Project 1: Explore and Prepare Data

CSE6242 - Data and Visual Analytics - Summer 2018 Due: Sunday, June 17, 2018 at 11:59 PM UTC-12:00 on T-Square

Gatech Id: rpothamsetty3

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's API (see www.omdbapi.com) and joining it with a separate dataset of movie budgets and gross earnings (unknown to you). The join key was the movie title. This data is available for personal use, but IMDb'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 "second window" streaming rights).

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(ggplot2)
library(GGally)
library(assertr)
library(splitstackshape)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(reshape2)
```

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$Type == "movie",]
movie_db <- movies_merged[which(movies_merged$Type == 'movie'),]
dim(movie_db)
```

```
## [1] 40000    39
```

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

A:40000

2. Process `Runtime` column

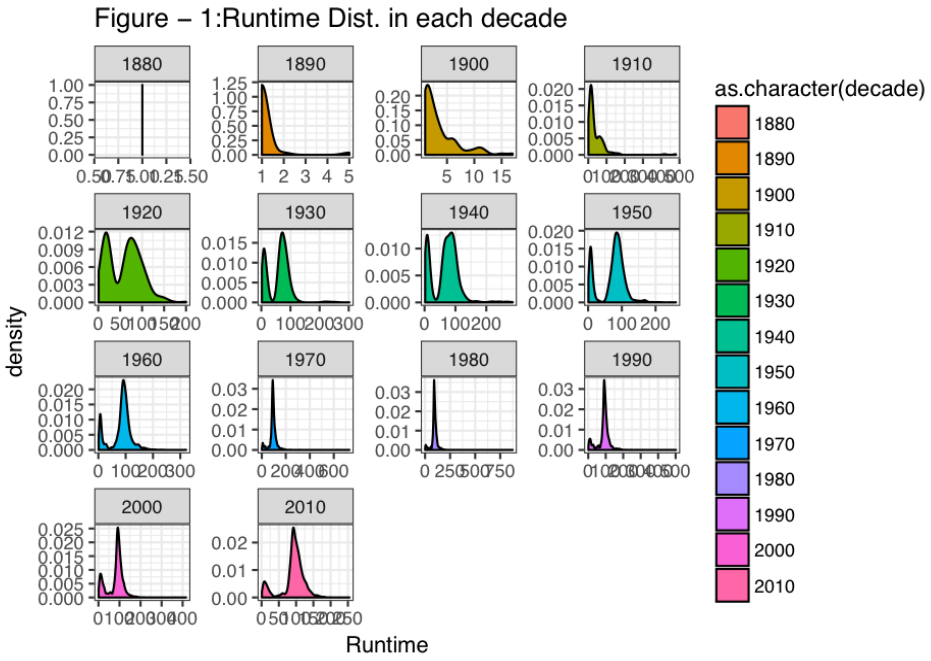
The variable `Runtime` represents the length of the title as a string. Write R code to convert it to a numeric value (in minutes) and replace `df$Runtime` with the new numeric column.

```
parse_runtime <- function(runtime){
  runtime <- strsplit(runtime, ' ')[[1]]
  if((length(runtime) == 2) && (runtime[2] == 'h')){
    return(as.integer(runtime[1])*60)
  }else if((length(runtime) == 2) && (runtime[2] == 'min')){
    return(as.integer(runtime[1]))
  }else if(length(runtime) == 4){
    hr_to_min = as.integer(runtime[1])*60
    return(hr_to_min + as.integer(runtime[3]))
  }else return(NA)
}
movie_db$Runtime <- sapply(movie_db$Runtime, parse_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.

```
# TODO: Investigate the distribution of Runtime values and how it varies by Year and Budget
# Bucketing years to decades
floor_year <- function(year){
  year <- as.integer(year)
  decade <- year - (year %% 10)
  return(decade)
}
movie_db$decade <- sapply(movie_db$Year, floor_year)

ggplot(data = movie_db, aes(Runtime, fill = as.character(decade))) +
  geom_density() +
  facet_wrap(~decade, scales = 'free') +
  ggtitle("Figure - 1:Runtime Dist. in each decade") +
  theme_bw()
```

```
## Warning: Removed 751 rows containing non-finite values (stat_density).
```

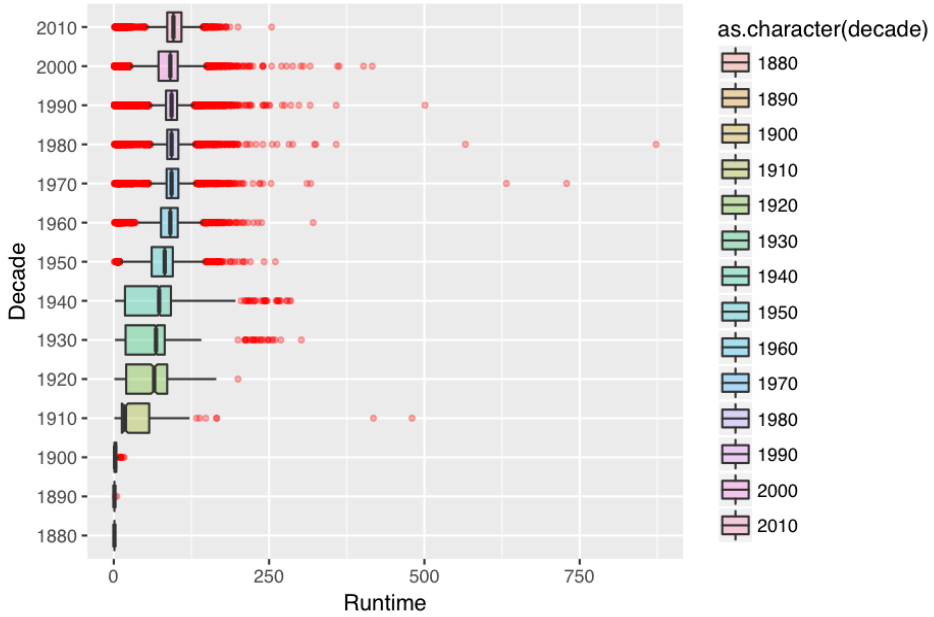



```
ggplot(movie_db, aes(x = as.character(decade), Runtime, fill = as.character(decade))) +
  geom_boxplot(alpha=0.3, notch=TRUE, notchwidth = 0.8, outlier.colour="red", outlier.fill="red", outli
  theme(legend.position="right") +
  ggtitle("Figure - 2: Runtime Dist. in each decade") +
  xlab("Decade") + coord_flip()
```

```
## Warning: Removed 751 rows containing non-finite values (stat_boxplot).
```

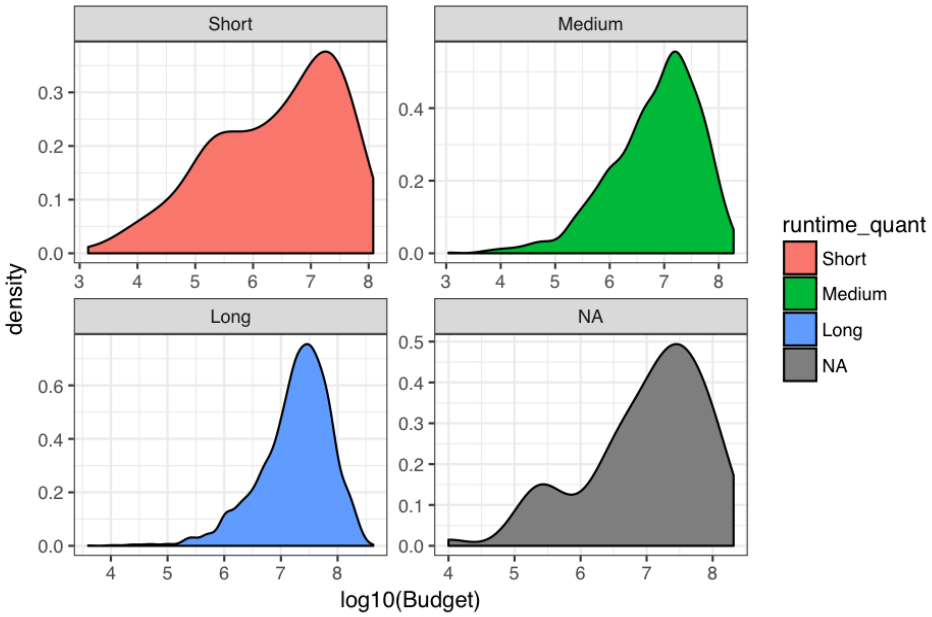
```
## notch went outside hinges. Try setting notch=FALSE.
```

Figure – 2: Runtime Dist. in each decade




