Code

Project 1: Explore and Prepare Data

Name: Fan Zhou

GT Account: fzhou32

CSE6242 - Data and Visual Analytics - Spring 2017 Due: Sunday, March 5, 2017 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, and will be released at a later date. Both projects will have equal weightage towards your grade.

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 IMDbâs 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 IMDbâs 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 âsecond 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

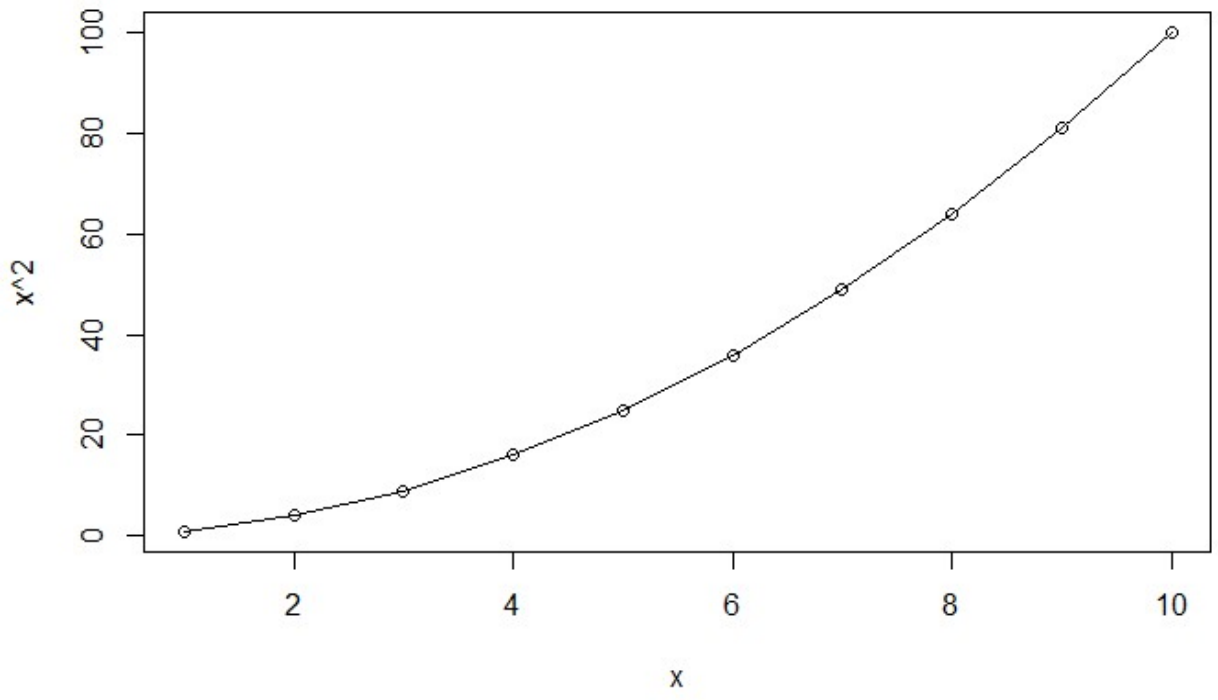
```
x = 1:10
print(x^2)
```

```
 [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*.

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 the tasks below and submit this R Markdown file (as **pr1.Rmd**) as well as a PDF export of it (as **pr1.pdf**). Both should contain all the code, output, plots and written responses for each task.

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')
```

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
cat("Dataset has", dim(df)[1], "rows and", dim(df)[2], "columns", end="\n", fil
e="")
```

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

Hide

```
colnames(df)
```

```
 [1] "Title"             "Year"              "Rated"             "Release
d"
 [5] "Runtime"           "Genre"             "Director"          "Write
r"
 [9] "Actors"            "Plot"              "Language"          "Countr
y"
[13] "Awards"            "Poster"            "Metascore"         "imdbRatin
g"
[17] "imdbVotes"         "imdbID"            "Type"              "tomatoMete
r"
[21] "tomatoImage"       "tomatoRating"      "tomatoReviews"     "tomatoFres
h"
[25] "tomatoRotten"      "tomatoConsensus"   "tomatoUserMeter"   "tomatoUserRat
ing"
[29] "tomatoUserReviews" "tomatoURL"         "DVD"               "BoxOffic
e"
[33] "Production"        "Website"           "Response"          "Budge
t"
[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

```
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 OK 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
df = subset(df, Type == 'movie')
```

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.

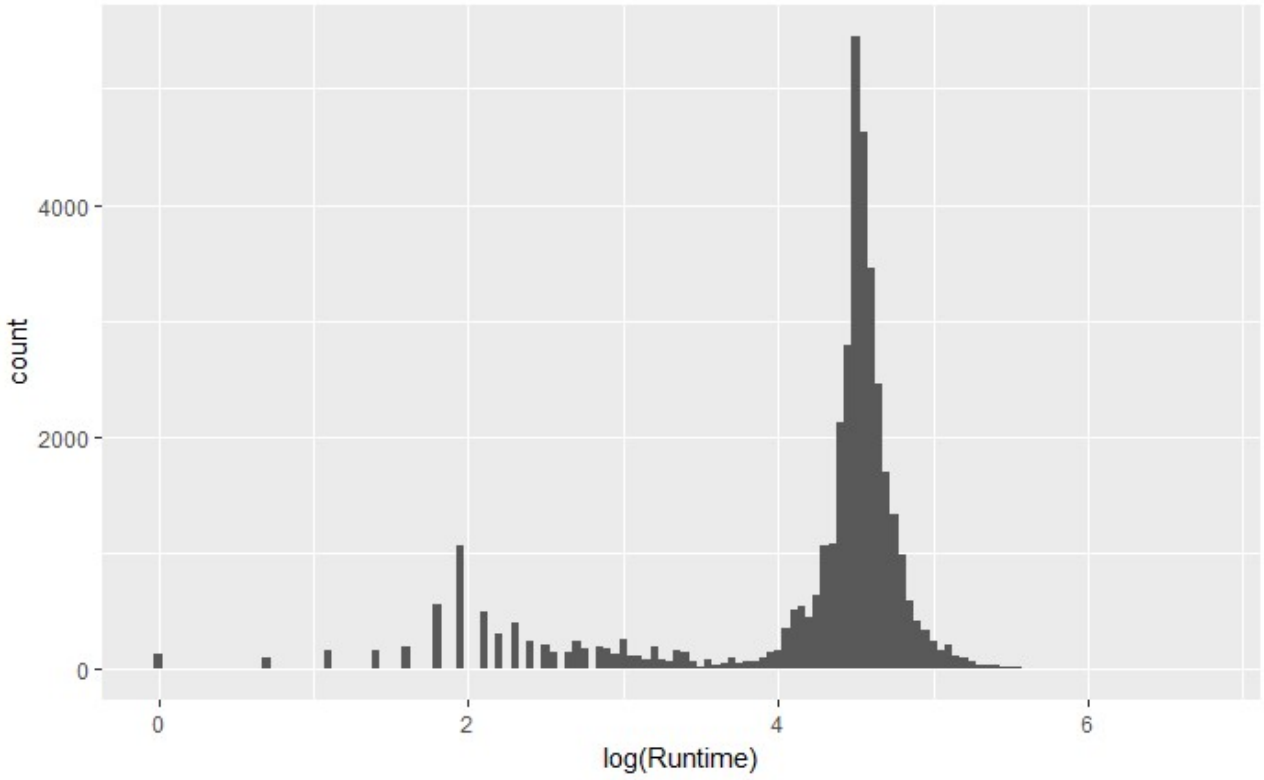
Hide

```
# TODO: Replace df$Runtime with a numeric column containing the runtime in minu
tes
names(df)[names(df)=='Runtime']='Old_Runtime'
convert_to_min = function(x) {
  if (x=='N/A') time = NA
  else {
    temp = unlist(strsplit(x, split=' '))
    if (length(temp)==4 & temp[2]=='h' & temp[4]=='min') {
      time=as.numeric(temp[1])*60+as.numeric(temp[3])
    } else if (temp[2]=='h' & length(temp)==2) {
      time=as.numeric(temp[1])*60
    } else {time=as.numeric(temp[1])
    }
  }
  return (time)
}
for (i in 1:nrow(df)) {
   df$Runtime[i] = convert_to_min(df$Old_Runtime[i])
}
#df = subset(df, select = -Old_Runtime)
```

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.

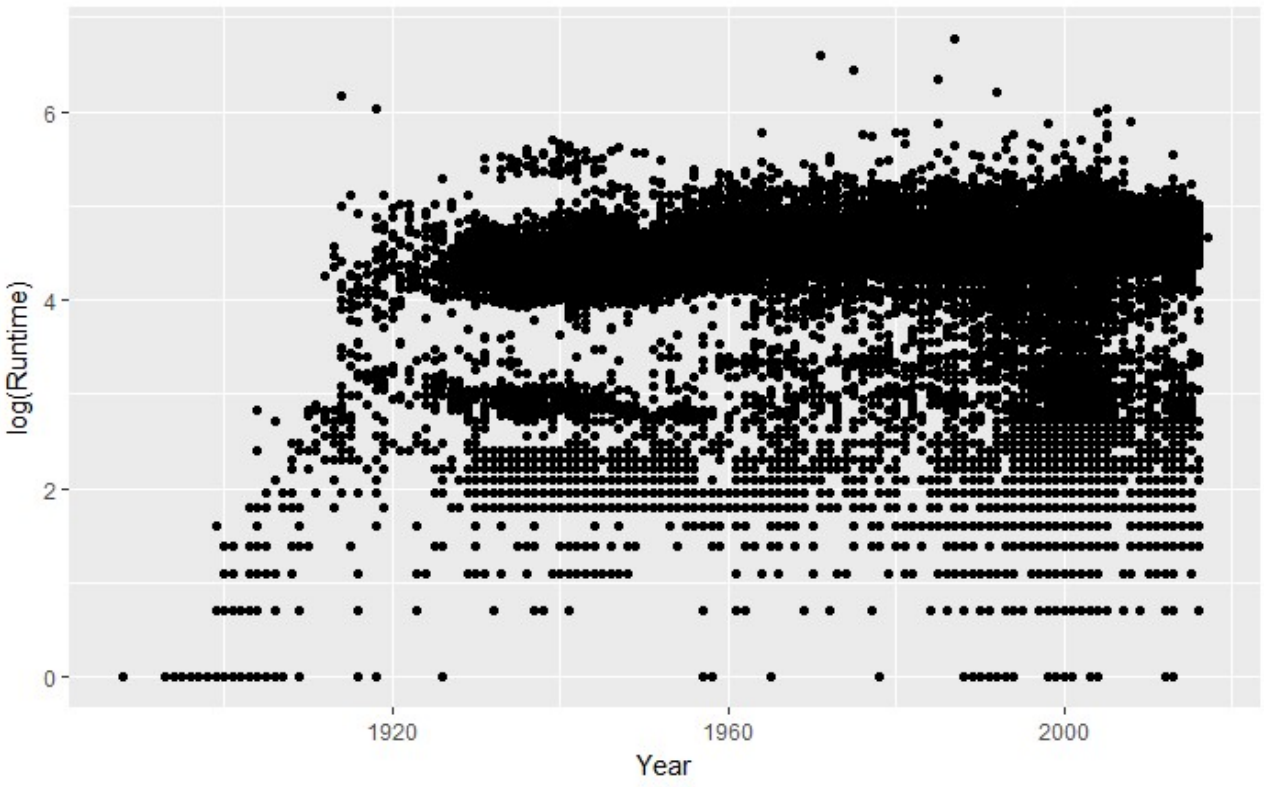
Hide

```
# TODO: Investigate the distribution of Runtime values and how it varies by Yea
r and Budget
# Runtime distribution
temp = subset(df, is.na(Runtime)==0)
runtime_dist = ggplot(temp, aes(x=log(Runtime)))+
  geom_histogram(binwidth = 0.05)
print (runtime_dist)
```



