Code

Project 1: Explore and Prepare Data - Submitted by GT User: sanne31 (sanne31@gatech.edu (mailto:sanne31@gatech.edu))

CSE6242 - Data and Visual Analytics - Summer 2018

Due: Sunday, June 17, 2018 at 11:59 PM UTC-12:00 on T-Square

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` (https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6242/project/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 (http://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 (http://rmarkdown.rstudio.com) 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*.

Hide

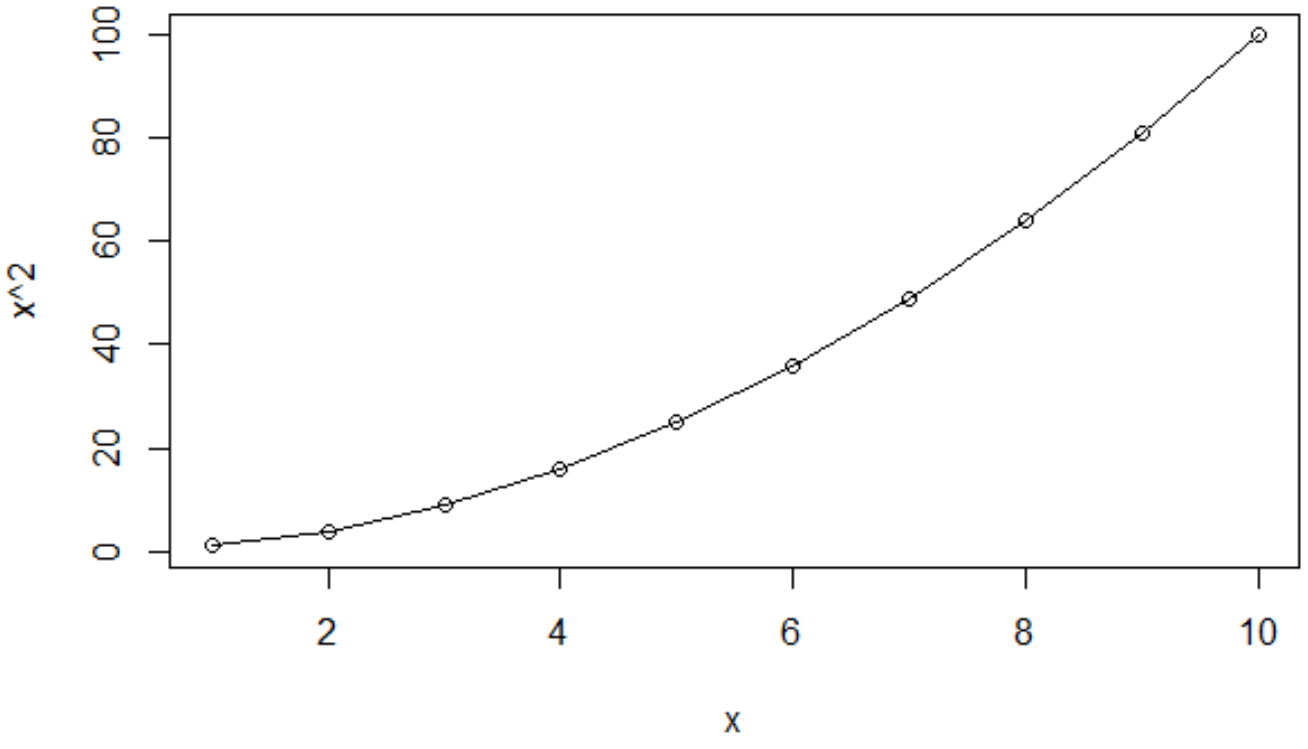
```
x = 1:10
print(x^2)
```

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

Plots appear inline too:

Hide

```
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.

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` (https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6242/project/movies_merged) file and it is in the current working directory. Now load it into memory:

Hide

```
load('movies_merged')
cat("Dataset has", dim(movies_merged)[1], "rows and", dim(movies_me
rged)[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:

Hide

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

```
Column names:
```

Hide

```
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.

Hide

```
install.packages("NLP",dependencies = TRUE)
```

```
Installing package into <U+393C><U+3E31>C:/Users/Anne/Documents/R/w
in-library/3.5<U+393C><U+3E32>
(as <U+393C><U+3E31>lib<U+393C><U+3E32> is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/NLP_0.
1-11.zip'
Content type 'application/zip' length 375672 bytes (366 KB)
downloaded 366 KB
```

```
package □NLP□ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\Anne\AppData\Local\Temp\RtmpMV3cg2\downloaded_packages
```

Hide

```
install.packages('reshape2')
```

```
Installing package into <U+393C><U+3E31>C:/Users/Anne/Documents/R/w
in-library/3.5<U+393C><U+3E32>
(as <U+393C><U+3E31>lib<U+393C><U+3E32> is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/reshap
e2_1.4.3.zip'
Content type 'application/zip' length 626669 bytes (611 KB)
downloaded 611 KB
```

```
package □reshape2□ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\Anne\AppData\Local\Temp\RtmpMV3cg2\downloaded_packages
```

Hide

```
install.packages("tm")
```

```
Installing package into <U+393C><U+3E31>C:/Users/Anne/Documents/R/w
in-library/3.5<U+393C><U+3E32>
(as <U+393C><U+3E31>lib<U+393C><U+3E32> is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/tm_0.7-
3.zip'
Content type 'application/zip' length 1360310 bytes (1.3 MB)
downloaded 1.3 MB
```

```
package □tm□ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\Anne\AppData\Local\Temp\RtmpMV3cg2\downloaded_packages
```

Hide

```
install.packages("ggplot2")
```

```
Installing package into <U+393C><U+3E31>C:/Users/Anne/Documents/R/w
in-library/3.5<U+393C><U+3E32>
(as <U+393C><U+3E31>lib<U+393C><U+3E32> is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/ggplot
2_2.2.1.zip'
Content type 'application/zip' length 3157706 bytes (3.0 MB)
downloaded 3.0 MB
```

```
package □ggplot2□ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\Anne\AppData\Local\Temp\RtmpMV3cg2\downloaded_packages
```

Hide

```
install.packages("GGally")
```

```
Installing package into <U+393C><U+3E31>C:/Users/Anne/Documents/R/w
in-library/3.5<U+393C><U+3E32>
(as <U+393C><U+3E31>lib<U+393C><U+3E32> is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/GGally
_1.4.0.zip'
Content type 'application/zip' length 1433954 bytes (1.4 MB)
downloaded 1.4 MB
```

```
package □GGally□ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\Anne\AppData\Local\Temp\RtmpMV3cg2\downloaded_packages
```

Hide

```
install.packages("tidyr")
```

```
Installing package into <U+393C><U+3E31>C:/Users/Anne/Documents/R/w
in-library/3.5<U+393C><U+3E32>
(as <U+393C><U+3E31>lib<U+393C><U+3E32> is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/tidyr_
0.8.1.zip'
Content type 'application/zip' length 943749 bytes (921 KB)
downloaded 921 KB
```

```
package □tidyr□ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\Anne\AppData\Local\Temp\RtmpMV3cg2\downloaded_packages
```

Hide

```
install.packages("SnowballC")
```

```
Installing package into <U+393C><U+3E31>C:/Users/Anne/Documents/R/w
in-library/3.5<U+393C><U+3E32>
(as <U+393C><U+3E31>lib<U+393C><U+3E32> is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/Snowba
llC_0.5.1.zip'
Content type 'application/zip' length 3082565 bytes (2.9 MB)
downloaded 2.9 MB
```

```
package □SnowballC□ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\Anne\AppData\Local\Temp\RtmpMV3cg2\downloaded_packages
```

Hide

```
install.packages("lubridate")
```

```
Installing package into <U+393C><U+3E31>C:/Users/Anne/Documents/R/w
in-library/3.5<U+393C><U+3E32>
(as <U+393C><U+3E31>lib<U+393C><U+3E32> is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/lubrid
ate_1.7.4.zip'
Content type 'application/zip' length 1568374 bytes (1.5 MB)
downloaded 1.5 MB
```

```
package □lubridate□ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
    C:\Users\Anne\AppData\Local\Temp\RtmpMV3cg2\downloaded_packages
```

Hide

```
library(ggplot2)
library(GGally)
library(tidyr)
library(NLP)
```

```
Attaching package: <U+393C><U+3E31>NLP<U+393C><U+3E32>

The following object is masked from <U+393C><U+3E31>package:ggplot2
<U+393C><U+3E32>:

    annotate
```

Hide

```
library(tm)
library(plyr)
library(reshape2)
```

```
Attaching package: <U+393C><U+3E31>reshape2<U+393C><U+3E32>

The following object is masked from <U+393C><U+3E31>package:tidyr<U
+393C><U+3E32>:

    smiths
```

Hide

```
library(SnowballC)
library(lubridate)
```

```
Attaching package: <U+393C><U+3E31>lubridate<U+393C><U+3E32>

The following object is masked from <U+393C><U+3E31>package:plyr<U+
393C><U+3E32>:

    here

The following object is masked from <U+393C><U+3E31>package:base<U+
393C><U+3E32>:

    date
```

Hide

```
library(stringr)
```

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:

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.

Hide

```
# 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.

Hide

```
# TODO: Replace df$Runtime with a numeric column containing the run
time in minutes
#NoNarun_time <- df2$Runtime
#NoNarun_time[is.character(NoNarun_time) & NoNarun_time == "N/A"] <-
 "0 min"
raw_run_time <- df2$Runtime
h_run_time <- as.numeric(gsub(".*?(\\d+) [Hh]r?.*", "\\1", raw_run_
time))
```

```
NAs introduced by coercion
```

Hide

```
h_run_time[is.na(h_run_time)] <- 0
m_run_time <- as.numeric(gsub(".*?(\\d+) [Mm]in?.*", "\\1", raw_run
_time))
```

```
NAs introduced by coercion
```

Hide

```
m_run_time[is.na(m_run_time)] <- 0
run_time <- (h_run_time * 60) + m_run_time
df2$Runtime <- run_time
typeof(df2$Runtime)
```

```
[1] "double"
```

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.

Hide

```
# TODO: Investigate the distribution of Runtime values and how it v
aries by Year and Budget
summary(df2$Year)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1888    1961    1989    1981    2001    2018
```

Hide

```
summary(df2$Budget)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      N
A's
     1100   5000000  18000000  31661108  40000000 425000000     354
42
```

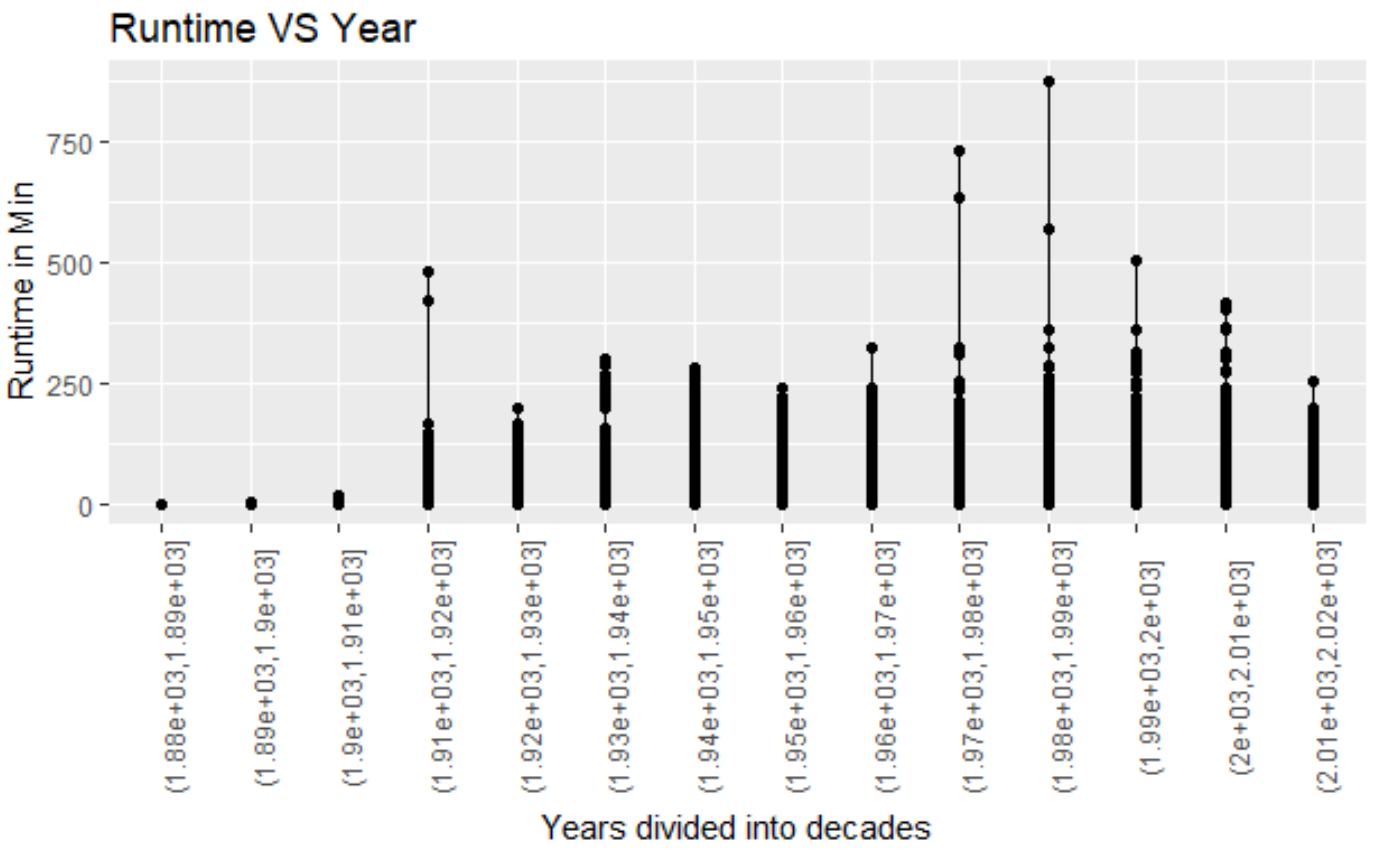
Hide

```
summary(df2$Runtime)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   69.00   90.00   80.26  100.00  873.00
```