```
## Runtime classification Vs Budget
runtime_quantiles <- quantile(movie_db$Runtime, probs = seq(0, 1, 0.33), na.rm = TRUE)
runtimeQuantiles <- function(runtime) {
  cut(runtime, breaks = runtime_quantiles, labels = c( "Short", "Medium", "Long"))
}
movie_db$runtime_quant <- sapply(movie_db$Runtime, runtimeQuantiles)

suppressWarnings(print(ggplot(data = movie_db, aes(log10(Budget), fill = runtime_quant)) +
                         geom_density() +
                         facet_wrap(~runtime_quant, scales = 'free') +
                         ggtitle("Figure - 3: Budget Dist. in each Runtime category") +
                         theme_bw()))
```

Figure – 3: Budget Dist. in each Runtime category



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: Runtime Vs Year(Decade): As you can see in Figure – 2, above, the median Runtime has been increasing from 1880s to 1960s, from there on the median level hasn't changed. The Runtime Range is large in from 1920s to 1940s. From 1960s the Runtime median stayed ~90-100 mins. If you observe Figure – 1, in the 1880s to 1920s, we can see a growth to bi-modal distribution from one mode distribution. From 1920s to 2010s, those bi-modal distribution have become leaner(thin tails) and concentrated around the mean.

Runtime Vs Budget: I have classified the Runtime in to three categories namely "Short", "Medium" and "Long"(Refer Figure-3). In the "Short" category, Budget distribution has fat tails, this could probably be from movies in early years but Budget were comparatively lower in the early years. As we move from "Short" to "Long" category, the Budget distribution has low tails and concentrated around the mean, this makes sense as longer runtime movies in generally have high budgets. If you look at the "N/A" category, understandably it has a dual distribution as it may have all ranges of Runtime.

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.

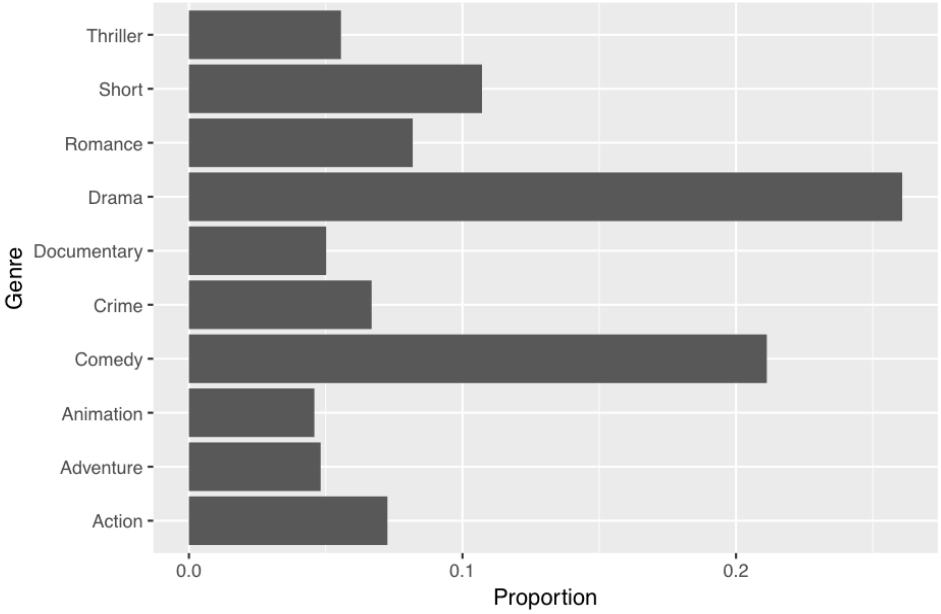
For example, if there are a total of 3 genres: Drama, Comedy, and Action, a movie that is both Action and Comedy should be represented by a binary vector <0, 1, 1>. Note that you need to first compile a dictionary of all possible genres and then figure out which movie has which genres (you can use the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

```r
# TODO: Replace Genre with a collection of binary columns
genre_df <- movie_db[,c('Title', 'Genre', 'Gross', 'Runtime')]
genre_df <- cSplit_e(genre_df, "Genre", ",", type = "character", fill = 0) #split Genre column in to mu
```

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

```r
# TODO: Select movies from top 10 most common genres and plot their relative proportions
most_cmon_10 <-  as.data.frame(sort(colSums(genre_df[,5:ncol(genre_df)]), decreasing = TRUE)[1:10], row
most_cmon_10$Genre <- rownames(most_cmon_10)
colnames(most_cmon_10) <- c("Count", "Genre")
most_cmon_10$Genre <- gsub("Genre_", "", most_cmon_10$Genre)
colnames(genre_df) <- gsub("Genre_", "", colnames(genre_df))

# Select movies from top 10 most common genres and plot their relative proportions
most_cmon_10$Proportion <- most_cmon_10$Count/sum(most_cmon_10$Count)
ggplot(most_cmon_10, aes(x = Genre, y = Proportion)) + geom_bar(stat = "identity") + coord_flip() +
                  ggtitle("Figure - 4: Top 10 Movie Genre's Proportion")
```

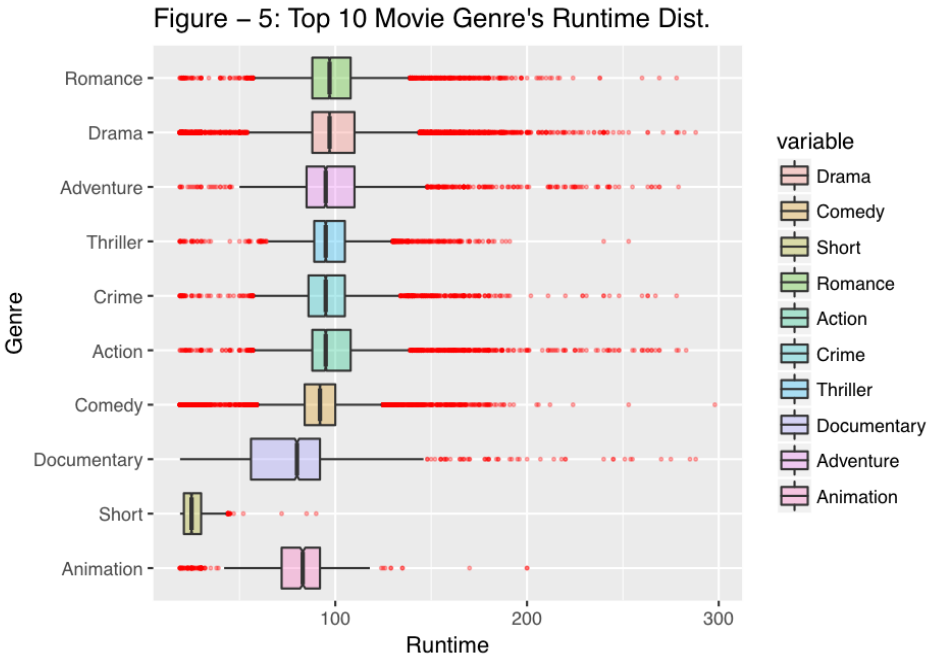
Figure – 4: Top 10 Movie Genre's Proportion



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

```r
# TODO: Plot Runtime distribution for top 10 most common genres
top_genre_runtime = melt(genre_df, id.vars = c("Title","Runtime"), measure.vars = most_cmon_10$Genre)
top_genre_runtime = top_genre_runtime[!is.na(top_genre_runtime$Runtime),]
top_genre_runtime = top_genre_runtime[top_genre_runtime$value!=0,]

ylim1 = boxplot.stats(top_genre_runtime$Runtime)$stats[c(1, 5)] # compute lower and upper whiskers

suppressWarnings(print(ggplot(top_genre_runtime, aes(reorder(variable, Runtime, median), Runtime, fill
```

```
geom_boxplot(alpha=0.3, notch=TRUE, notchwidth = 0.8, outlier.colour="red", outlier.fill="red", outli
theme(legend.position="right") +
ggtitle("Figure - 5: Top 10 Movie Genre's Runtime Dist.") +
xlab("Genre") +
ylim(ylim1*1.9) +
coord_flip()))
```



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

A: • As expected we see from Figure – 4, general movie Genre's like Drama, Romance, Comedy, Action e.t.c in the top 10 Genres. Drama and Comedy have the highest proportion in terms of number of movies whereas least being the Animation and Adventure. • Unexpectedly Short movies is the 3rd highest category. • From Figure – 5, we can say that except for Animation, Short and Documentary, other Genres have a median close to 100mins. Documentary has widest range of runtime whereas the other Genre's have fairly contained range in 2nd and 3rd quartiles.

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.

