# Project 1: Explore and Prepare Data

yyao45@gatech.edu

*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 IMDb 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 IMDb 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 鈥渆cond window 鈥? streaming rights).

## Instructions

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

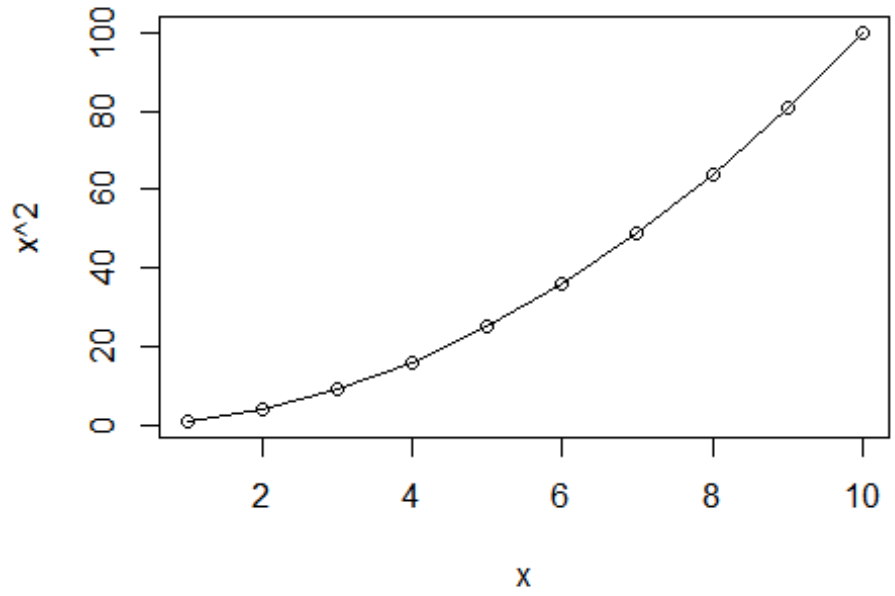
When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

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

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

Plots appear inline too:

```r
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 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)
library(reshape2)
library(plyr)
library(stringr)
```

```
library(grid)
library(gridExtra)
```

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$Runtime with a numeric column containing the runtime
 in minutes
class(df2$Runtime)
```

```
## [1] "character"
```

```
head(sort(unique(df2$Runtime)))
```

```
## [1] "1 h"        "1 h 1 min"  "1 h 10 min" "1 h 11 min" "1 h 12 min"
## [6] "1 h 14 min"
```

```
tail(sort(unique(df2$Runtime)))
```

```
## [1] "95 min" "96 min" "97 min" "98 min" "99 min" "N/A"
```

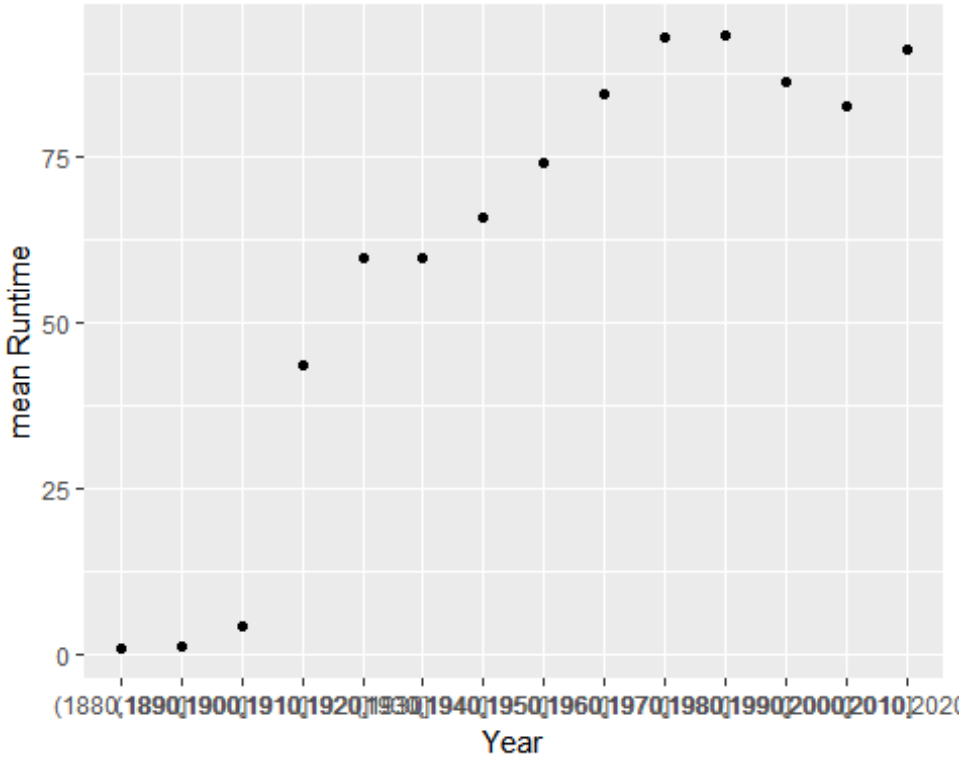
```
# there are 4 type of Runtime, 1)"N/A", 2) "XX h xx min", 3) "xx min" 4)
"xx h"

Runtime_num = c()
for(i in seq_along(df2$Runtime)){

        x = strsplit(df2$Runtime[i], ' ')[[1]]
        if (length(x) ==  2){
                if(x[2] == "min") #3) "xx min"
                {Runtime_num = c(Runtime_num, suppressWarnings(as.numer
ic(x[1])))}
                # 4)"xx h"
                else{Runtime_num <- c(Runtime_num, suppressWarnings(as.
numeric(x[1])) * 60)}
        }
        # 2) "XX h xx min"
        else if( length((x) == 4)) {
                Runtime_num = c(Runtime_num, suppressWarnings(as.numeri
c(x[1])) * 60  + suppressWarnings(as.numeric(x[3])))
        }
        # 1)"N/A"
        else Runtime_num = c(Runtime_num, NA)
}
df2$Runtime = Runtime_num
class(df2$Runtime)
```

```
## [1] "numeric"
```

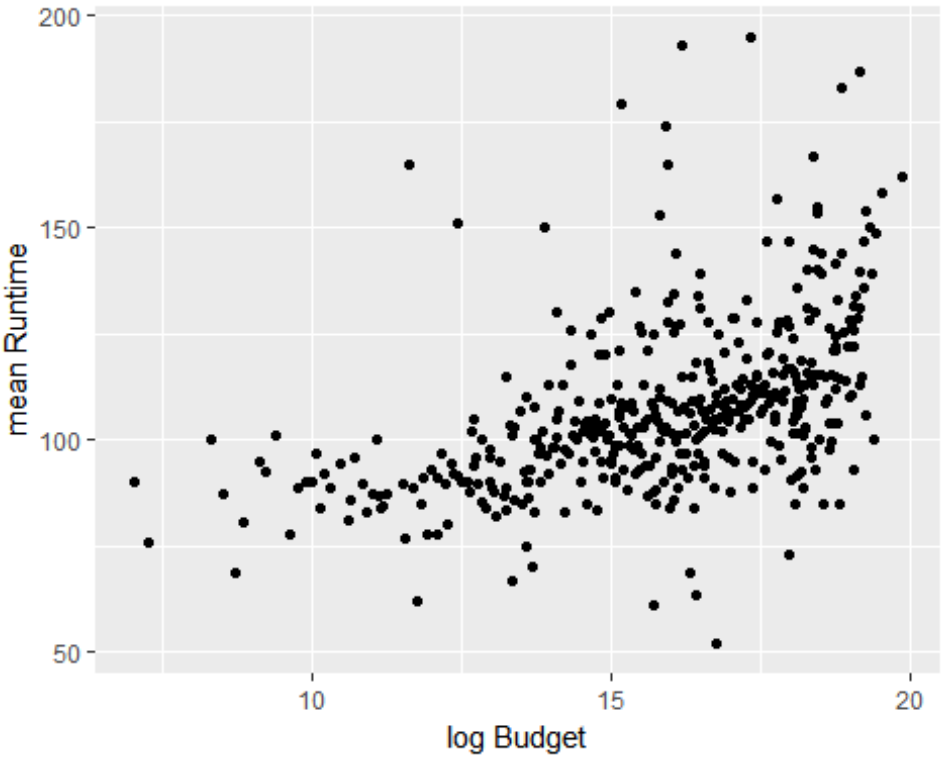
Now investigate the distribution of `Runtime` values and how it changes over years (variable `Year`, which you can bucket into decades) and in relation to the budget (variable `Budget`). Include any plots that illustrate.