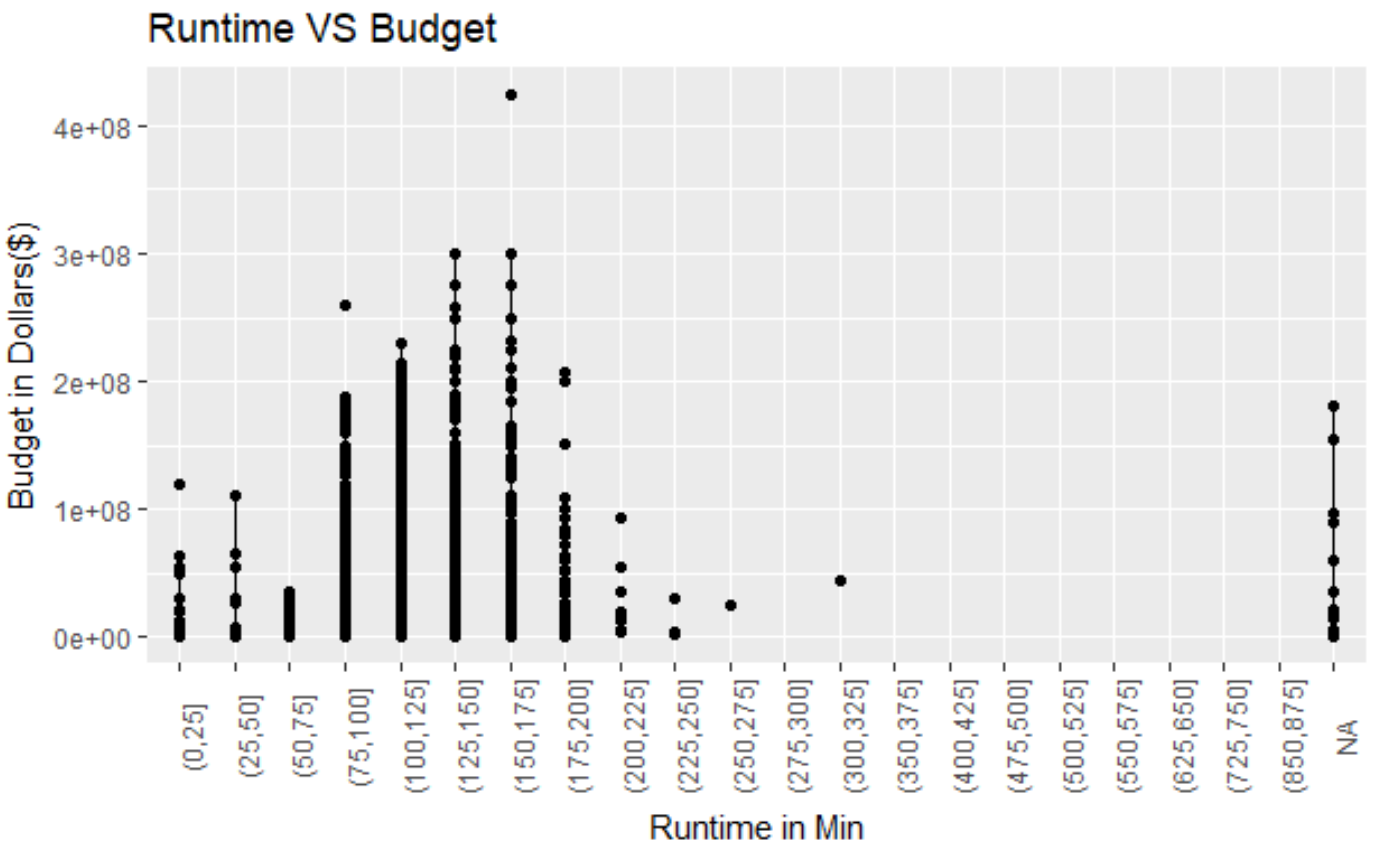
Hide

```
df2$Period = cut(x = df2$Year, breaks = c(1880, seq(1890,2020, by=1
0)))
ggplot(df2, aes(x = Period, y = Runtime)) + geom_point() + geom_lin
e() +#geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90)) + ggtitle("Runtime
 VS Year") +
  xlab("Years divided into decades") + ylab("Runtime in Min")
```



Hide

```
df2$Div_runtime = cut(x = df2$Runtime, breaks = c(0, seq(25,875, by
=25)))
ggplot(df2, aes(x = Div_runtime, y = Budget)) + geom_point() + geom
_line() +
  theme(axis.text.x = element_text(angle = 90)) + ggtitle("Runtime
 VS Budget") +xlab("Runtime in Min") + ylab("Budget in Dollars($)")
```



Feel free to insert additional code chunks as necessary.

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

A: It can be observed from both of the graphs that runtime varies greatly with both Year and the Budget.

Taking a look at the plot between Runtime and Year, we can see that it is not a straightforward relationship. It is almost as if the runtime is negligible until 1910 where the runtime increases rather quickly. Thereafter, the runtime appears to be constant during the years until 1970 where it increases again till 1980 and then appears to be dropping gradually indicating that over the course of the next couple years, it can be assumed that the runtime length of films might decrease.

At a quick glance at the relationship between runtime and budget, we can almost guess that it is linear which it is. As we know, short films and too lengthy films will not really have much of an entertaining aspect and fails to draw a larger audience which in turn affects the budget rightly so. It appears that the best runtime to expect a high budget will be from around 125 min to 160 min.

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).

Hide

```
# TODO: Replace Genre with a collection of binary columns
summary(df2$Genre)
```

```
   Length     Class      Mode
    40000 character character
```

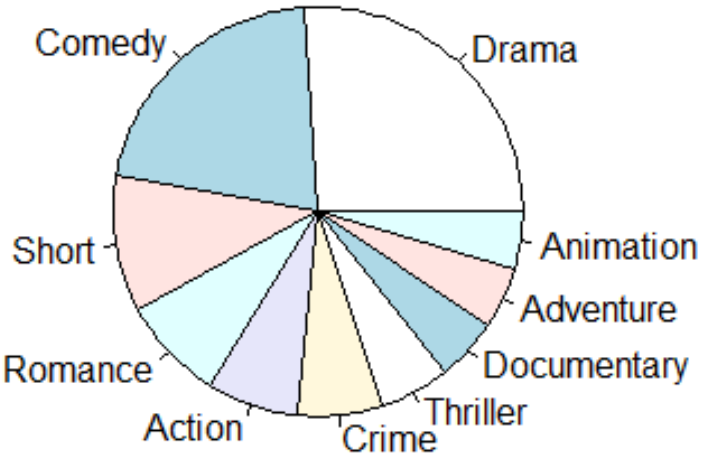
Hide

```
View(df2)
genre_dictionary <- unique(unlist(strsplit(df2$Genre, ", ")))
genre = data.frame(matrix(nrow = nrow(df2), ncol = length(genre_dic
tionary)))
colnames(genre) = genre_dictionary
genre[] = 0
for (x in genre_dictionary)
{
  genre[grep(x, df2[, "Genre"]), x] = 1
  #here we compare each and every genre value to the original datas
et and the dictionary,
  #searching for x in df2 dataframe$genre, will put 1 if it encount
ers any of the genres in genre data frame or will put a zero.
  #total number of genres is 29, so there will be 29 extra rows.
}
new_genre = cbind(df2, genre)
#final step is to attach the set of binary columns to the original
 dataframe
```

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

Hide

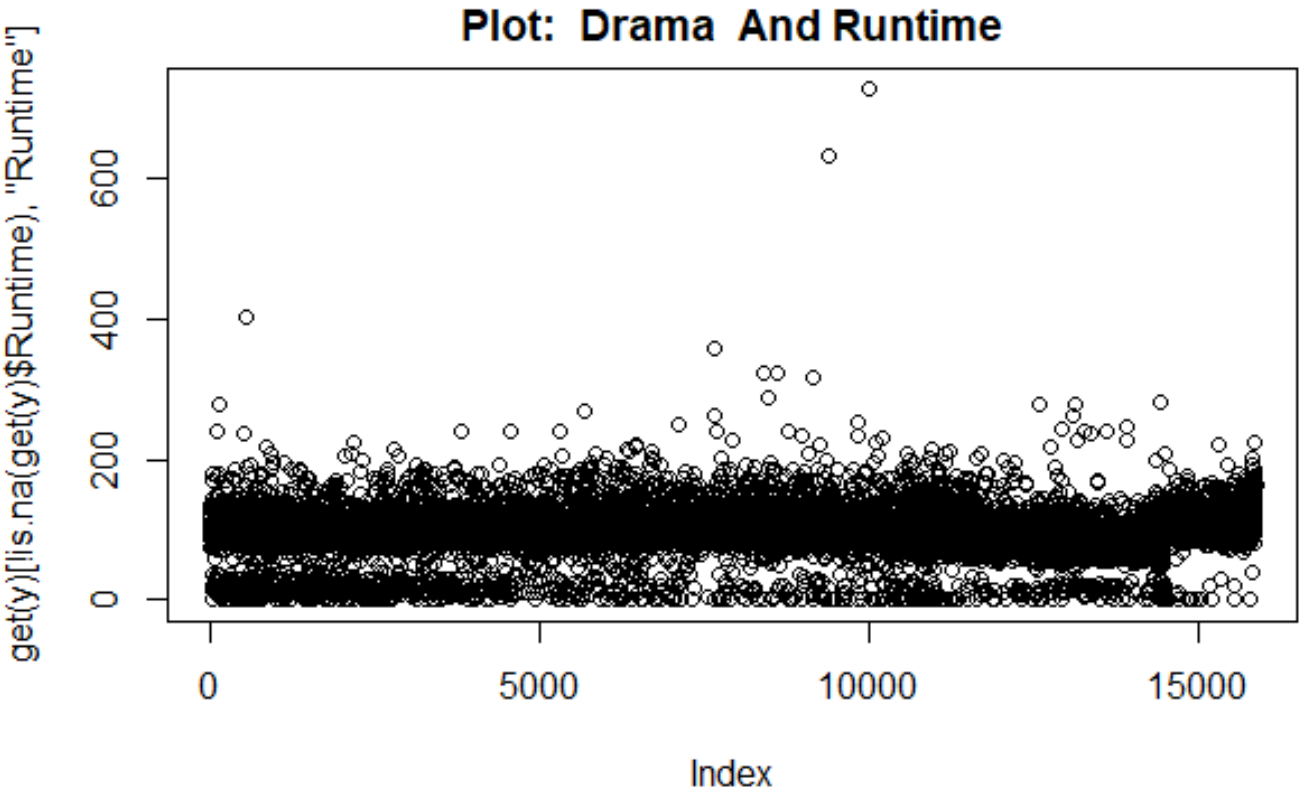
```
# TODO: Select movies from top 10 most common genres and plot their
 relative proportions
metrics = colSums(genre)
sorted = sort(metrics, decreasing = TRUE)
#relative proportions = total count / number of rows
#taking top 10 genres
rel_prop = (sorted[1:10]) / nrow(df2)
#rel_prop <- data.frame(rel_prop)
pie(rel_prop)
```

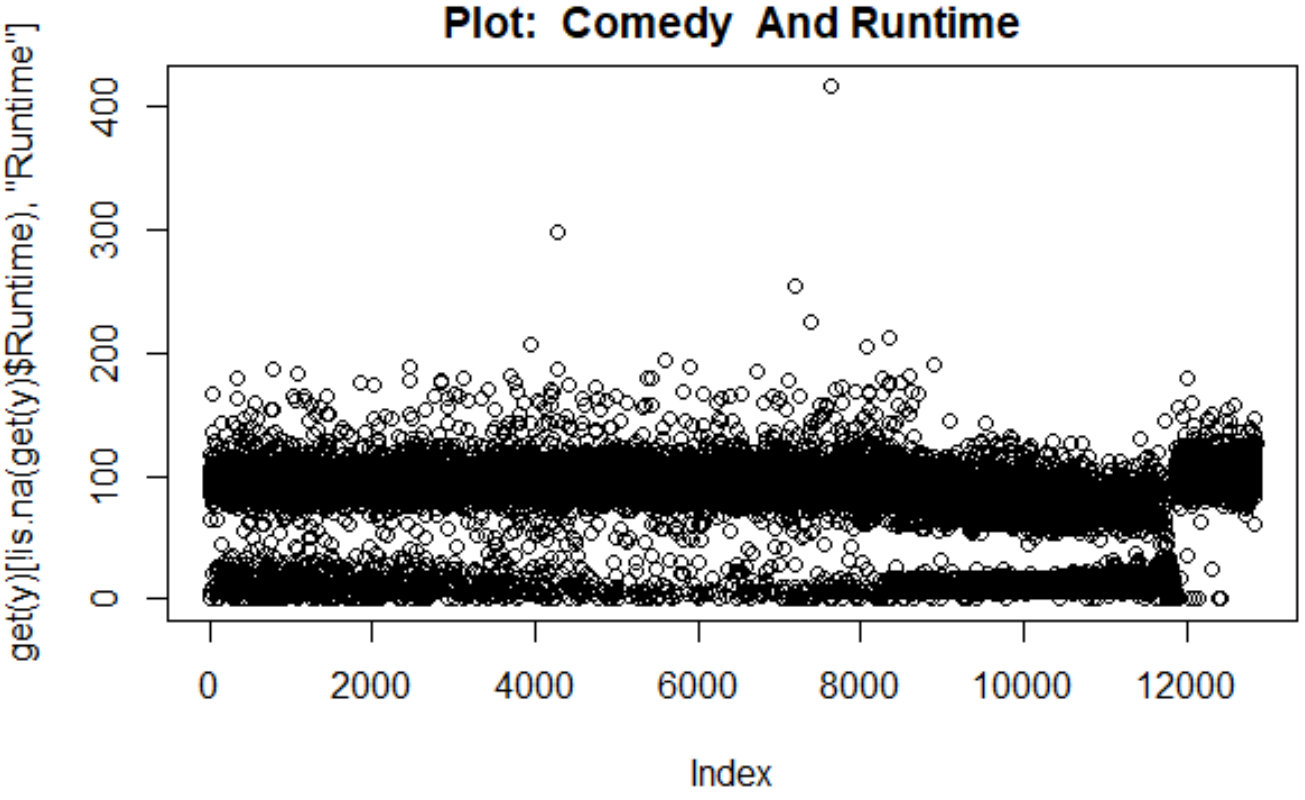


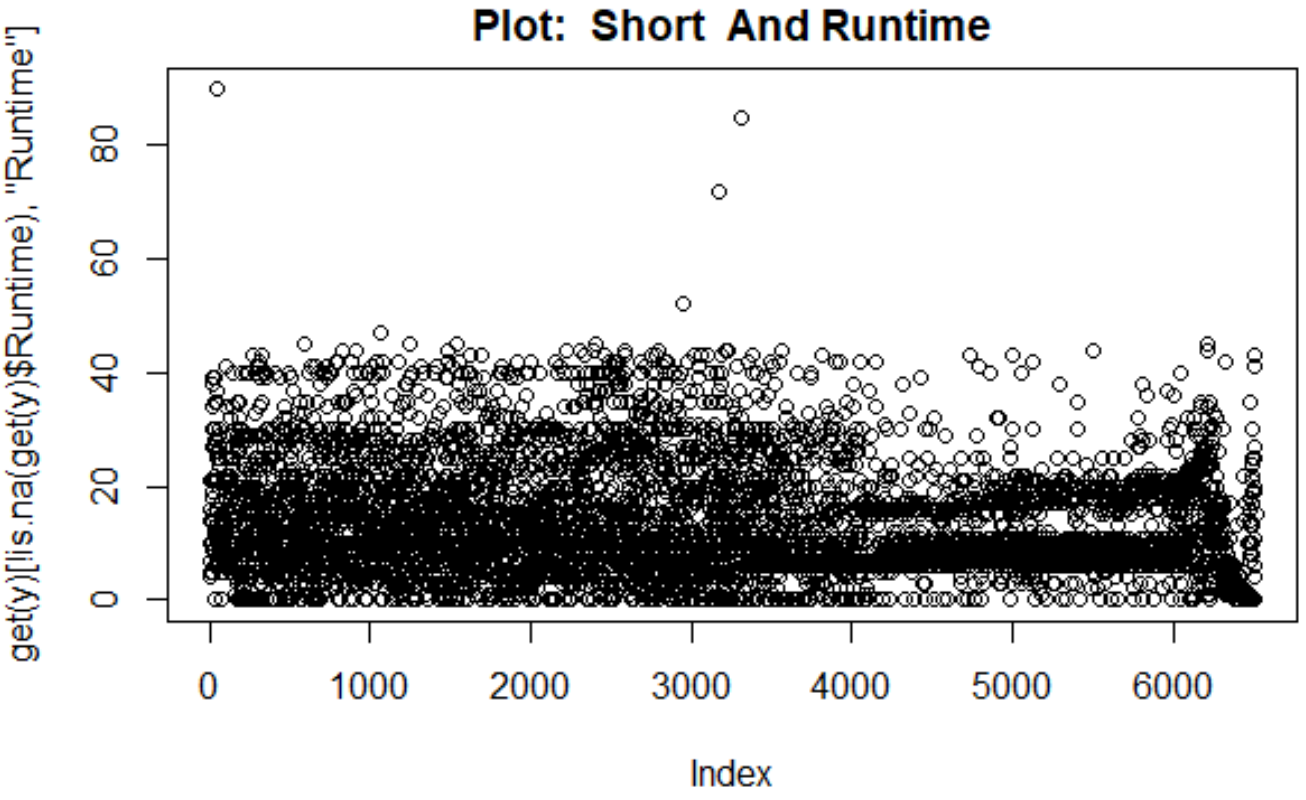
Examine how the distribution of `Runtime` changes across genres for the top 10 most common genres.