```
temp <- movie_db[,c('Title', 'Year', 'Released', 'Date')]
temp <- movie_db[!is.na(movie_db$Gross),] #4558
total_rows <- nrow(movie_db)
print("Percent of rows remaining after removing NA from 'Year'")
```

```
## [1] "Percent of rows remaining after removing NA from 'Year'"
```

```
((total_rows - nrow(movie_db[is.na(movie_db$Year),]))/total_rows) * 100
```

```
## [1] 100
```

```
#100
print("Percent of rows remaining after removing NA from 'Released'")
```

```
## [1] "Percent of rows remaining after removing NA from 'Released'"
```

```
((total_rows - nrow(movie_db[is.na(movie_db$Released),]))/total_rows) * 100
```

```
## [1] 87.6275
```

```
# 87.6275 implies loosing more than 10% od data
print("Percent of rows remaining after removing NA from 'Released' with value in 'Gross'")
```

```
## [1] "Percent of rows remaining after removing NA from 'Released' with value in 'Gross'"
```

```
((nrow(temp) - nrow(temp[is.na(temp$Released),]))/nrow(temp)) * 100
```

```
## [1] 99.01272
```

```
# 99.0 implies loosing only 1% of Gross information with values
print("Percent of rows remaining after removing NA from 'Date'")
```

```
## [1] "Percent of rows remaining after removing NA from 'Date'"
```

```
((total_rows - nrow(movie_db[is.na(movie_db$Date),]))/total_rows) * 100
```

```
## [1] 11.395
```

```
# 11.395 implies loosing significant amount of valuable data
print("Percent of rows remaining after removing NA from 'Date' with value in 'Gross'")
```

```
## [1] "Percent of rows remaining after removing NA from 'Date' with value in 'Gross'"
```

```
((nrow(temp) - nrow(temp[is.na(temp$Date),]))/nrow(temp)) * 100
```

```
## [1] 100
```

```
#100 -> Date ==Gross== NA is Zero count
# removing all NA Released implies loss of more than 10% data

temp_df <- movie_db
temp_df$Released_year <- year(ymd(temp_df$Released))

temp_df$Released_year <- ifelse(is.na(temp_df$Released_year), temp_df$Year, temp_df$Released_year)
# | (is.na(temp_df$Released_year) & !is.na(temp_df$Year))
sub_temp_df <- temp_df[(temp_df$Year != temp_df$Released_year)  &
                         (temp_df$Date != temp_df$Released_year) &
                         (temp_df$Date != temp_df$Year),]
#sub_temp_df <- sub_temp_df[(sub_temp_df$Date != sub_temp_df$Released_year), ]
```

```r
#sub_temp_df <- sub_temp_df[(sub_temp_df$Date != sub_temp_df$Year), ]
print("Number of Total Rows Removed")
```

```
## [1] "Number of Total Rows Removed"
```

```r
print(nrow(sub_temp_df))
```

```
## [1] 5005
```

```r
temp <- sub_temp_df[,c('Title', 'Year', 'Released_year', 'Date')]
print("Number of rows removed with value in 'Gross is 48, i.e less than 5%")
```

```
## [1] "Number of rows removed with value in 'Gross is 48, i.e less than 5%"
```

```r
nrow(sub_temp_df[!is.na(sub_temp_df$Gross),]) #815 to 770 to 459
```

```
## [1] 48
```

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

A: Observations: Refer the above print statements, 'Year' has no NA values, Released has ~12.5% NA values out of which less than 1% has values in 'Gross'. To keep the most amount of Data, I have filled Released_year with 'Year' if NA in 'Released'. 1. Fill Released_year with Year if Released == NA 2. Remove Year != Released_year 3. Remove Date != Released_year 4. Remove Date != Year After all the above rules, a total of 5005 rows were removed, leaving 34995 Rows intact. Within the remaining rows less than 1% of rows with 'Gross' value were removed, i.e 48 rows of total 4558 rows with value in 'Gross'.

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.

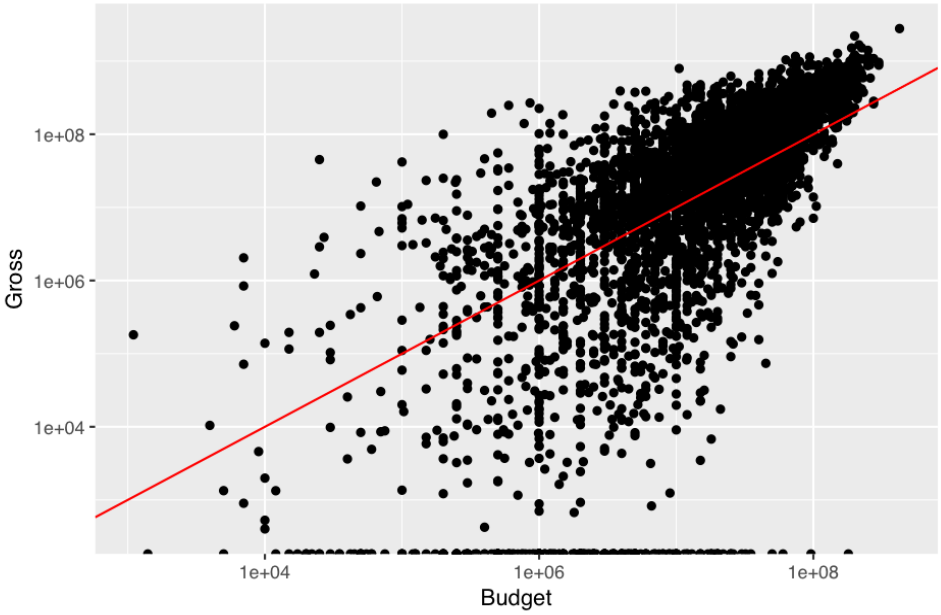
Note: To get a meaningful relationship, you may have to partition the movies into subsets such as short vs. long duration, or by genre, etc.

```r
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
# Plot Gross distribution Vs Budget
ggplot(movie_db, aes(y = Gross, x = Budget)) + geom_point() +
  scale_x_continuous(trans = 'log10') +
  scale_y_continuous(trans = 'log10') + geom_abline(color = "red") +
  ggtitle("Figure - 6: Gross Revenue Vs Budget on Log10 scale")
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 35442 rows containing missing values (geom_point).
```

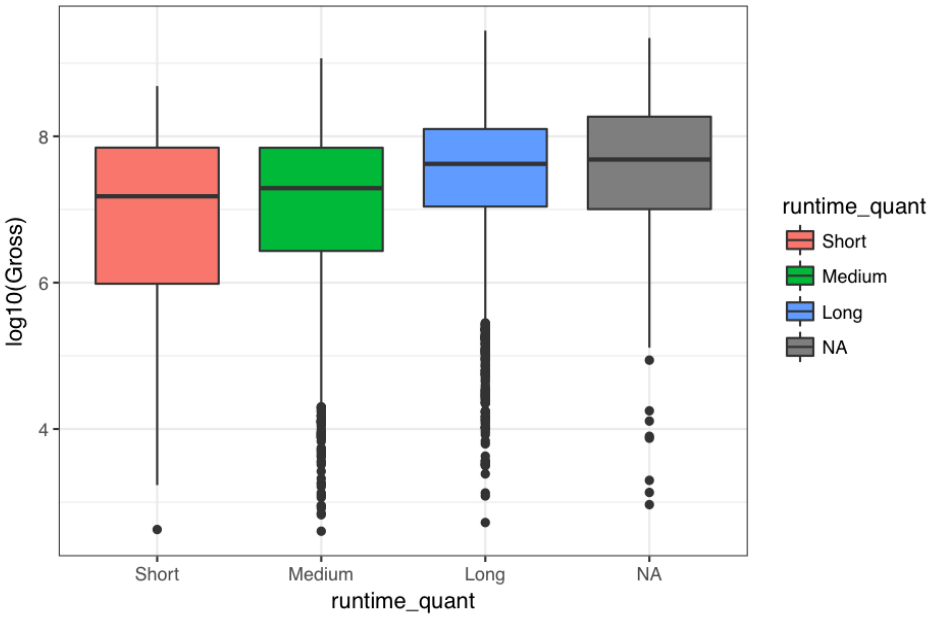
Figure – 6: Gross Revenue Vs Budget on Log10 scale




```
# Plot Gross Vs Runtime Categories (Short to Long Duration)
####### parse Runtime to categories ####
runtime_quantiles <- quantile(movie_db$Runtime, probs = seq(0, 1, 0.33), na.rm = TRUE)
runtimeQuantiles <- function(runtime) {
  cut(runtime, breaks = runtime_quantiles, labels = c( "Short", "Medium", "Long"))
}
movie_db$runtime_quant <- sapply(movie_db$Runtime, runtimeQuantiles)
ggplot(data = movie_db, aes(runtime_quant, log10(Gross), fill = runtime_quant)) +
  geom_boxplot() +
  ggtitle("Figure - 7: Gross Dist. in each Runtime Duration Category") +
  theme_bw()
```