```
# TODO: Investigate the distribution of Runtime values and how it varie
s by Year and Budget
# combine the row of Runtime with Year to a new dataframe
Runtime_Year= data.frame(Runtime = df2$Runtime , df2$Year)
colnames(Runtime_Year) = c("Runtime", "Year")

# remove all the NA columns of Runtime_Year_merged dataframe
Runtime_Year = Runtime_Year[complete.cases(Runtime_Year),]
```

```
Runtime_Year$YearBin = cut(Runtime_Year$Year, seq(1880, 2020, by=10), d
ig.lab=4)
R_Y = data.frame(Runtime_Year$Runtime, Runtime_Year$YearBin)
colnames(R_Y) = c("Runtime", "Year")
R_YMod = ddply(R_Y, "Year", summarise, mean.time = mean(Runtime, na.rm
= TRUE))

qplot(x = Year, y = mean.time, data = R_YMod,
      xlab = "Year", ylab = "mean Runtime")
```




```
# combine the row of Runtime with Budget to a new dataframe
Runtime_Budget= data.frame(Runtime = df2$Runtime , df2$Budget)
colnames(Runtime_Budget) = c("Runtime", "Budget")

# remove all the NA columns of Runtime_Year_merged dataframe
Runtime_Budget = Runtime_Budget[complete.cases(Runtime_Budget),]
R_BMod = ddply(Runtime_Budget, "Budget", summarise, mean.time = mean(Ru
ntime, na.rm = TRUE))

qplot(x = log(Budget), y = mean.time, data = R_BMod,
      xlab = "log Budget", ylab = "mean Runtime")
```




*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**: We could find before 1920s, the movie time are very short that less than 10 min. Then from 1920s to 1970s, the run time increased from 40 min to more than 90 min. Then the run time became stable between 90 min to 100 min. For the budget, we could find the linear relation between the mean run time and the log of Budget. And when the budget increases, the run time also become longer.

## 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
# Split individual genre in a cell by comma and convert into list
```

```
genre = unlist(strsplit(df2$Genre, ','))
# Trim whitespace
genre = gsub(' ', '', genre)
# Change to lower case
genre = tolower(genre)
# Remove punctuation
genre = gsub("[[:punct:]]", "", genre)
# Get unique set of genre
genre = unique(genre)
# List all the unique genre
print(genre)
```

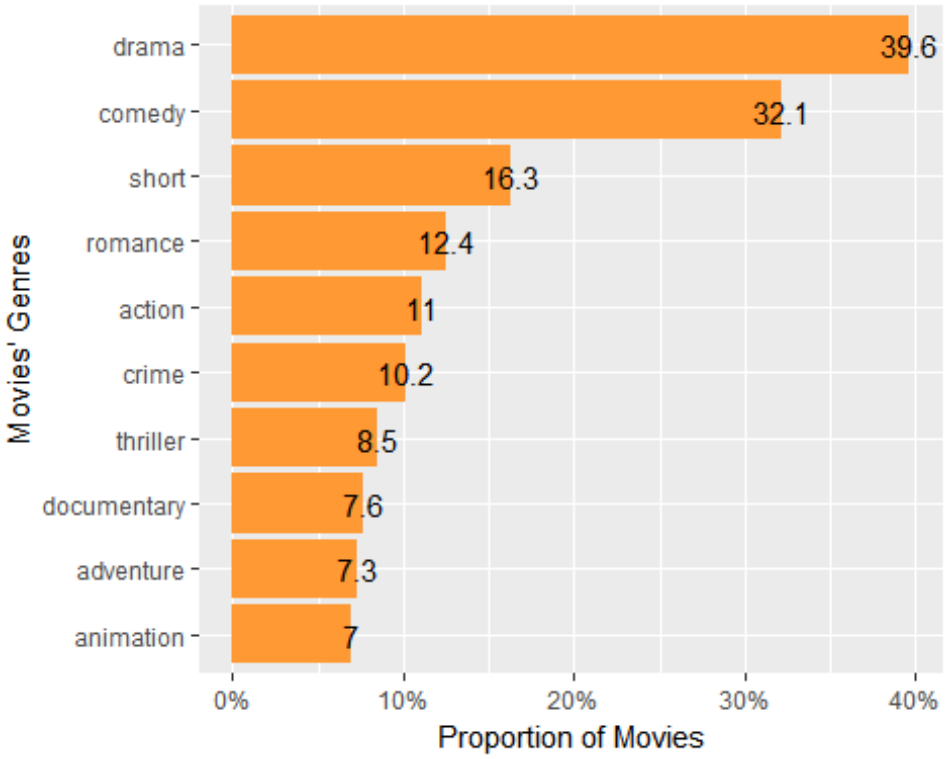
```
##  [1] "documentary" "biography"   "romance"     "short"       "thrill
er"
##  [6] "drama"       "war"         "comedy"      "horror"      "scifi"

## [11] "adventure"   "family"      "history"     "crime"       "action
"
## [16] "music"       "mystery"     "fantasy"     "sport"       "animat
ion"
## [21] "musical"     "na"          "talkshow"    "adult"       "wester
n"
## [26] "filmnoir"    "realitytv"   "news"        "gameshow"
```

```
# change Genre column under
df2$GenreModified = tolower(df2$Genre)
# trim whitespace
df2$GenreModified = gsub(" ", "", df2$GenreModified)
# replace comma by space
df2$GenreModified = gsub(",", " ", df2$GenreModified)
# remove punctuation
df2$GenreModified = gsub("[[:punct:]]", "", df2$GenreModified)
# Create binary vector for each type of genre
genre_binary_matrix = NULL
for (i in 1:length(genre)){
  g = genre[i]
  binary_v = as.numeric(grepl(paste('\\b',g,'\\b', sep=""), df2$GenreMo
dified))
  genre_binary_matrix = cbind(genre_binary_matrix, binary_v)
}
genre_binary_df = data.frame(genre_binary_matrix)
colnames(genre_binary_df) = paste("Genre_", genre, sep="")
df2 = cbind(df2, genre_binary_df)
df3 = subset(df2, select = -GenreModified)
```

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 rel
ative proportions
genreSummary = data.frame(genre=names(genre_binary_df),num_movies=colSu
ms(genre_binary_df))
```

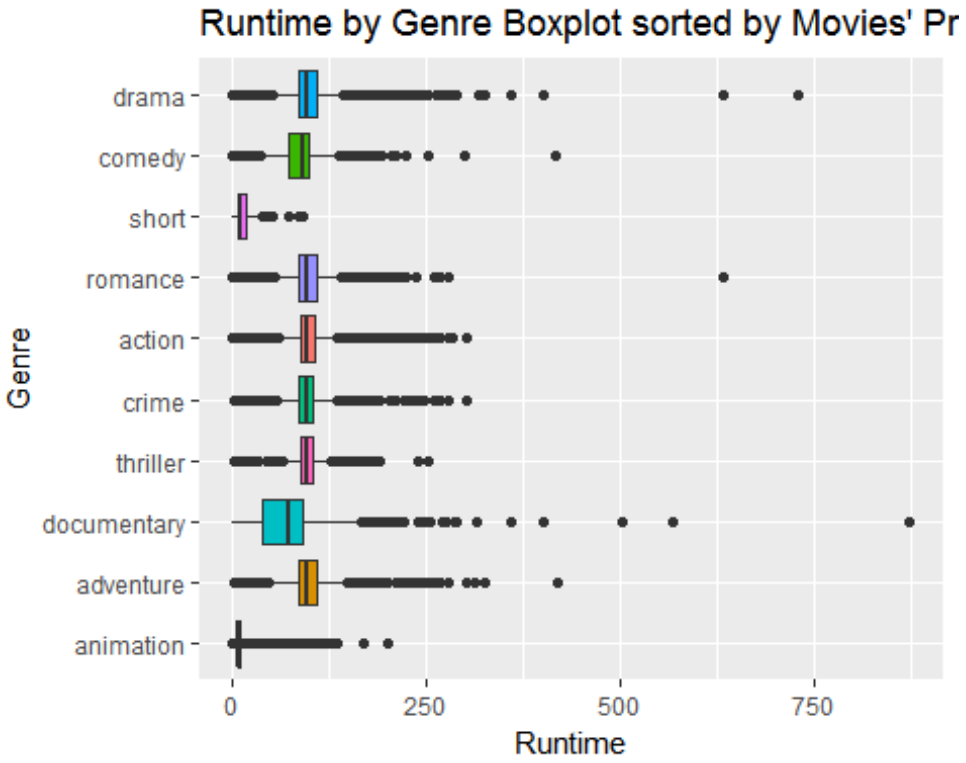
```
genreSummary$genre = gsub("Genre_", "", genreSummary$genre)
genreSummary$per_movies = genreSummary$num_movies / nrow(df3)
genreSummary = genreSummary[order(-genreSummary$per_movies),]
genreSummary$rank = seq(1:nrow(genreSummary))
genreSummary$rank2 = sprintf("%02d", genreSummary$rank)
row.names(genreSummary) = NULL

ggplot(genreSummary[1:10,], aes(x=reorder(genre, per_movies), y=per_mov
ies)) +
  geom_bar(stat="identity", fill="#ff9933") +
  geom_text(aes(label=round(per_movies*100,1))) +
  scale_y_continuous(labels = scales::percent) +
  coord_flip() +
  labs(x="Movies' Genres", y="Proportion of Movies")
```




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
top_genre = head(genreSummary,10)
genre = top_genre$genre
newGenreDF = NULL
for (g in genre){
  # Filter dataset to include 1 if genre present in the record
  genre_data = df3[df3[[paste("Genre_",g,sep="")]]==1,]
  genre_data = genre_data[,c("Title","Runtime")]
  genre_data$genre = g
```

```
  newGenreDF = rbind(newGenreDF, genre_data)
}

newGenreDF = merge(newGenreDF, genreSummary[,c("genre","rank")], by="ge
nre")

ggplot(newGenreDF, aes(x=reorder(genre, -rank), y=Runtime, fill=genre))
 +
  geom_boxplot(show.legend=FALSE) +
  coord_flip() +
  labs(x="Genre", y="Runtime",
       title="Runtime by Genre Boxplot sorted by Movies' Proportion")

## Warning: Removed 809 rows containing non-finite values (stat_boxplo
t).
```