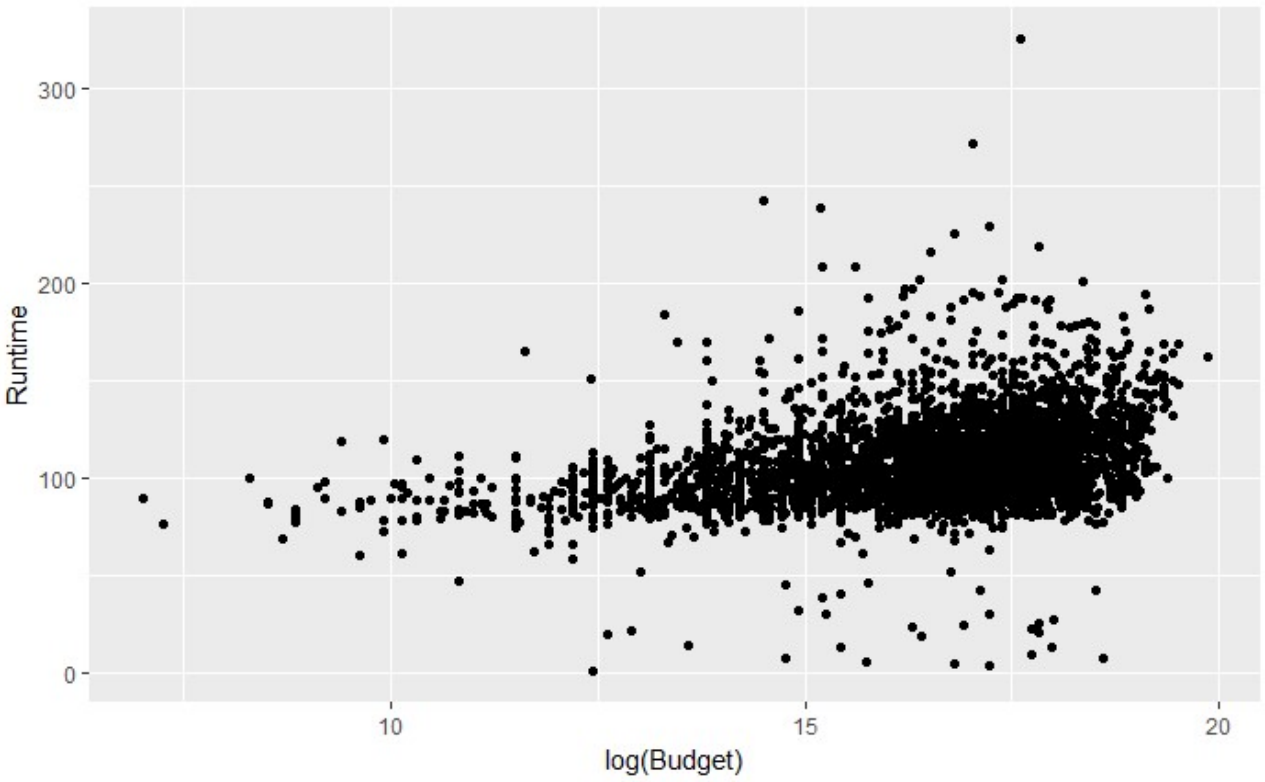
Hide

```
# Runtime vs. Year
temp = subset(df, is.na(Runtime)==0)
g_runtime_year = ggplot(temp, aes(x=Year, y=log(Runtime)))+
  geom_point()
print (g_runtime_year)
```



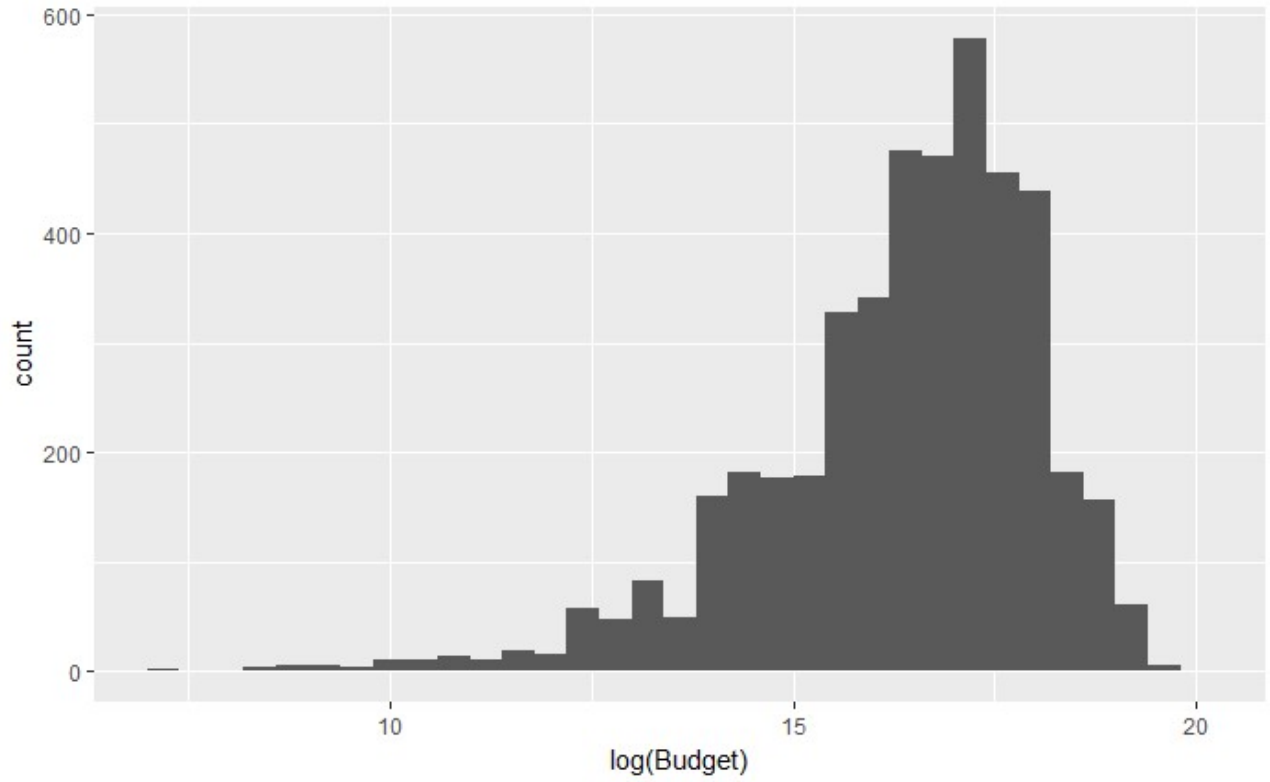
Hide

```
# Runtime vs. Budget
temp = subset(df,is.na(Budget)==0 & is.na(Runtime)==0)
g_runtime_budget = ggplot(temp, aes(x=log(Budget), y=Runtime))+
  geom_point()
print (g_runtime_budget)
```



Hide

```
budget_dist = ggplot(temp, aes(x=log(Budget)))+
  geom_histogram(binwidth = 0.4)
print (budget_dist)
```



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: In Runtime-count plot, the histogram is skewed to the right and is not very informative, so I applied the log-scale to the Runtime. From the log(Runtime)-count plot, it is found two obvious peaks at 1.9 and 4.6, which corresponds approximately to 7 min and 100 min. In the Year vs. log(Runtime) plot, we see before 1910 the Runtime is low and after 1910 movies are divided into two Runtime range and this clear gap continues to ~1990. From 1990 to 2003, this gap is somewhat filled since movies with more various Runtime appeared. After 2003, the Runtime is still tend to have two regions. In the log(Budget) vs. Runtime plot, we notice that when the Budget increases the Runtime distribution becomes wider. From the log(Budget) histogram, movies with budget of ~2.4e9 have the highest count.

