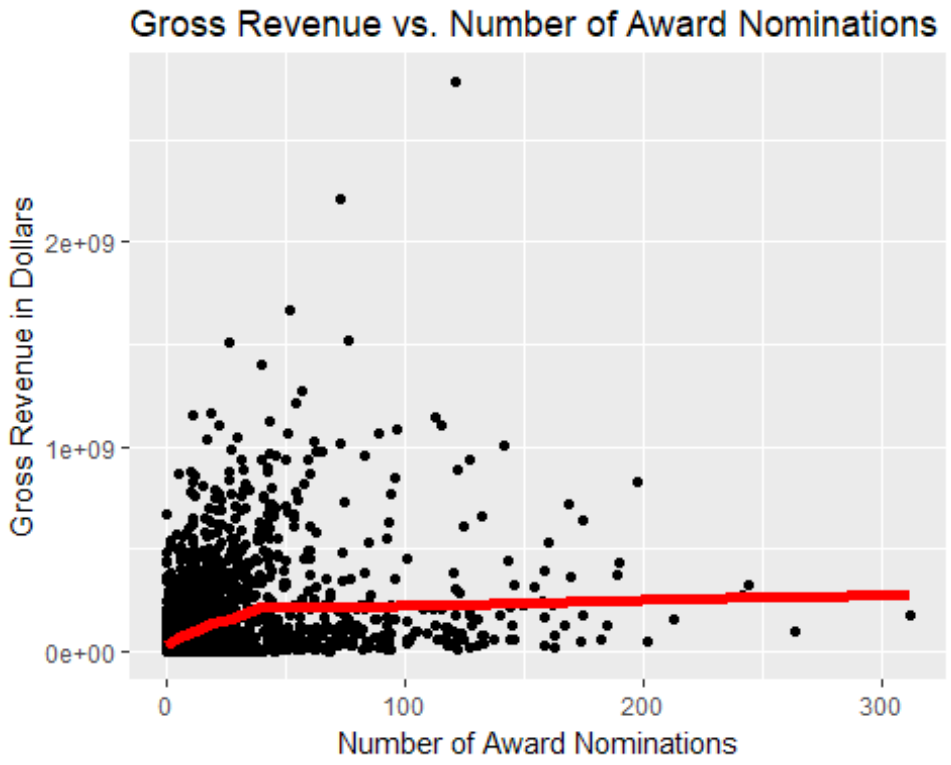
I also found another pattern in which the Awards were described as “Won X” awards. In this case, the text that followed “Won” was always numeric and so it could be added to the total number of awards won. In addition, many descriptions included the phrase “Nominated for” in which case the text following was always a numeric value.

By locating these patterns using grepl and adding the numbers that corresponded with them, I was able to sum the wins and nominations and insert them into their own columns in df2. I also included a check to determine whether the Awards text was “N/A”, in which case the wins and nominations values were both set to zero.

```
# TODO: Plot Gross revenue against wins and nominations
qplot(Wins, Gross, data=df2, main="Gross Revenue vs. Number of Awards Won",
xlab="Number of Awards Won", ylab="Gross Revenue in Dollars", na.rm=TRUE) +
stat_smooth(method="loess", method.args = list(degree=0), span=0.5, se=FALSE,
color="red", size=2, na.rm=TRUE)
```




```
qplot(Nominations, Gross, data=df2, main="Gross Revenue vs. Number of Award
Nominations", xlab="Number of Award Nominations", ylab="Gross Revenue in
Dollars", na.rm=TRUE) + stat_smooth(method="loess", method.args =
list(degree=0), span=0.5, se=FALSE, color="red", size=2, na.rm=TRUE)
```




```
ggpairs(df2, columns=c("Gross","Wins","Nominations"))
```

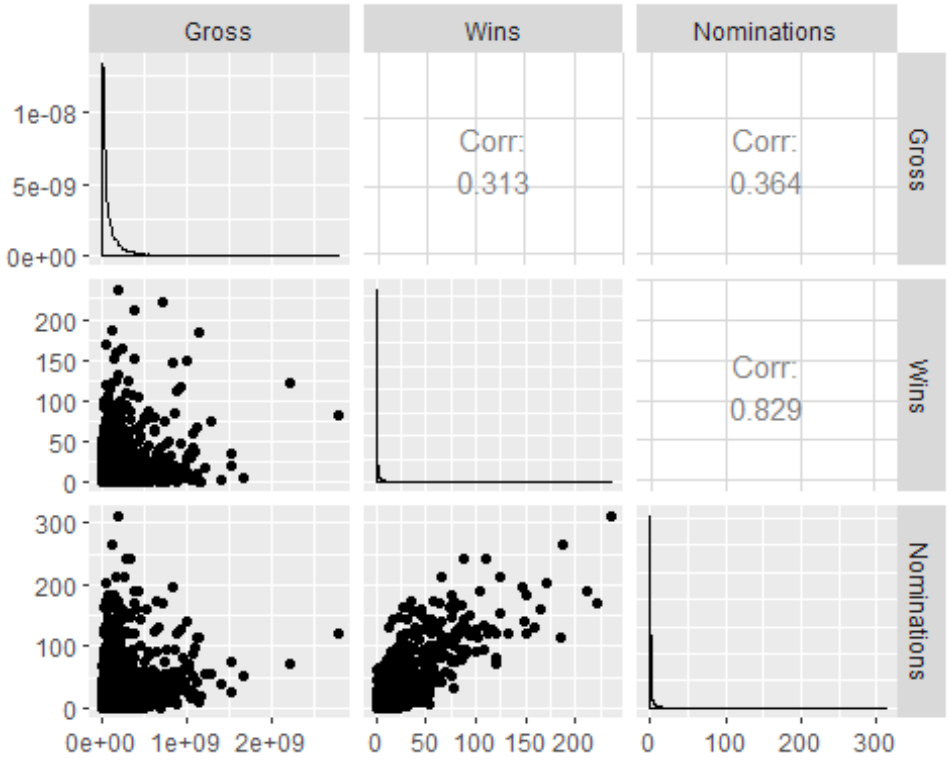
```
## Warning: Removed 35113 rows containing non-finite values (stat_density).
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 35113 rows containing missing values
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 35113 rows containing missing values
```

```
## Warning: Removed 35113 rows containing missing values (geom_point).
```

```
## Warning: Removed 35113 rows containing missing values (geom_point).
```



Q: How does the gross revenue vary by number of awards won and nominations received?