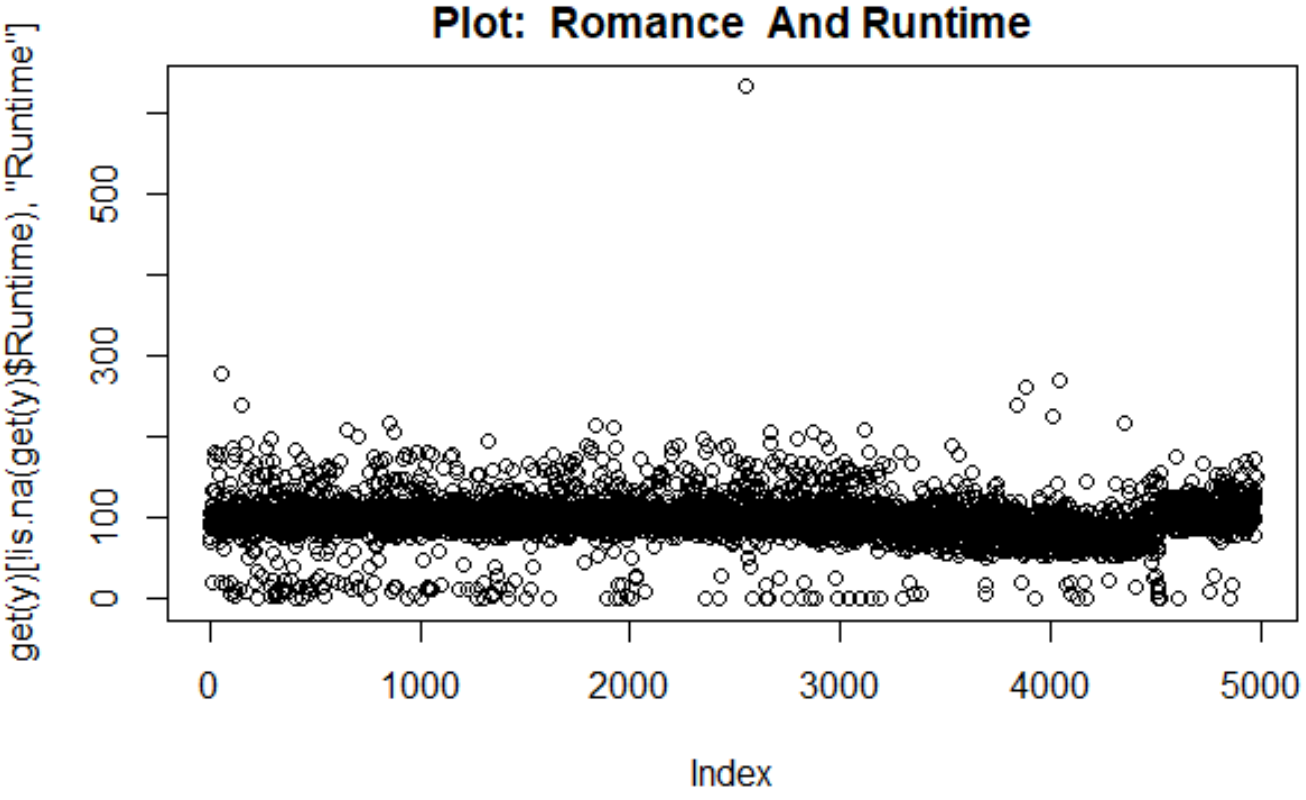
Hide

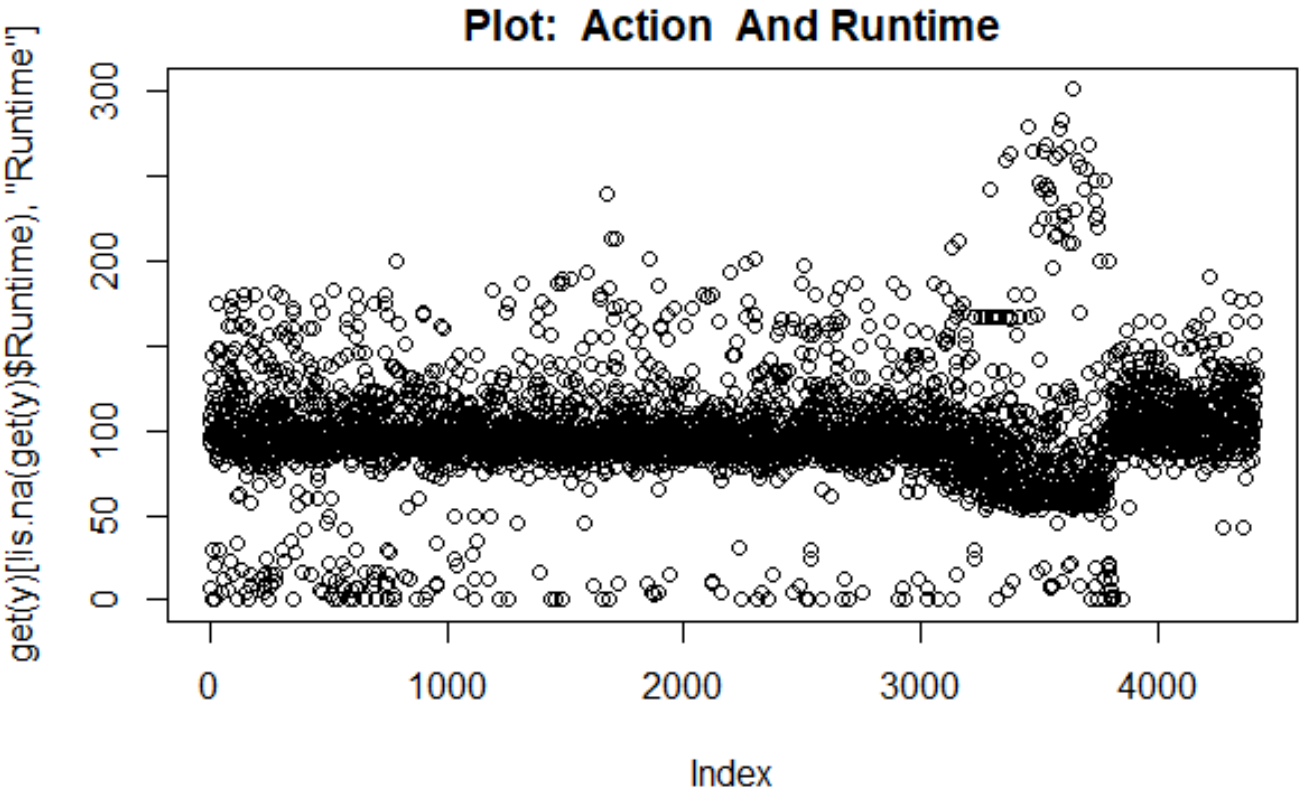
```
# TODO: Plot Runtime distribution for top 10 most common genres
for (y in names(sorted[1:10]))
{
  assign(y, new_genre[new_genre[, y]==1,])

}
#assinging a dataframe consisting of each specific genre to each dr
ama for easier plotting
for (y in names(sorted[1:10]))
{
  plot(get(y)[!is.na(get(y)$Runtime), "Runtime"], main = paste("Plo
t: ", y, " And Runtime"))

}
```

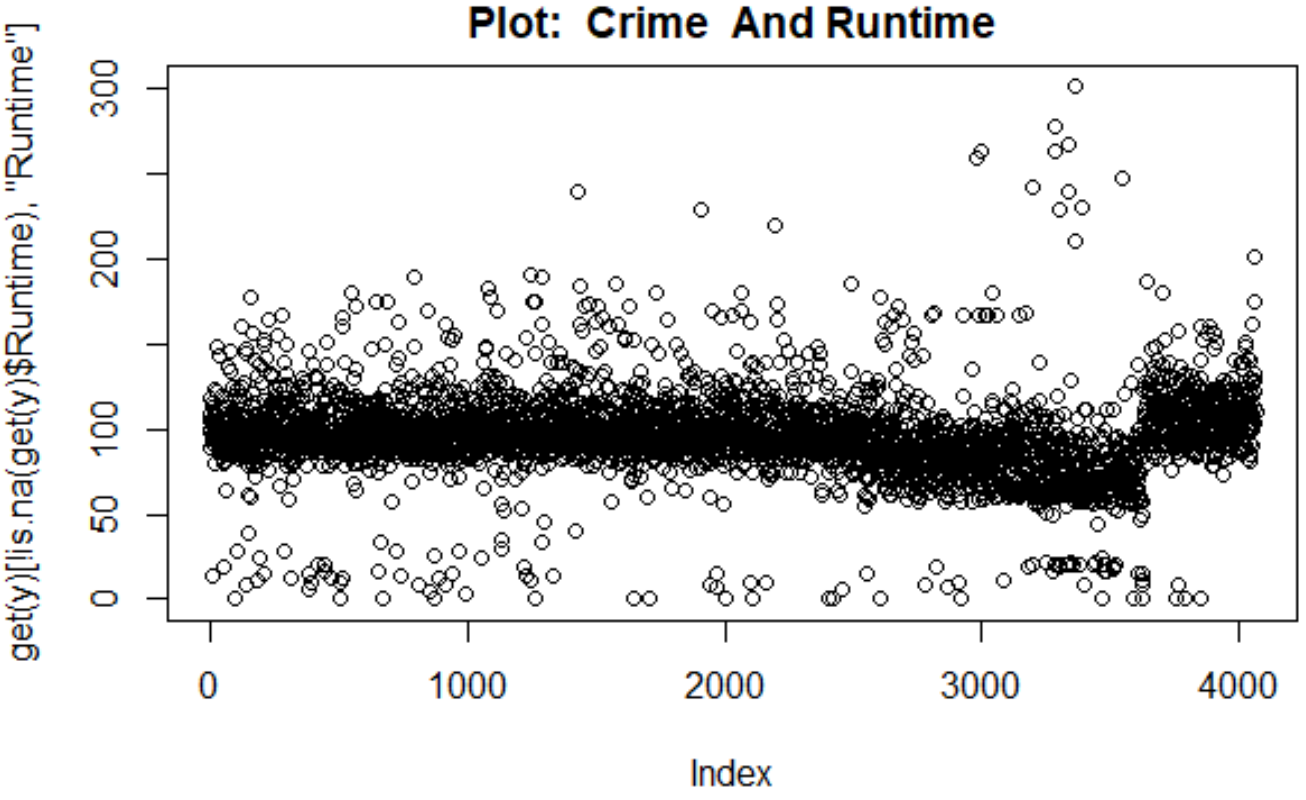

Plot: Drama And Runtime
get(y)[!is.na(get(y)$Runtime), "Runtime"]
0
200
400
600
0
5000
10000
15000
Index


Plot: Comedy And Runtime
get(y)[!is.na(get(y)$Runtime), "Runtime"]
0
100
200
300
400
0
2000
4000
6000
8000
10000
12000
Index

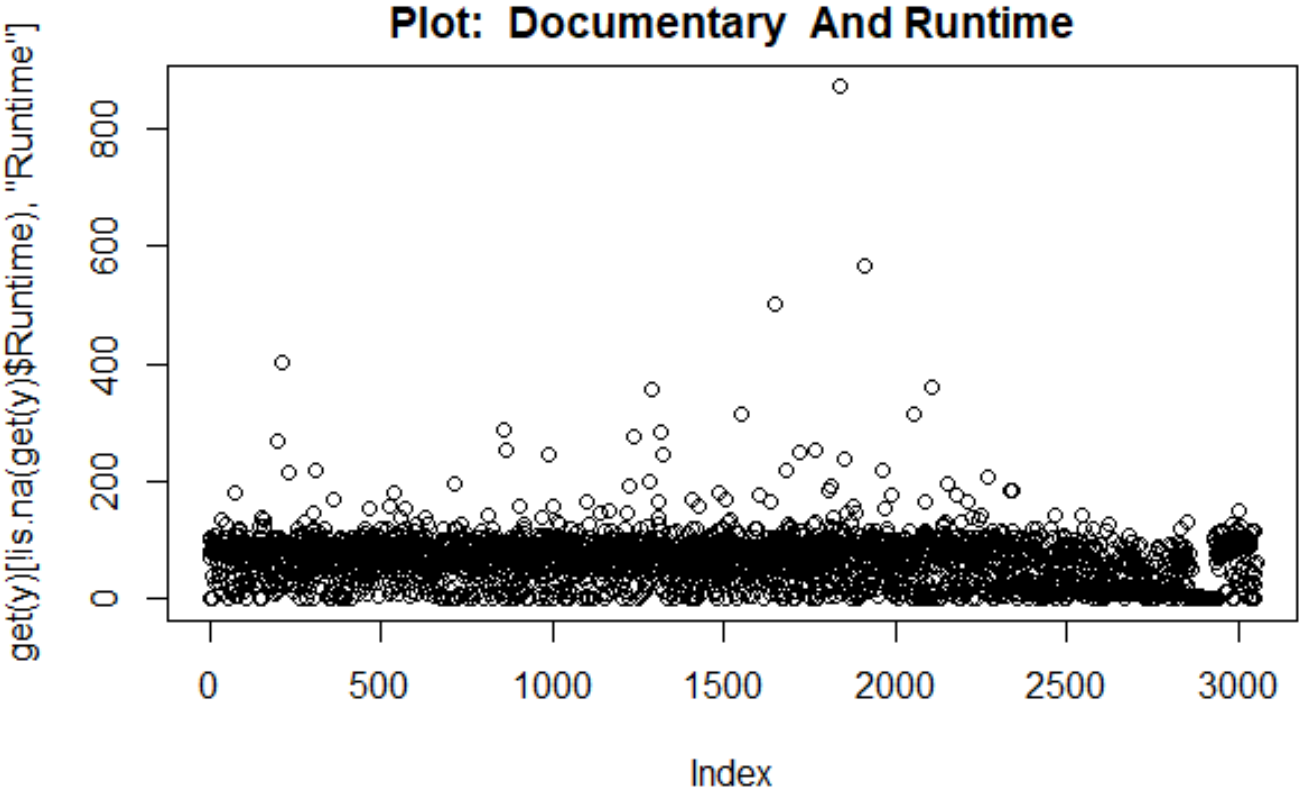

Plot: Short And Runtime
get(y)[!is.na(get(y)$Runtime), "Runtime"]
0
20
40
60
80
0
1000
2000
3000
4000
5000
6000
Index

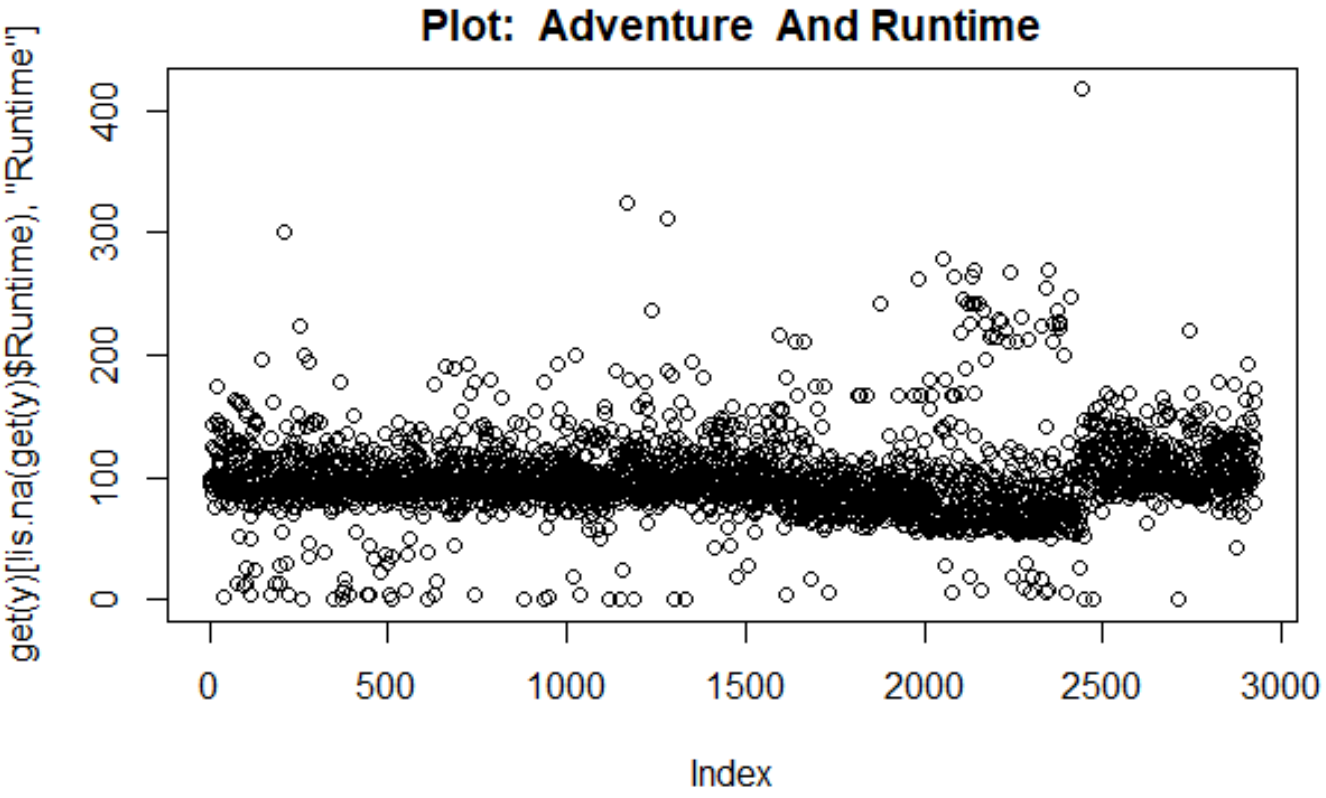

Plot: Romance And Runtime
get(y)[!is.na(get(y)$Runtime), "Runtime"]
0
100
300
500
0
1000
2000
3000
4000
5000
Index