```
## Warning: Removed 35727 rows containing non-finite values (stat_boxplot).
```

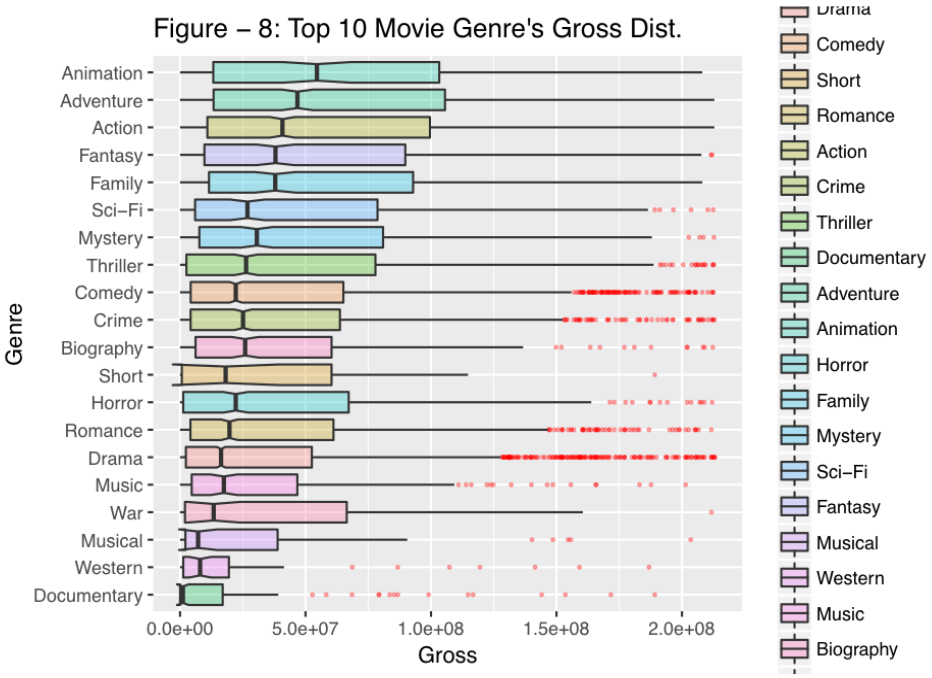
Figure – 7: Gross Dist. in each Runtime Duration Category




```
# Plot Gross distribution for top 10 most common genres
most_cmon_20 <-  as.data.frame(sort(colSums(genre_df[,5:ncol(genre_df)]), decreasing = TRUE)[1:20], row
most_cmon_20$Genre <- rownames(most_cmon_20)
colnames(most_cmon_20) <- c("Count", "Genre")
most_cmon_20$Genre <- gsub("Genre_", "", most_cmon_20$Genre)
#colnames(genre_df) <- gsub("Genre_", "", colnames(genre_df))

top_gross_runtime = melt(genre_df, id.vars = c("Title","Gross"), measure.vars = most_cmon_20$Genre)
top_gross_runtime = top_gross_runtime[!is.na(top_gross_runtime$Gross),]
top_gross_runtime = top_gross_runtime[top_gross_runtime$value!=0,]

ylim1 = boxplot.stats(top_gross_runtime$Gross)$stats[c(1, 5)] # compute lower and upper whiskers
ggplot(top_gross_runtime, aes(reorder(variable, Gross, median), Gross, fill = variable)) +
  geom_boxplot(alpha=0.3, notch=TRUE, notchwidth = 0.8, outlier.colour="red", outlier.fill="red", outli
  theme(legend.position="right") +
  ggtitle("Figure - 8: Top 10 Movie Genre's Gross Dist.") +
  xlab("Genre") +
  ylim(ylim1*0.8) +
  coord_flip()
```

```
## Warning: Removed 1448 rows containing non-finite values (stat_boxplot).
```

```
## notch went outside hinges. Try setting notch=FALSE.
## notch went outside hinges. Try setting notch=FALSE.
## notch went outside hinges. Try setting notch=FALSE.
```



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: Budget: From Figure 6, we can say that movies with higher budget has higher revenues. The correlation is ~0.74.

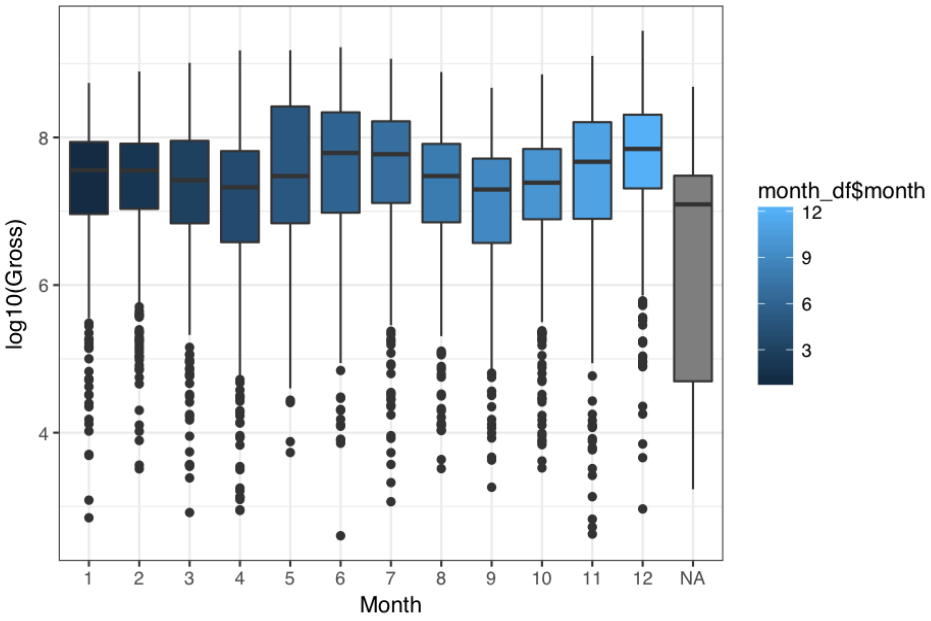
Runtime: From Figure 7, Longer movies tend to generate higher gross revenue. One observation is 2nd and 3rd quartile range is smaller for Long Runtime movies. We have to be aware of the outliers in the Long Runtime category, which says Runtime is not the only criteria for a successful movie.

Genre: From Figure 8, I looked at top 20 most common genre's. Animation, Adventure and Action are highest Grossing movie genre's whereas the Documentary, Western and Musical are lowest grossing movie genre's. Drama, Comedy, Crime and Romance does have higher number of outliers in the 4th quartile.

```
# TODO: Investigate if Gross Revenue is related to Release Month
month_df <- movie_db[,c('Released', 'Gross')]
month_df$month <- month(ymd(month_df$Released))
#month_df <- month_df[!is.na(month_df$month),]

ggplot(data = month_df, aes(reorder(month_df$month, Gross, median), log10(Gross), fill = month_df$month
  geom_boxplot() +
  ggtitle("Figure - 9: Gross Dist. Vs Month") +
  xlab("Month") +
  theme_bw()
```

```
## Warning: Removed 35727 rows containing non-finite values (stat_boxplot).
```

Figure – 9: Gross Dist. Vs Month



Q: Investigate if Gross Revenue is related to Release Month

A: Based on the Figure 9, months of (January, February) in the beginning fo the year and (June, July) in the mid year (Summer) and (November, December) at the end of year(Holiday season) have the highest Grossing Revenues.

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
awards_df <- movie_db
awards_df <- awards_df[which(awards_df$Awards != 'N/A'), ]
# awards_df <- df[sample(nrow(awards_df), 5000), ] #sampling 5000 to reduce runtime for submission
parse_awards <- function(movie_db){

  regex_awards <- function(awards) {
    temp_x <- regmatches(awards, gregexpr("+[0-9]+", awards))[[1]]
    temp_x <- as.numeric(unlist(temp_x))
  }
  movie_db$numeric_awards <- sapply(movie_db$Awards, regex_awards)
  #function(awards) as.numeric(unlist(regmatches(awards, gregexpr("+[0-9]+", awards))[[1]]) )
```

```
  movie_db$Wins = 0
  movie_db$Nominations = 0
  Count_InvalidAwards <- 0 #
  cv <- nrow(movie_db)
  for(i in 1:cv){
    if( (length(movie_db[i,]$numeric_awards[[1]])!=3) & (length(movie_db[i,]$numeric_awards[[1]])!=2) &
      if(length(grep("win",movie_db[i,]$Awards))>0){
        movie_db[i,]$Wins = as.numeric(movie_db[i,]$numeric_awards[[1]][1])
      }
      if(length(grep("nomination",movie_db[i,]$Awards))>0){
        movie_db[i,]$Nominations = as.numeric(movie_db[i,]$numeric_awards[[1]][1])
      }
    }
    if(length(movie_db[i,]$numeric_awards[[1]])==2){
      if((length(grep("win",movie_db[i,]$Awards))>0) || (length(grep("won",movie_db[i,]$Awards))>0)|| (
        movie_db[i,]$Wins = as.numeric(movie_db[i,]$numeric_awards[[1]][1])
      }
      if(length(grep("nomination",movie_db[i,]$Awards))>0){
        movie_db[i,]$Nominations = as.numeric(movie_db[i,]$numeric_awards[[1]][2])
      }
    }
    if(length(movie_db[i,]$numeric_awards[[1]])==3){
      if((length(grep("win",movie_db[i,]$Awards))>0) || (length(grep("won",movie_db[i,]$Awards))>0)|| (
        movie_db[i,]$Wins = as.numeric(movie_db[i,]$numeric_awards[[1]][2])
      }
      if(length(grep("nomination",movie_db[i,]$Awards))>0){
        movie_db[i,]$Nominations = as.numeric(movie_db[i,]$numeric_awards[[1]][3])
      }
    }
    if(length(movie_db[i,]$numeric_awards[[1]])==0){
      movie_db[i,]$Wins = 0
      movie_db[i,]$Nominations = 0
      Count_InvalidAwards = Count_InvalidAwards + 1
    }

    if(length(unlist(movie_db[i,]$Wins))==1)
      movie_db[i,]$Wins = as.numeric(unlist(movie_db[i,]$Wins))
    else
      print(paste(movie_db[i,]$Awards, movie_db[i,]$Wins, sep = " ==> " ))
    if(length(unlist(movie_db[i,]$Nominations))==1)
      movie_db[i,]$Nominations = as.numeric(unlist(movie_db[i,]$Nominations)[[1]])
    else
      print(paste(movie_db[i,]$Nominations, movie_db[i,]$Nominations, sep = " ==> " ))
  }
  print("Invalid Number of Rows after excluding 'N/A' in Awards")
  print(Count_InvalidAwards)
  return(movie_db)
}
awards_df <- parse_awards(awards_df)
```

```
## [1] "Invalid Number of Rows after excluding 'N/A' in Awards"
## [1] 0
```