A: According to the plots of gross revenue by award wins and nominations, there is a very weak relationship in general. For awards won, revenue does seem to trend upwards with more wins, but only slightly. For nominations, there is also a very slight increase as the number of nominations increases. According to the GGpairs plot of revenue, wins, and nominations, there is a correlation of 0.313 between revenue and wins and a correlation of 0.364 between revenue and nominations.

7. Movie ratings from IMDb and Rotten Tomatoes

There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by several variables pre-fixed by `tomato`). Read up on such ratings on the web (for example rottentomatoes.com/about and www.imdb.com/help/show_leaf?votestopfaq).

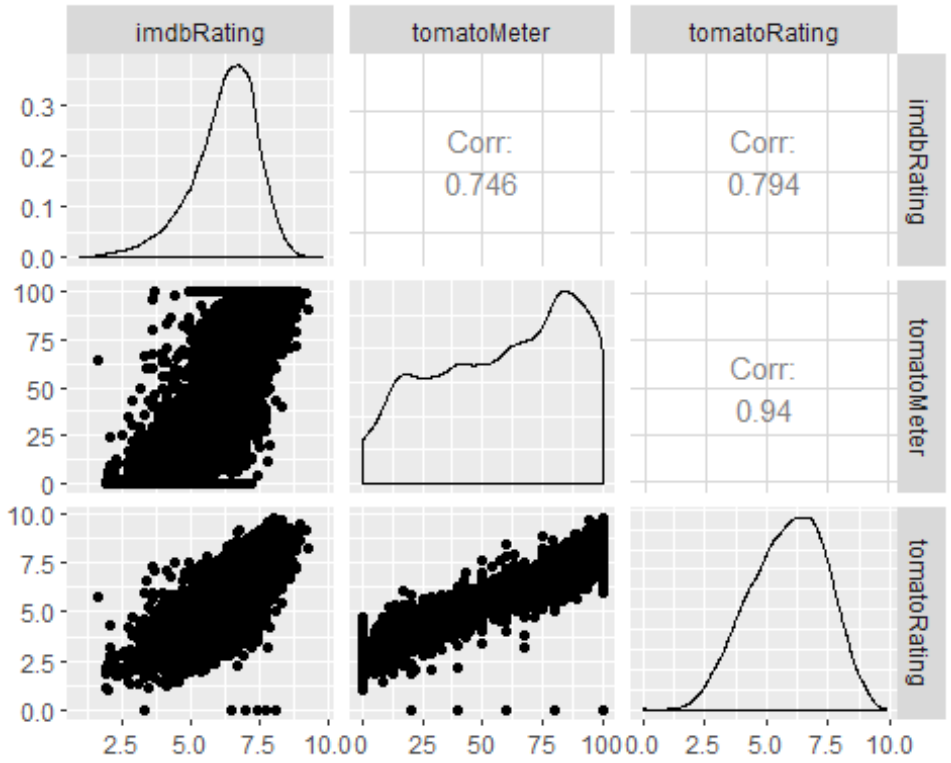
Investigate the pairwise relationships between these different descriptors using graphs.

```
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
ggpairs(df2, columns=c("imdbRating", "tomatoMeter", "tomatoRating"))
```

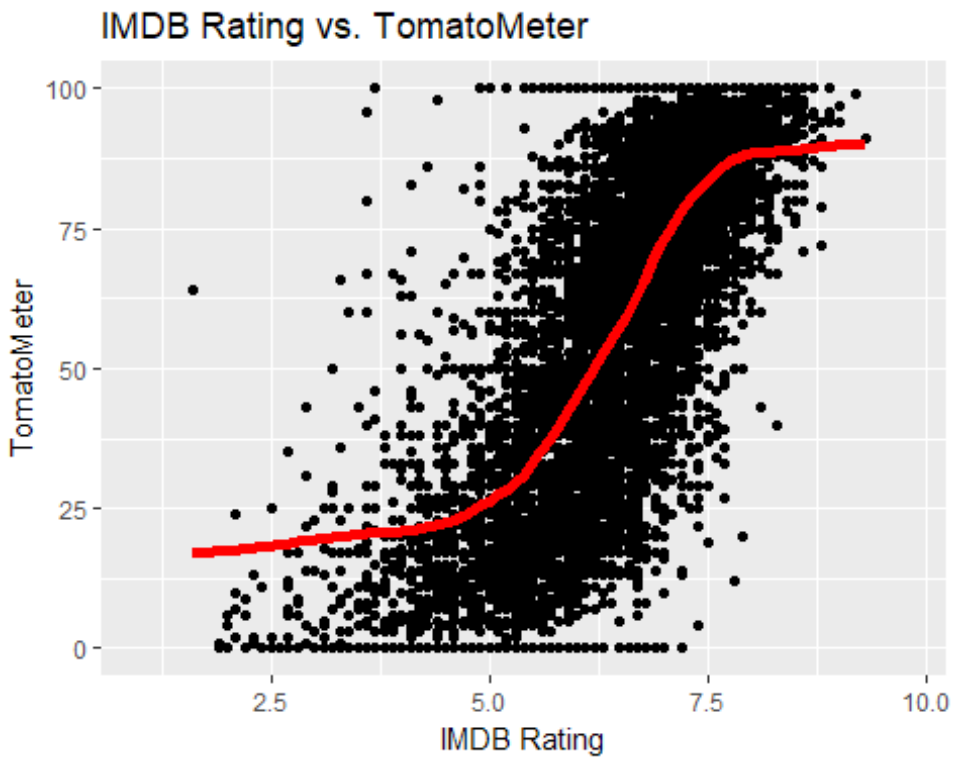
```
## Warning: Removed 1133 rows containing non-finite values (stat_density).
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30248 rows containing missing values
```