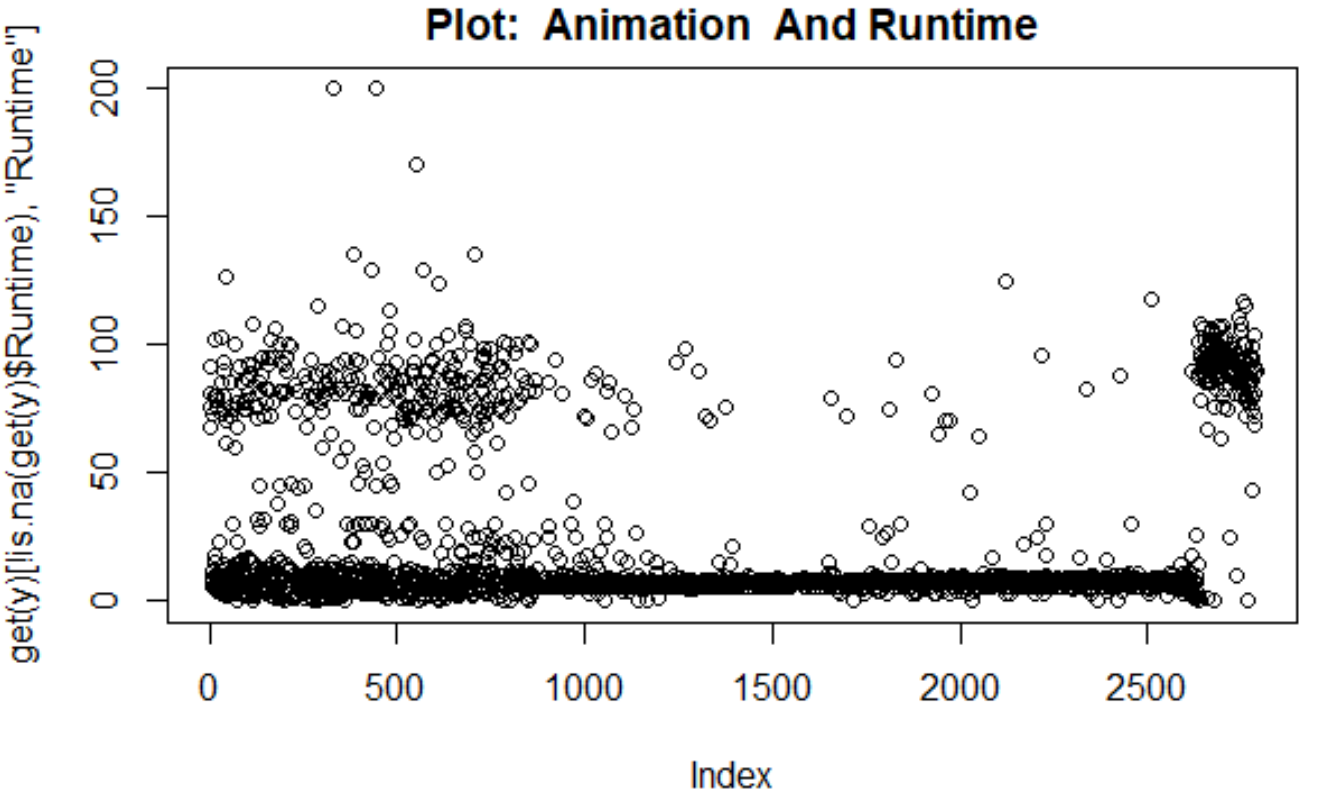

Plot: Action And Runtime
get(y)[!is.na(get(y)$Runtime), "Runtime"]
0
50
100
200
300
0
1000
2000
3000
4000
Index


Plot: Crime And Runtime
get(y)[!is.na(get(y)$Runtime), "Runtime"]
0
50
100
200
300
0
1000
2000
3000
4000
Index


Plot: Thriller And Runtime
get(y)[!is.na(get(y)$Runtime), "Runtime"]
0
50
100
150
200
250
0
500
1000
1500
2000
2500
3000
3500
Index


Plot: Documentary And Runtime
get(y)[!is.na(get(y)$Runtime), "Runtime"]
0
200
400
600
800
0
500
1000
1500
2000
2500
3000
Index





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

A: I used a basic histogram plot to demonstrate the relationship between the runtimes and the genres. From all the above plots, we can see that the genre Short has the lowest runtime of them all and interestingly enough, the Documentary genre has the highest of all runtimes.

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.

Hide

```
# TODO: Remove rows with Year/Date/Released mismatch
time_df <- data.frame(new_genre$Title, new_genre$Year, new_genre$Re
leased, new_genre$Date, new_genre$Gross)
df3 <- new_genre
df3$Released = as.Date(df3$Released, "%Y-%m-%d")
df3$Year_of_Release = as.numeric(format(df3$Released, "%Y"))
bad_data <- df3[(df3$Year == df3$Year_of_Release) | ((df3$Year + 1)
 == df3$Year_of_Release),]
before = length(df3$Gross[!is.na(df3$Gross)])
after = length(bad_data$Gross[!is.na(bad_data$Gross)])
percentage = ((before - after)/after) * 100
```

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

A: 1. I identified that the Released Column was actually a string and converted into a date format. 2. Since majority of the Date column seems to be N/A values, I decided to obtain the mismatched rows based on the Release Date and more specifically the year in the release date. Subset only the year and assigned it to a new variable within the same dataset. 3. Then I removed all the entries from the set where the Year of Release doesn't equal to the Year and also where the Year was 1 year greater than the Year of Release. 4. This new dataset had 38169 rows in it. 5. Number of rows remaining with Gross not equal to NA/(N/A) 6. 3.07 % of rows with the Gross not equal to NA have been removed.

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.

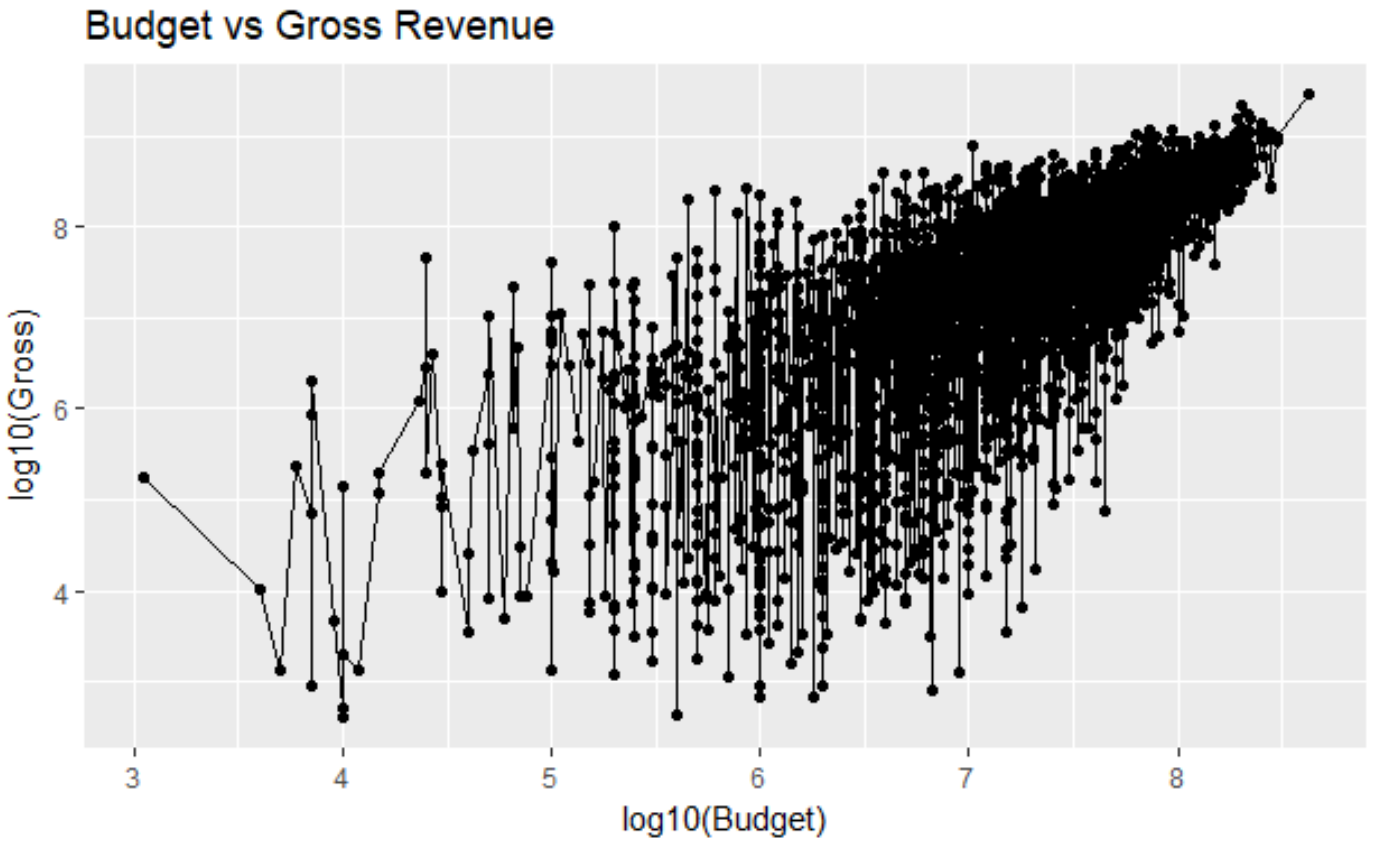
Hide

```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime
 or Genre
gross_revenue = subset(new_genre, !is.na(new_genre$Gross))
gross_revenue <- gross_revenue[gross_revenue$Gross != 0, ]
summary(new_genre$Runtime)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   69.00   90.00   80.26  100.00  873.00
```

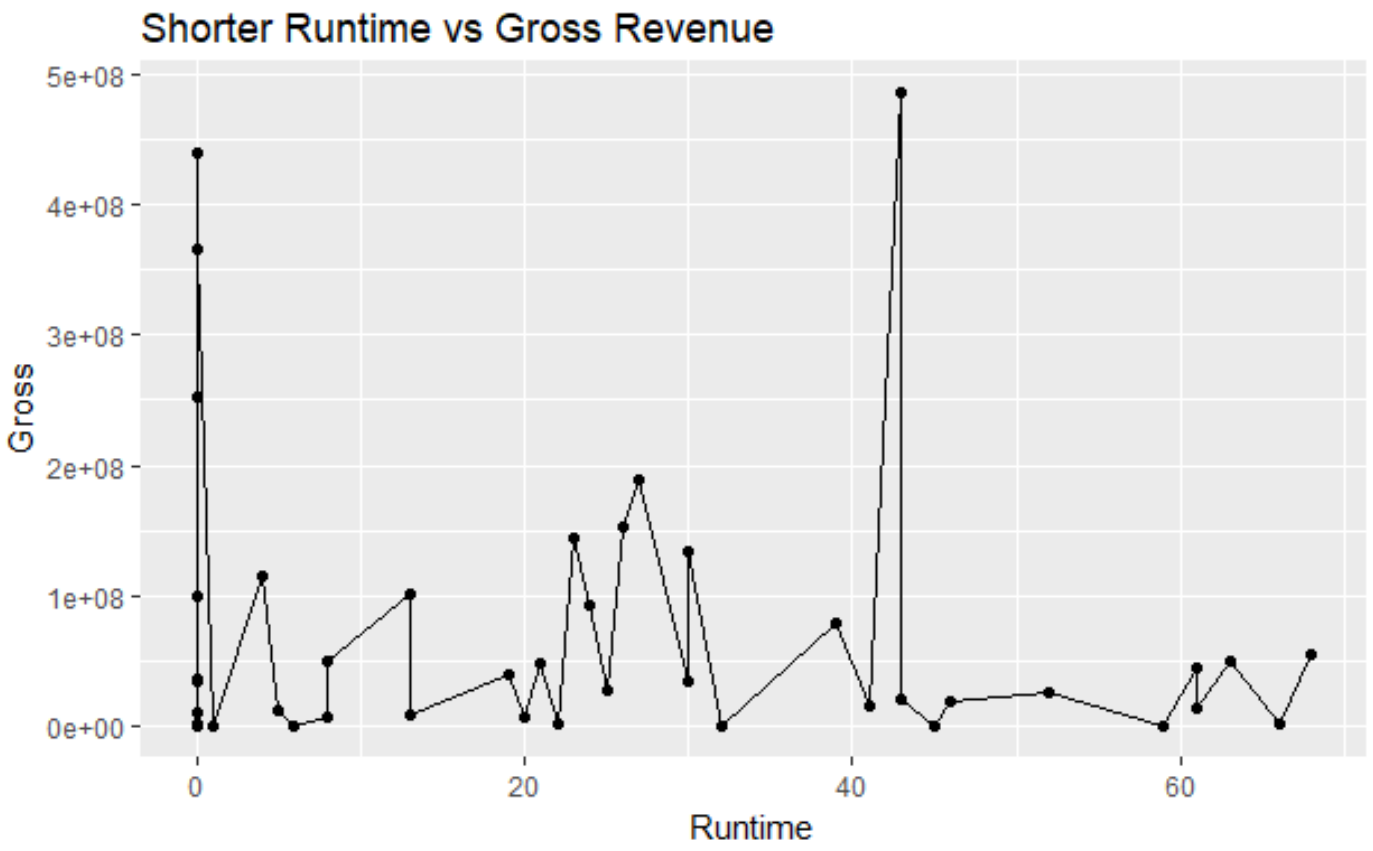
Hide

```
small_runtime = subset(gross_revenue, gross_revenue$Runtime < 69) #
1st quartile value
long_runtime = subset(gross_revenue, gross_revenue$Runtime > 100) #
3rd quartile value
ggplot(gross_revenue, aes(x = log10(Budget), y = log10(Gross))) + g
eom_point() + geom_line() + ggtitle("Budget vs Gross Revenue")
```