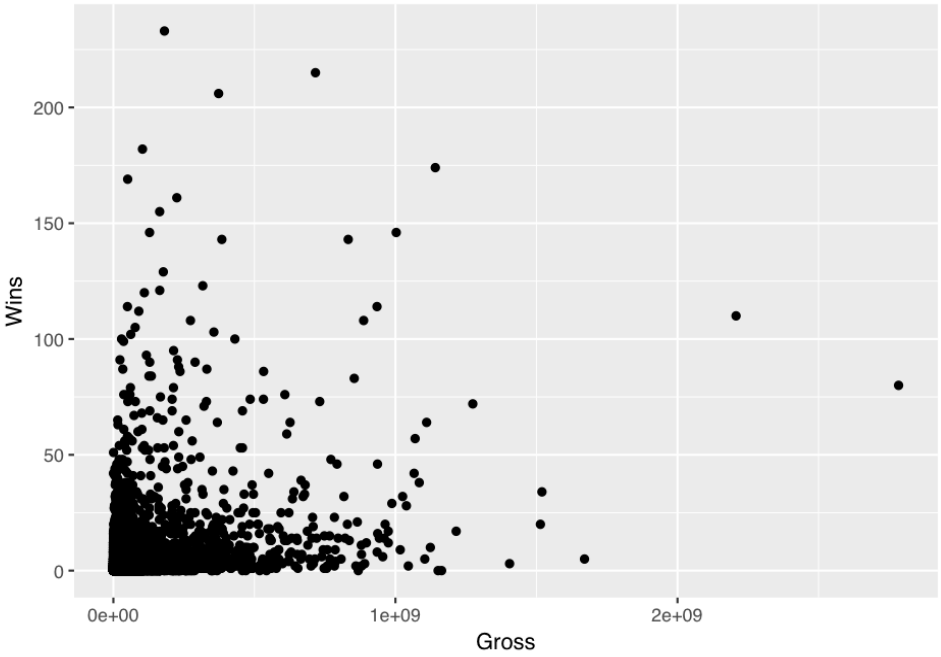
Q: How did you construct your conversion mechanism? How many rows had valid/non-zero wins or

nominations?

A: • Total Number of Rows is 40,000 out of which only 14,727 rows had non-N/A values. • "Wins", "win", "wins" etc are considered 'Wins', similarly for 'Nominations'. • Find number of numerics in the Awards o If only one numeric, assign it to either 'Wins' or 'Nominations' depending on the presence of similar forms of 'win' or 'nomination' words. o If two numerics, either assign it to (Wins, Nominations) in the order. - Considering terms like 'win', 'wins', 'won', 'nominations' e.t.c are present o If 3 numerics, assign the second to Wins and third to Nominations - Considering terms like 'win', 'wins', 'won', 'nominations' e.t.c are present

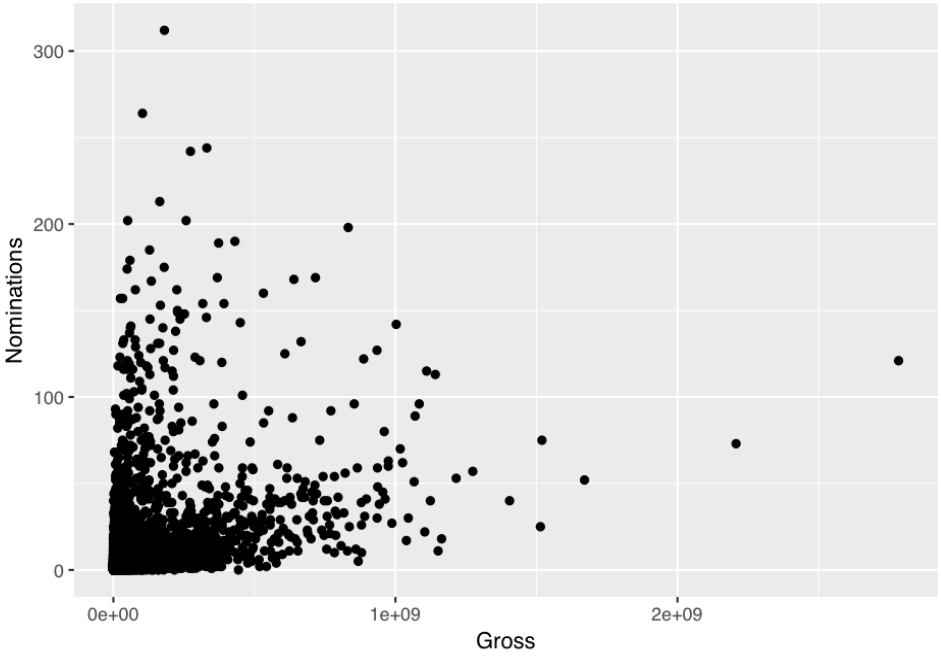
```
# TODO: Plot Gross revenue against wins and nominations
ggplot(awards_df, aes(x = Gross, y = Wins)) + geom_point()
```

```
## Warning: Removed 10948 rows containing missing values (geom_point).
```




```
ggplot(awards_df, aes(x = Gross, y = Nominations)) + geom_point()
```

```
## Warning: Removed 10948 rows containing missing values (geom_point).
```



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

A: From the above two figures Awards Vs Wins/Nominations, it is not very evident of the relationship. There is slight positive correlation is seen in the above plots. The othe way visualize this would be to categorize awards in to 3 or 4 buckets which will show a clear relation between Gross and Awards.

7. Movie ratings from IMDb and Rotten Tomatoes

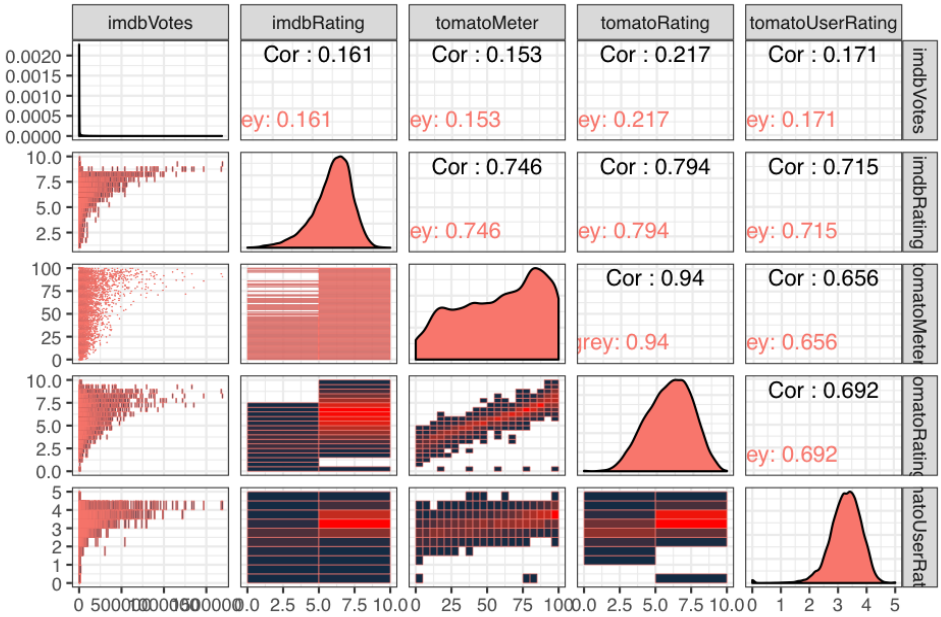
There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by several variables pre-fixed by `tomato`). Read up on such ratings on the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