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

**A**: For the top 10 genre, we could find runtimes of short and animation are much shorter than others, then the documentary. The rest 7 genre have similar mean run time about 90 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). 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
# first investigate the data
head(df3[, c("Released", "Year", "Date", "Gross")])
```

```
##     Released Year Date Gross
## 1 2005-04-08 2005   NA    NA
## 2 2005-01-25 2005   NA    NA
## 3 2005-02-24 2005   NA    NA
## 4 2005-01-20 2005   NA    NA
## 5 2005-03-12 2005   NA    NA
## 6 2005-01-23 2005   NA    NA
```

```
sum(is.na(df3$Released))
```

```
## [1] 4949
```

```
sum(is.na(df3$Year))
```

```
## [1] 0
```

```
sum(is.na(df3$Date))
```

```
## [1] 35442
```

```
sum(!is.na(df3$Gross))
```

```
## [1] 4558
```

```
# 4949 NA is Released, 0 in Year, 35442 NA in Date. Thus we don't remov
e NA value rows.

df3$Released = as.Date(df3$Released, "%Y-%m-%d")
df3$ReleasedYear = as.numeric(format(df3$Released, "%Y"))
df3$YearReleasedDiff = ifelse(abs(df3$ReleasedYear - df3$Year) <= 1 | i
s.na(df3$ReleasedYear), "Y", "N")
#released year is None or equal to year+-1 is seen as matched data.
df3$YearDateDiff = ifelse(abs(df3$Year - df3$Date) <= 1 | is.na(df3$Dat
```

```
e), "Y", "N")
#Date is None or equal to year+-1 or is seen as matched data.
df4 = subset(df3, YearReleasedDiff == "Y")
df4 = subset(df4, YearDateDiff == "Y")
nrow(df4)

## [1] 38099

sum(!is.na(df4$Gross))

## [1] 4385
```

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

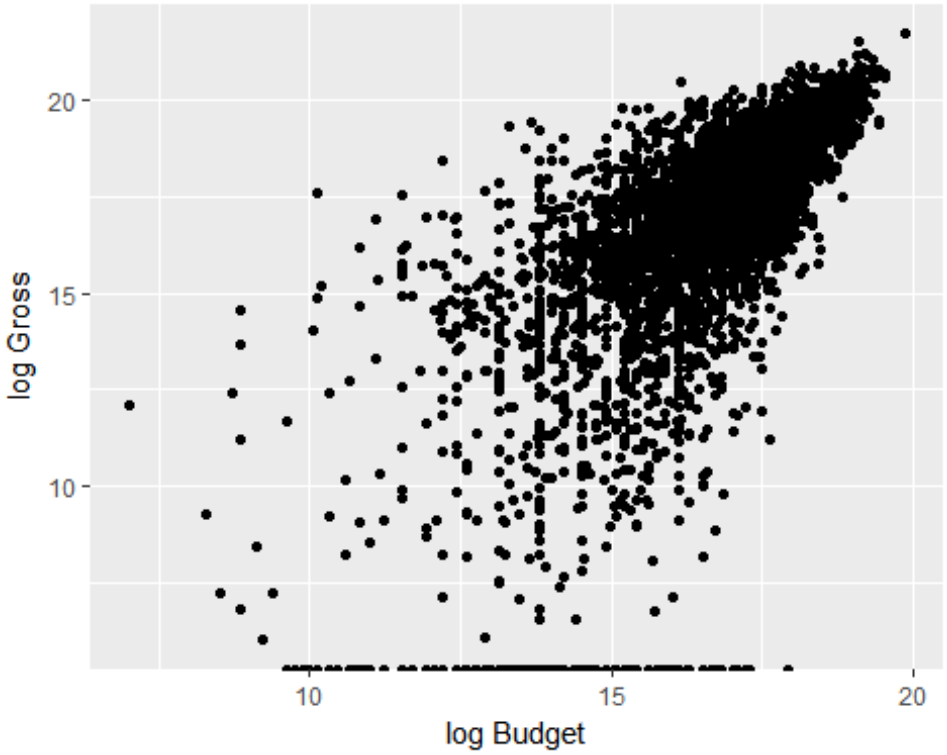
**A**: Since there's too much NA is Gross, Released and Date (larger than 10%), then I don't remove rows if there's NA in Date or Released. Since there's maybe liitle difference between Year and released and Date, then I only remove mismatched that Year and Date difference or Year and Released difference larger than 1. There are 38099 rows remain in the resulting dataset.

## 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.

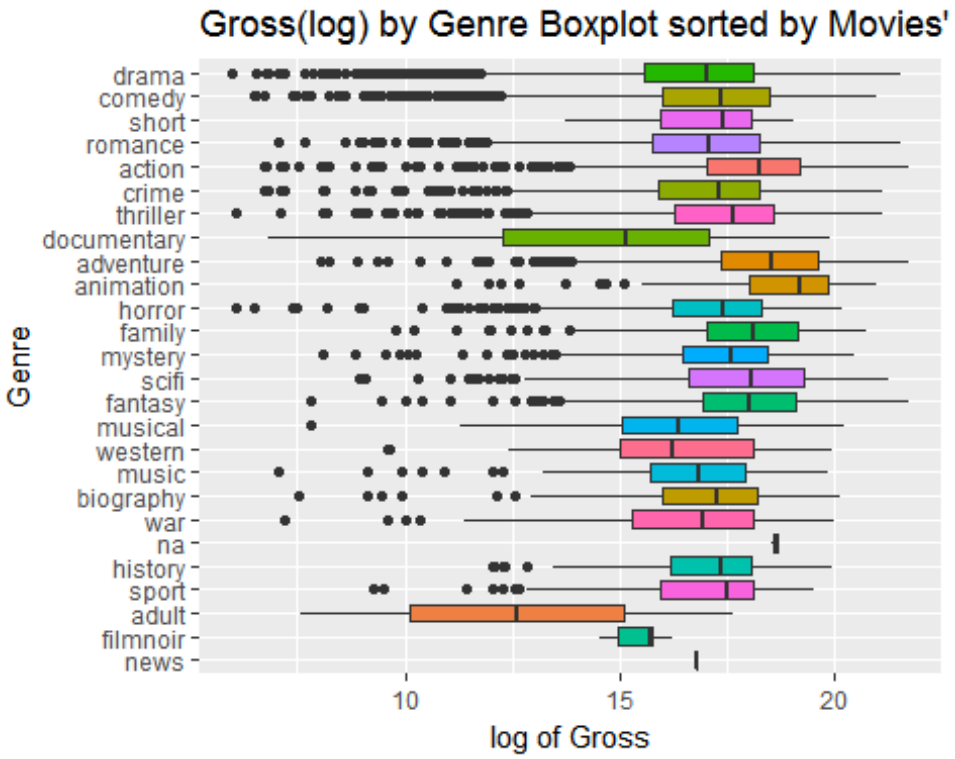
*Note: To get a meaningful relationship, you may have to partition the movies into subsets such as short vs. long duration, or by genre, etc.*

```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or G
enre
sub_df = df4[, c("Budget", "Runtime", "Genre", "Gross")]
sum(is.na(sub_df$Gross))

## [1] 33714

sub_df = sub_df[complete.cases(sub_df),]
nrow(sub_df)

## [1] 4353

#plot log Gross distribution for log Budget
qplot(x = log(Budget), y = log(Gross), data = sub_df,
      xlab = "log Budget", ylab = "log Gross")
```




```
#plot log Gross distribution for runtime
sub_df$RuntimeBin = cut(sub_df$Runtime, seq(50, 200, by=10), dig.lab=4)
G_R = data.frame(sub_df$Gross, sub_df$RuntimeBin)
colnames(G_R) = c("Gross", "Runtime")
G_RMod = ddply(G_R, "Runtime", summarise, mean.gross = mean(Gross, na.r
m = TRUE))

qplot(x = Runtime, y = log(mean.gross), data = G_RMod,
      xlab = "Runtime", ylab = "mean Gross")
```


```
# Plot Goss distribution for genres
genre = genreSummary$genre
newGenreDF = NULL
sub_df1 = df4[!is.na(df4$Gross),]
for (g in genre){
  # Filter dataset to include 1 if genre present in the record
  genre_data = sub_df1[sub_df1[[paste("Genre_",g,sep="")]]==1,]
  if (nrow(genre_data) == 0) next
  genre_data = genre_data[,c("Title","Gross")]
  genre_data$genre = g
  newGenreDF = rbind(newGenreDF, genre_data)
}

newGenreDF = merge(newGenreDF, genreSummary[,c("genre","rank")], by="ge
nre")

ggplot(newGenreDF, aes(x=reorder(genre, -rank), y=log(Gross), fill=genr
e)) +
  geom_boxplot(show.legend=FALSE) +
  coord_flip() +
  labs(x="Genre", y="log of Gross",
       title="Gross(log) by Genre Boxplot sorted by Movies' Proportion")