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

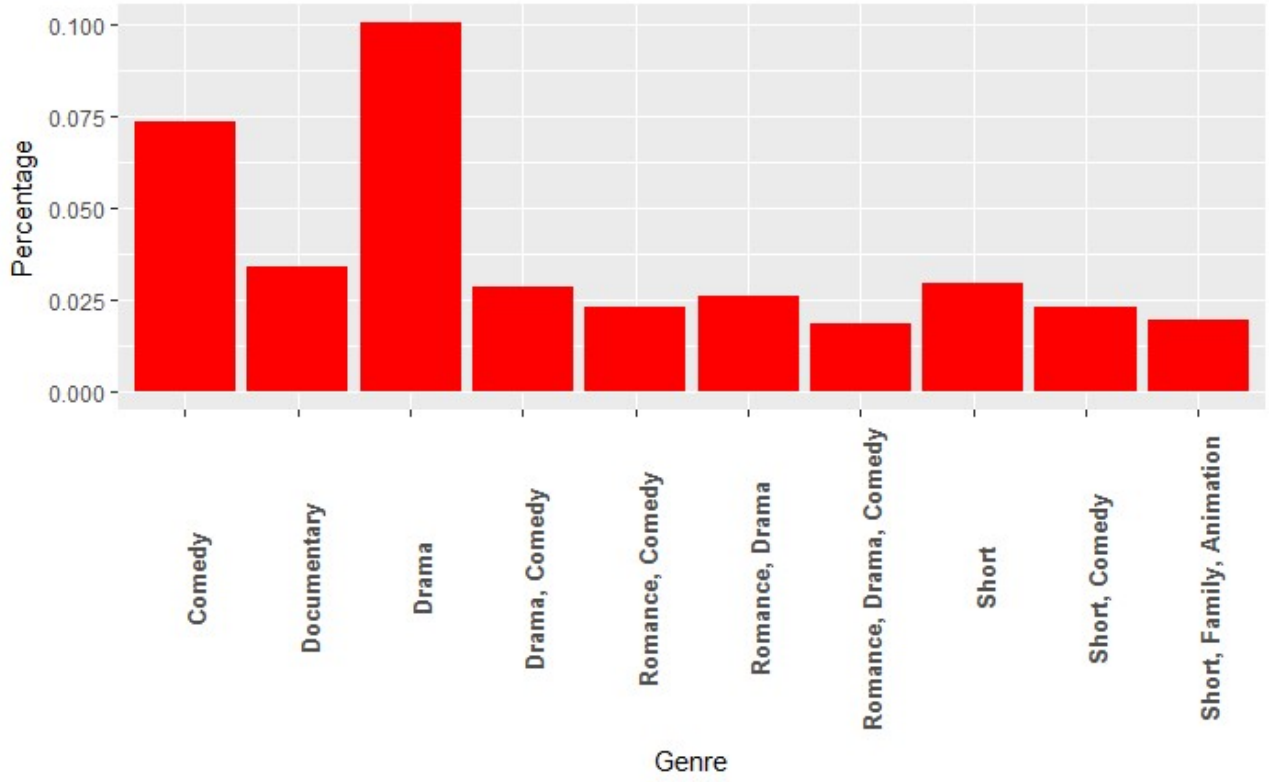
```
# TODO: Replace Genre with a collection of binary columns
# Find all types and place them in a vector
genre=c()
n=1
for (i in 1:nrow(df)) {
  temp = unlist(strsplit(df$Genre[i],', '))
  for (j in 1:length(temp)){
    genre[n]=temp[j]
    n=n+1
  }
}
index=duplicated(genre) # remove repeated types
genre=genre[!index]
genre=genre[genre!='N/A'] # remove 'N/A' type
library(hash)
genre_hash = hash(keys=genre, values=1:length(genre)) # a dictionary for all ty
pes
for (i in 1:nrow(df)) {
  temp_vector = vector(length=length(genre), mode='numeric') # vector [0,0,
0...]
  if (df$Genre[i]!='N/A') {
    temp = unlist(strsplit(df$Genre[i],', '))
    for (j in 1:length(temp)) {
        temp_vector[values(genre_hash,temp[j])] = 1
    }
  }
  df$genre_vector[i]=paste(temp_vector,collapse='')
}
df$genre_vector = unlist(df$genre_vector)
```

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 pr
oportions
# Find non-repeated genres
unique_genre_vector = df$genre_vector[!duplicated(df$genre_vector)]
# Find top 10 common genre
common_genre_count = c()
for (i in 1:length(unique_genre_vector)) {
  common_genre_count[i] = length(df$Title[df$genre_vector==unique_genre_vector
[i]])
}
common_genre_count_hash = hash(keys=common_genre_count, values=unique_genre_vec
tor)
sorted_common_genre_count = sort(common_genre_count, decreasing=TRUE)
top_common = c()
for (i in 1:11) {
  temp = as.character(sorted_common_genre_count[i])
  top_common = c(top_common,common_genre_count_hash[[temp]])
}
# top_common[7]='' due to no genre was input, so delete it
top_common = top_common[-7]
# Switch genre indicator format from '0000...' to c(0,0,0,0...), and output gen
re
reverse_genre_hash = hash(keys=values(genre_hash), values=keys(genre_hash))
vector_to_genre = function(str) {
  genre = c()
  temp = as.numeric(unlist(strsplit(str,split='')))
  for (i in 1:length(temp)) {
    if (temp[i]==1) genre = c(genre,values(reverse_genre_hash, as.character
(i)))
  }
  genre = paste(genre,collapse=', ')
  return (genre)
}
# Proportions of top10 commmon genre movies
top_common_proportion = c()
for (i in 1:length(top_common)) {
  top_common_proportion[i] = length(df$genre_vector[df$genre_vector==top_common
[i]])/nrow(df)
}
top_common_name = c()
for (i in 1:length(top_common)) {
  top_common_name[i] = vector_to_genre(top_common[i])
}
temp_df=data.frame(top_common_name, top_common_proportion)
ggplot(temp_df, aes(x=top_common_name,y=top_common_proportion))+
  geom_bar(stat = "identity",fill="red")+
  xlab("Genre")+
```

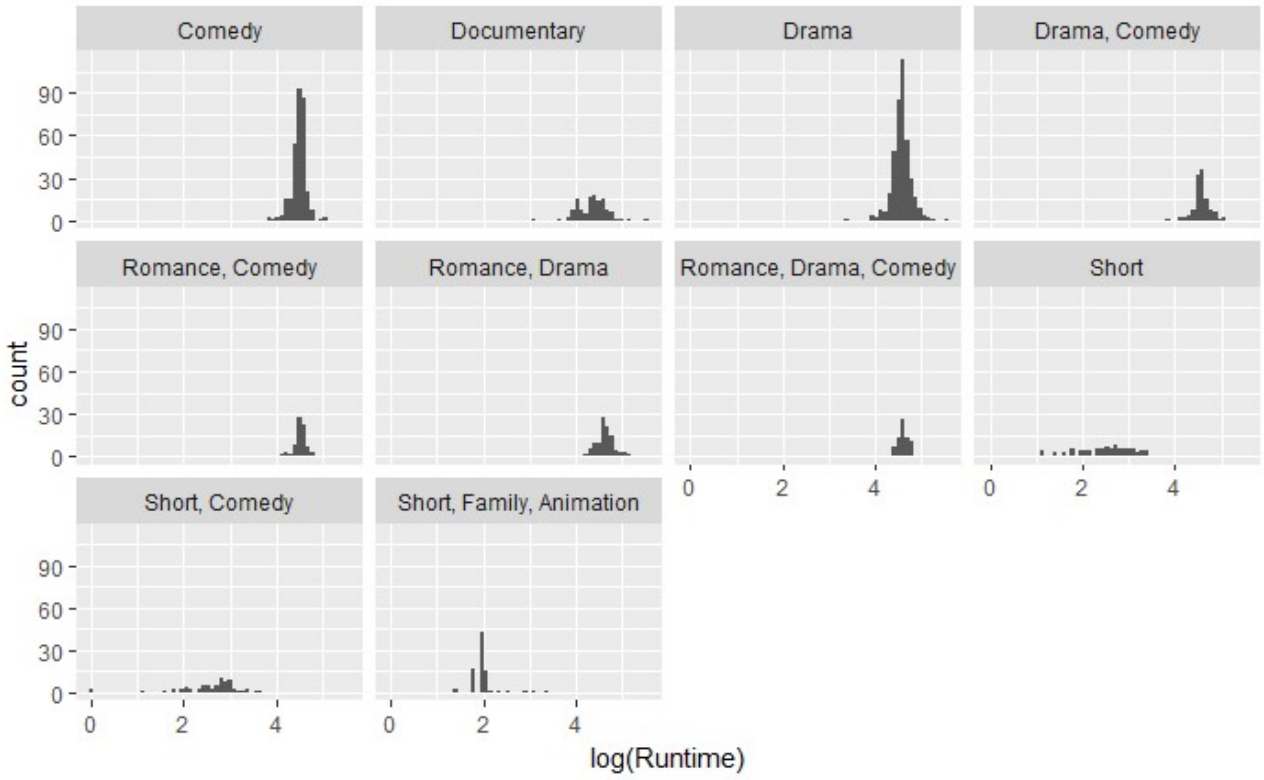
```
  ylab("Percentage")+
  theme(axis.text.x=element_text(face="bold",size=10,angle=90))
```



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

Hide

```
standard_genre=c()
for (i in 1:nrow(df)) {
  standard_genre[i] = vector_to_genre(df$genre_vector[i])
}
df$standard_genre = standard_genre
for (i in 1:nrow(df)) {
  if (df$genre_vector[i] =='00000000000000000000000000') df$standard_genre[i]
='N/A'
}
# Create multiple columns for each genre
for (i in 1:length(keys(genre_hash))) {
  df[,keys(genre_hash)[i]]=0
}
for (i in 1:nrow(df)) {
  temp = unlist(strsplit(df$standard_genre[i],split=', '))
  for (j in 1:length(temp)) {
    df[i,temp[j]]=1
  }
}
# TODO: Plot Runtime distribution for top 10 most common genres
temp = subset(df, is.na(Runtime)==0 & standard_genre==top_common_name)
qplot(x=log(Runtime), data = temp, bins=60, facets = ~ standard_genre)
```



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

A: From the Genre vs. Percentage plot, we see the Drama and Comedy have much higher percentage than other genres in the top 10 movie genres. In the log(Runtime) distributions, it is obvious to find both Comedy and Drama have high counts. Most genres has the median Runtime at ~90-100 min except Short, Short/Comedy, and Short/Family/Animation. Short genre doesn't have an obvious peak but spreads from 3 to 30 min. Short/Comedy has a small peak at ~20 min. Short/Family/Animation has an peak at ~7 min.

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). The first sourceâs release time was represented by the column `Year` (numeric representation of the year) and the second by the column `Released` (string representation of release date).

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

Hide

```
# TODO: Remove rows with Released-Year mismatch
year_indicator = rep(1, times=nrow(df))
for (i in 1:nrow(df)) {
  if (is.na(df$Released[i])==0) {
    release_year = unlist(strsplit(as.character(df$Released[i]),split='-'))[1]
    year = as.character(df$Year[i])
    if (release_year != year) year_indicator[i]=0
  }
}
df$year_indicator = year_indicator
df_ry_mismatch = subset(df, year_indicator==1)
nrow(df_ry_mismatch)
```