Investigate the pairwise relationships between these different descriptors using graphs.

```
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
my_bin <- function(data, mapping, ..., low = "#132B43", high = "#56B1F7") {
  ggplot(data = data, mapping = mapping) +
    geom_bin2d(...) +
    scale_fill_gradient(low = low, high = high)
}

# including "tomatoUserRating" increased the Runtime
pm <- ggpairs(
  movie_db, columns = c("imdbVotes","imdbRating", "tomatoMeter","tomatoRating", "tomatoUserRating"),
  mapping=aes(color = "grey"),
  lower = list(
    combo = wrap("facethist", binwidth = 1),
```

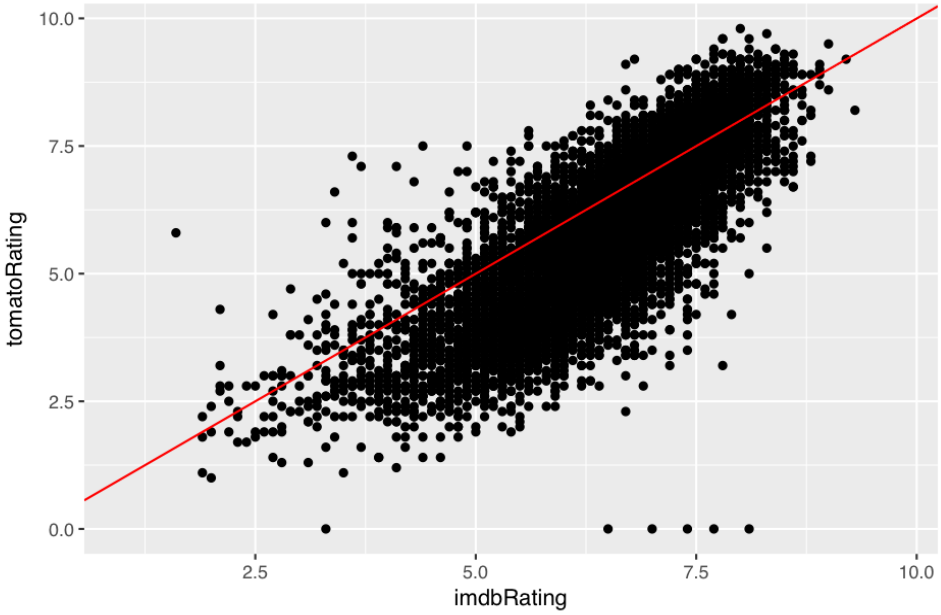
```
    continuous = wrap(my_bin, binwidth = c(5, 0.5), high = "red")
  )
) + ggtitle(" Figure - 9: ggpairs of imdb and tomato variables")
suppressWarnings(print(pm + theme_bw()))
```

Figure – 9: ggpairs of imdb and tomato variables




```
suppressWarnings(print(ggplot(movie_db, aes(y = tomatoRating, x = imdbRating)) + geom_point() +
                         geom_abline(color = "red")+
                         ggtitle("Figure - 10: imdbRating Vs tomatoRating")
))
```

Figure – 10: imdbRating Vs tomatoRating



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

A: Based on Figure -10, imdbRatings and tomatoRatings has positive linear correlation ~0.794 as expected. Based on the Figure 10 (ggpairs plot) above, tomatometer and imdbRatings also have positive correlation of ~0.75. All 3 of tomatoRating, tomatoUserRating and imdbRatings have have a bell shaped curve. imdbRating and tomatoUserRating are positively correlated(~0.715) since both are reviews are provide by users and not by only critics. imdbVotes and imdbRatings doesn't have a relation as expected, generally good movies does have more user reviews as it is watched by many.

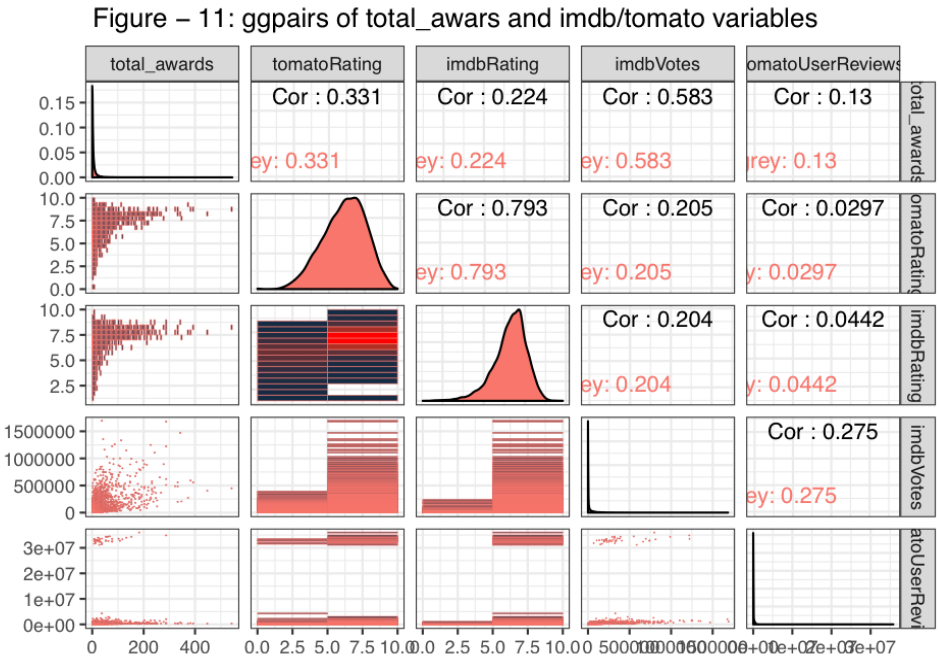
8. Ratings and awards

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
awards_df$total_awards <- awards_df$Wins + awards_df$Nominations

pm <- ggpairs(
  awards_df, columns = c("total_awards","tomatoRating","imdbRating", "imdbVotes", "tomatoUserReviews"),
  mapping=aes(color = "grey"),
  lower = list(
    combo = wrap("facethist", binwidth = 1),
    continuous = wrap(my_bin, binwidth = c(5, 0.5), high = "red")
```