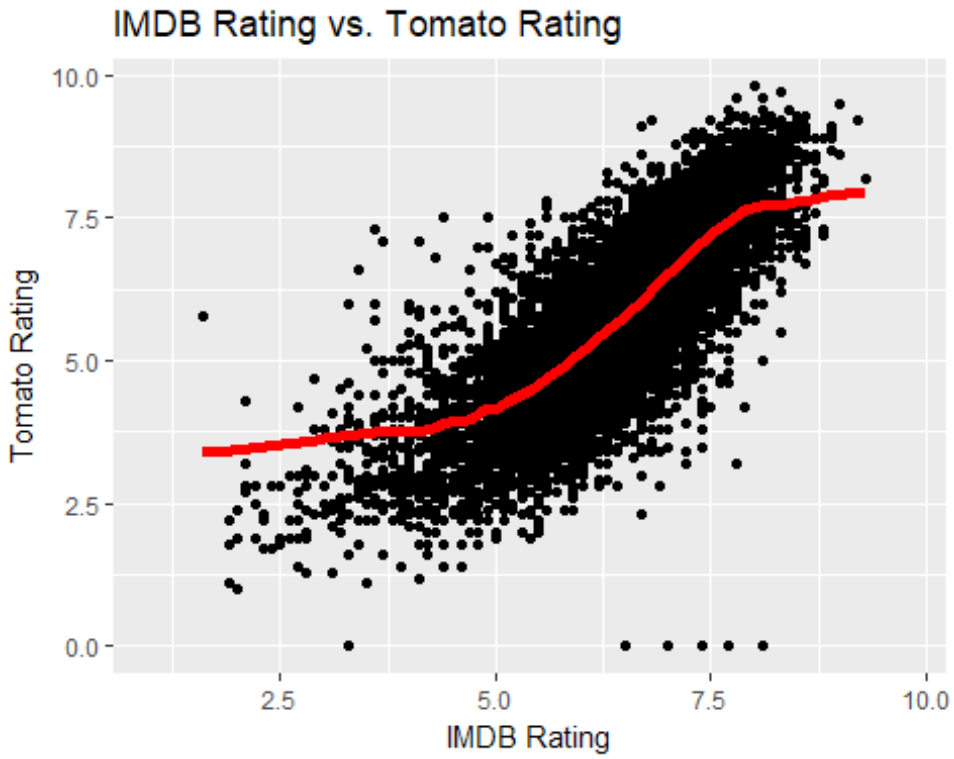
```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30257 rows containing missing values

## Warning: Removed 30248 rows containing missing values (geom_point).

## Warning: Removed 30246 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30255 rows containing missing values

## Warning: Removed 30257 rows containing missing values (geom_point).

## Warning: Removed 30255 rows containing missing values (geom_point).

## Warning: Removed 30255 rows containing non-finite values (stat_density).
```




```
qplot(imdbRating, tomatoMeter, data=df2, main="IMDB Rating vs. TomatoMeter",
xlab="IMDB Rating", ylab="TomatoMeter",na.rm=TRUE) +
stat_smooth(method="loess", method.args=list(degree=0), span=0.2, se=FALSE,
color="red", size=2, na.rm=TRUE)
```




```
qplot(imdbRating, tomatoRating, data=df2, main="IMDB Rating vs. Tomato
Rating", xlab="IMDB Rating", ylab="Tomato Rating",na.rm=TRUE) +
stat_smooth(method="loess", method.args=list(degree=0), span=0.2, se=FALSE,
color="red", size=2, na.rm=TRUE)
```



Q: Comment on the similarities and differences between the user ratings of IMDb and the critics ratings of Rotten Tomatoes.

A: The comparison between the IMDB Rating and with TomatoMeter and Tomato Rating appear generally similar in the way they both have an S-shaped correlation. Both show a strong relationship where the TomatoMeter and Tomato Rating both increase as the IMDB Rating increases, especially when the IMDB Rating is above 5.0. However, they also show an leveling-off of the relationship when the IMDB Rating is around 7.5-8.0, the TomatoMeter is above 80, and the Tomato Rating is above 7.5. This indicates that as the IMDB Rating rises above this level, the correlation weakens. Most likely, this is due to a small percentage of viewers disliking a movie, even when the majority of viewers rate a movie strongly.

The pairwise plot shows the correlation between the IMDB Rating and the TomatoMeter at 0.746 and the correlation between the IMDB Rating and the Tomato Rating at 0.794. The lower correlation for the TomatoMeter is likely due to the inclusion of critics' reviews as well as user reviews. The Tomato Rating, however, shows a stronger correlation due to relying on user reviews alone.

Also, both plots show a number of outliers that have high ratings on one website and low ratings on the other. The plot of IMDB Rating vs. Tomato Rating shows five such movies that are rated around a 7.5 on the IMDB, but apparently have a rating of zero on RottenTomatoes while the TomatoMeter plot also shows a number of films that have either a zero of a 100 rating on RottenTomatoes. There are also some points showing low ratings on the IMDB and high ratings on RottenTomatoes. These outliers are likely to be films that

have a very low number of reviews on one or both sites, resulting in a weak relationship between their ratings.