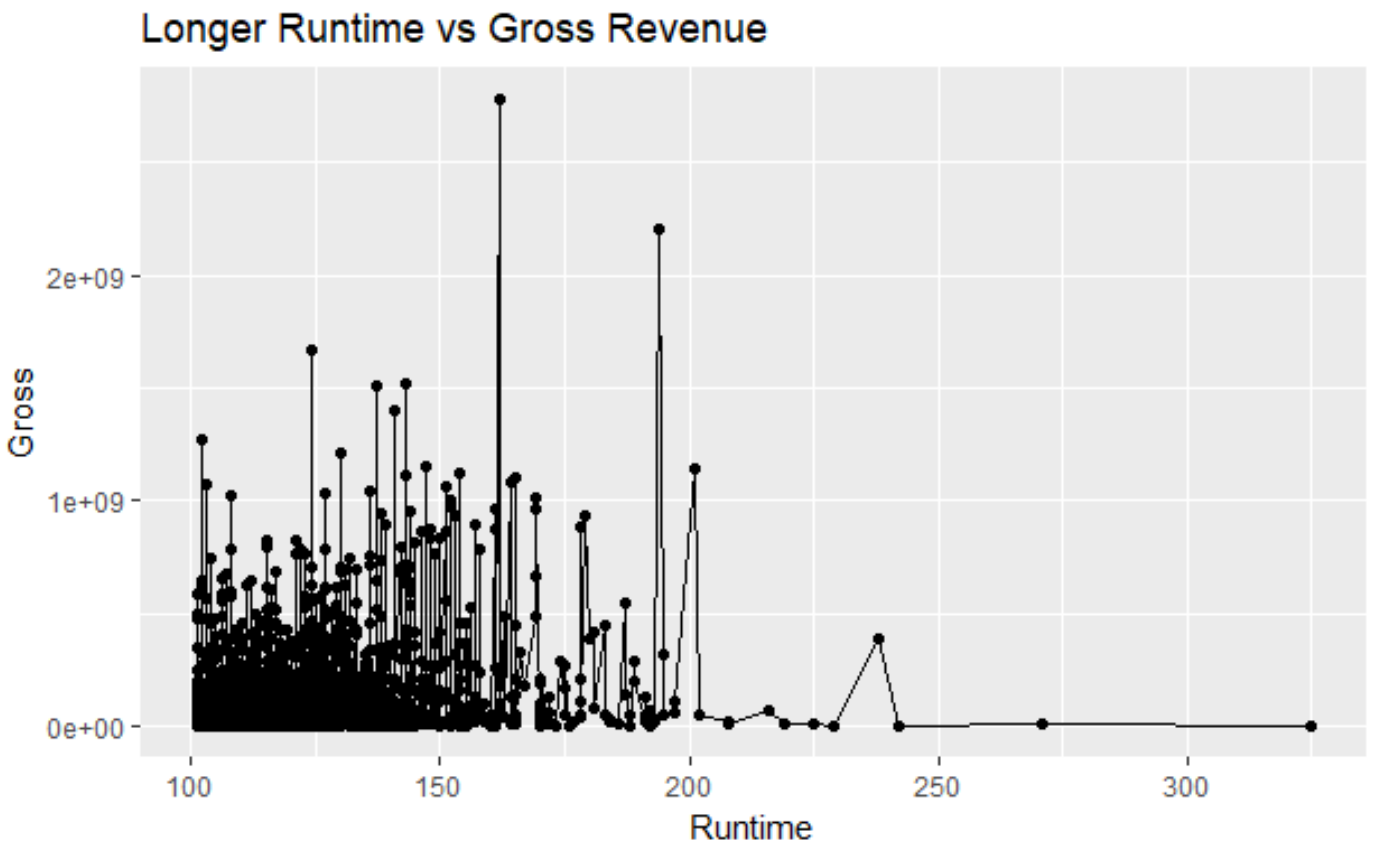
Hide

```
ggplot(small_runtime, aes(Runtime, Gross)) + geom_point() + geom_li
ne() + ggtitle("Shorter Runtime vs Gross Revenue")
```



Hide

```
ggplot(long_runtime, aes(Runtime, Gross)) + geom_point() + geom_lin
e() + ggtitle("Longer Runtime vs Gross Revenue")
```



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: From all the above graphs, we are able to observer several characteristics. Between gross revenue and budget, the graph depicts that as the budget increases, so will the runtime of the movie. There is also a positive correlation among the 2.

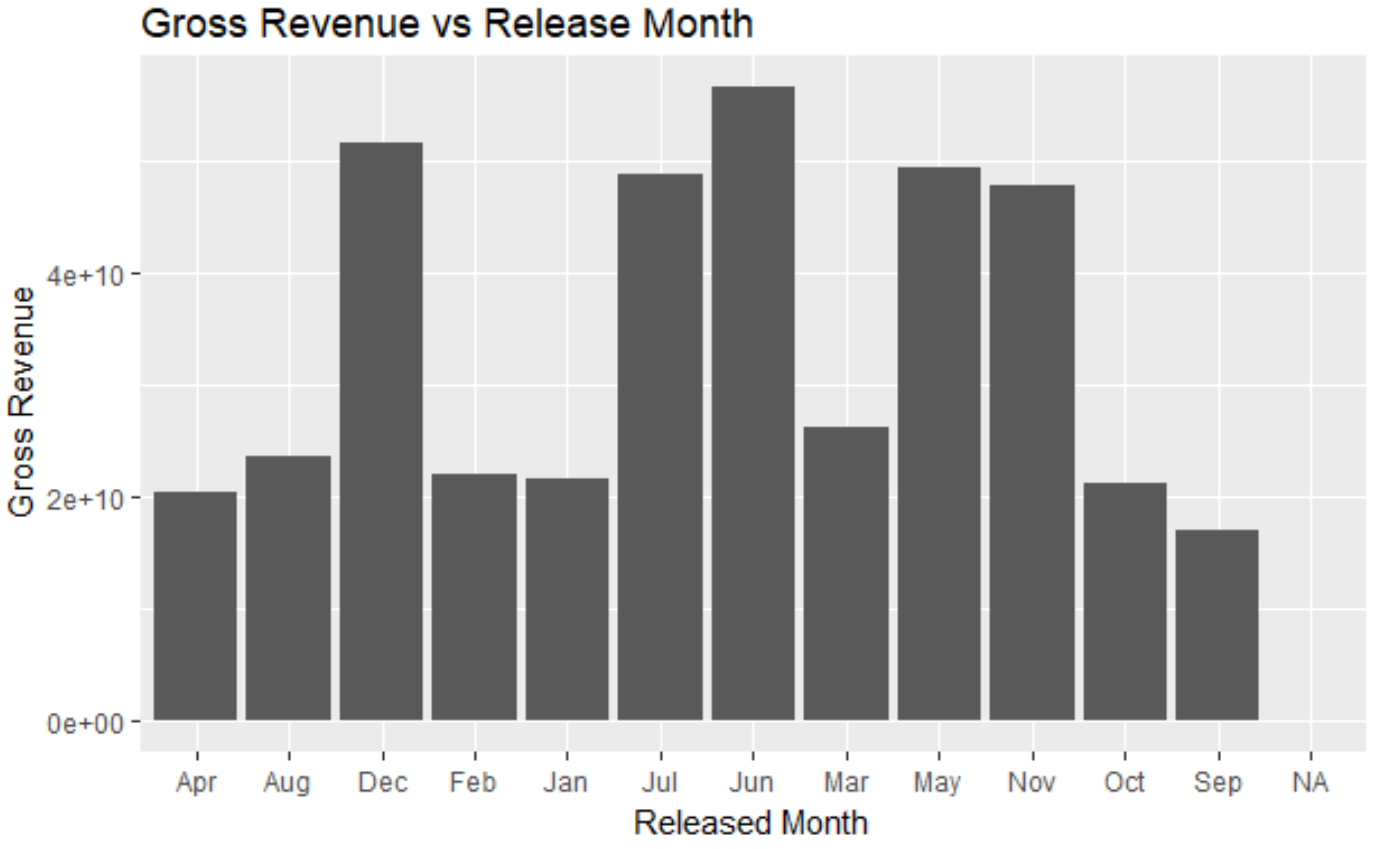
I then divided the movies based on runtime namely, runtime < 69 and runtime > 100. (The first and third quartiles repectively). Icould obtain no solid evidence or enough facts about the relationship between gross revenue and shorter film run times.

Finally, I divided it into a longer duration which led me to observe 2 things. 1) there were many outliers which have affected our calculations, and 2) the plot is linear in a way albeit some outliers.

Below, we can see the relationship between the gross revenue and the releae month. From the chart, it is observed that it will be most profitable to release a movie in the month of June and would also bring in good profits during the months of December, July, May, November. Some of the least profitable months would be September and April.

Hide

```
# TODO: Investigate if Gross Revenue is related to Release Month
release_month <- month(as.POSIXct(bad_data$Released, format = "%y-%
m-%d"))
bad_data$release_month <- month.abb[release_month]
ggplot(bad_data, aes(bad_data$release_month, bad_data$Gross)) +
  geom_bar(stat = "identity") + xlab("Released Month") + ylab("Gros
s Revenue") +
  ggtitle("Gross Revenue vs Release Month")
```



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.

Hide

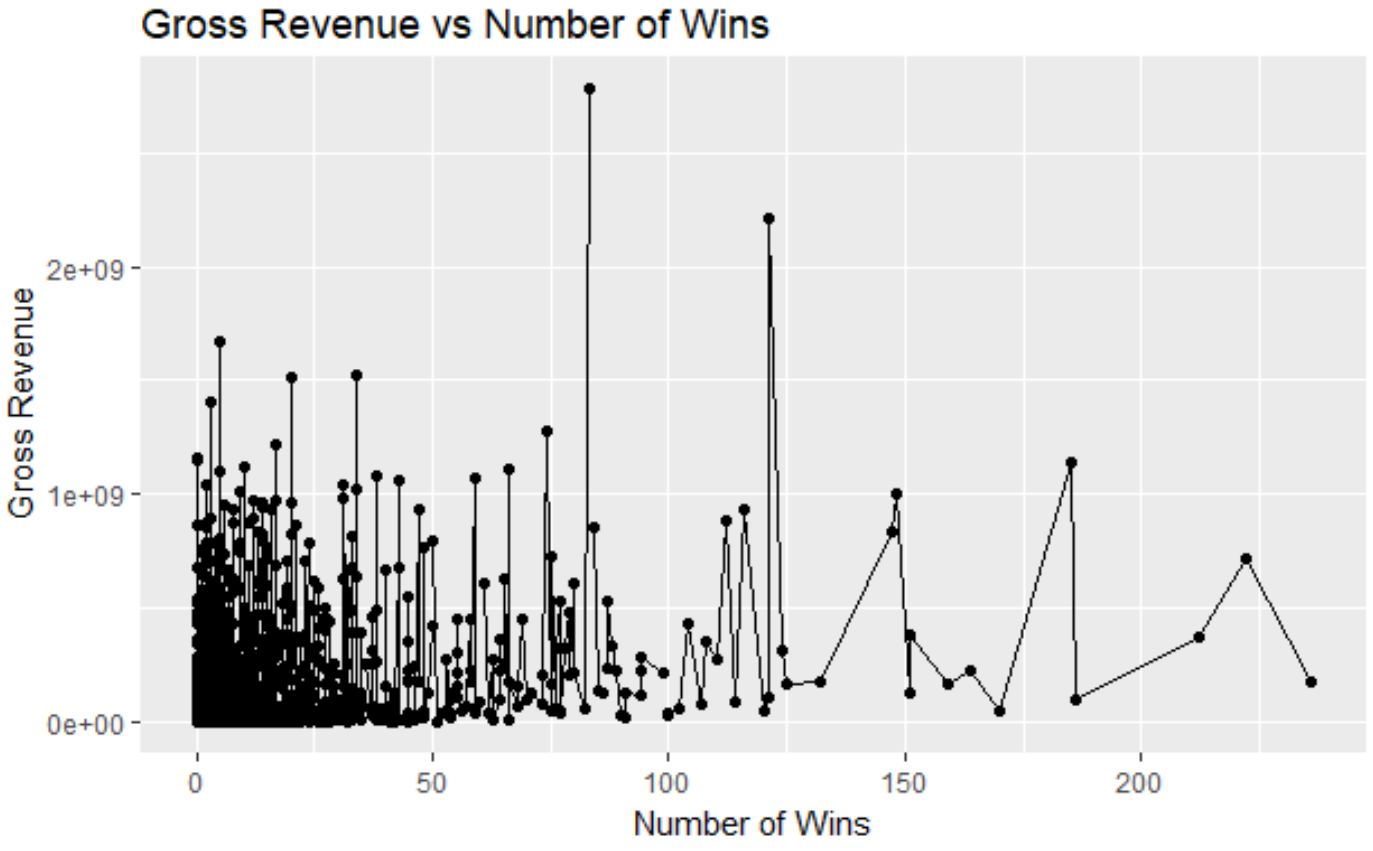
```
# TODO: Convert Awards to 2 numeric columns: wins and nominations
for (a in c(1:nrow(new_genre)))
{
  aw1 = 0
  aw2 = 0
  if(grepl("Won", new_genre[a, "Awards"])) {
    aw1=as.numeric(str_extract(str_extract(new_genre[a,"Awards"],
"Won ([\\d]+)"), "[[:digit:]]+"))
    aw1=aw1+as.numeric(str_extract(str_extract(new_genre[a,"Awards"
], "([\\d]+) win"), "[[:digit:]]+"))
    aw2=aw2+as.numeric(str_extract(str_extract(new_genre[a,"Awards"
], "([\\d]+) nomination"), "[[:digit:]]+"))
  }
  else if(grepl("Nominated",new_genre[a, "Awards"])){
    aw2=as.numeric(str_extract(str_extract(new_genre[a,"Awards"],
"Nominated for ([\\d]+)"), "[[:digit:]]+"))
    aw1=aw1+as.numeric(str_extract(str_extract(new_genre[a,"Awards"
], "([\\d]+) win"), "[[:digit:]]+"))
    aw2=aw2+as.numeric(str_extract(str_extract(new_genre[a,"Awards"
], "([\\d]+) nomination"), "[[:digit:]]+"))
  }
  else{
    aw1=aw1+as.numeric(str_extract(str_extract(new_genre[a,"Awards"
], "([\\d]+) win"), "[[:digit:]]+"))
    aw2=aw2+as.numeric(str_extract(str_extract(new_genre[a,"Awards"
], "([\\d]+) nomination"), "[[:digit:]]+"))
  }
    new_genre[a,"wins"]=aw1
    new_genre[a, "nominations"]=aw2
    if(is.na(new_genre[a,"wins"])){new_genre[a,"wins"]=0}
    if(is.na(new_genre[a,"nominations"])){new_genre[a,"nominations"
]=0}
  }
```

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

A: I first explored the awards column of the dataset and saw that there were several options that I could use regex for. There was a “won _ oscar”“,”_ wins“,”_ nominations“, and”nominated _ for“. The logic that I used does the following: If it encounters the word Won, its going to add 1 to the aw1 variable(award 1 - meaning a win). If it encounters wins or win somewhere, it will add the numeric value right before the word appears to the aw1 variable. Similarily for the nominations as well. After that is completed, it adds it to a separate column for wins and nominations separately.