```
  )
) + ggtitle(" Figure - 11: ggpairs of total_awars and imdb/tomato variables")
suppressWarnings(print(pm + theme_bw()))
```



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?

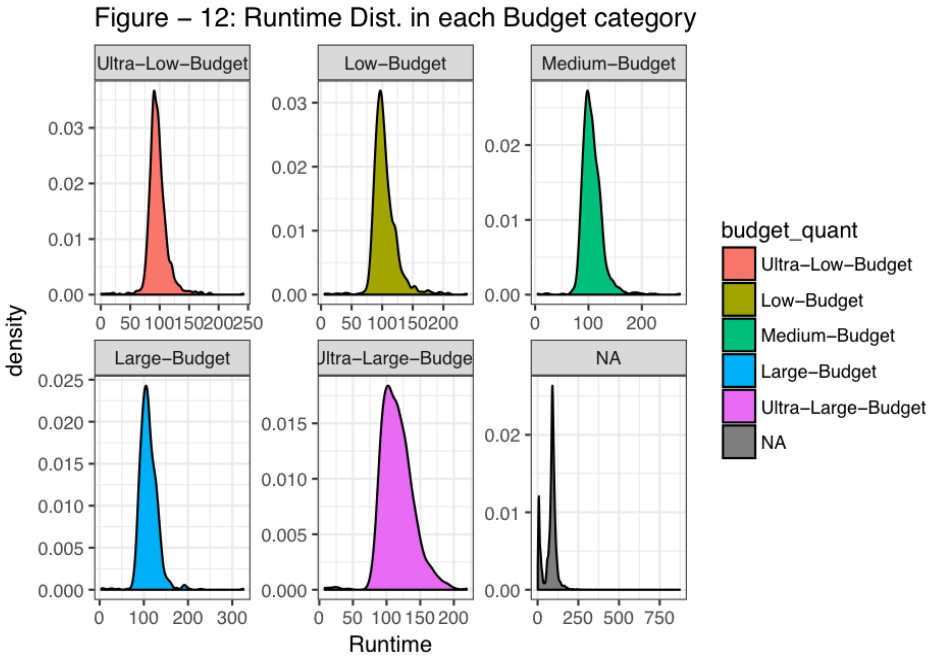
A: Based on Figure 11, a correlation ~0.6 is observed between total_awards and imdbVotes. Other than imdbVotes, total_awards has no strong positive correlation with ratings variables such as imdbRating and tomatoRating. So we can conclude that more Users do not actually care about the number of awards a movie has achieved.

9. Expected insights

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

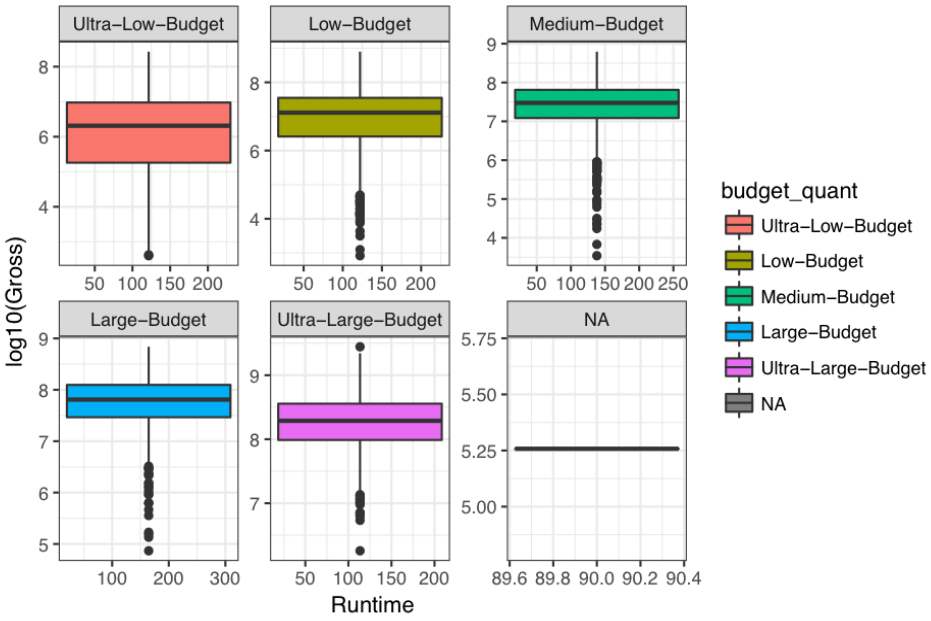
```
# TODO: Find and illustrate two expected insights
# Insight - 1; Runtime Dist. Vs budget classification
####### parse budget to categories ####
budget_quantiles <- quantile(movie_db$Budget, probs = seq(0, 1, 0.20), na.rm = TRUE)
budgetQuantiles <- function(budget) {
  cut(budget, breaks = budget_quantiles, labels = c("Ultra-Low-Budget", "Low-Budget", "Medium-Budget",
}
movie_db$budget_quant <- sapply(movie_db$Budget, budgetQuantiles)
### plots for Runtime Vs Budget Category
```

```
suppressWarnings(print(ggplot(data = movie_db, aes(Runtime, fill = budget_quant)) +
  geom_density() +
  facet_wrap(~budget_quant, scales = 'free') +
  ggtitle("Figure - 12: Runtime Dist. in each Budget category") +
  theme_bw()))
```




```
suppressWarnings(print(ggplot(data = movie_db, aes(Runtime, log10(Gross), fill = budget_quant)) +
  geom_boxplot() +
  facet_wrap(~budget_quant, scales = 'free') +
  ggtitle("Figure - 13: Gross Dist. in each Runtime Duration Category") +
  theme_bw()))
```

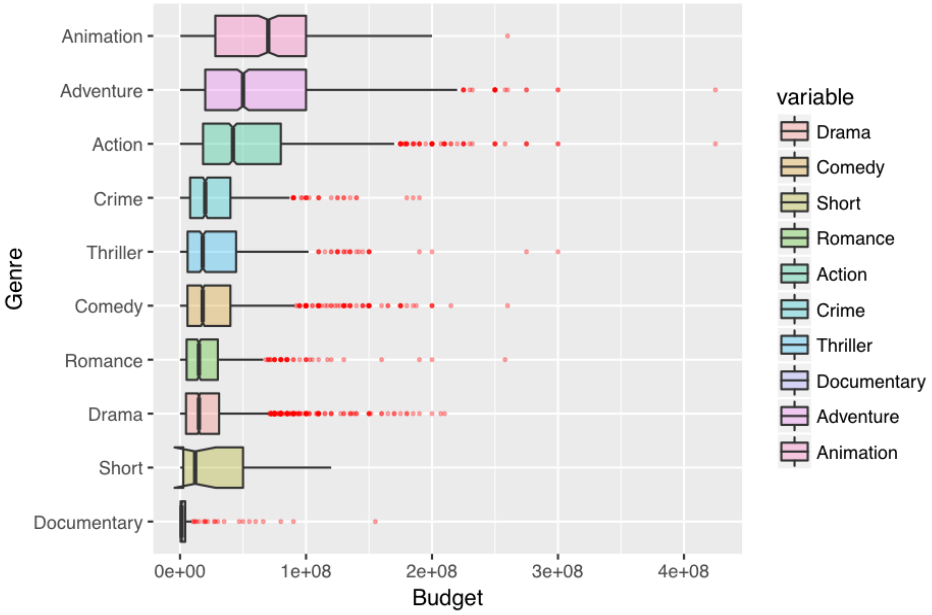
Figure – 13: Gross Dist. in each Runtime Duration Category




```
# Insight - 2; Top 10 Genres with Gross
genre_df <- movie_db[,c('Title', 'Genre', 'Budget', 'Runtime')]
genre_df <- cSplit_e(genre_df, "Genre", ",", type = "character", fill = 0) #split Genre column in to mu
########## 10 most common genres
most_cmon_10 <-  as.data.frame(sort(colSums(genre_df[,5:ncol(genre_df)]), decreasing = TRUE)[1:10], row
most_cmon_10$Genre <- rownames(most_cmon_10)
colnames(most_cmon_10) <- c("Count", "Genre")
most_cmon_10$Genre <- gsub("Genre_", "", most_cmon_10$Genre)
colnames(genre_df) <- gsub("Genre_", "", colnames(genre_df))

# Plot Runtime distribution for top 10 most common genres
top_genre_runtime = melt(genre_df, id.vars = c("Title","Budget"), measure.vars = most_cmon_10$Genre)
top_genre_runtime = top_genre_runtime[!is.na(top_genre_runtime$Budget),]
top_genre_runtime = top_genre_runtime[top_genre_runtime$value!=0,]

ggplot(top_genre_runtime, aes(reorder(variable, Budget, median), Budget, fill = variable)) +
  geom_boxplot(alpha=0.3, notch=TRUE, notchwidth = 0.8, outlier.colour="red", outlier.fill="red", outli
  theme(legend.position="right") +
  ggtitle("Figure - 14: Top 10 Movie Genre's Budget Dist.") +
  xlab("Genre") +
  coord_flip()
```

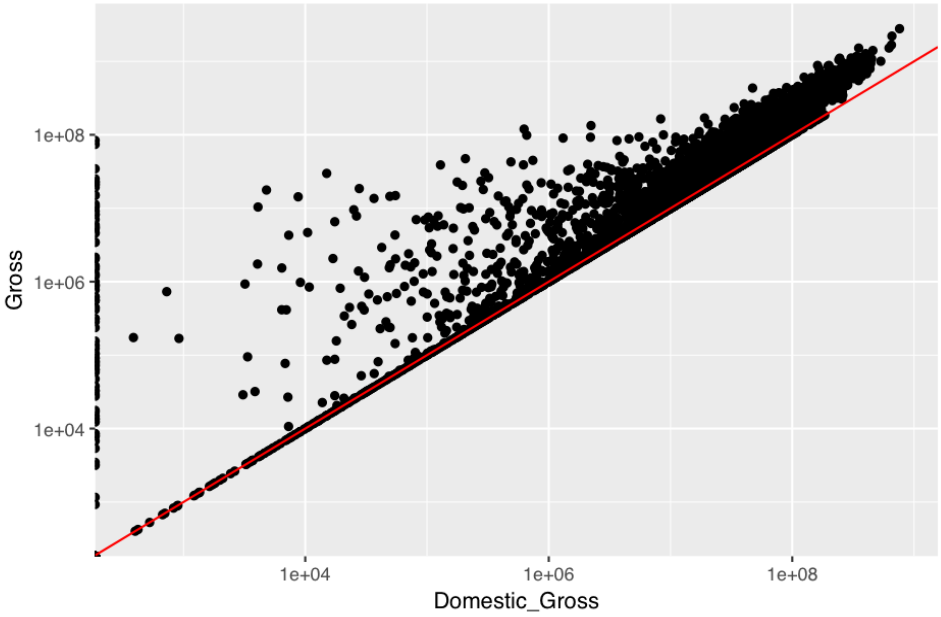
```
## notch went outside hinges. Try setting notch=FALSE.
```

Figure – 14: Top 10 Movie Genre's Budget Dist.