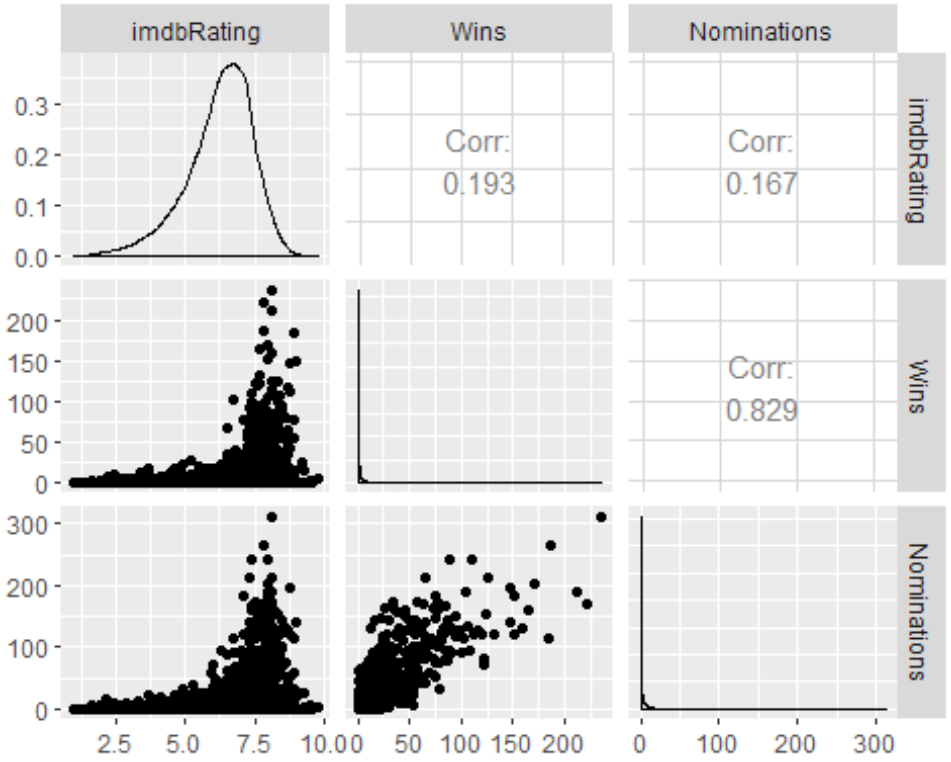
8. Ratings and awards

These ratings typically reflect the general appeal of the movie to the public or gather opinions from a larger body of critics. Whereas awards are given by professional societies that may evaluate a movie on specific attributes, such as artistic performance, screenplay, sound design, etc.

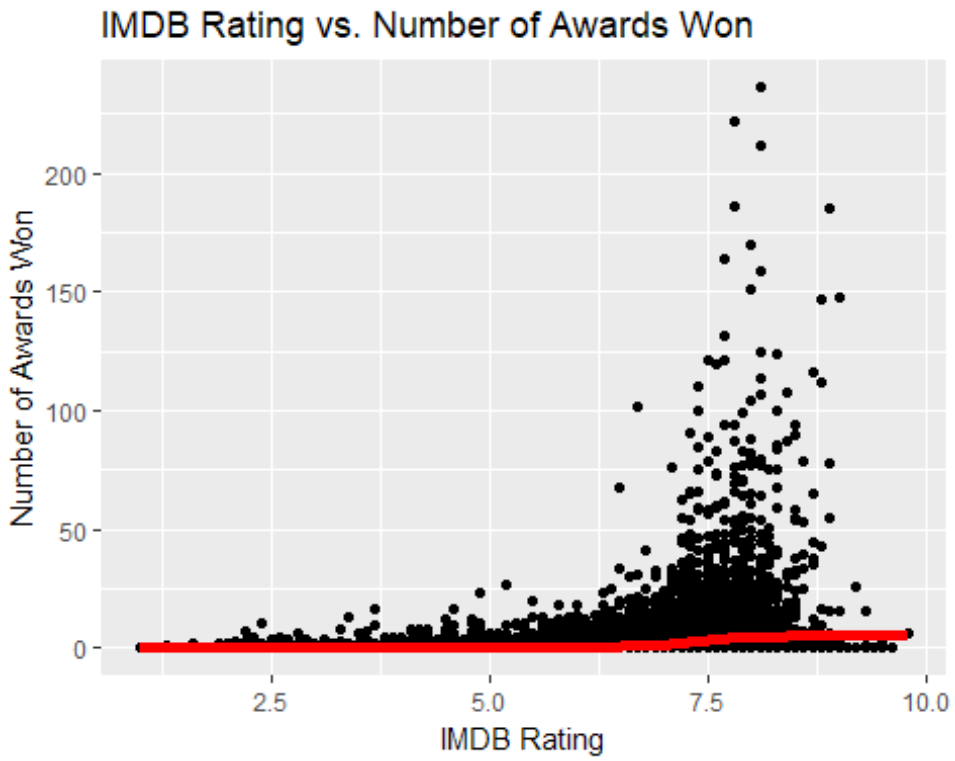
Study the relationship between ratings and awards using graphs (awards here refers to wins and/or nominations).

```
# TODO: Show how ratings and awards are related
ggpairs(df2, columns=c("imdbRating", "Wins", "Nominations"))
```

```
## Warning: Removed 1133 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1133 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1133 rows containing missing values

## Warning: Removed 1133 rows containing missing values (geom_point).

## Warning: Removed 1133 rows containing missing values (geom_point).
```

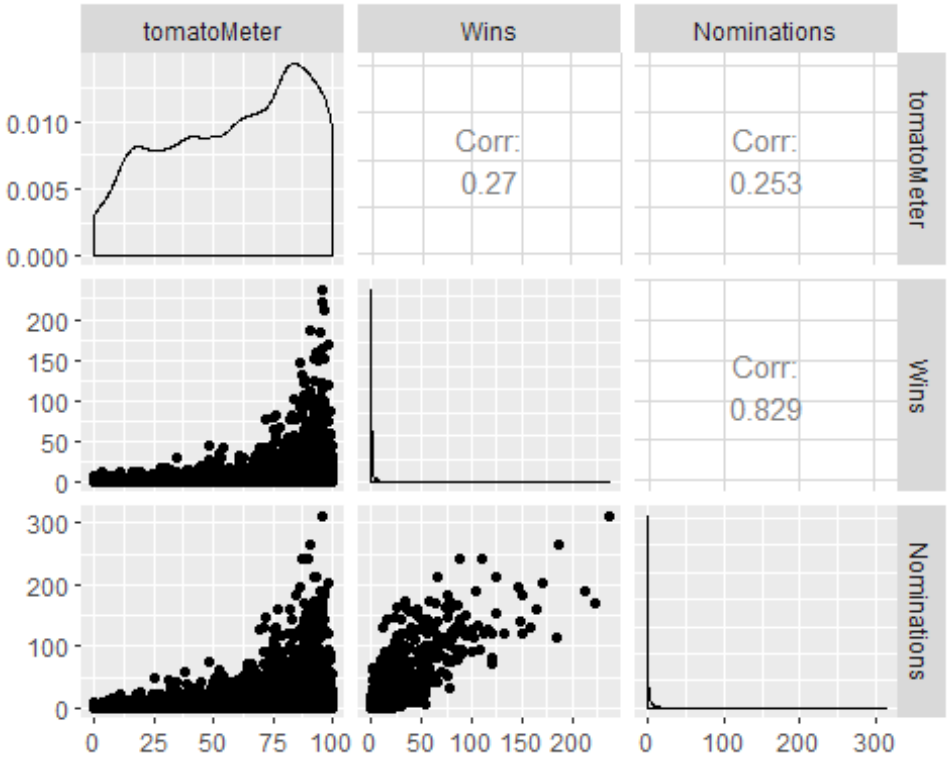


```
qplot(imdbRating, Wins, data=df2, main="IMDB Rating vs. Number of Awards
Won", xlab="IMDB Rating", ylab="Number of Awards Won", na.rm=TRUE) +
stat_smooth(method="loess", method.args = list(degree=0), span=0.2, se=FALSE,
color="red", size=2, na.rm=TRUE)
```