## Warning: Removed 497 rows containing non-finite values (stat_boxplo
t).
```

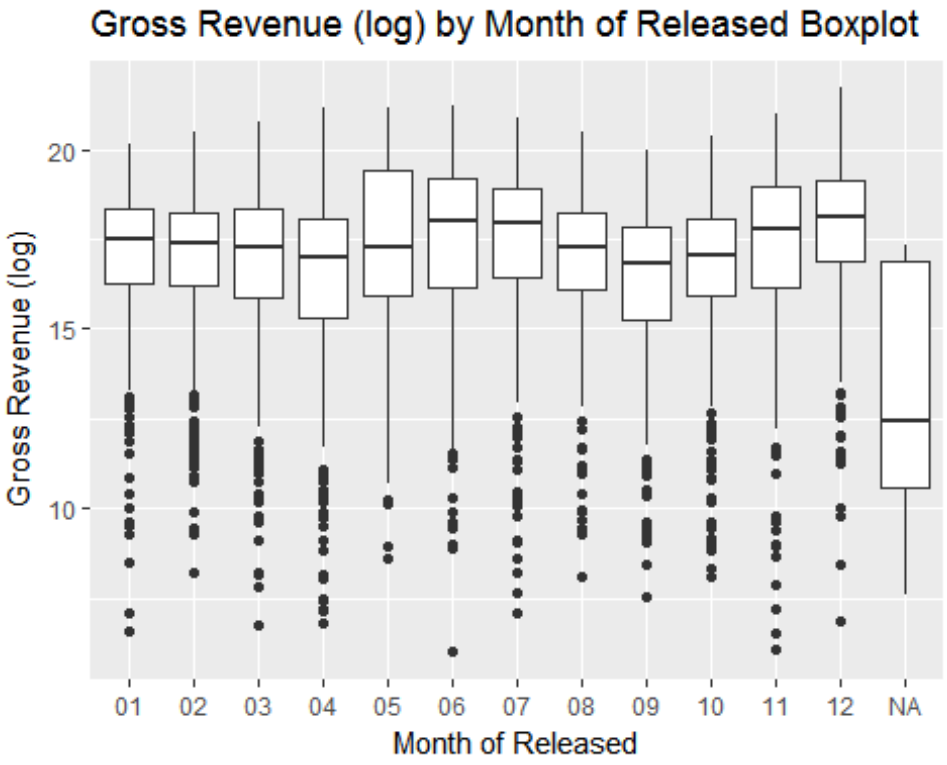



**Q**: Did you find any observable relationships or combinations of Budget/Runtime/Genre that result in high Gross revenue? If you divided the movies into different subsets, you may get different answers for them - point out interesting ones.

**A**: For Genre, the top 3 genre with highest Gross Revenue are Animation, Adventure, and Action. Basically, log Budget has linear relationship with log Gross, thus more buget could produce more gross revenue. And after calculate the mean gross during Runtime with 10 min as interval, I found the highest gross revenue happened during 160 - 170 runtime minutes.

```
# TODO: Investigate if Gross Revenue is related to Release Month
sub_df1$ReleasedMonth = format(sub_df1$Released, "%m")
ggplot(sub_df1, aes(x=ReleasedMonth, y=log(Gross))) +
  geom_boxplot() +
  labs(x="Month of Released", y="Gross Revenue (log)",
       title="Gross Revenue (log) by Month of Released Boxplot")
```

```
## Warning: Removed 242 rows containing non-finite values (stat_boxplo
t).
```




```
#Result display that the Gross revenue in June, July and December are h
igher 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: 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
# Create New column name AwardsModified by lower case Awards
df4$AwardsModified = tolower(df4$Awards)
# Convert values with "n/a" to NA
df4$AwardsModified[df4$AwardsModified=="n/a"] = NA
# Examine all unique values of awards to see pattern
# unique(movies$AwardsModified)
# Remove punctuation from Awards column
df4$AwardsModified = gsub("[[:punct:]]","",df4$AwardsModified)
df4$AwardsModified = gsub(" ","",df4$AwardsModified)
#df4$AwardsNum = gregexpr("[0-9]+wins", df4$AwardsModified)
# get all numeric values from AwardsModified
```

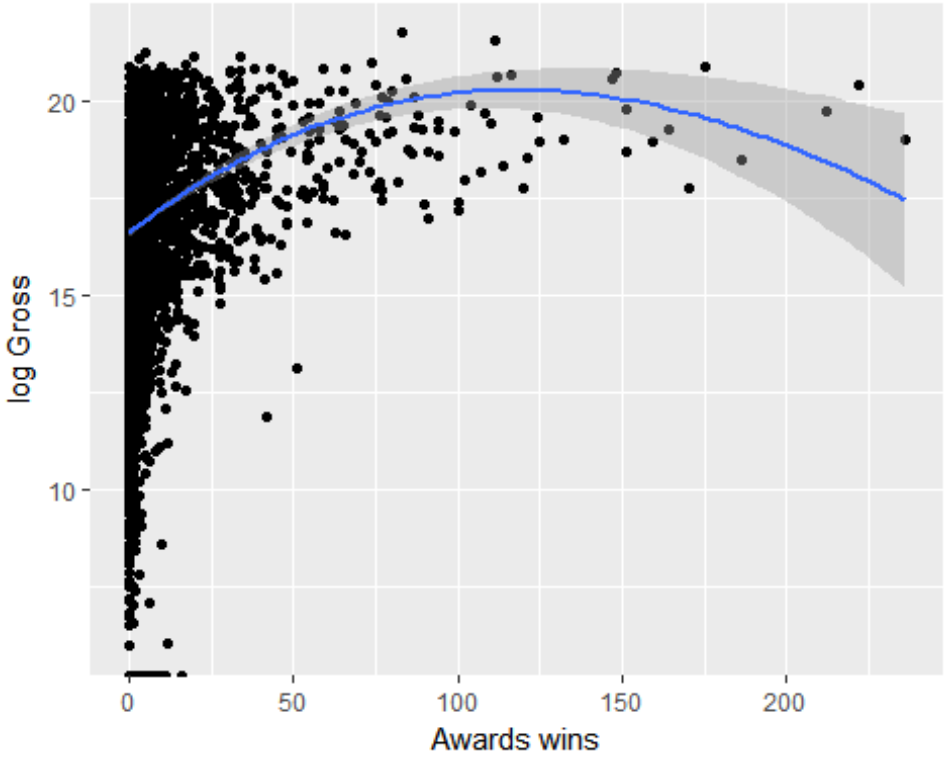
```
awards = c()
nominations = c()
for( i in seq_along(df4$AwardsModified)){
        if(is.na(df4$AwardsModified[i])){
                awards = c(awards, 0)
                nominations = c(nominations, 0)
        }else{
                x = df4$AwardsModified[i]
                temp1 = gregexpr("[0-9]+win", x)
                temp2 = gregexpr("won+[0-9]", x)
                y1 = unlist(regmatches(x, temp1))
                y2 = unlist(regmatches(x, temp2))
                temp1 = gregexpr("[0-9]+", y1)
                temp2 = gregexpr("[0-9]+", y2)
                # extract the awards and sum them
                total_awards = sum(as.numeric(unlist(regmatches(y1, tem
p1)))) + sum(as.numeric(unlist(regmatches(y2, temp2))))
                awards = c(awards, total_awards)

                temp3 = gregexpr("[0-9]+nominat", x)
                temp4 = gregexpr("nominatedfor+[0-9]", x)
                y3 = unlist(regmatches(x, temp3))
                y4 = unlist(regmatches(x, temp4))
                temp3 = gregexpr("[0-9]+", y3)
                temp4 = gregexpr("[0-9]+", y4)
                # extract the nominations and sum them
                total_nominations = sum(as.numeric(unlist(regmatches(y3,
 temp3)))) + sum(as.numeric(unlist(regmatches(y4, temp4))))
                nominations = c(nominations, total_nominations)
        }
}
df4$TotalAwards = awards
df4$TotalNominations = nominations
nrow(df4[df4$TotalAwards != 0 | df4$TotalNominations != 0,])