```
[1] 34273
```

Q: What is your precise removal logic and how many rows did you end up removing?

A: I applied string split to each item in the "Releasd" column and extracted the year, which is compared with the item in "Year" column. If two year strings are the same, I assume the merging is correct and mark 1 for this movie in the year_indicator; if two year strings are different, I assume the merging is incorrect and mark 0 in the year_indicator. Removing these wrong merging items was done by selecting the "df" with year_indicaotr of 1. After removing, there are 34273 rows.

5. Explore `Gross` revenue

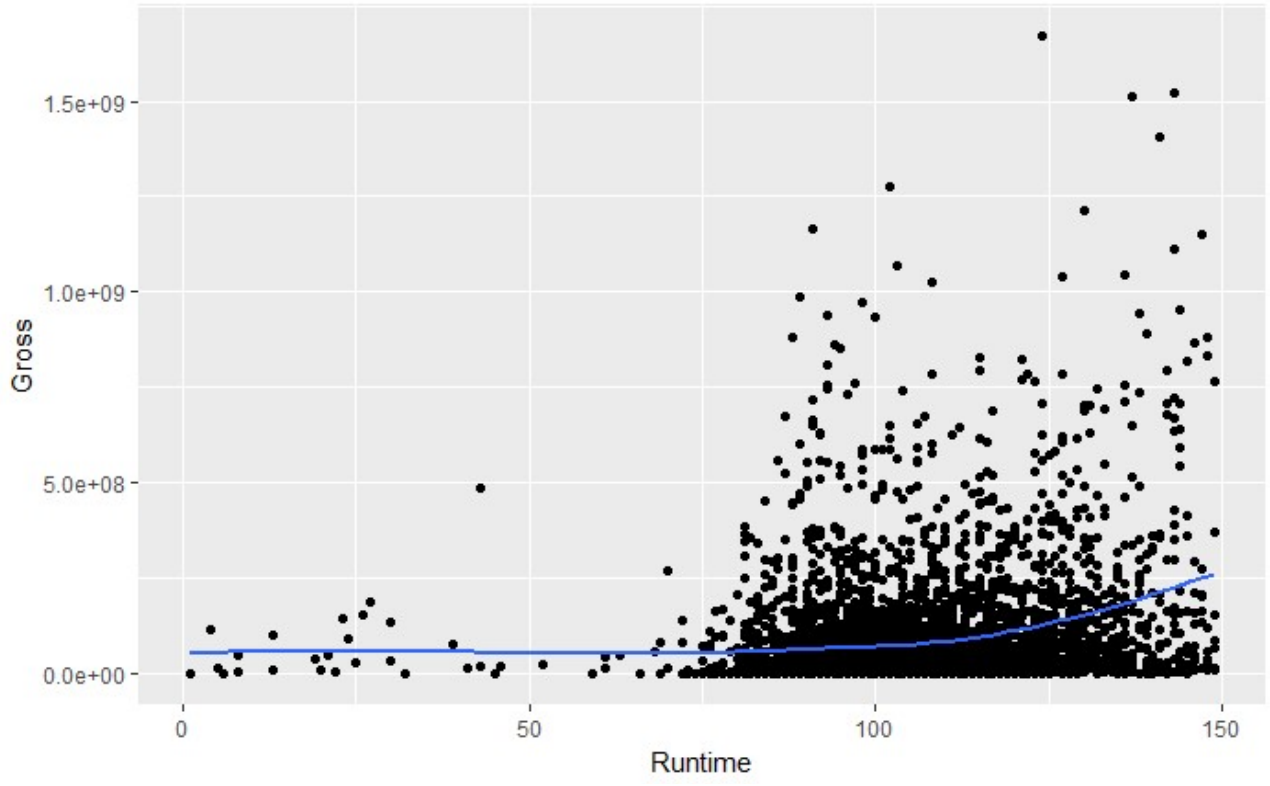
For the commercial success of a movie, production houses want to maximize Gross revenue.

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.

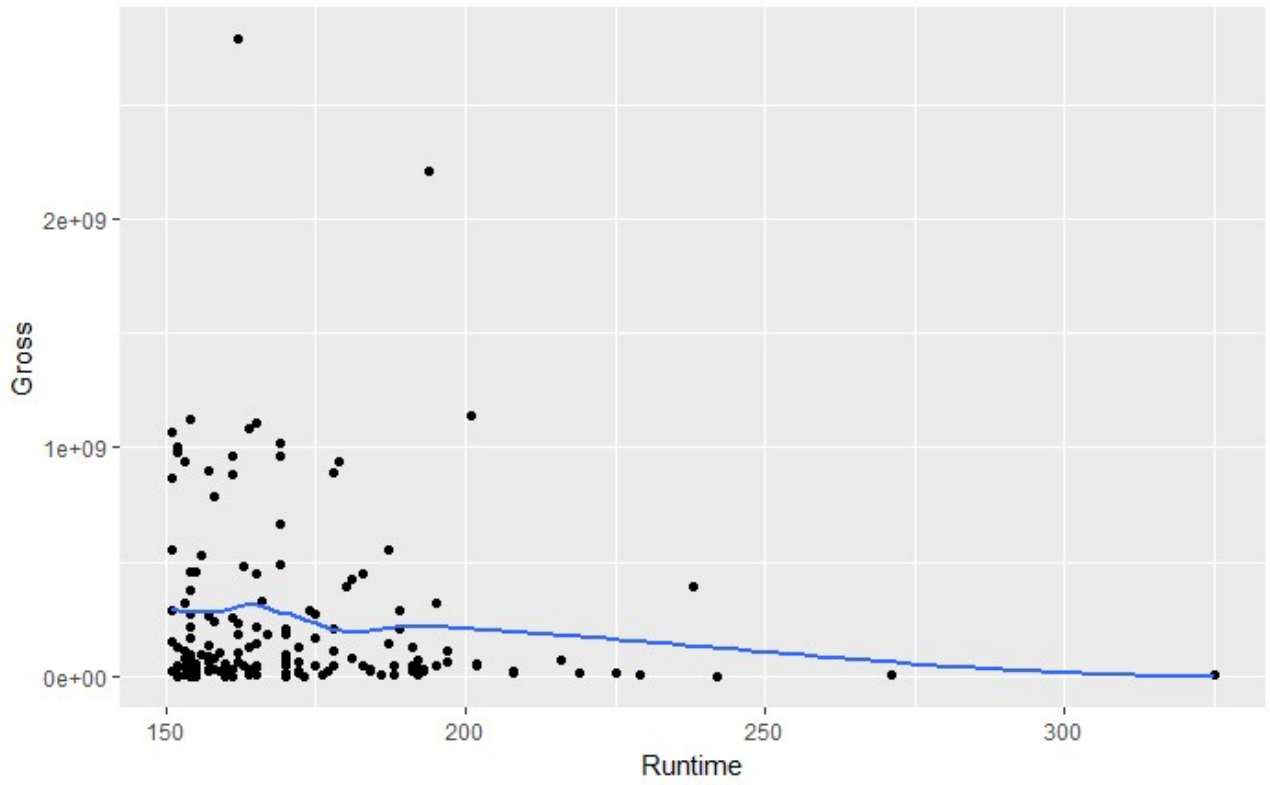
Hide

```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
t=subset(df,is.na(Gross)==0 & Gross!=0)
# Runtime vs. Gross. The movies are divided into short (<=150 min) and long (>1
50 min) movies
qplot(x=Runtime, y=Gross, data=subset(t,t$Runtime<150))+stat_smooth(se=F)
```



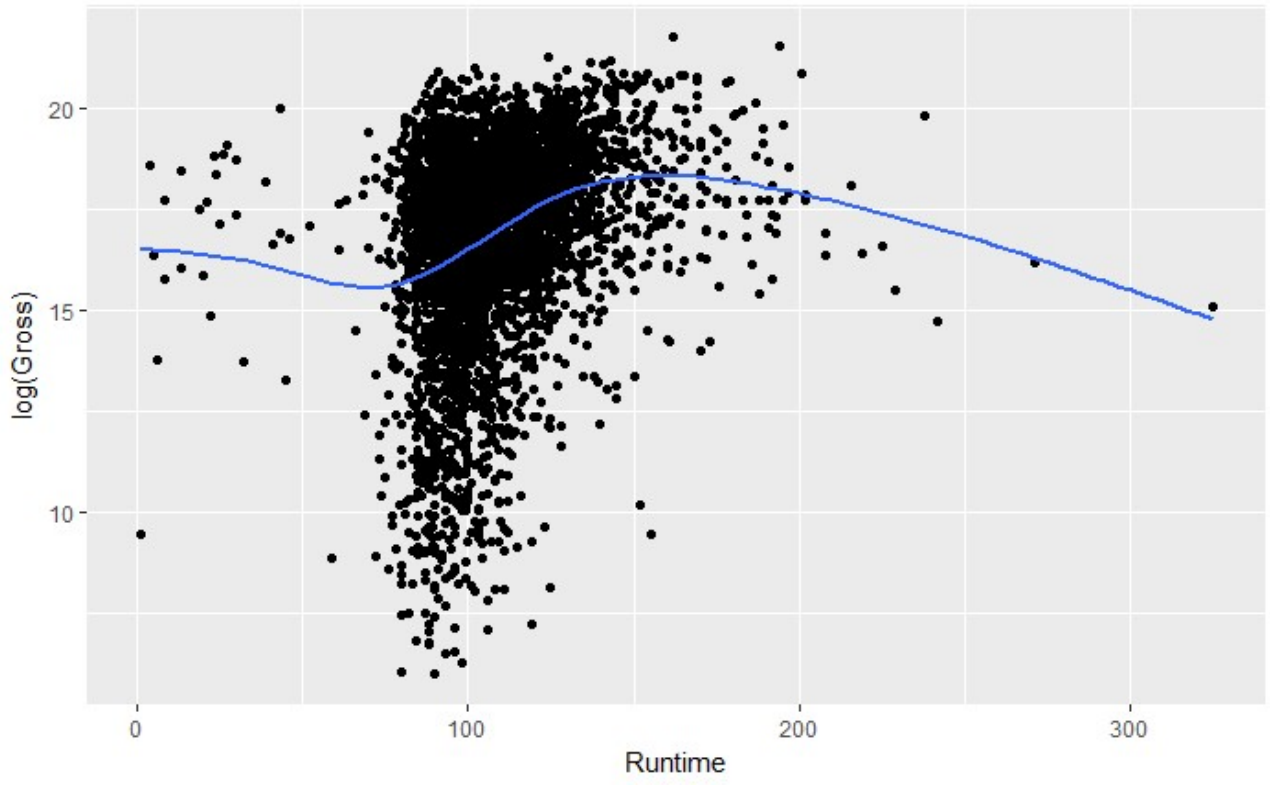
Hide