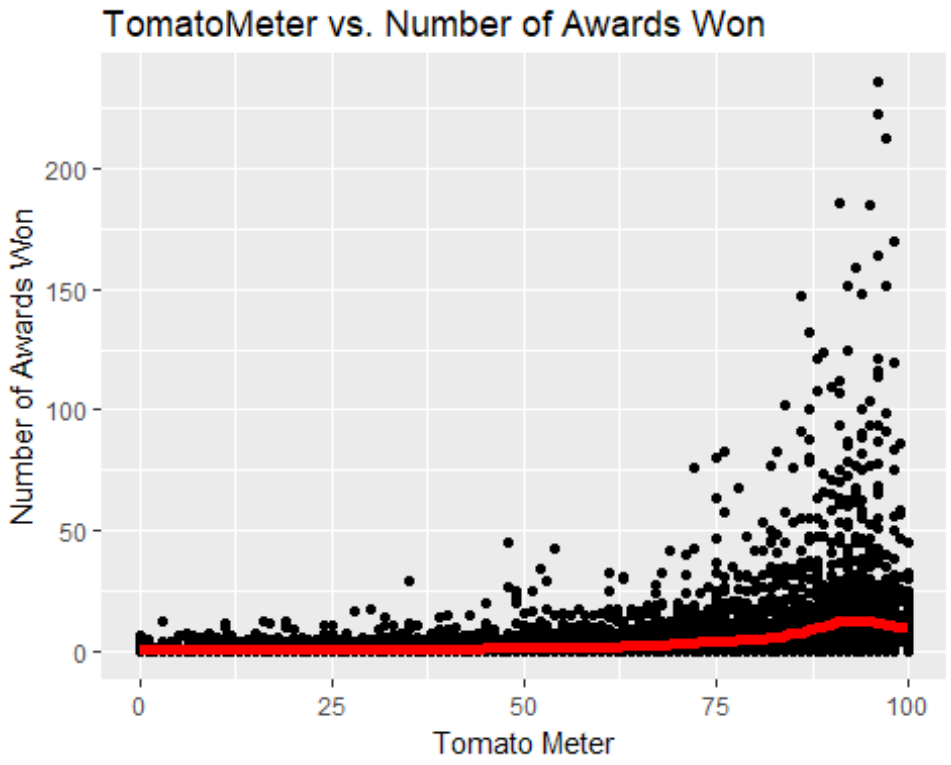



```
ggpairs(df2, columns=c("tomatoMeter", "Wins", "Nominations"))
```

```
## Warning: Removed 30246 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30246 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30246 rows containing missing values

## Warning: Removed 30246 rows containing missing values (geom_point).

## Warning: Removed 30246 rows containing missing values (geom_point).
```




```
qplot(tomatoMeter, Wins, data=df2, main="TomatoMeter vs. Number of Awards
Won", xlab="Tomato Meter", ylab="Number of Awards Won", na.rm=TRUE) +
stat_smooth(method="loess", method.args = list(degree=0), span=0.2, se=FALSE,
color="red", size=2, na.rm=TRUE)
```



Q: How good are these ratings in terms of predicting the success of a movie in winning awards or nominations? Is there a high correlation between two variables?

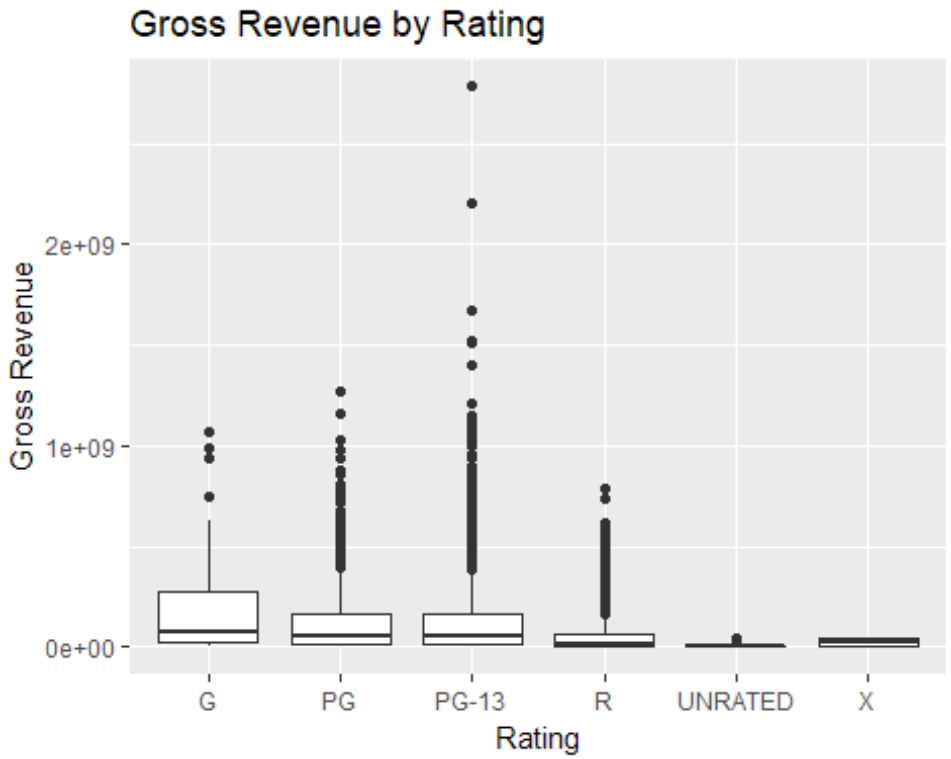
A: Both the IMDB Ratings and the Tomato Meter show a very low correlation with the number of awards and nominations a movie receives. The IMDB Rating has the lowest correlations at 0.193 for wins and 0.167 for nominations. The Tomato Meter has slightly higher correlations with 0.27 for wins and 0.253 for nominations. The slightly higher correlation can be seen in the smoothed line curve of TomatoMeter vs. Number of Awards Won, where there is a slight rise in the curve as the TomatoMeter rises above 75. At this point, there is a larger number of movies with more awards and nominations.
Overall, the correlations are too low to say that ratings can predict a movie's success at winning awards or nominations. However, there is a slightly higher likelihood that the Tomato Meter can predict awards and nominations for movies with a high rating. A likely explanation is that the Tomato Meter relies on critics' reviews and many of those critics are also members of the organizations that grant the awards and nominations.