## [1] 13905
```

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

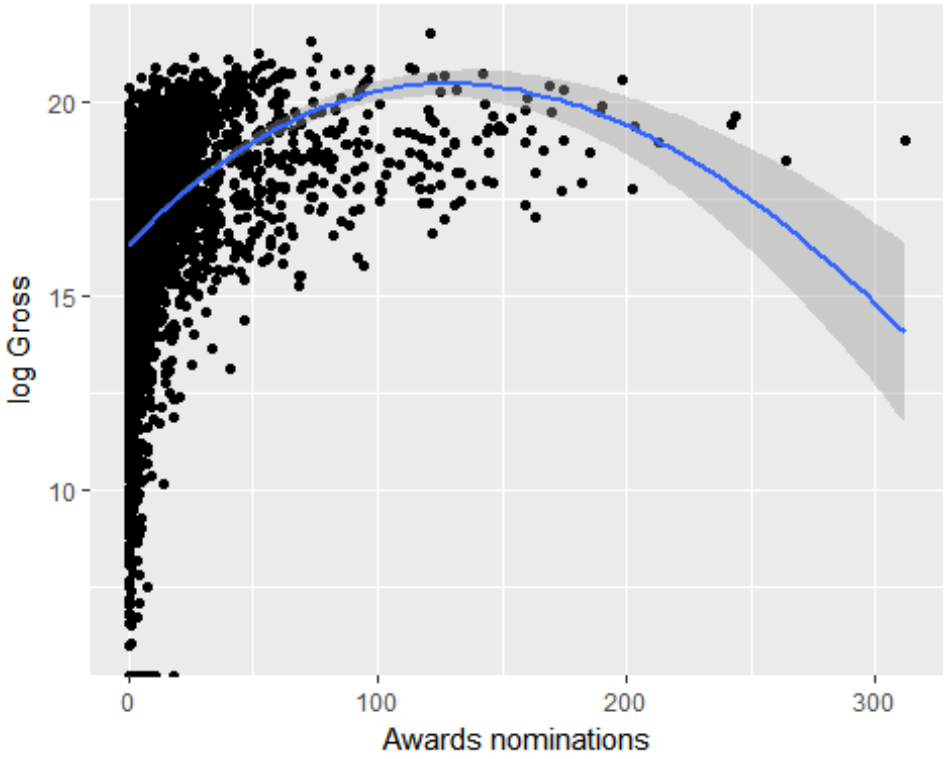
**A**: I construct the conversion using that the “numb + win” and “won + num” to determined the number of wins and sum them. For nominations, I use “num + nomination” and “nominated for + num” as number for nominations and sum them. There’s total 13905 rows had valid/non-zero wins or nominations

```
# TODO: Plot Gross revenue against wins and nominations
#plot log Gross distribution for wins
qplot(x = TotalAwards, y = log(Gross), data = df4,
      xlab = "Awards wins", ylab = "log Gross") + stat_smooth(method =
      "loess", span = 2, se = TRUE)
```

```
## Warning: Removed 33956 rows containing non-finite values (stat_smooth).

## Warning: Removed 33714 rows containing missing values (geom_point).
```




```
#plot log Gross distribution for nominations
qplot(x = TotalNominations, y = log(Gross), data = df4,
      xlab = "Awards nominations", ylab = "log Gross")+ stat_smooth(method =
      "loess", span = 2, se = TRUE)

## Warning: Removed 33956 rows containing non-finite values (stat_smooth).

## Warning: Removed 33714 rows containing missing values (geom_point).
```




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

**A**: The basic trend between gross revenue with number of awards and nominations are positive which means more awards and nominations will increase gross revenue.

## 7. Movie ratings from IMDb and Rotten Tomatoes

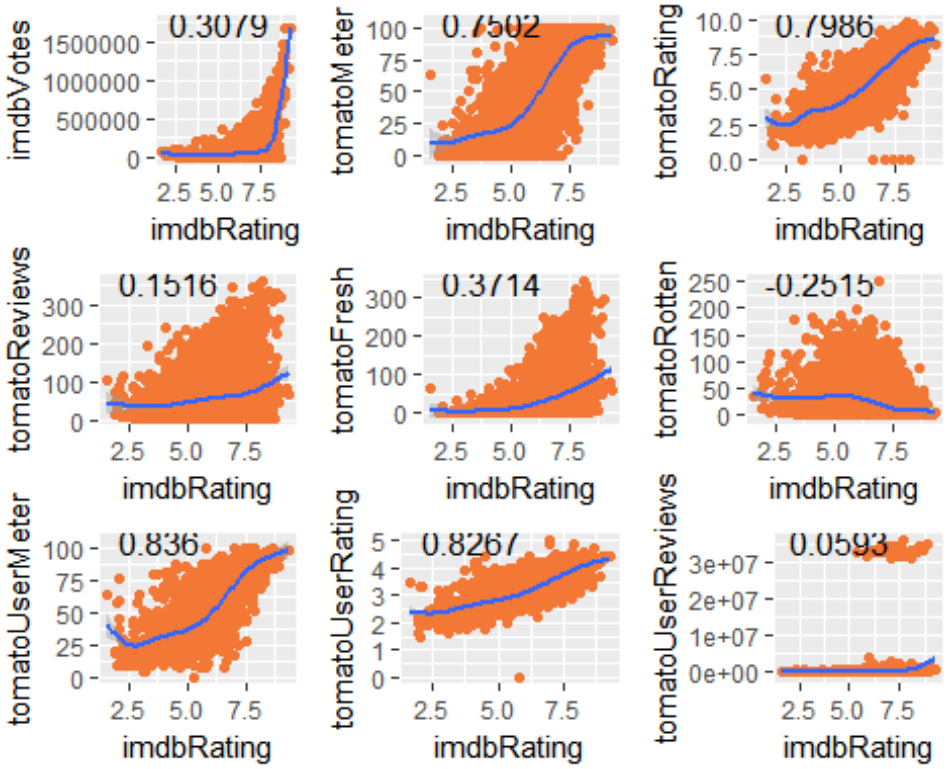
There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by several variables pre-fixed by `tomato`). Read up on such ratings on the 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 relate
d
rating_col = c("imdbRating","imdbVotes", "tomatoMeter","tomatoRating",
               "tomatoReviews","tomatoFresh","tomatoRotten","tomatoUser
Meter",
               "tomatoUserRating","tomatoUserReviews")
sub_df2 = df4[rating_col]
sub_df2 = sub_df2[complete.cases(sub_df2),]
rating_col_combn = combn(rating_col, 2)
```

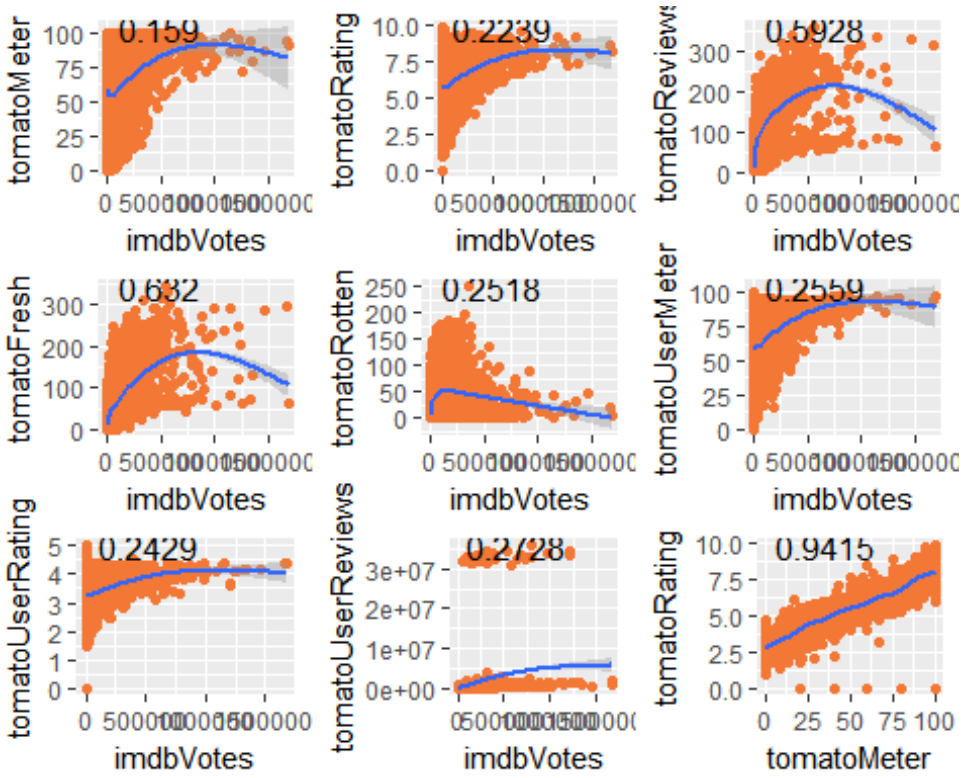
```
corr_ls = NULL
pairwise_graph = NULL
for (i in 1:ncol(rating_col_combn)){
  rating_var1 = rating_col_combn[1,i]
  rating_var2 = rating_col_combn[2,i]
  corr = round(cor(sub_df2[[rating_var1]], sub_df2[[rating_var2]], use=
"complete.obs"),4)
  corr_ls[[i]] = corr
  corr_grob = grobTree(textGrob(corr , x=0.1,  y=0.95, hjust=0))
  pairwise_graph[[i]] = ggplot(sub_df2, aes_string(x=rating_var1, y=rat
ing_var2)) +
    geom_point(color="#f37735") +
    geom_smooth() +
    annotation_custom(corr_grob)
}

do.call(grid.arrange, c(pairwise_graph[1:9], ncol=3))
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```



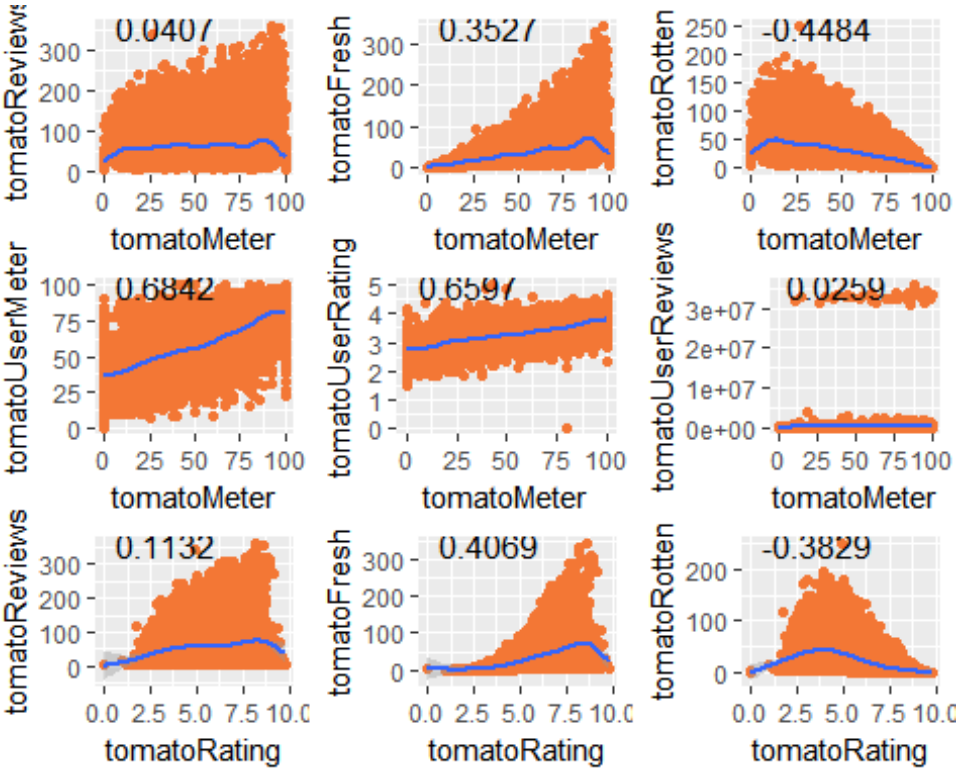
```
do.call(grid.arrange, c(pairwise_graph[10:18], ncol=3))
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```