```
qplot(x=Runtime, y=Gross, data=subset(t,t$Runtime>150))+stat_smooth(se=F)
```



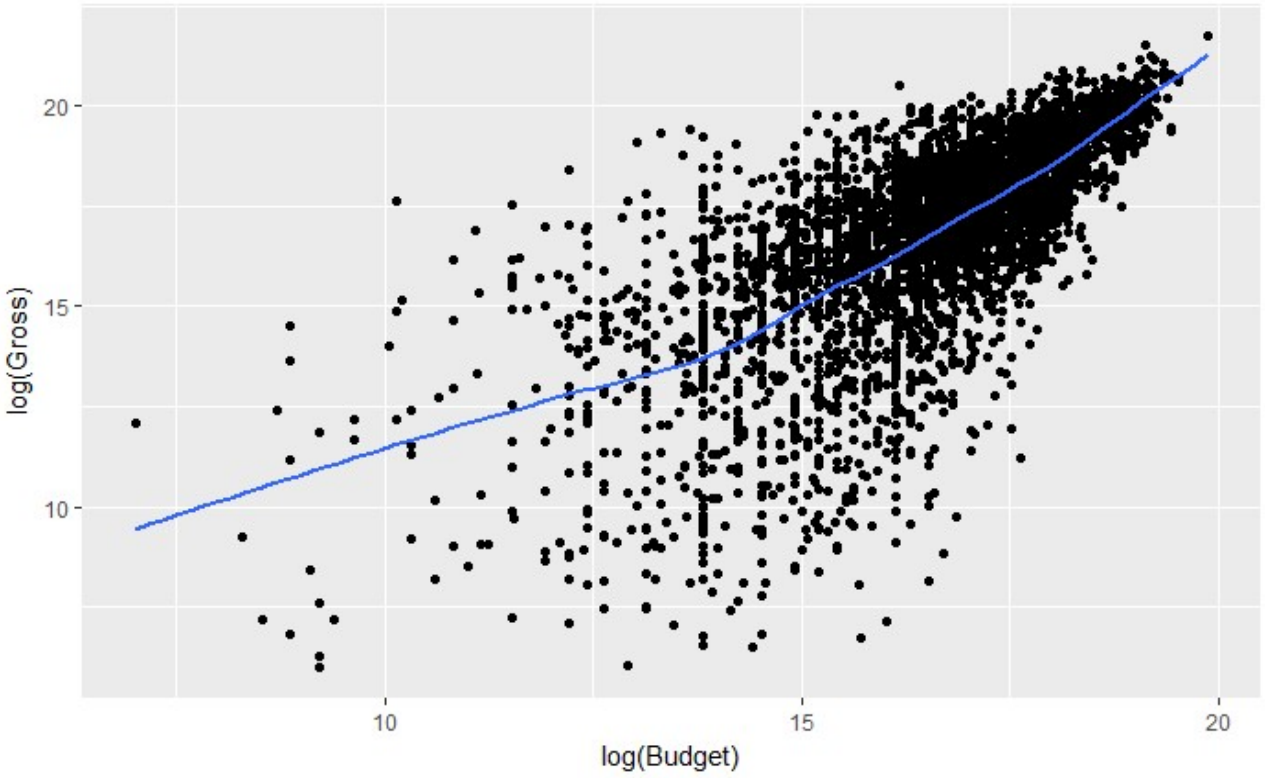
Hide

```
qplot(x=Runtime, y=log(Gross), data=t)+stat_smooth(se=F)
```



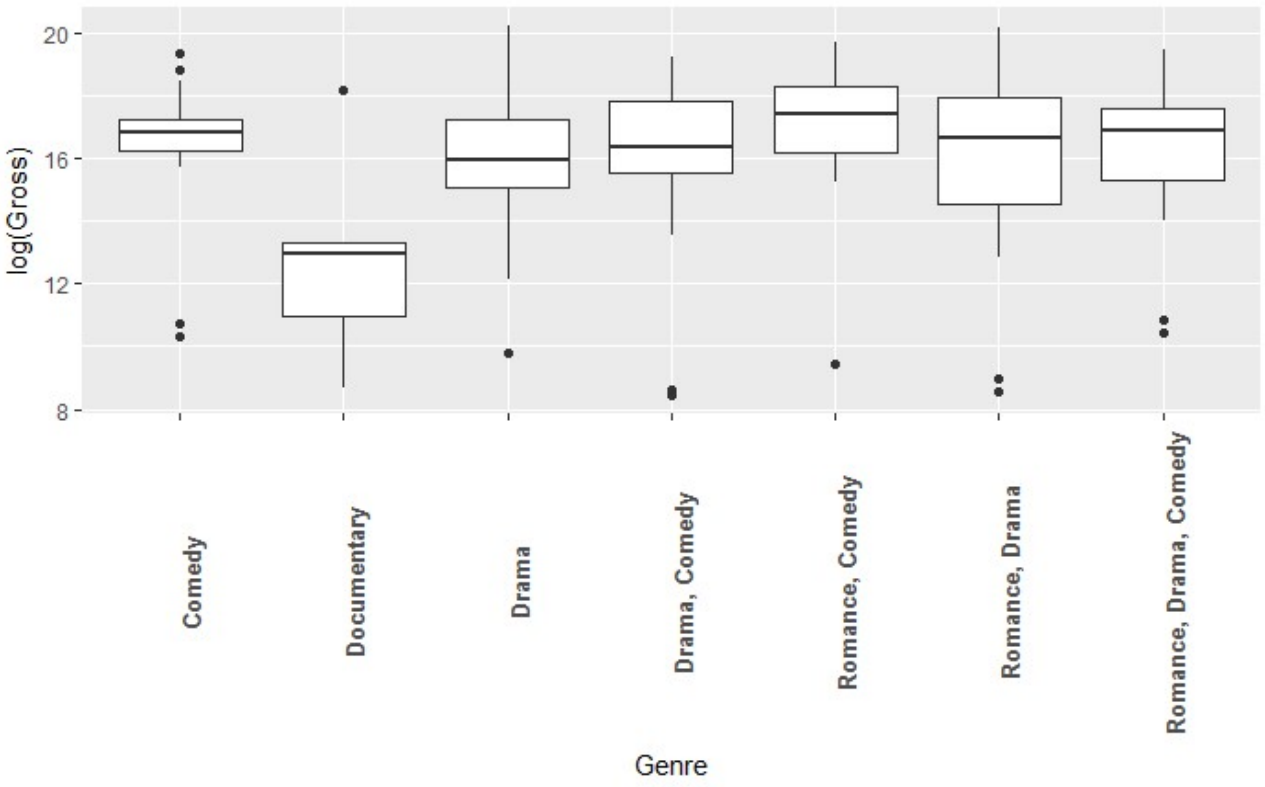
Hide

```
# Budget vs. Gross
qplot(x=log(Budget), y=log(Gross), data=t)+stat_smooth(se=F)
```



Hide

```
# Genre vs. Gross
t1=subset(df, is.na(Gross)==0 & Gross!=0 & standard_genre==top_common_name)
ggplot(t1, aes(x=standard_genre, y=log(Gross))) + geom_boxplot()+
  xlab("Genre")+
  ylab("log(Gross)")+
  theme(axis.text.x=element_text(face="bold",size=10,angle=90))
```



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.

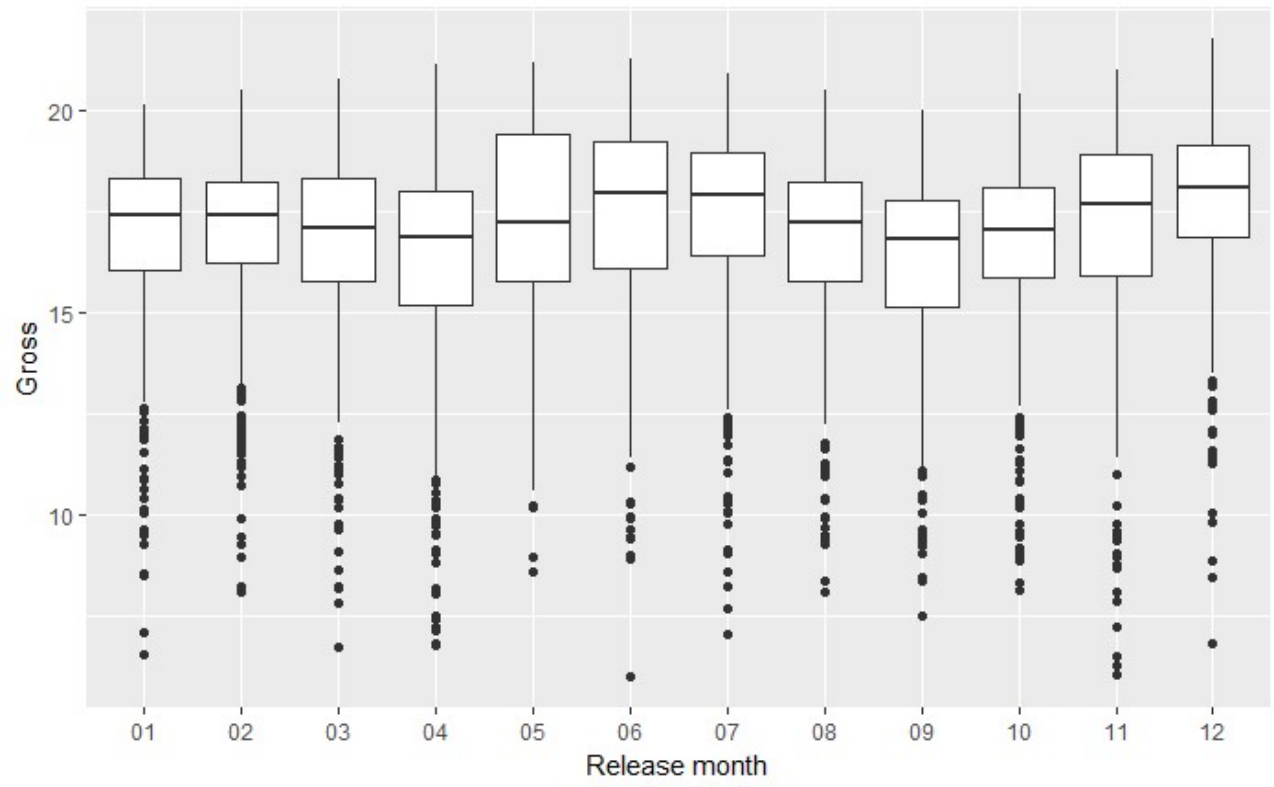
A: The movies are divided into short (<150 min) and long (<150 min) movies. From the two plots, we can see for short movies the Gross gradually increases with the increase of Runtime, while for long movies the Gross decreases with the increase of Runtime. From the log-scale plot, we see more details, when movies' Runtime is shorter than 60 min, the Gross decreases a bit but the data points are less. The log(Gross) spread width becomes narrower from 60 min upto 150 min.

From the log(Budget) vs. log(Gross) plot, it is clear that the Gross increases with the increase of Budget. The spread width of log(Gross) decreases when log(Budget) increases.

documentary tpye has hte lowest gross.

Hide

```
# TODO: Investigate if Gross Revenue is related to Release Month
release_month = c()
for (i in 1:nrow(t)) {
  release_month[i] = unlist(strsplit(as.character(t$Released[i]),split='-'))[2]
}
t$release_month = release_month
t1=subset(t,is.na(release_month)==0)
ggplot(t1, aes(x=release_month,y=log(Gross)))+geom_boxplot()+
  xlab("Release month")+
  ylab("Gross")
```



It is clear to see movies realsed during the summer (June, July) and end year time (December) have higher median Gross. Gross in May has higher spread than other months.

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 that 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
wins = c()
nominations = c()
for (i in 1:nrow(df)) {
  temp_wins=0
  temp_nominations=0
  if (df$Awards[i] !='N/A') {
    temp = unlist(strsplit(df$Awards[i],' '))
    j=1
    for (j in 1:length(temp)) {
      if (temp[j] == 'wins' | temp[j] == 'win' | temp[j] =='wins.') temp_wins
= temp_wins+as.numeric(temp[j-1])
      if (temp[j] == 'Won') temp_wins = temp_wins+as.numeric(temp[j+1])
      if (temp[j] == 'nominations.' | temp[j] == 'nomination.') temp_nomination
s = temp_nominations+as.numeric(temp[j-1])
      if (temp[j] == 'for') temp_nominations = temp_nominations+as.numeric(temp
[j+1])
    }
  }
  wins = c(wins,temp_wins)
  nominations = c(nominations, temp_nominations)
}
df$wins = wins
df$nominations = nominations
t2 = subset(df, wins!=0 | nominations!=0)
nrow(t2)
```