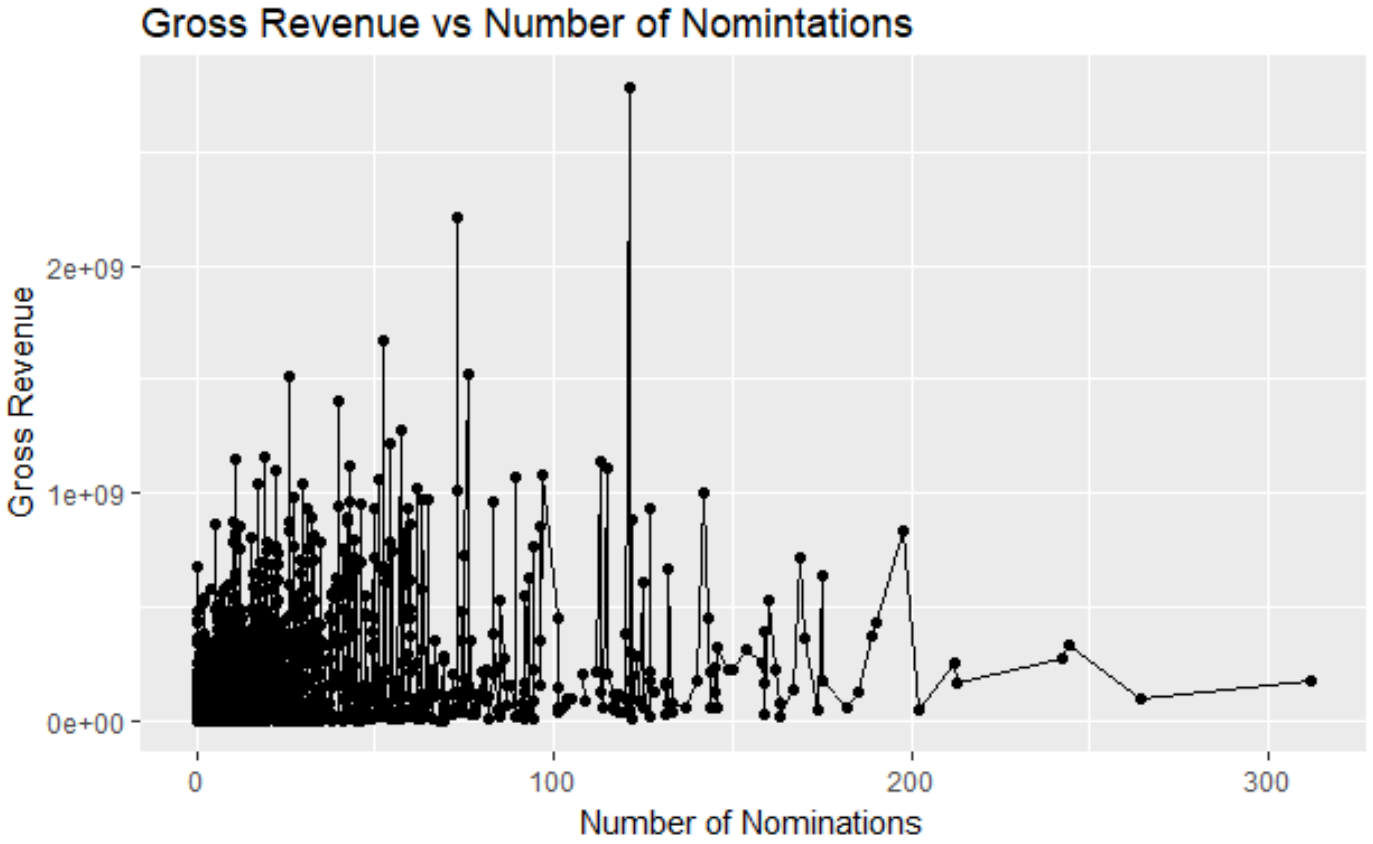
Hide

```
# TODO: Plot Gross revenue against wins and nominations
ggplot(new_genre, aes(x = wins, y = Gross)) + geom_point() + geom_l
ine() + ggtitle("Gross Revenue vs Number of Wins") + xlab("Number o
f Wins") + ylab("Gross Revenue")
```



Hide

```
ggplot(new_genre, aes(x = nominations, y = Gross)) + geom_point() +
geom_line() + ggtitle("Gross Revenue vs Number of Nomintations") +
 xlab("Number of Nominations") + ylab("Gross Revenue")
```



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

A: Just by looking at the graph, we can observe that as the number of wins/nominations increases, so does the gross revenue. It also goes down as the number of wins/nominations decreases. Thus we can say it has a linear relationship.

7. Movie ratings from IMDb and Rotten Tomatoes

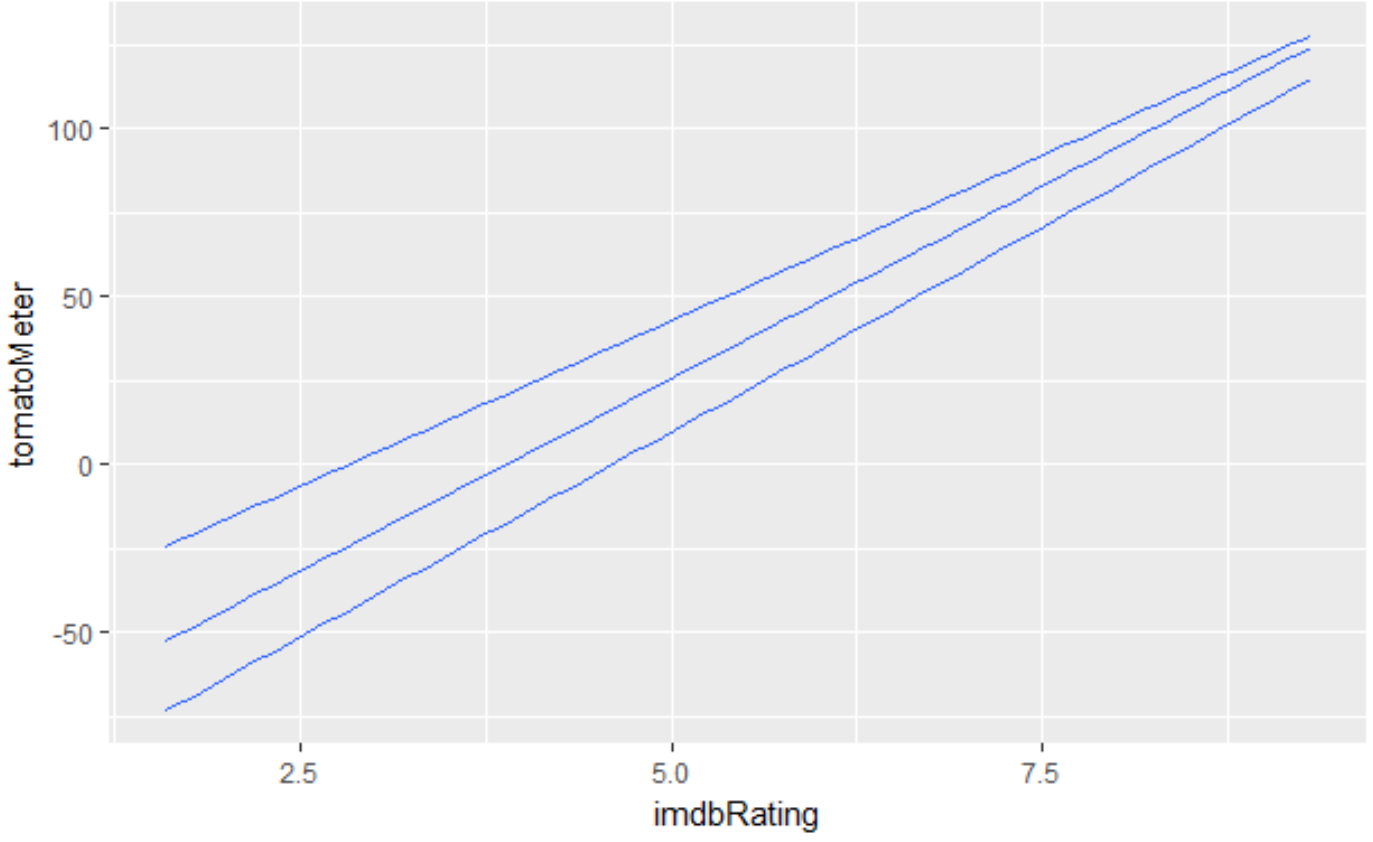
There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by

several variables pre-fixed by `tomato`). Read up on such ratings on the web (for example rottentomatoes.com/about (https://www.rottentomatoes.com/about) and www.imdb.com/help/show_leaf?votestopfaq (http://%20www.imdb.com/help/show_leaf?votestopfaq)).

Investigate the pairwise relationships between these different descriptors using graphs.

Hide

```
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are re
lated
ggplot(new_genre, aes(x = imdbRating, y = tomatoMeter)) + geom_quan
tile()
```

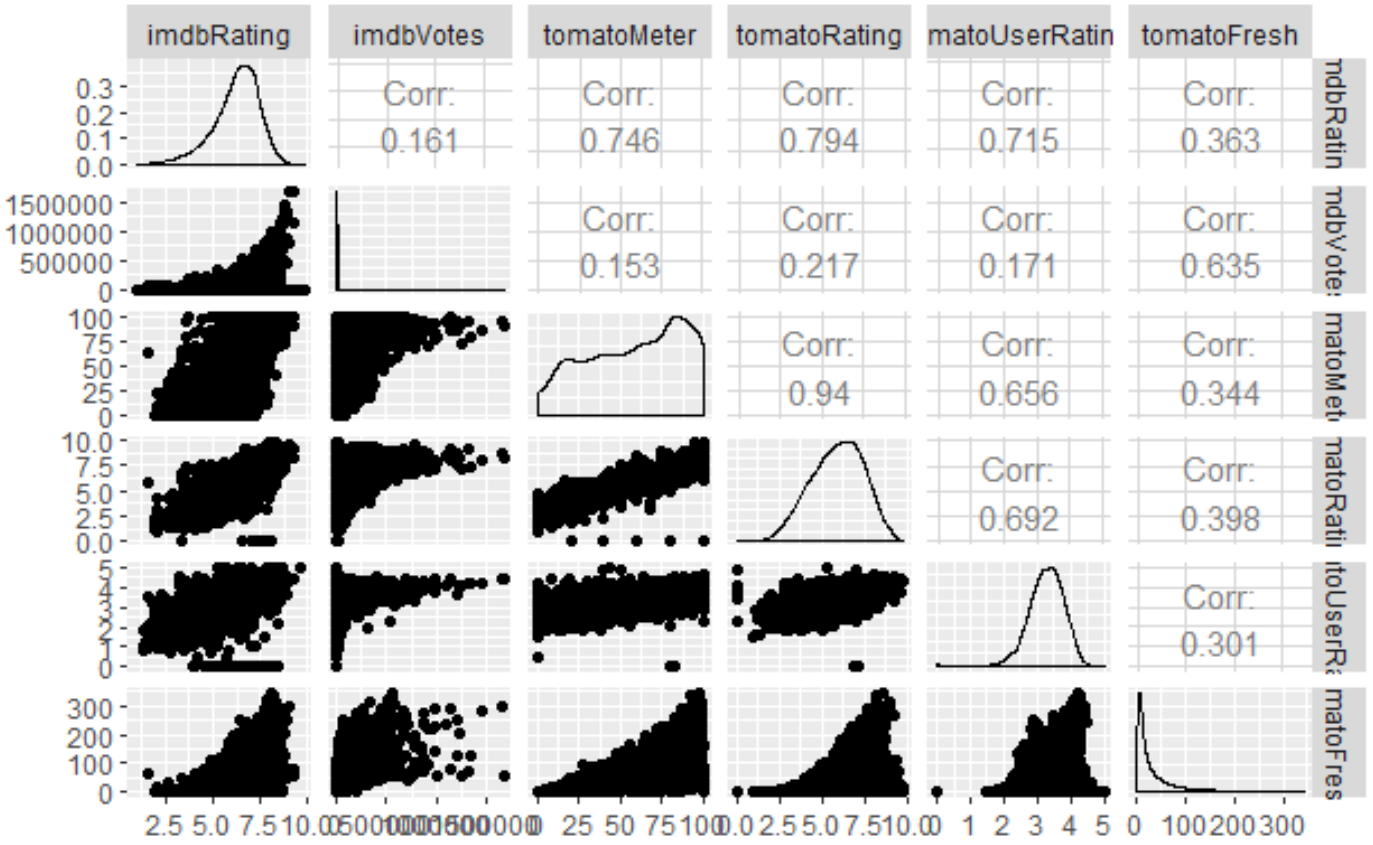


Hide

```
ggpairs(new_genre[,c("imdbRating", "imdbVotes", "tomatoMeter", "tom
atoRating", "tomatoUserRating", "tomatoFresh")])
```

```
 plot: [1,1] [=-----------------------------------------]  3% es
t: 0s
 plot: [1,2] [==----------------------------------------]  6% es
t: 2s
 plot: [1,3] [====--------------------------------------]  8% es
t: 2s
 plot: [1,4] [=====-------------------------------------] 11% es
t: 2s
 plot: [1,5] [======------------------------------------] 14% es
t: 2s
 plot: [1,6] [========----------------------------------] 17% es
t: 2s
 plot: [2,1] [=========---------------------------------] 19% es
t: 2s
 plot: [2,2] [==========--------------------------------] 22% es
t: 3s
 plot: [2,3] [===========-------------------------------] 25% es
t: 2s
 plot: [2,4] [============------------------------------] 28% es
t: 2s
 plot: [2,5] [==============----------------------------] 31% es
t: 2s
 plot: [2,6] [===============---------------------------] 33% es
t: 2s
 plot: [3,1] [================--------------------------] 36% es
t: 2s
 plot: [3,2] [==================------------------------] 39% es
t: 2s
 plot: [3,3] [===================-----------------------] 42% es
t: 2s
 plot: [3,4] [====================----------------------] 44% es
t: 2s
 plot: [3,5] [=====================---------------------] 47% es
t: 2s
 plot: [3,6] [======================--------------------] 50% es
t: 2s
 plot: [4,1] [========================------------------] 53% es
```

```
t: 1s
 plot: [4,2] [=========================-------------------] 56% es
t: 1s
 plot: [4,3] [==========================------------------] 58% es
t: 1s
 plot: [4,4] [============================----------------] 61% es
t: 1s
 plot: [4,5] [==============================--------------] 64% es
t: 1s
 plot: [4,6] [================================------------] 67% es
t: 1s
 plot: [5,1] [==================================----------] 69% es
t: 1s
 plot: [5,2] [====================================--------] 72% es
t: 1s
 plot: [5,3] [======================================------] 75% es
t: 1s
 plot: [5,4] [========================================----] 78% es
t: 1s
 plot: [5,5] [==========================================--] 81% es
t: 1s
 plot: [5,6] [============================================] 83% es
t: 1s
 plot: [6,1] [==========================================--] 86% es
t: 0s
 plot: [6,2] [========================================----] 89% es
t: 0s
 plot: [6,3] [======================================------] 92% es
t: 0s
 plot: [6,4] [====================================--------] 94% es
t: 0s
 plot: [6,5] [==================================----------] 97% es
t: 0s
 plot: [6,6] [==================================================]100% es
t: 0s
```



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

A: IMDB uses 2 separate ratings: ‘imdbrating’ and ‘imdbvotes’ Rotten tomatoes however uses 4 different ratings: ‘tomatometer’, ‘tomatofresh’, ‘tomatorating’ and ‘tomatouserratings’ The pairs graph can be seen above. It can be discerned that there is a high correlation between the ratings. Imdb does not have any self correlation whereas, the opposite can be said about Rotten tomatoes(which has a much stronger self correlation). The least correlation is observed between tomatoMeter and imdbVotes. However, the main similarity both ratings share is that they are centralized over a range of rating scores.