9. Expected insights

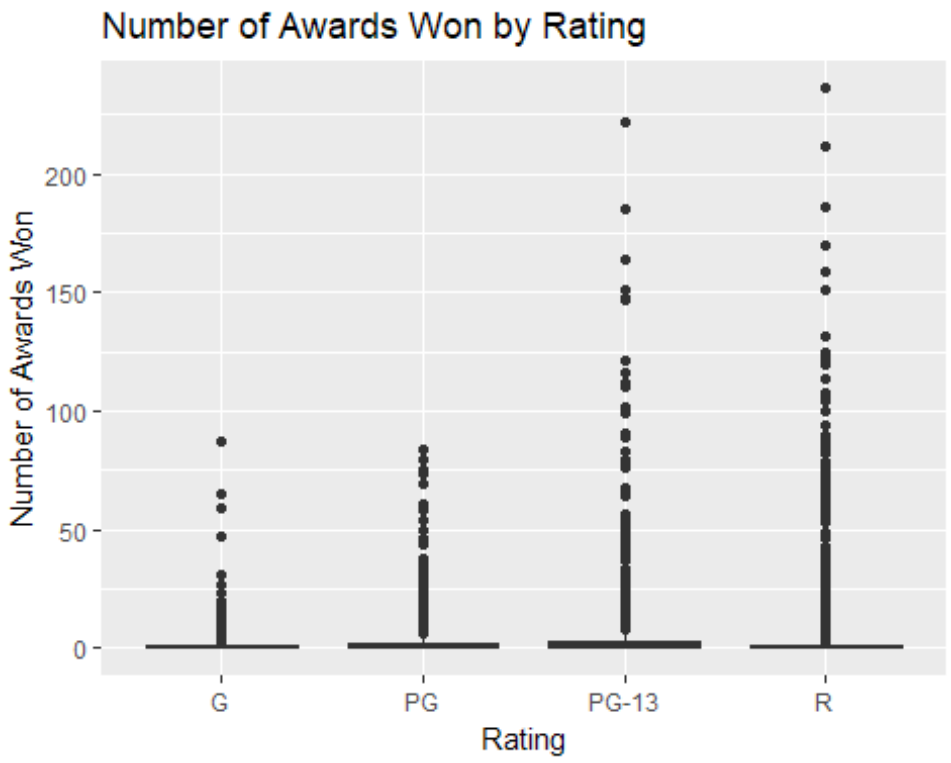
Come up with two new insights (backed up by data and graphs) that is expected. Here â??newâ?? means insights that are not an immediate consequence of one of the above tasks. You may use any of the columns already explored above or a different one in the dataset, such as `Title`, `Actors`, etc.

```
# TODO: Find and illustrate two expected insights
ggplot(df2, aes(Rated, Gross)) + geom_boxplot(na.rm=TRUE) +
```

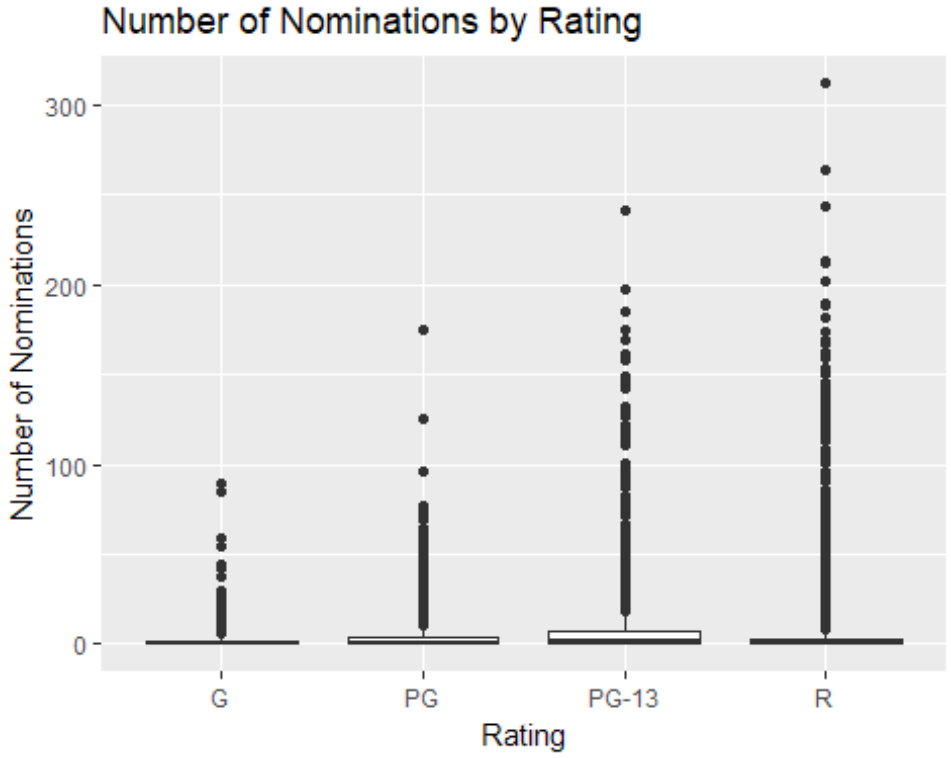
```
scale_x_discrete(limits=c("G","PG","PG-13","R","UNRATED","X")) +
xlab("Rating") + ylab("Gross Revenue") + ggtitle("Gross Revenue by Rating")
```




```
movies_rated <- subset(df2, Rated=="G" | Rated=="PG" | Rated=="PG-13" |
Rated=="R")
ggplot(movies_rated, aes(Rated, Wins)) + geom_boxplot() + xlab("Rating") +
ylab("Number of Awards Won") + ggtitle("Number of Awards Won by Rating")
```




```
ggplot(movies_rated, aes(Rated, Nominations)) + geom_boxplot() +
xlab("Rating") + ylab("Number of Nominations") + ggtitle("Number of
Nominations by Rating")
```



Q: Expected insight #1.