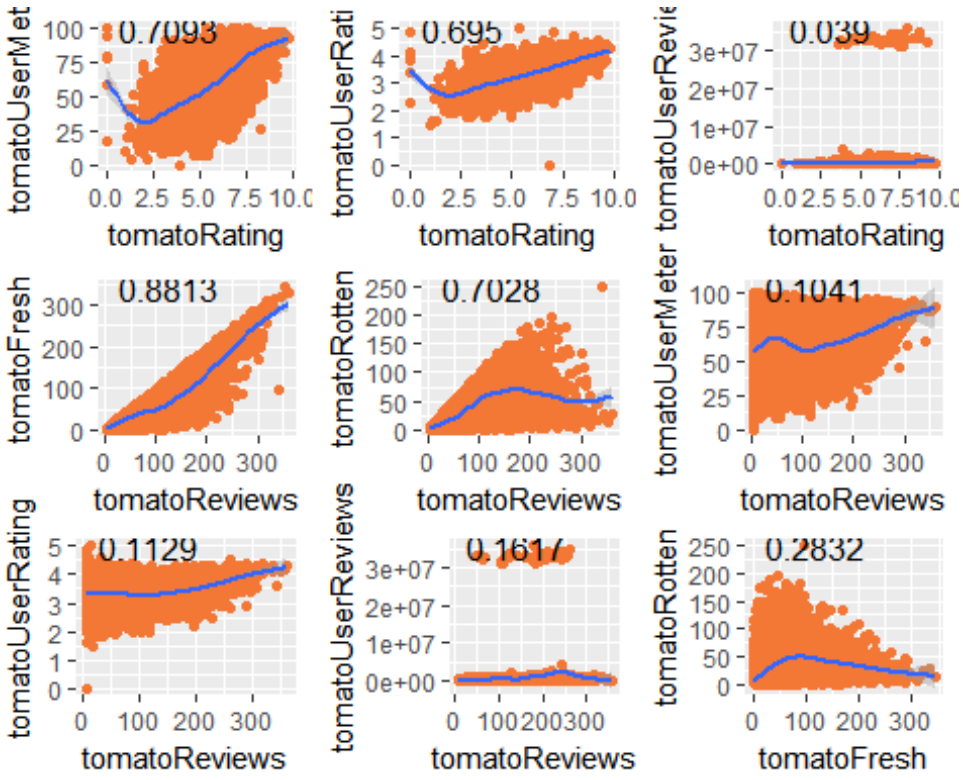

```
do.call(grid.arrange, c(pairwise_graph[19:27], ncol=3))
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```



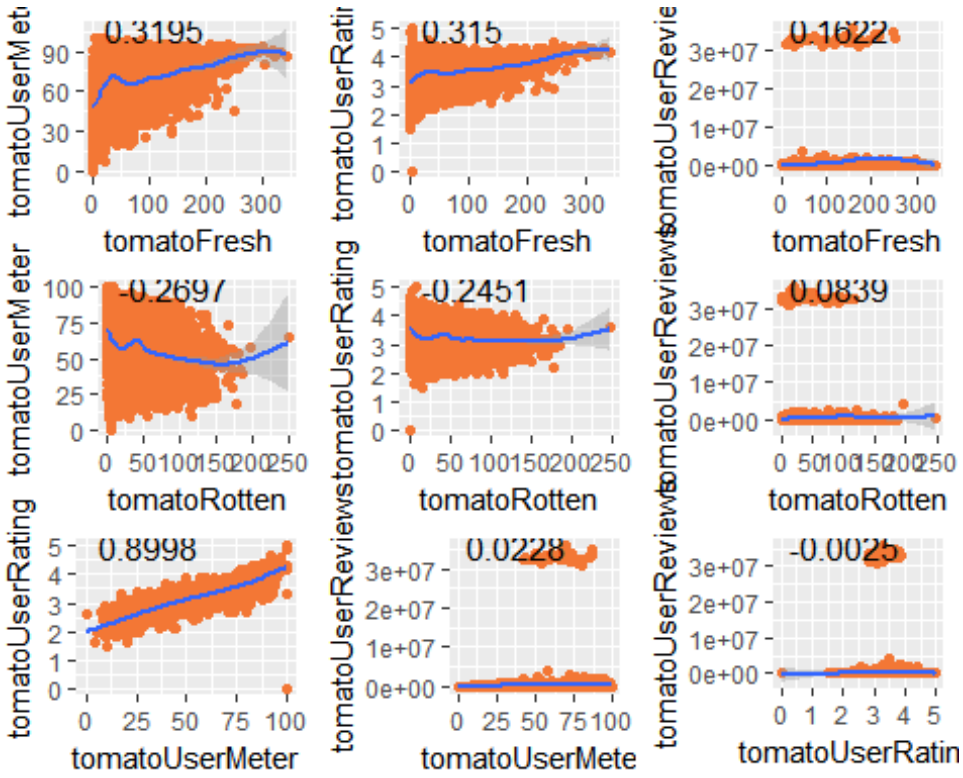