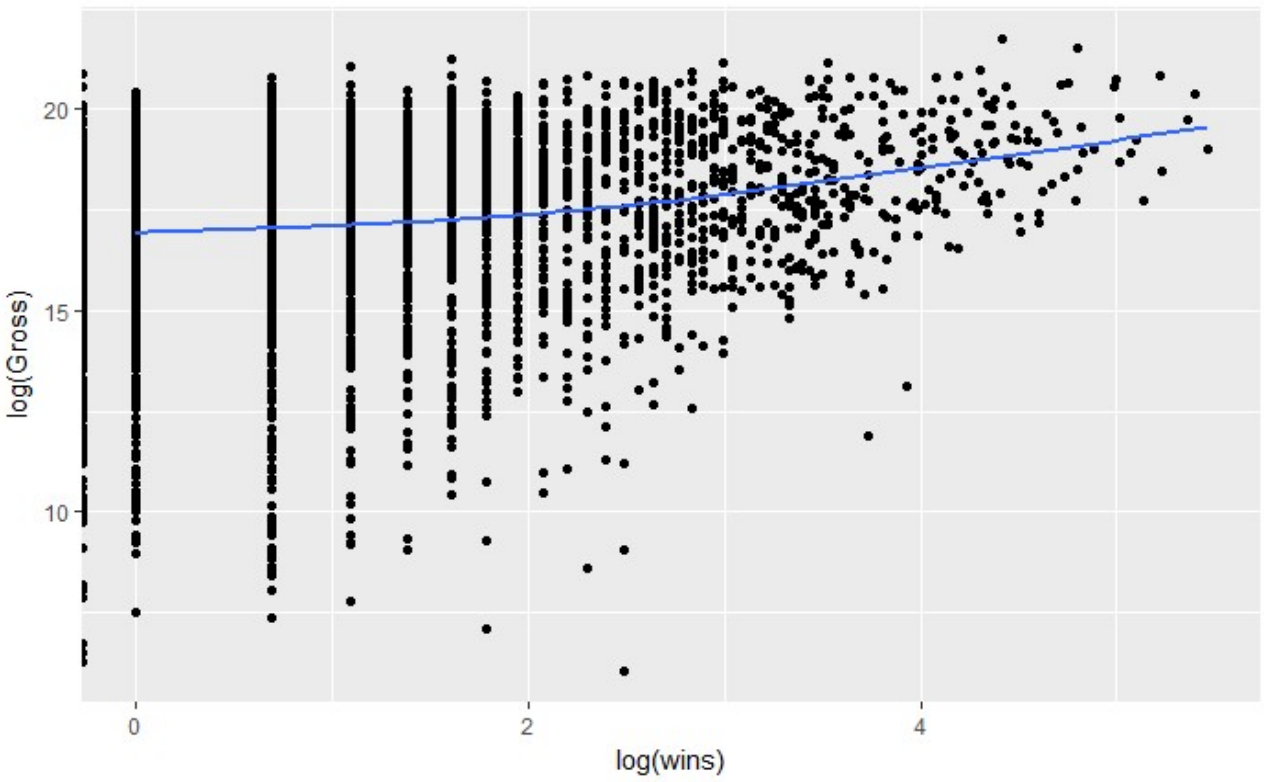
```
[1] 12780
```

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

A: For each Award item, I first check if it is ‘N/A’, if not, I split and unlist the string, so I can get the charactor array for each item. Then I check the key words in this array. For example, for the number of wins, the first case is if there is ‘wins’ or ‘win’ in the array, I can extract the number before this key word. The second case is if there is “won” key word, I can extract the number after it. When there is no record for Awards. Both “wins” and “nominations” will be zero. The non-zero “wins”" or “nominations” can be selected by setting the condition “wins!=0 | nominations!=0”. There are 12780 rows with non-zero “wins”" or “nominations”.

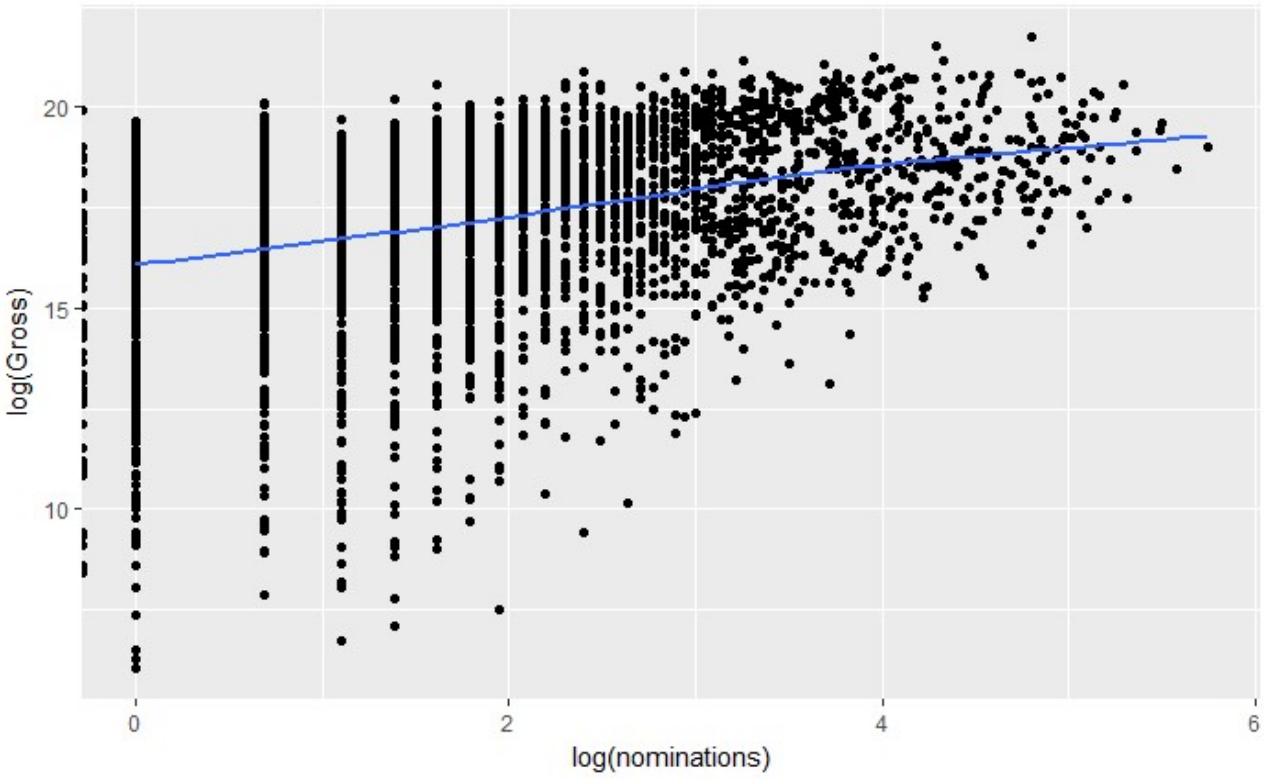
Hide

```
# TODO: Plot Gross revenue against wins and nominations
t3 = subset(t2,is.na(Gross)==0 & Gross!=0)
qplot(x=log(wins), y=log(Gross), data=t3)+stat_smooth(se=F)
```



Hide

```
qplot(x=log(nominations), y=log(Gross), data=t3)+stat_smooth(se=F)
```



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

A: From the log(wins) vs. log(Gross) plot, gross revenue increases and the distribution width decreases with the number of awards won and nominations received.

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)).

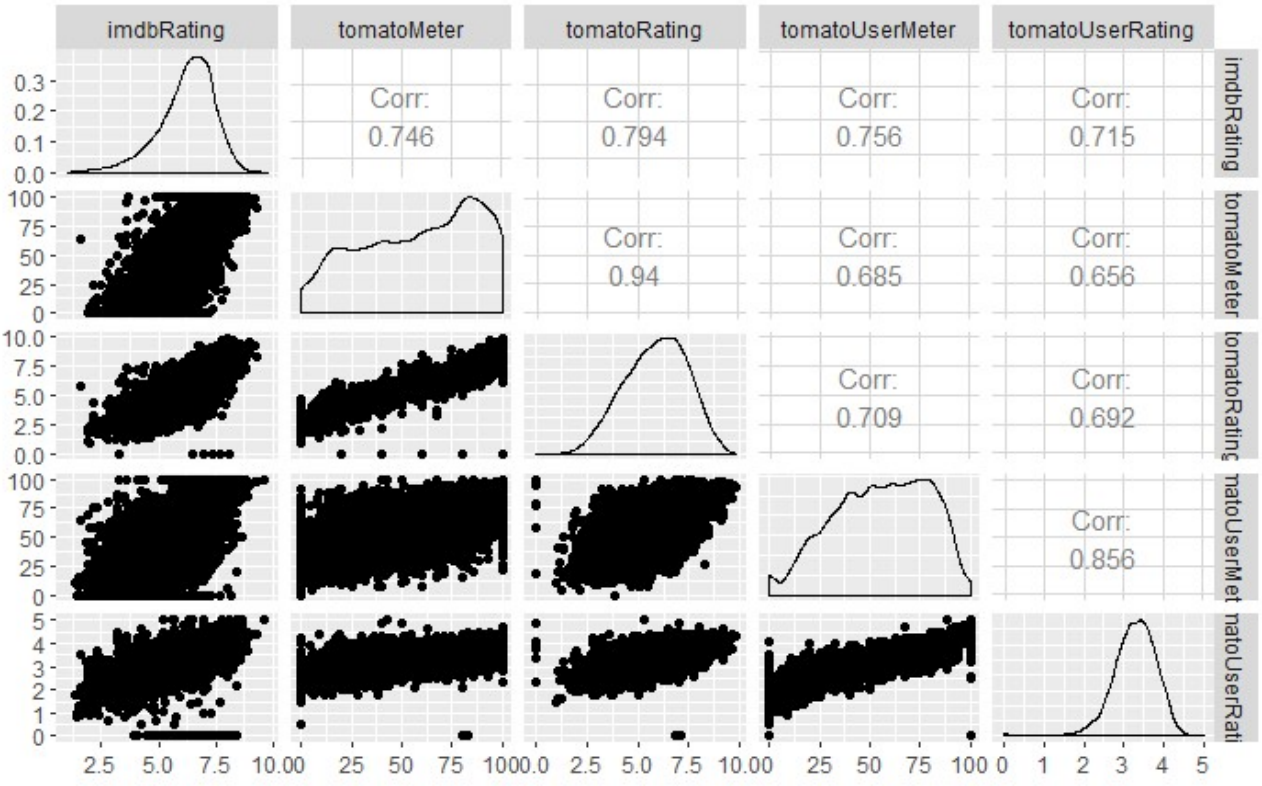
Investigate the pairwise relationships between these different descriptors using graphs.