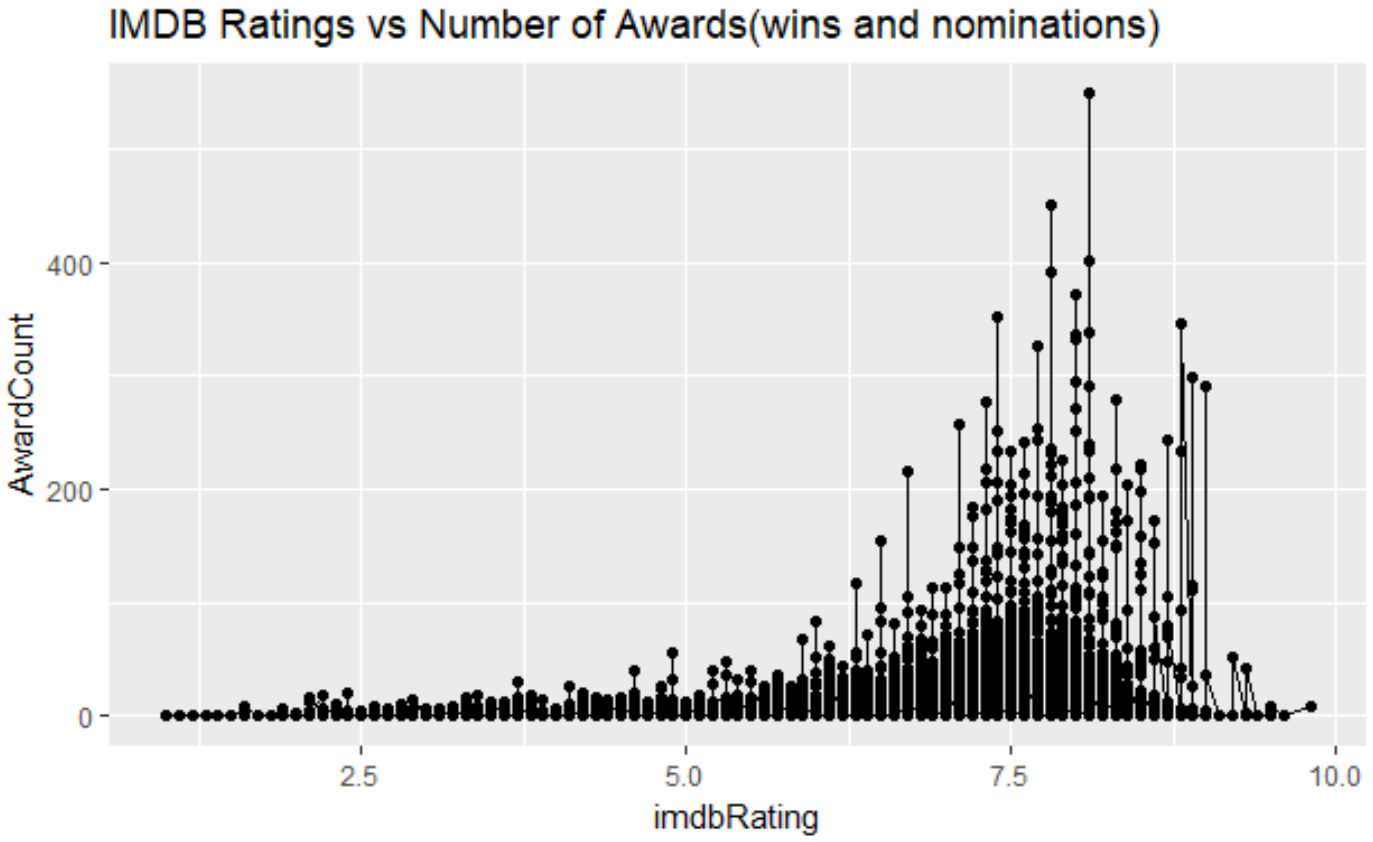
8. Ratings and awards

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).

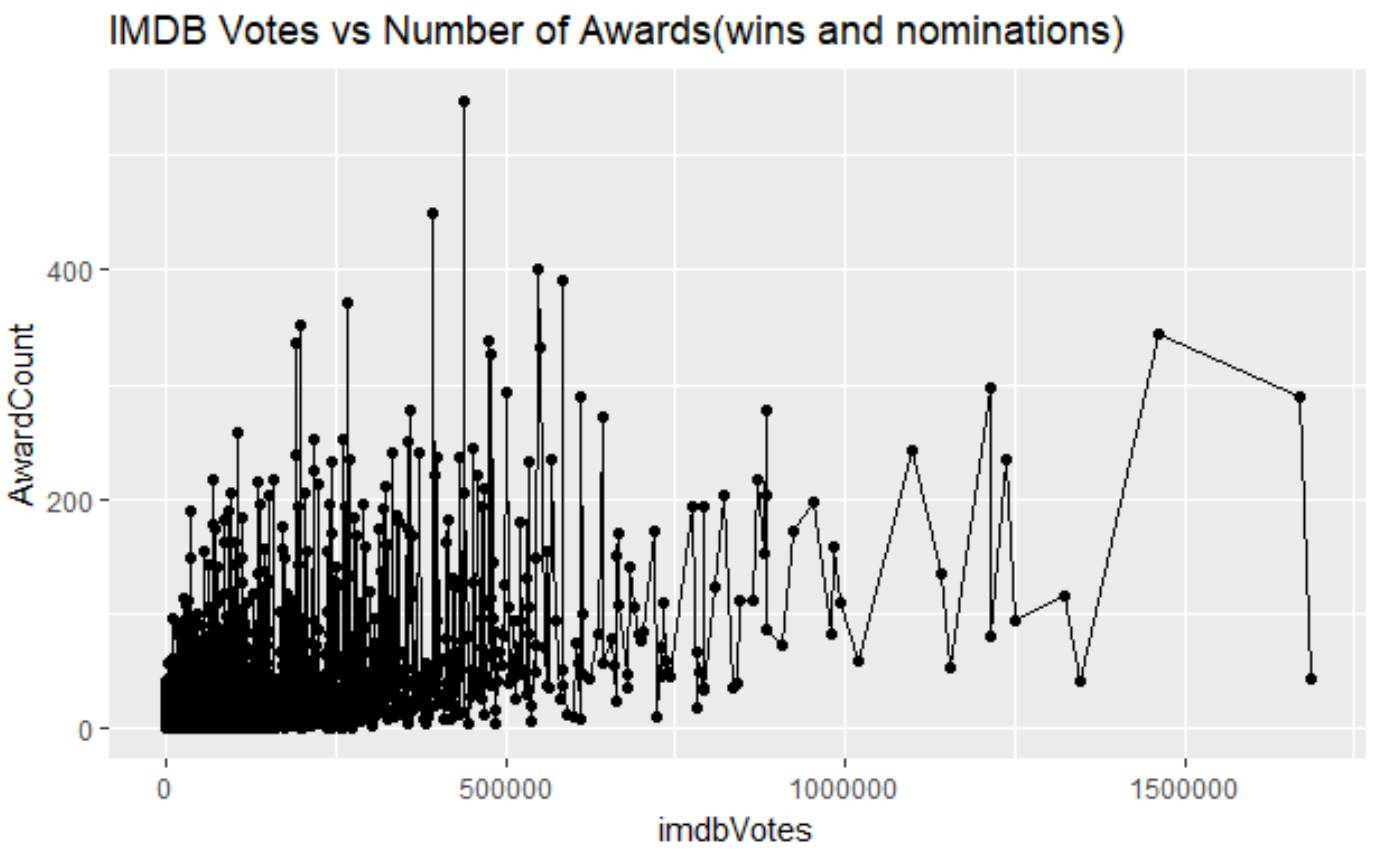
Hide

```
# TODO: Show how ratings and awards are related
new_genre$AwardCount <- new_genre$wins + new_genre$nominations
ggplot(new_genre, aes(x = imdbRating, y = AwardCount)) + geom_point
() + geom_line() + ggtitle("IMDB Ratings vs Number of Awards(wins a
nd nominations)")
```



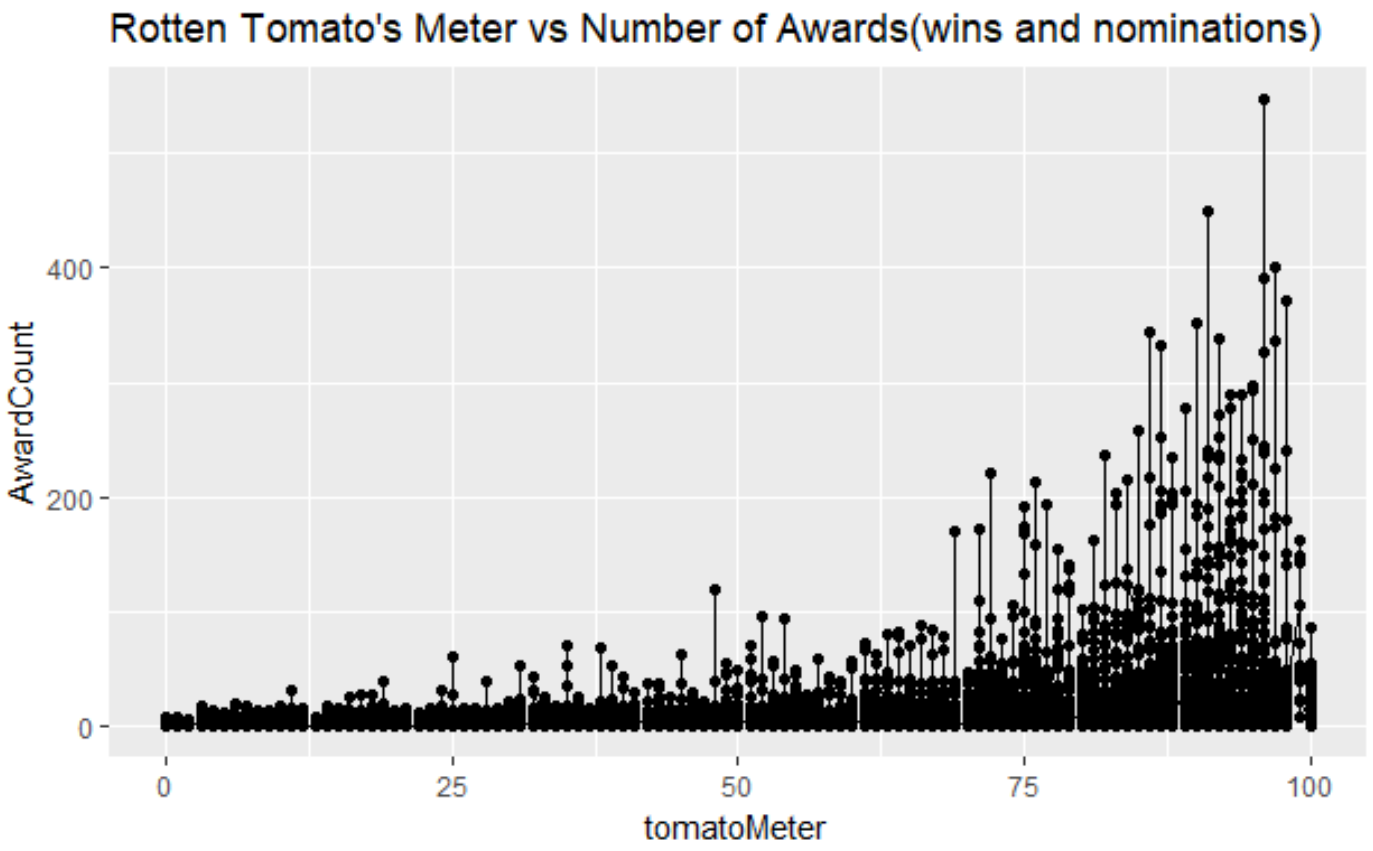
Hide

```
ggplot(new_genre, aes(x = imdbVotes, y = AwardCount)) + geom_point()
+ geom_line() + ggtitle("IMDB Votes vs Number of Awards(wins and no
minations)")
```



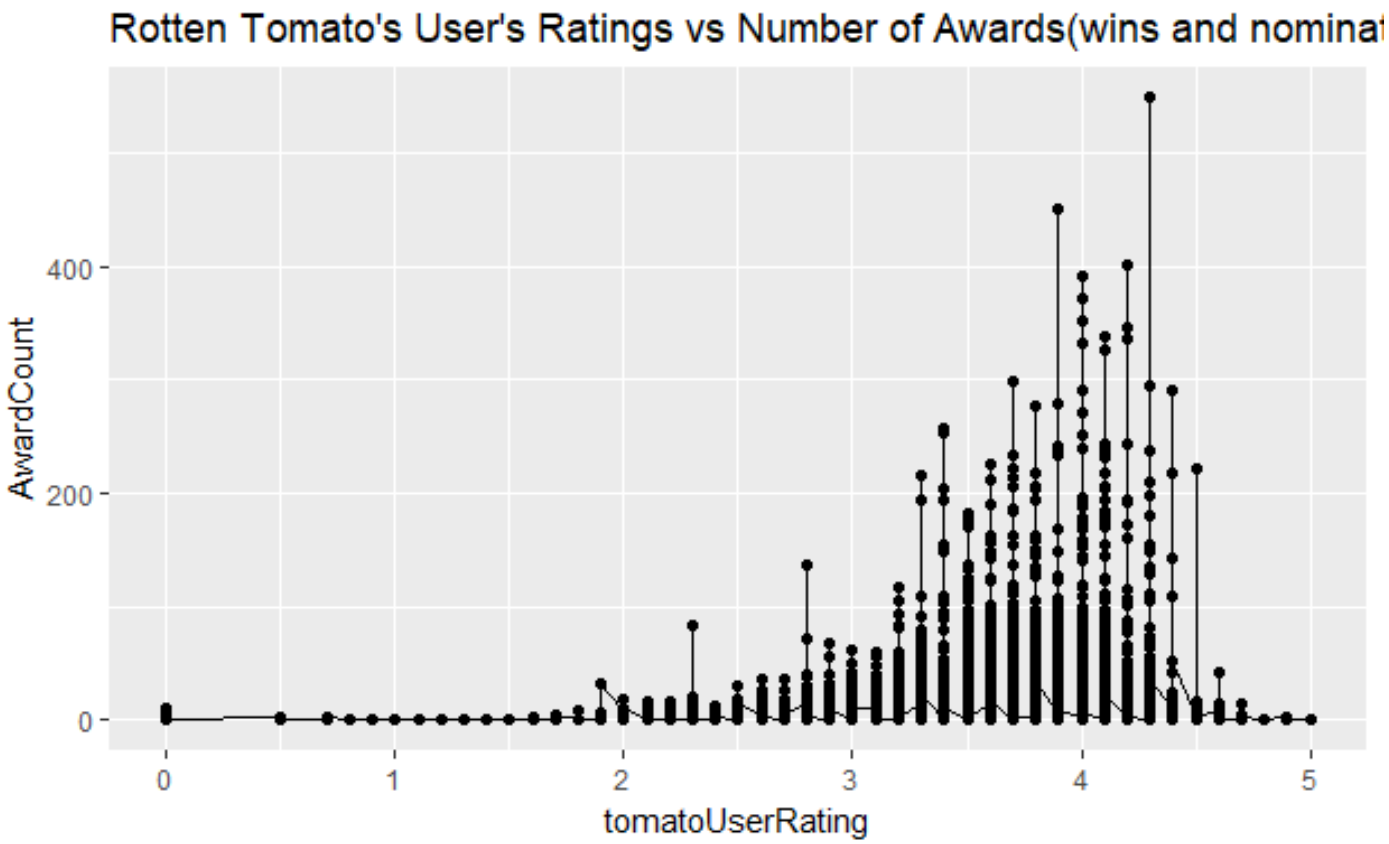
Hide

```
ggplot(new_genre, aes(x = tomatoMeter, y = AwardCount)) + geom_poin
t() + geom_line() + ggtitle("Rotten Tomato's Meter vs Number of Awa
rds(wins and nominations)")
```



Hide

```
ggplot(new_genre, aes(x = tomatoUserRating, y = AwardCount)) + geom
_point() + geom_line() + ggtitle("Rotten Tomato's User's Ratings vs
 Number of Awards(wins and nominations)")
```



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?