```
do.call(grid.arrange, c(pairwise_graph[28:36], ncol=3))
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```




```
do.call(grid.arrange, c(pairwise_graph[37:45], ncol=3))
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```




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

**A**: There are 2 IMDb related ratings: imdbRating and imdbVotes and there are 8 rotten tomato related rating: tomatoMeter, tomatoRating, tomatoReviews, tomatoFresh, tomatoRotten, tomatoUserMeter, tomatoUserRating, tomatoUserReviews.

TomatoMeter is the percentage of critics give positive reviews and it can be calculated as tomatoFresh/(tomatoFresh + tomatoRotten). tomatoReviews is the count of total critic reviews and it equals to tomatoFresh + tomatoRotten. TomatoRating is the average of critics ratings( range from 0 to 10). TomatoUserMeter is the percentage of users give positive reviews. TomatoUserReviews is the total counts of using ratings. TomatoUserRating is the average user rating (range from 0 to 5).

We can see tomatoMeter is based on tomatoFresh, tomatoRotten and tomatoReview. TomatoUserMeter is based on tomatoUserRewiews. The largest corr is between tomatoRating and tomatoMeter, which is 0.9415. The imdbRating and tomatoRating has a good corr which is 0.8986.

## 8. Ratings and awards

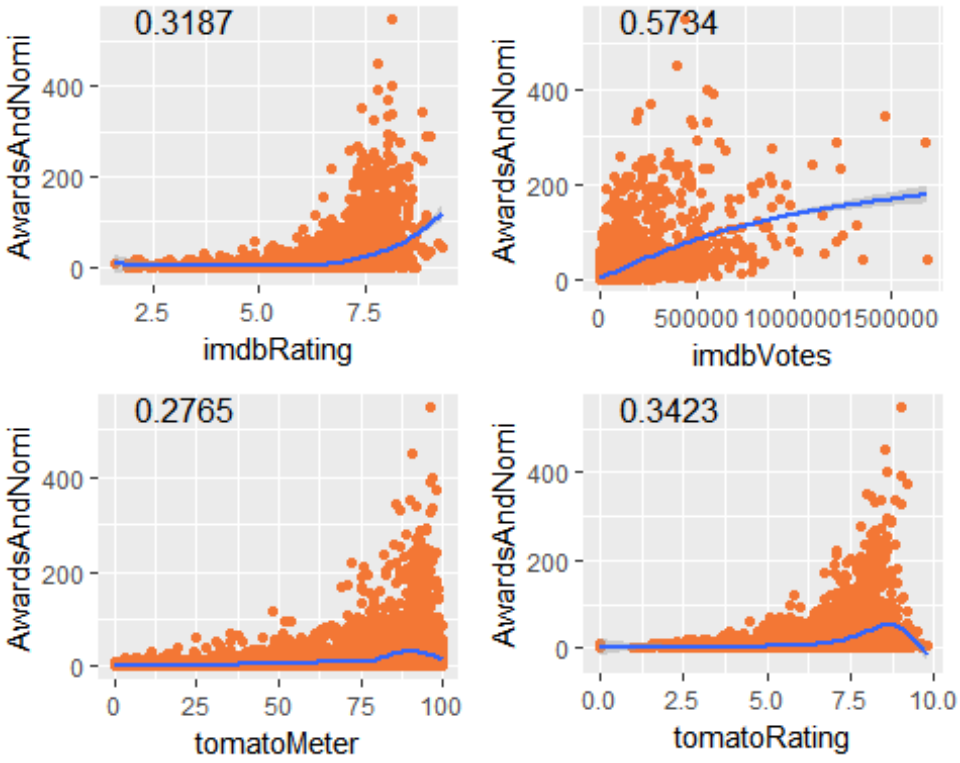
These ratings typically reflect the general appeal of the movie to the public or gather opinions from a larger body of critics. Whereas awards are given by professional

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).

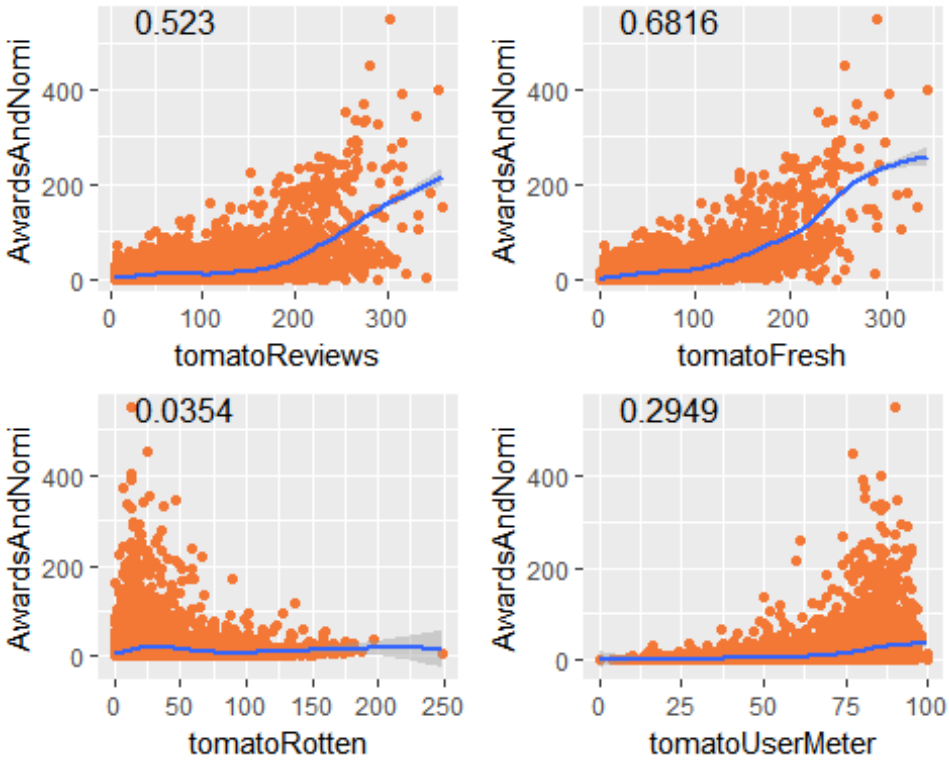
```
# TODO: Show how ratings and awards are related
df4$AwardsAndNomi = df4$TotalAwards + df4$TotalNominations
rating_award_col = c("imdbRating","imdbVotes", "tomatoMeter","tomatoRat
ing",
                "tomatoReviews","tomatoFresh","tomatoRotten","tomatoUser
Meter",
                "tomatoUserRating","tomatoUserReviews", "AwardsAndNomi")
sub_df3 = df4[rating_award_col]
sub_df3 = sub_df3[complete.cases(sub_df3),]
corr_ls = NULL
pairwise_graph = NULL
for (i in 1:10){
  rating_var1 = rating_award_col[i]
  rating_var2 = rating_award_col[11]
  corr = round(cor(sub_df3[[rating_var1]], sub_df3[[rating_var2]], use=
"complete.obs"),4)
  corr_ls[[i]] = corr
  corr_grob = grobTree(textGrob(corr , x=0.1,  y=0.95, hjust=0))
  pairwise_graph[[i]] = ggplot(sub_df3, aes_string(x=rating_var1, y=rat
ing_var2)) +
    geom_point(color="#f37735") +
    geom_smooth() +
    annotation_custom(corr_grob)
}

do.call(grid.arrange, c(pairwise_graph[1:4], ncol=2))
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```



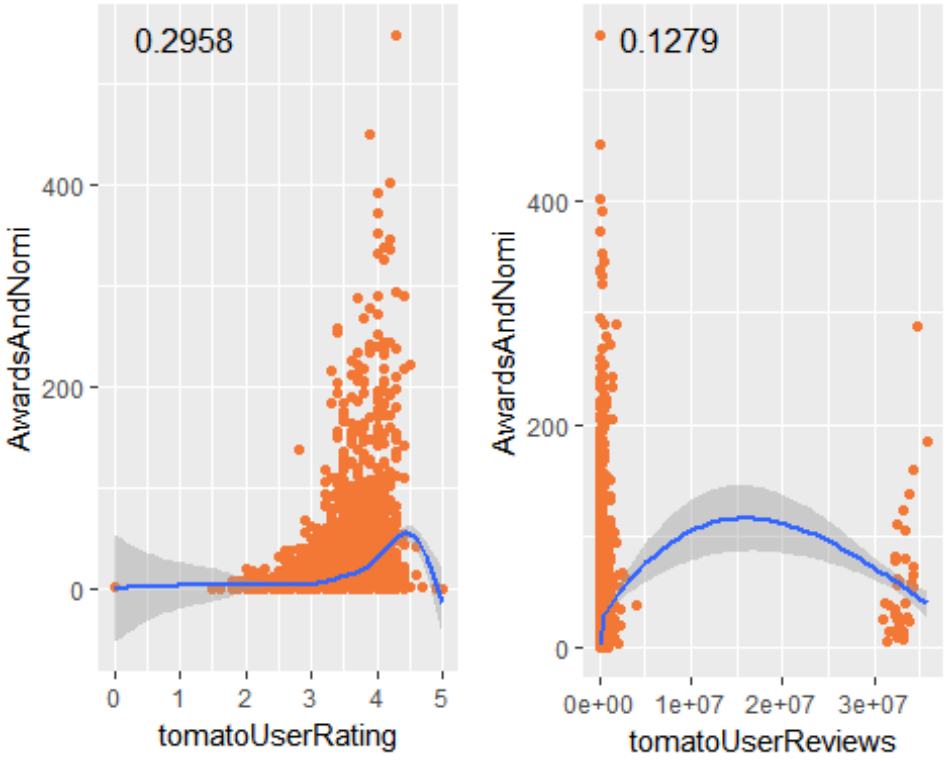

```
do.call(grid.arrange, c(pairwise_graph[5:8], ncol=2))
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```