Hide

```
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
pair = df[,c('imdbRating','tomatoMeter','tomatoRating","tomatoUserMeter", "toma
toUserRating")]
ggpairs(pair)
```

```
 plot: [1,1] [===-------------------------------------------------------------
------]  4% est: 0s
 plot: [1,2] [======----------------------------------------------------------
------]  8% est: 1s
 plot: [1,3] [=========-------------------------------------------------------
------] 12% est: 1s
 plot: [1,4] [==========------------------------------------------------------
------] 16% est: 1s
 plot: [1,5] [=============---------------------------------------------------
------] 20% est: 1s
 plot: [2,1] [================----------------------------------------------
------] 24% est: 1s
 plot: [2,2] [====================------------------------------------------
------] 28% est: 1s
 plot: [2,3] [=======================---------------------------------------
------] 32% est: 1s
 plot: [2,4] [===========================-----------------------------------
------] 36% est: 1s
 plot: [2,5] [==============================--------------------------------
------] 40% est: 1s
 plot: [3,1] [=================================-----------------------------
------] 44% est: 1s
 plot: [3,2] [====================================--------------------------
------] 48% est: 1s
 plot: [3,3] [=======================================-----------------------
------] 52% est: 1s
 plot: [3,4] [==========================================------------------
------] 56% est: 1s
 plot: [3,5] [=============================================---------------
------] 60% est: 1s
 plot: [4,1] [================================================------------
------] 64% est: 1s
 plot: [4,2] [====================================================--------
------] 68% est: 0s
 plot: [4,3] [=======================================================-----
------] 72% est: 0s
 plot: [4,4] [==========================================================--
------] 76% est: 0s
 plot: [4,5] [=============================================================
------] 80% est: 0s
 plot: [5,1] [=============================================================
------] 84% est: 0s
 plot: [5,2] [=============================================================
------] 88% est: 0s
 plot: [5,3] [=============================================================
------] 92% est: 0s
 plot: [5,4] [=============================================================
```

```
======---] 96% est: 0s
 plot: [5,5] [=================================================================
======]100% est: 0s
```



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

A: 1. From the plot we can see tomatoMeter vs. tomatoRating and tomatoUserRating vs. tomatoUserRating have linear relationships and their correlation correlation coefficients are 0.84 and 0.856 respectively. 2. It is also seen that imdbRaing increases with the tomatoRating somewhat linearly and the correlation coefficient is 0.794. 3. Comparing tomatoRating and tomatoUserRating, we see although they are in linear-like relationship, tomatoUserRating is usually lower than tomatoRating. This means users tend to give lower rating than critics. 4. Comparing imdbRating with tomatoUserRating, we find imdbRaing also increases with tomatoUserRating in a linear-like behavior, but compared with 2, data points of tomatoUserRating spread more widely at the same imdbRating point.

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.

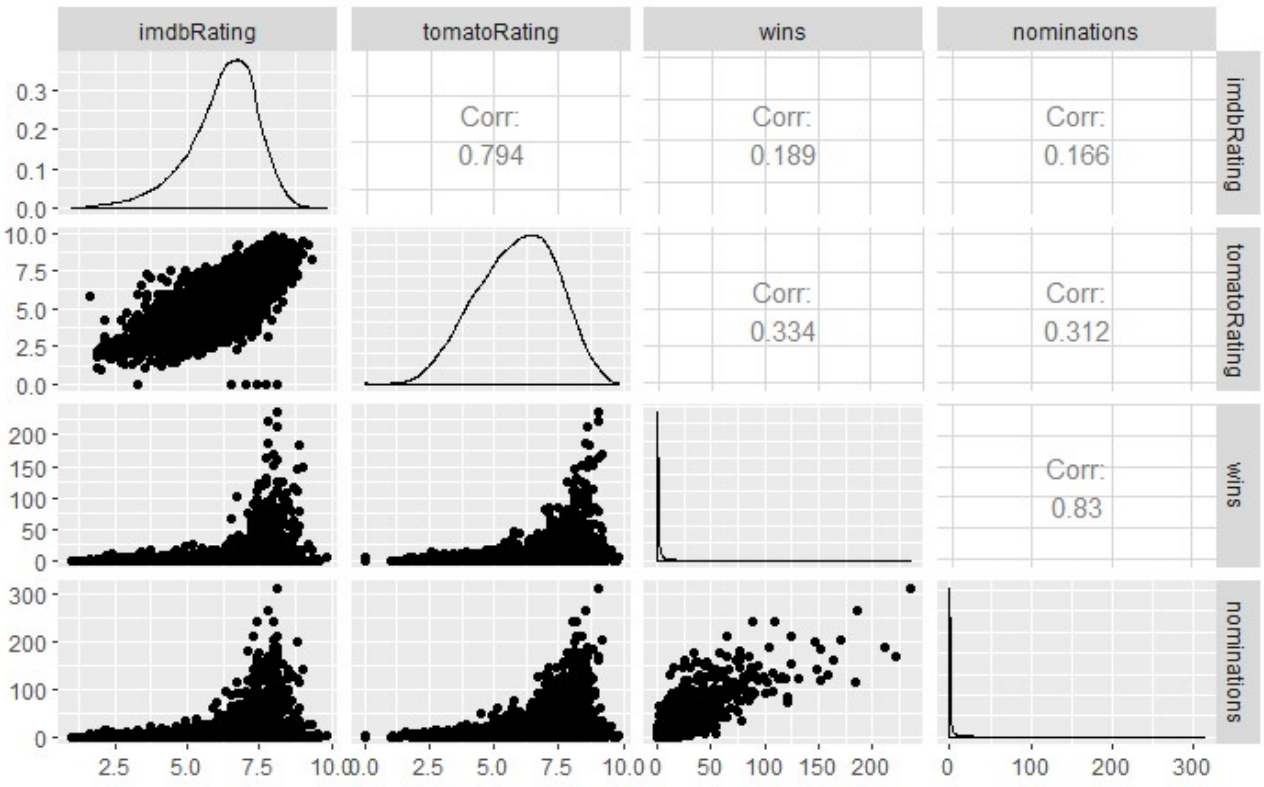
Study the relationship between ratings and awards using graphs (awards here refers to wins and/or

nominations).

Hide

```
# TODO: Show how ratings and awards are related
win_nomination = df[,c('imdbRating','tomatoRating','wins','nominations')]
ggpairs(win_nomination)
```

```
 plot: [1,1] [====------------------------------------------------------
------]  6% est: 0s
 plot: [1,2] [=========-----------------------------------------------------
------] 12% est: 0s
 plot: [1,3] [==============--------------------------------------------------
------] 19% est: 0s
 plot: [1,4] [==================-----------------------------------------------
------] 25% est: 1s
 plot: [2,1] [=======================-------------------------------------------
------] 31% est: 1s
 plot: [2,2] [============================----------------------------------------
------] 38% est: 1s
 plot: [2,3] [================================------------------------------------
------] 44% est: 1s
 plot: [2,4] [====================================----------------------------
------] 50% est: 0s
 plot: [3,1] [=========================================-------------------------
------] 56% est: 0s
 plot: [3,2] [=============================================-------------------
------] 62% est: 0s
 plot: [3,3] [=================================================----------------
------] 69% est: 0s
 plot: [3,4] [=====================================================-----------
------] 75% est: 0s
 plot: [4,1] [==========================================================-------
------] 81% est: 0s
 plot: [4,2] [==============================================================---
------] 88% est: 0s
 plot: [4,3] [==================================================================
==----] 94% est: 0s
 plot: [4,4] [==================================================================
======]100% est: 0s
```



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?

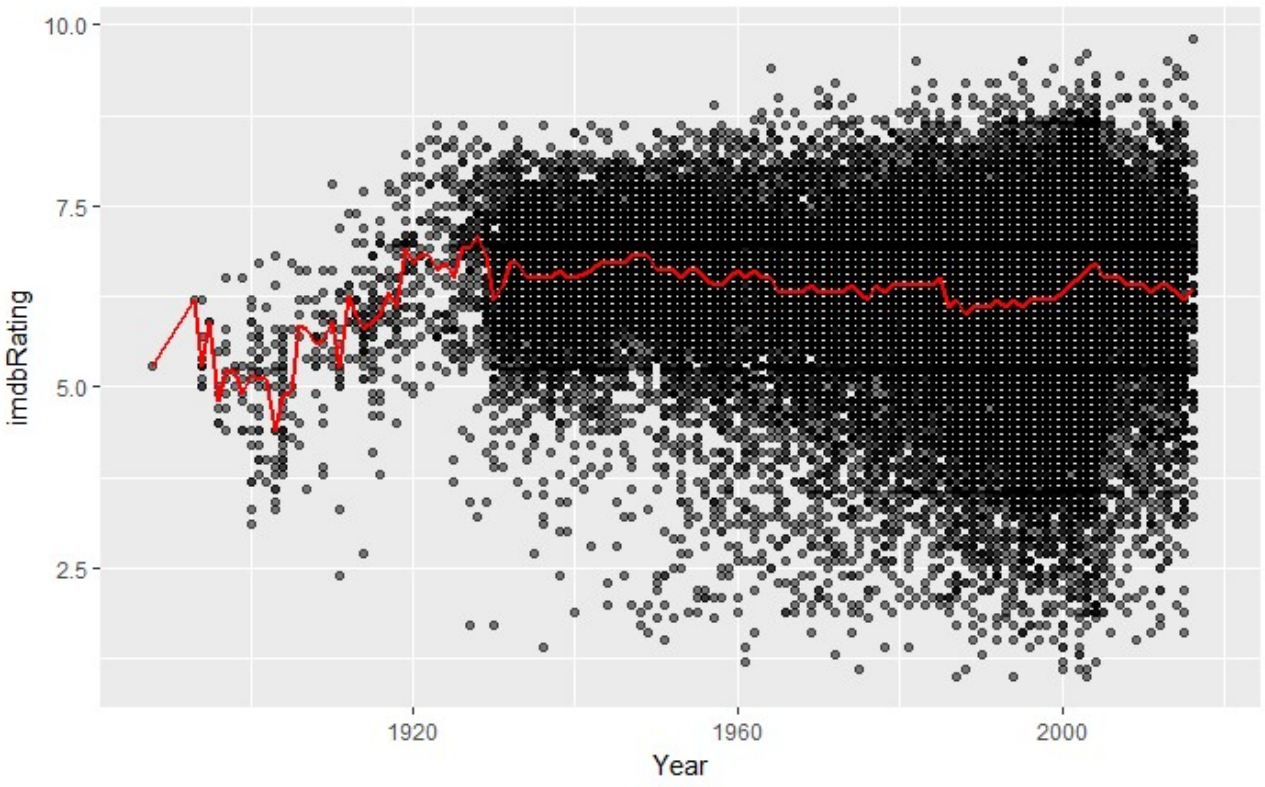
A: From the plot we see both imdbRating and tomatoRating can povide a prediction to wins and nominations to a centain extent because both their correlation coefficients are not high. But wins/nomination vs. tomoatoRatnig has higher corr. From the plot we notice at a certain range of rating, the corr may become higher enough to make a good prediction. Movies with high number of wins/nomiations are more likely in mid-high rating range. But in this same range, there are also lots of movies with less wins/nomiations. Rating close to 10 is very few since very few people would give a movie a full score.