A: I explored the relationship between a movie's rating and its gross revenue. Not surprisingly, I found the highest average revenue and the largest Interquartile Range belonged to G-rated movies. This isn't surprising because of the continuous popularity of children's movies. The next highest average revenues belonged to PG and PG-13 rated films, with R-rated films having a significantly lower average along with a lower range of outliers. It makes sense that as a film's rating is more restrictive that less people will likely see the film and its revenue will be lower. Continuing with that trend, unrated and X-rated films had the lowest gross revenues.

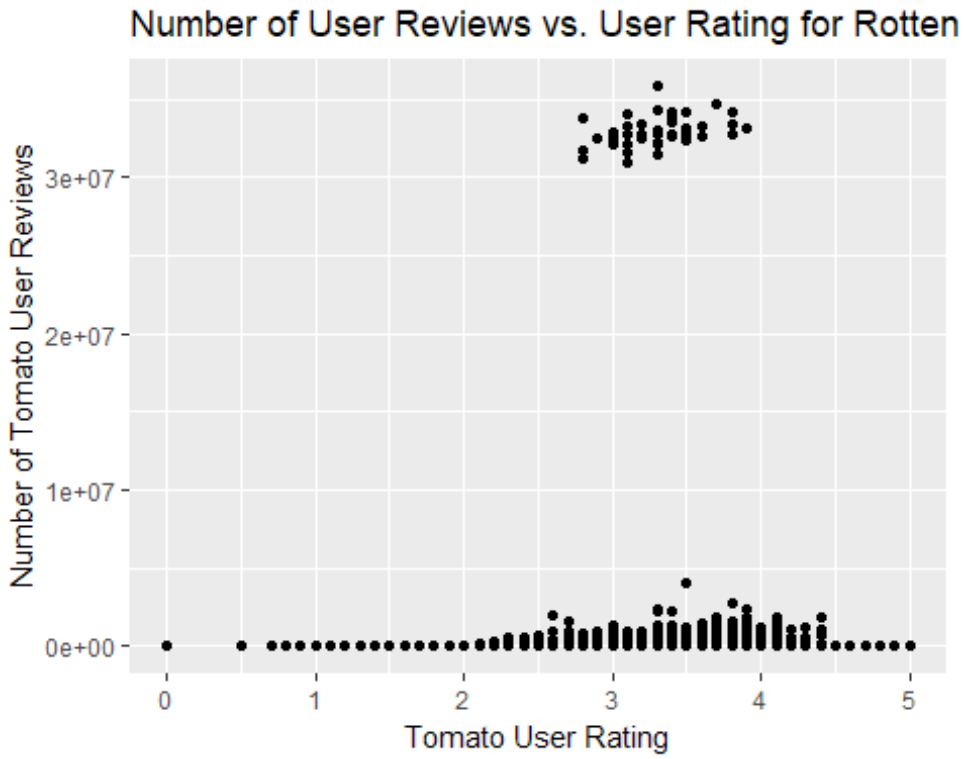
Q: Expected insight #2.

A: The next expected relationship I found was that higher rated movies, especially PG-13 and R, had higher rates of nominations and awards. While the mean and IQR's for each rating are very low, it is clear that PG-13 has the largest IQR of the ratings. Also, PG-13 and R have a much larger number of outliers, especially those above 100 awards and nominations. This pattern is to be expected because most critically-acclaimed films tend to be more mature and therefore have higher ratings. Films that are aimed at children or families are less likely to be seen as films that are worthy of Oscars or other awards.

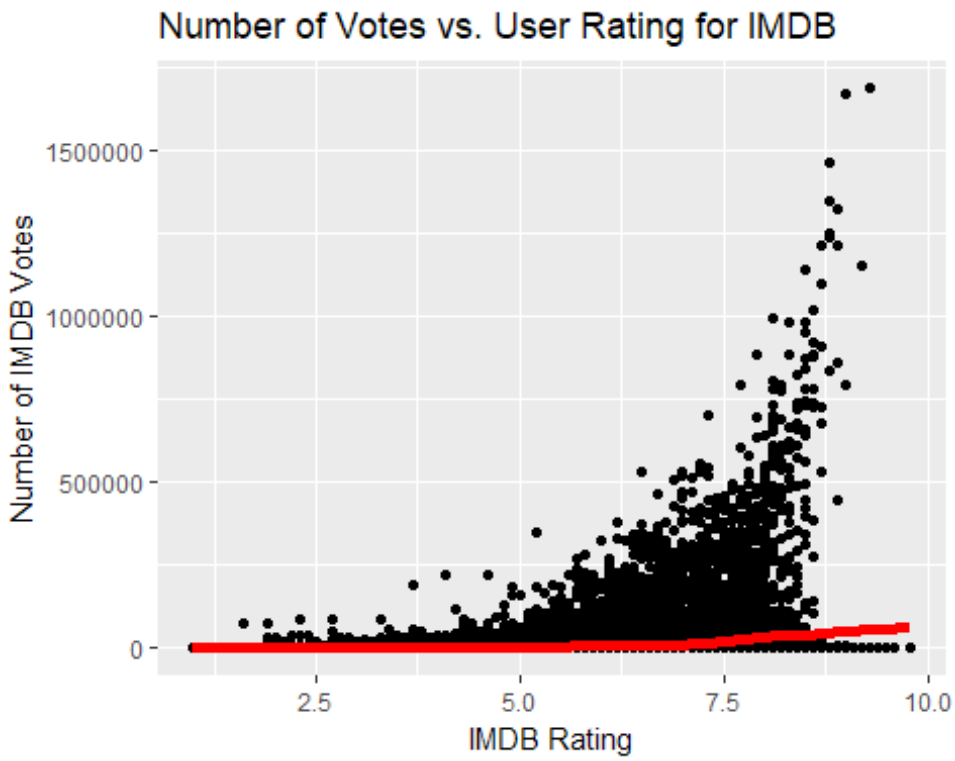
10. Unexpected insight

Come up with one new insight (backed up by data and graphs) that is unexpected at first glance and do your best to motivate it. Same instructions apply as the previous task.