```
do.call(grid.arrange, c(pairwise_graph[9:10], ncol=2))
```

```
## `geom_smooth()` using method = 'gam'
## `geom_smooth()` using method = 'gam'
```



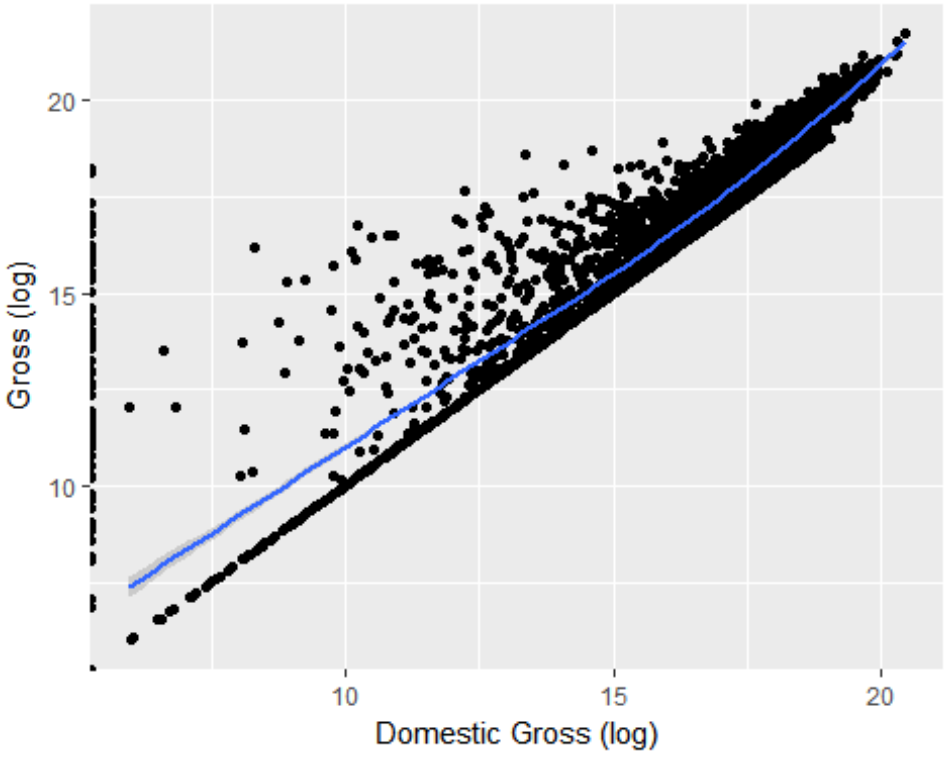

**Q**: How good are these ratings in terms of predicting the success of a movie in winning awards or nominations? Is there a high correlation between two variables?

**A**: These ratings somehow can prdicting the success of a movie such lick the correlation between tomatoFresh and total number for wins and nominations can reach 0.6816 as highest correlation.

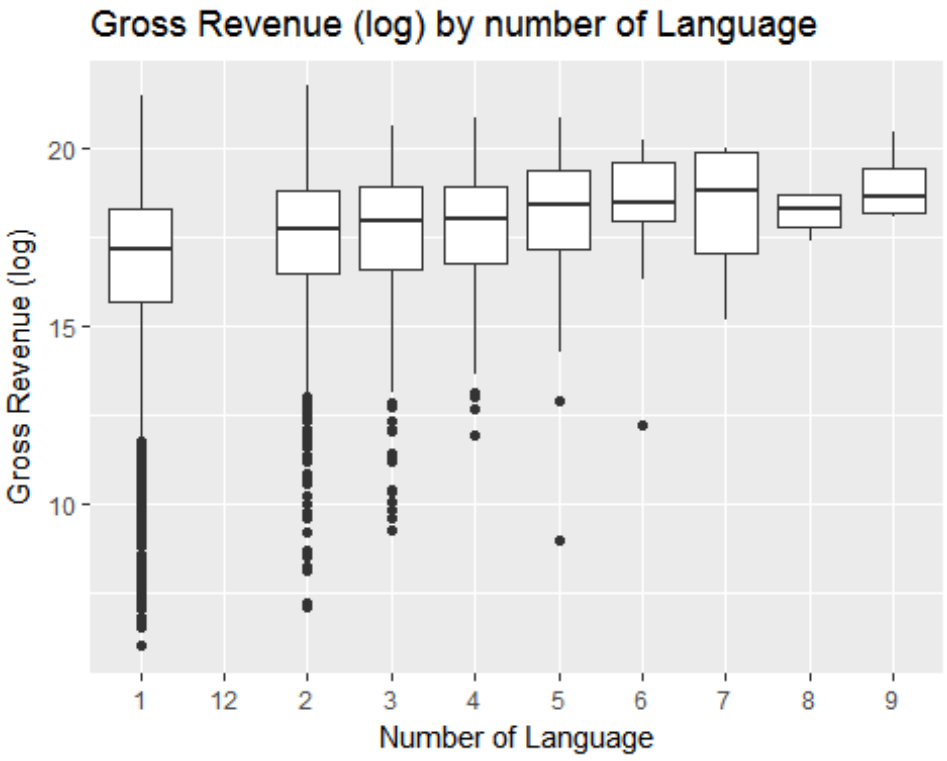
## 9. Expected insights

Come up with two new insights (backed up by data and graphs) that is expected. Here 欽涇 ew 欽? 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
#Plot Domestic_Gross with Gross
qplot(x = log(Domestic_Gross), y = log(Gross), data = df4,
      xlab = "Domestic Gross (log)", ylab = "Gross (log)")+ stat_smooth
(method =
      "loess", span = 1, se = TRUE)

## Warning: Removed 34032 rows containing non-finite values (stat_smoot
h).

## Warning: Removed 33714 rows containing missing values (geom_point).
```




```
#plot number of countries with gross
number_language = c()
for (i in 1:nrow(df4)){
  number = lengths(strsplit(df4$Language[i], ","))
  number_language = c(number_language, number)
}
df4$number_language = as.character(number_language)

ggplot(df4, aes(x=number_language, y=log(Gross))) +
  geom_boxplot() +
  labs(x="Number of Language", y="Gross Revenue (log)",
       title="Gross Revenue (log) by number of Language")

## Warning: Removed 33956 rows containing non-finite values (stat_boxpl
ot).
```




**Q**: Expected insight #1.

**A**: We could find the domestic gross (log) and total gross (log) have very clear linear relationship that the movie with higher total gross always has high domestic gross.

**Q**: Expected insight #2.

**A**: We could find the gross revenue increase with the more number of language belong to movies. That means if the movie has more version of different language, it may has more gross revenue.

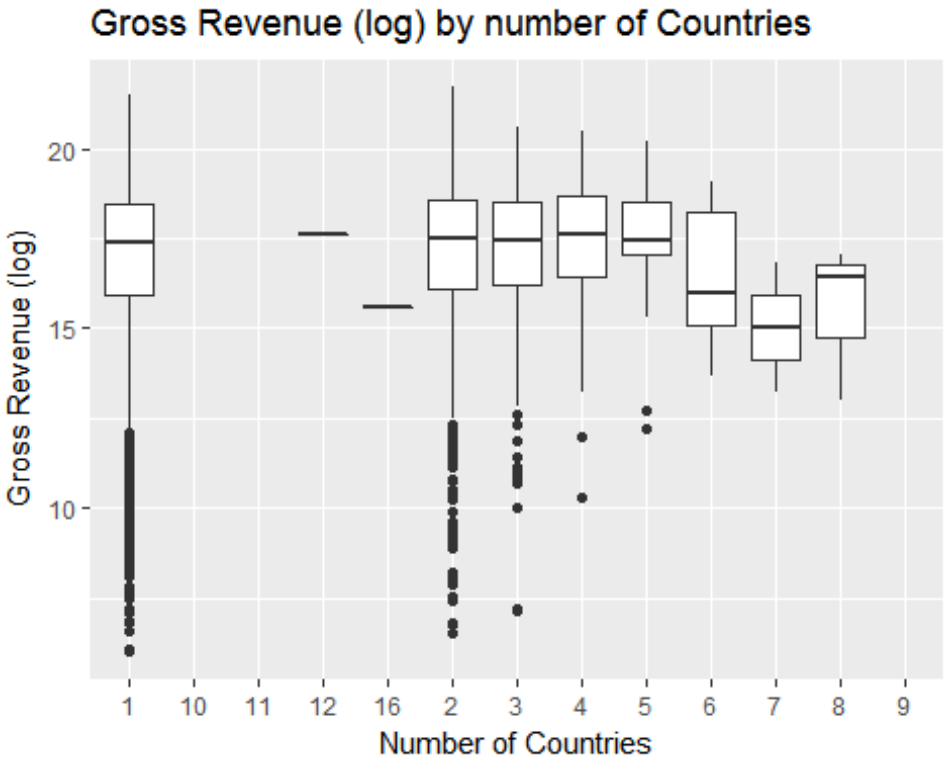
## 10. Unexpected insight

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
#plot number of countries with gross
number_countries = c()
for (i in 1:nrow(df4)){
  number = lengths(strsplit(df4$Country[i], ","))
  number_countries = c(number_countries, number)
}
df4$number_countires = as.character(number_countries)

ggplot(df4, aes(x=number_countires, y=log(Gross))) +
```

```
  geom_boxplot() +
  labs(x="Number of Countries", y="Gross Revenue (log)",
       title="Gross Revenue (log) by number of Countries")
```

```
## Warning: Removed 33956 rows containing non-finite values (stat_boxpl
ot).
```




**Q**: Unexpected insight.

**A**: At first glance, I think the total gross revenue should be correlate with the number of release countries. Becasue I think the more contries the movie was released, the gross revenue should be higher. But After plot the Gross Revenue with number of countries, I found the medians of gross revenue are very similar that the statistic of gross of movie only released in one country is almost same as the movie shown in 4 or 5 countries. Moreover, the movie shown in 7 countries has the lowest gross in the results.