```
### Insight - 3
suppressWarnings(print(ggplot(movie_db, aes(y = Gross, x = Domestic_Gross)) + geom_point() +
                         scale_x_continuous(trans = 'log10') +
                         scale_y_continuous(trans = 'log10') + geom_abline(color = "red") +
                         ggtitle("Figure - 15: Gross Vs. Domestic_Gross")
                         ))
```

Figure – 15: Gross Vs. Domestic_Gross



Q: Expected insight #1.

A: Based on Figure 12, movies with Ultra-large Budget do have higher Runtimes. The mean Ultra category is close to 110+ whereas other categories are close to 100mins.

Q: Expected insight #2.

A: Based on Figure 14, as expected Genres Animation, Adventure and Action with highest Grossing are also the movies with highest budgets.

Q: Expected insight #3.

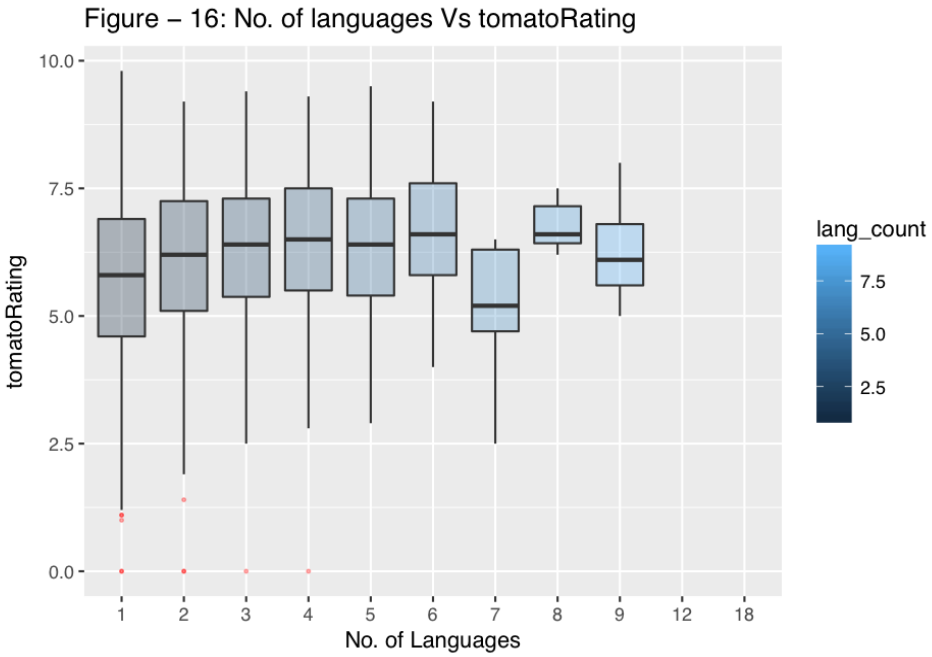
A: Based on Figure 15, as expected Gross Revenue and Domestic_Gross have high positive correlation.

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.

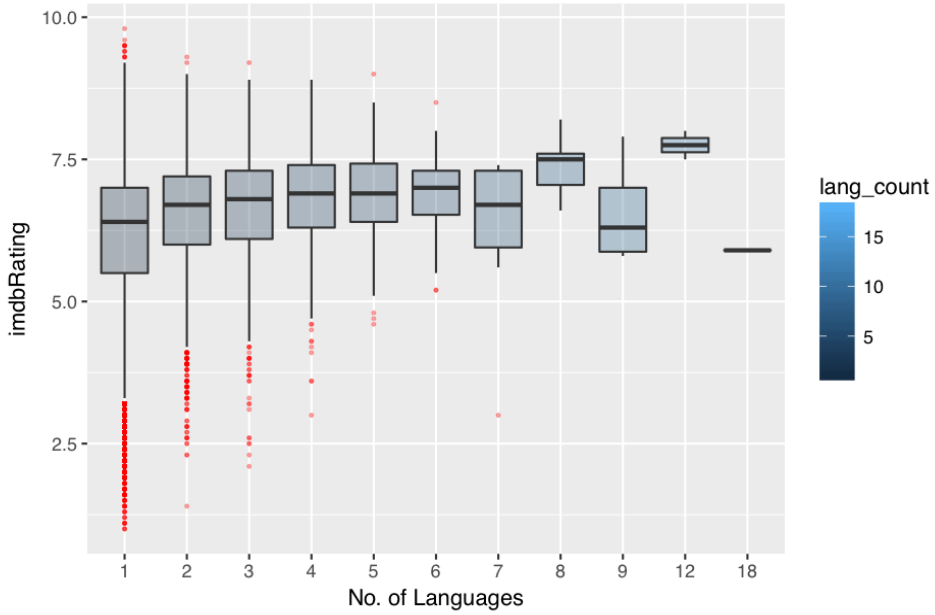
```
# TODO: Find and illustrate one unexpected insight
##### Unexpected Insight - 1 ###############
language_df <- movie_db[,c('Title', 'Language', "tomatoRating","imdbRating")]
language_df <- cSplit_e(language_df, "Language", ",", type = "character", fill = 0)
language_df$lang_count <- rowSums(language_df[,5:ncol(language_df)])
language_df <- language_df[,c('Title', 'Language', "tomatoRating","imdbRating", "lang_count")]

suppressWarnings(print(ggplot(language_df, aes(reorder(lang_count, lang_count, median), tomatoRating, f
  geom_boxplot(alpha=0.3, notch=FALSE, notchwidth = 0.8, outlier.colour="red", outlier.fill="red", outl
  theme(legend.position="right") +
```

```
  ggtitle("Figure - 16: No. of languages Vs tomatoRating") +
  xlab("No. of Languages")))
```



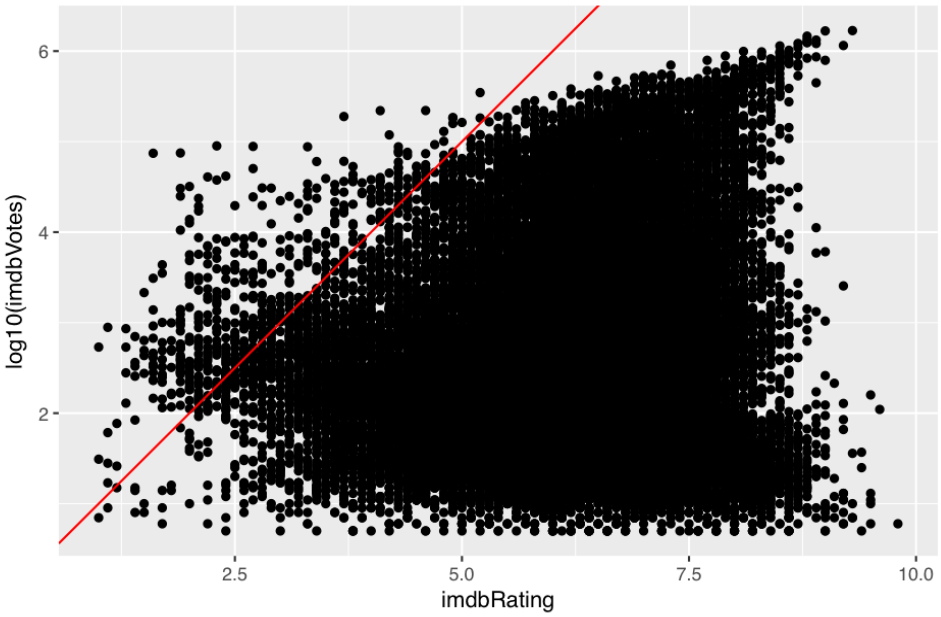

```
suppressWarnings(print(ggplot(language_df, aes(reorder(lang_count, lang_count, median), imdbRating, fil
                       geom_boxplot(alpha=0.3, notch=FALSE, notchwidth = 0.8, outlier.colour="red", o
                       theme(legend.position="right") +
                       ggtitle("Figure - 17: No. of languages Vs imdbRating") +
                       xlab("No. of Languages")))
```

Figure – 17: No. of languages Vs imdbRating




```
##### Unexpected Insight - 2 ###############
## Relation between indbVotes Vs imdbReviews
suppressWarnings(print(ggplot(movie_db, aes(y = log10(imdbVotes), x = imdbRating)) + geom_point() +
                         ggtitle("Figure - 18: imdbVotes Vs imdbRating") +
                         geom_abline(color = "red")))
```

Figure – 18: imdbVotes Vs imdbRating



Q: Unexpected insight - 1.

A: Based on Figure 16/17, I expected that higher the number of languages a movie is released, higher is the Rating. But from the plots we can see, Ratings increase from 1 to 6 languages. After 6 languages, it doesnt have a clear relation. Both imdbRating and tomatoRating exhibhit same relationship with No. of languages.

Q: Unexpected insight - 2.

A: Based on Figure 9/18, I expected that higher the number of imdb votes higher is the imdbRating. As a good movie is watched by more people and thus more votes. But the relationship has very low correlation as evident from the visuals.