9. Expected insights

Come up with two new insights (backed up by data and graphs) that is expected. Here ânewâ 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.

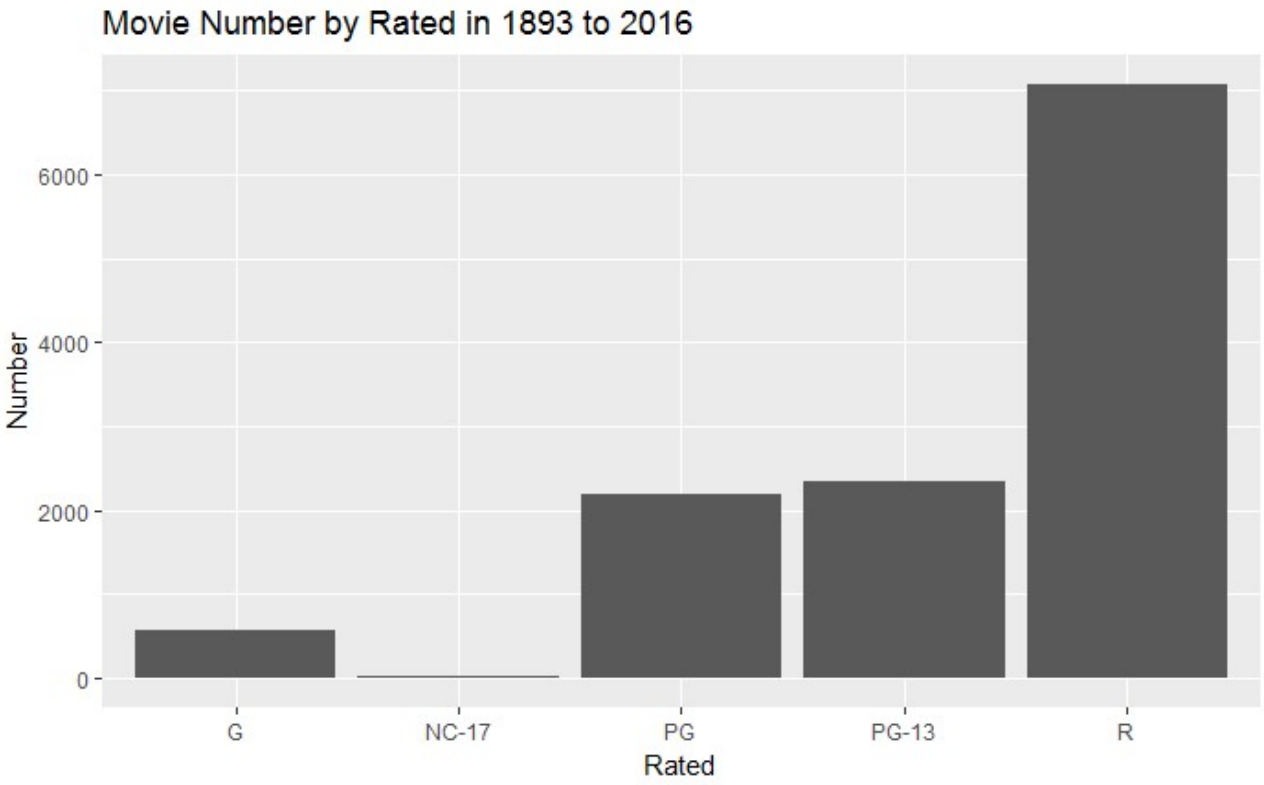
Hide

```
# TODO: Find and illustrate two expected insights
# Year vs. imdbVotes
ggplot(df, aes(x=Year,y=imdbRating))+geom_point(alpha=0.5)+
  geom_line(stat="summary",fun.y='median',col=2,lwd=1)
```



Hide

```
# Movie amount by rated
x=c('R','PG-13','PG','G','NC-17')
y=c()
for (i in 1:length(x)) {
 y[i] = sum(df$Rated==x[i])
}
number=data.frame(x,y)
ggplot(number,aes(x=x,y=y))+geom_bar(stat = "identity")+
 xlab("Rated") + ylab("Number") +ggtitle("Movie Number by Rated in 1893 to 201
6")
```



Q: Expected insight #1.

A: In the Year vs. imdbRating plot, the median line (red) is plotted. From 1900 to 1930 the median rating increases becuase moive is in its early develop stage and soundless. I imagine people don't have high requirements on these movies but moive' progress is obvious from year to year during this period, so the median goes up. From 1930 to 1990, we see a slowly downward of the rating median. During this period, many classic sound movie appeared and becomes classic today so they have a high rating. But people's expectation become higher and higher, so the overall trend is going down slowly. After 1990, technologies such as computer graphics and special effects made moives more attarctive and highly rated.

Q: Expected insight #2.

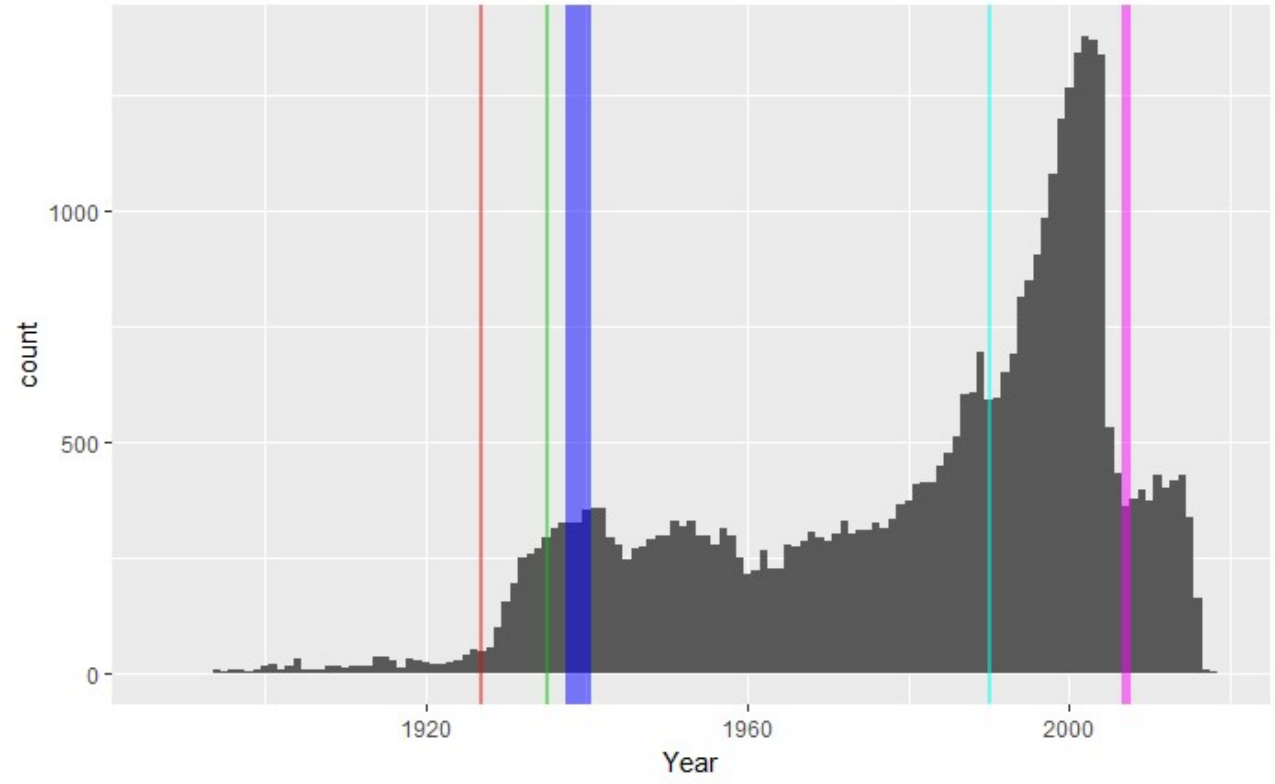
A: From the Rated vs. Number plot we see the amount of movies in different genres. NC-17 has the least number due to it is very restrictive and only open to adults. R has the most amount since it includes most virious moive topics and has most audiences from teen to adults. Although other three genres also open to large amount of audience, the movie topics or plot may not be as attractive and excited as R movies.

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(df,aes(x=Year))+
  geom_histogram(binwidth=1)+
  geom_vline(xintercept=1927,lwd=1,col=2,alpha=0.5)+
  geom_vline(xintercept=1935,lwd=1,col=3,alpha=0.5)+
  geom_vline(xintercept=1939,lwd=5,col=4,alpha=0.5)+
  geom_vline(xintercept=1990,lwd=1,col=5,alpha=0.5)+
  geom_vline(xintercept=2007,lwd=2,col=6,alpha=0.5)
```



Q: Unexpected insight.

A: We usually thought the movie growth is in a gradual trend and somewhat predictable. But in reality the interesting but unexpected insight is the movie growth is not constant due to impacts from society. Sometimes it increases suddenly and fast, but sometimes the growth becomes unchanged and even moves downside. In 1927 (red), the first sound movie was born. In 1935 (green), colored moives came into market. This impacts can possibly expalin the sudden growth from 1930 to 1940. From 1939 to 1945 (blue), WarII reduces the growth of movies and after the war the movie grew negtively. From 1980, moive market entered the high-speed development period. There is a bump at 1990 (cyan) due to US economy recession. It is not clear why after 2005 the movie growth suddenly dropped, but it is somewhat reasonable that the movie production is low during 2007-2009 (pink) US economy recession.