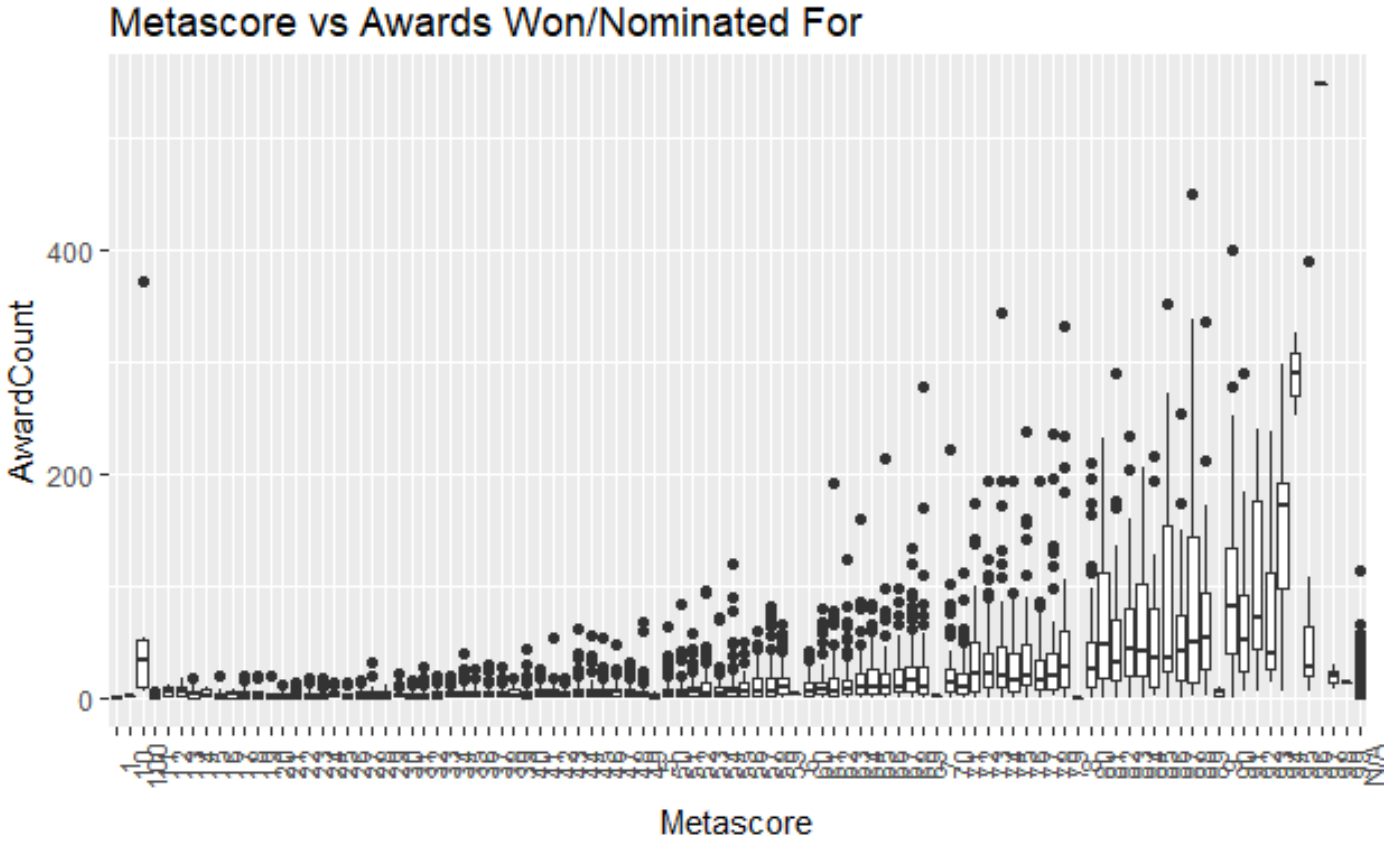
A: All the ratings appear to have a strong correlation between the total awards count (both wins and nominations). This proves that they can be used quite extensively for predicting the success of a movie. The highest correlation observed was between imdbVotes and the AwardsCount.

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.

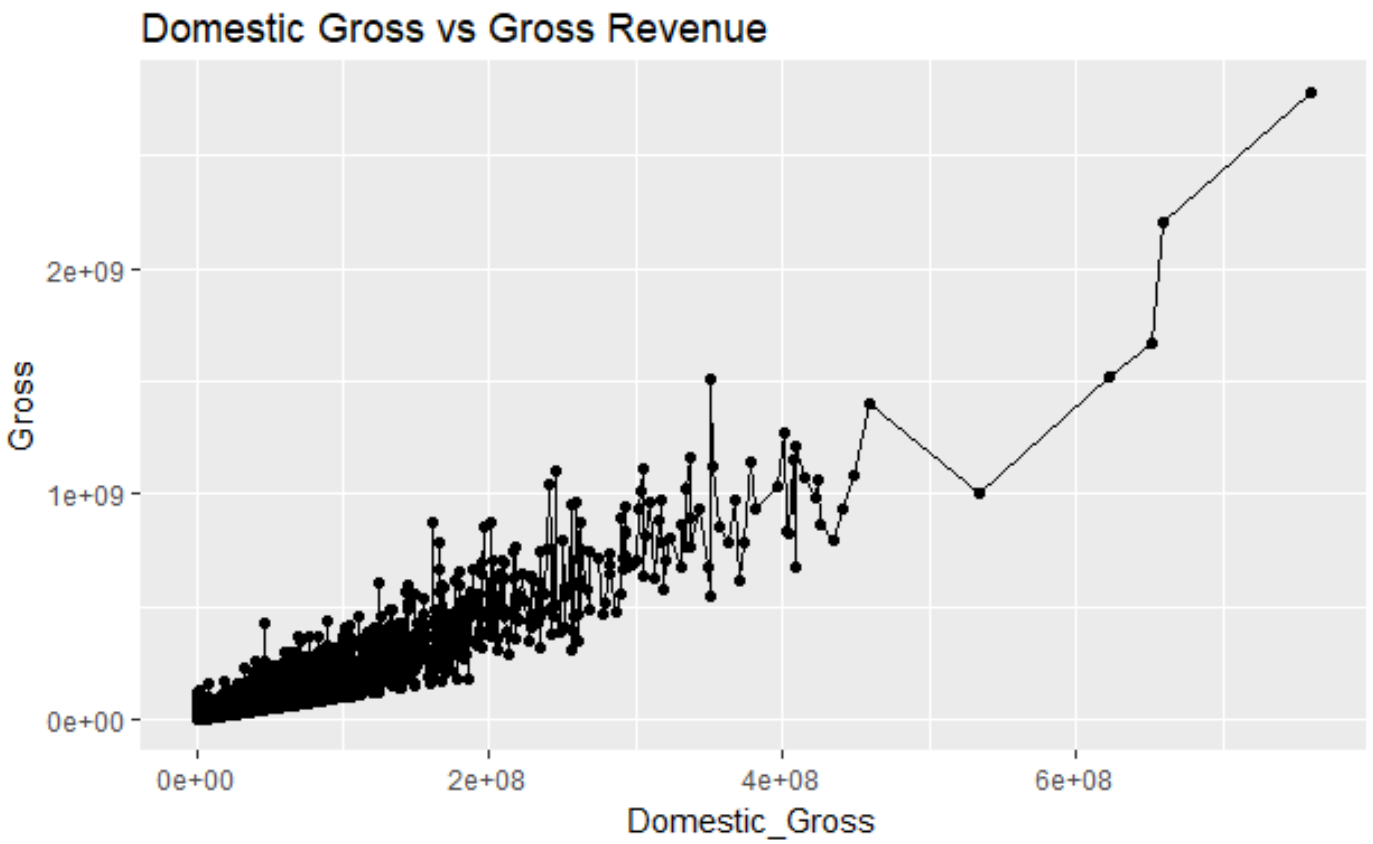
Hide

```
# TODO: Find and illustrate two expected insights
#new_genre$Metascore[is.na(new_genre$Metascore)]
#INSIGHT1
#RELATIONSHIP BETWEEN METASCORE AND AWARD COUNT
ggplot(new_genre, aes(x = Metascore, y = AwardCount)) + geom_boxplo
t() +
  theme(axis.text.x = element_text(angle = 90)) + ggtitle("Metascor
e vs Awards Won/Nominated For")
```



Hide

```
#INSIGHT2
#RELATIONSHIP BETWEEN DOMESTIC GROSS AND  GROSS REVENUE
ggplot(new_genre, aes(x = Domestic_Gross, y = Gross)) + geom_point()
 + geom_line() + ggtitle("Domestic Gross vs Gross Revenue")
```



Q: Expected insight #1.

A: A relationship between Metascore and the total award count (win/nominations) is taken into affect above. As we can see clearly, there is some positive correlation between the two quantities leading us to the conclusion that the higher the metascore, the more number of awards will be won or nominated for. Due to the presence of some outliers, the relationship can't be said as a strictly linear relationship. As we can clearly expect from the dataset, as the score keeps getting higher, the film will prove to be much more productive in terms of producing more revenue or even getting more awards, etc.

Q: Expected insight #2.

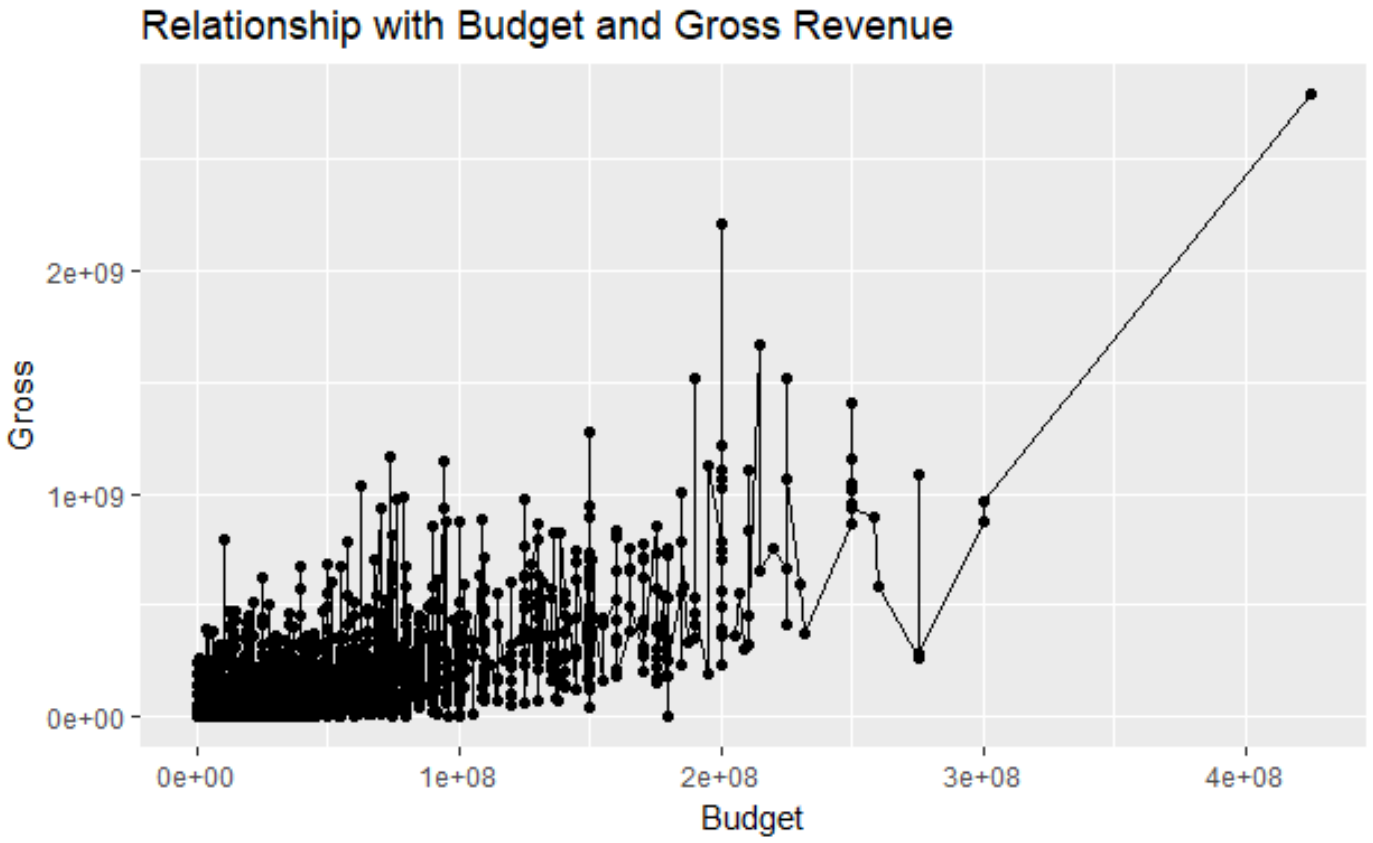
A: At a single glance at the graph between the domestic gross and the gross revenue, we can see that it is a pure linear relationship with a high positive correlation. Also tells us something around the lines of as the domestic gross revenue for a film increases, so will the total gross revenue, which is already a known fact to us.

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.

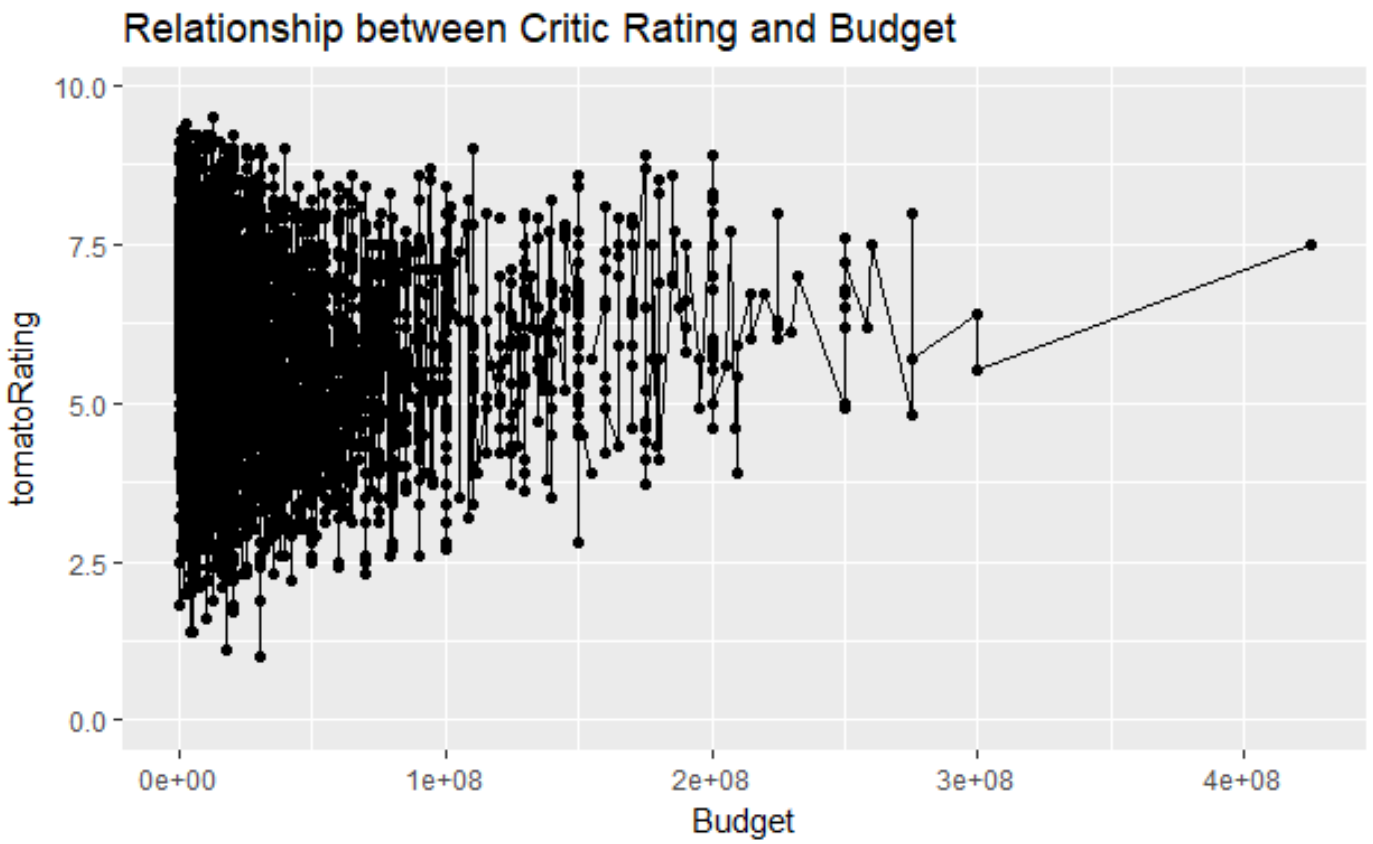
Hide

```
# TODO: Find and illustrate one unexpected insight
ggplot(new_genre, aes(x = Budget, y = Gross)) + geom_point() + geom
_line() + ggtitle("Relationship with Budget and Gross Revenue")
```



Hide

```
ggplot(new_genre, aes(x = Budget, y = tomatoRating)) + geom_point()
+ geom_line() + ggtitle("Relationship between Critic Rating and Bud
get")
```



Q: Unexpected insight.

A: From the above 2 charts, we clearly notice that as the budget for a film keeps increases, so does the gross revenue proving that it is indeed a linear relationship. However, when a graph is plotted against the budget and the critic rating, it appears that no matter how high the budget might be, it doesn't necessarily prove that the rating of the film will be high.