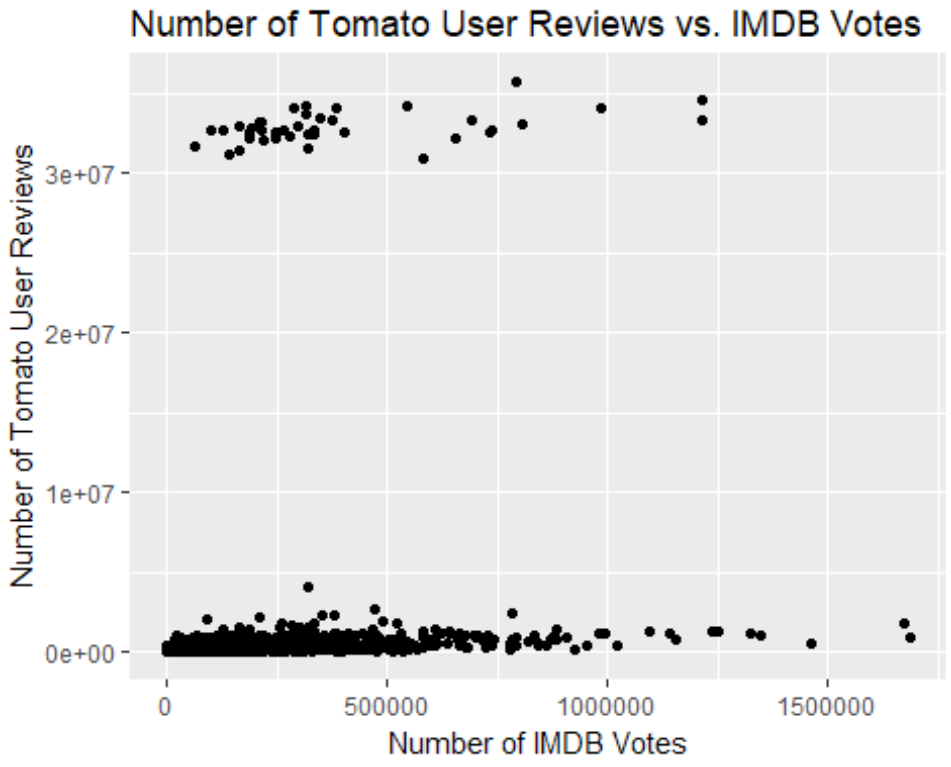
```
# TODO: Find and illustrate one unexpected insight
qplot(tomatoUserRating, tomatoUserReviews, data=df2, na.rm=TRUE, main =
"Number of User Reviews vs. User Rating for Rotten Tomatoes", xlab="Tomato
User Rating", ylab="Number of Tomato User Reviews")
```




```
qplot(imdbRating, imdbVotes, data=df2, na.rm=TRUE, xlab="IMDB Rating",
ylab="Number of IMDB Votes", main="Number of Votes vs. User Rating for IMDB")
+ stat_smooth(method="loess", method.args = list(degree=0), span=0.2,
se=FALSE, color="red", size=2, na.rm=TRUE)
```




```
qplot(imdbVotes, tomatoUserReviews, data=df2, na.rm=TRUE, main="Number of
Tomato User Reviews vs. IMDB Votes", xlab="Number of IMDB Votes",
ylab="Number of Tomato User Reviews")
```



Q: Unexpected insight.

A: One unexpected pattern I found was in the relationship between the number of user reviews and ratings on Rotten Tomatoes, as well as the different numbers of reviews and votes between Rotten Tomatoes and the IMDB. In the first plot, I compared the Tomato User Rating to the number of Tomato User Reviews. While I expected that more popular movies would have more reviews, I was surprised to see a cluster of movies with 30 million or more reviews that is completely separated from the vast majority of films. When examining the data, there are about 45 films with more than 30 million reviews while the rest have about 4 million or less. Most of these films are very mainstream, with many being large budget blockbusters, and most were made in the 1990s or later.

The second plot is a similar plot for the IMDB, plotting the IMDB Rating by the number of IMDB votes. On this plot, there is a similar trend for more votes being cast for certain highly-rated movies. However, the movies with the most votes don't form a cluster separate from the majority as they do in the Rotten Tomatos plot. I was also surprised to find that the number of votes on the IMDB was much lower than Rotten Tomatos, with the films with the most votes receiving about 1.5 million, as opposed to the cluster of 45 films with more than 30 million reviews on Rotten Tomatos.

In the third plot, I compared the number of IMDB votes with the number of Tomato User Reviews. In this plot, the cluster of films with more than 30 million Tomato User Reviews clearly stands out as it does in the first plot. However, I was surprised to see that some of these films have a relatively low number of votes on the IMDB, with a fair amount having less than 500,000. In addition, there are some outliers for IMDB votes with over 1 million, yet have a relatively low number of Tomato User Reviews.

The explanation for why Rotten Tomatoes and the IMDB would have a difference in the number of votes or reviews is unclear. However, I suspect it may have to do with the sites having different kinds of users. Since the 45 films with more than 30 million Tomato Reviews are mostly blockbusters or very well known films, I suspect that Rotten Tomatoes may have a larger user base, but one that is more focused on mainstream films. The IMDB, meanwhile, has a number of films with a relatively large number of votes, but very few Tomato Reviews, indicating that IMDB users may be more focused on certain films that are less mainstream, or may be popular in certain niche markets.