project1

Weifeng She

September 13, 2016

R Markdown

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
##
## The following objects are masked _by_ '.GlobalEnv':
##
##     diamonds, mpg
```

```
library(stringr)
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(GGally)
library(reshape)
load("movies_merged")
# dimention of the data
dim(movies_merged)
```

```
## [1] 40789    39
```

1. The variable Type captures whether the row is a movie, a TV series, or a game. Remove all rows that do not correspond to movies. How many rows did you remove?

Answer:

```
#colnames(movies_merged)
table(movies_merged$Type)
```

```
##
##   game  movie series
##     64  40000    725
```

```r
# we can see there are 64 game, 725 series and 40000 movies
movies_merged = subset(movies_merged, Type == "movie")
dim(movies_merged) # after subset, there are only 40000 movies left
```

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

By table function, we can tell there are 40000 movies, 64 games and 725 Series. Then we can simply subset only movies.

2. 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 Runtime with the new numeric column. Investigate and describe the distribution of that value and comment on how it changes over years (variable Year) and how it changes in relation to the budget (variable Budget).

Answer:

```r
class(movies_merged$Runtime) # character
```

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

```r
head(sort(unique(movies_merged$Runtime)))#""1 h"        "1 h 1 min"  "1 h 10 min" "1 h 11 min" "1 h 12 m
```

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

```r
tail(sort(unique(movies_merged$Runtime))) #"95 min" "96 min" "97 min" "98 min" "99 min" "N/A"
```

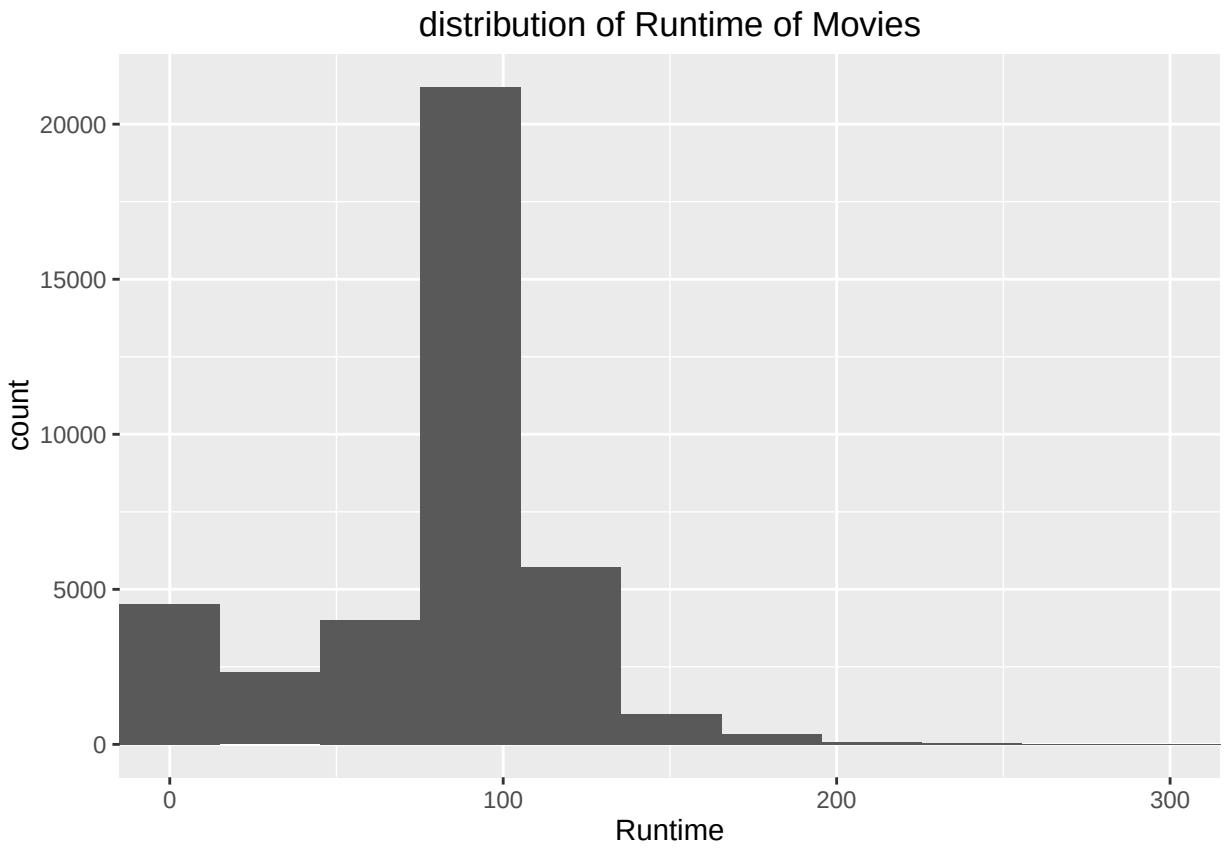
```
## [1] "95 min" "96 min" "97 min" "98 min" "99 min" "N/A"
```

We can see there are four types of Runtime variables: “N/A”, “XX min”, “X h”, “xx min X h”. We need to consider all situations

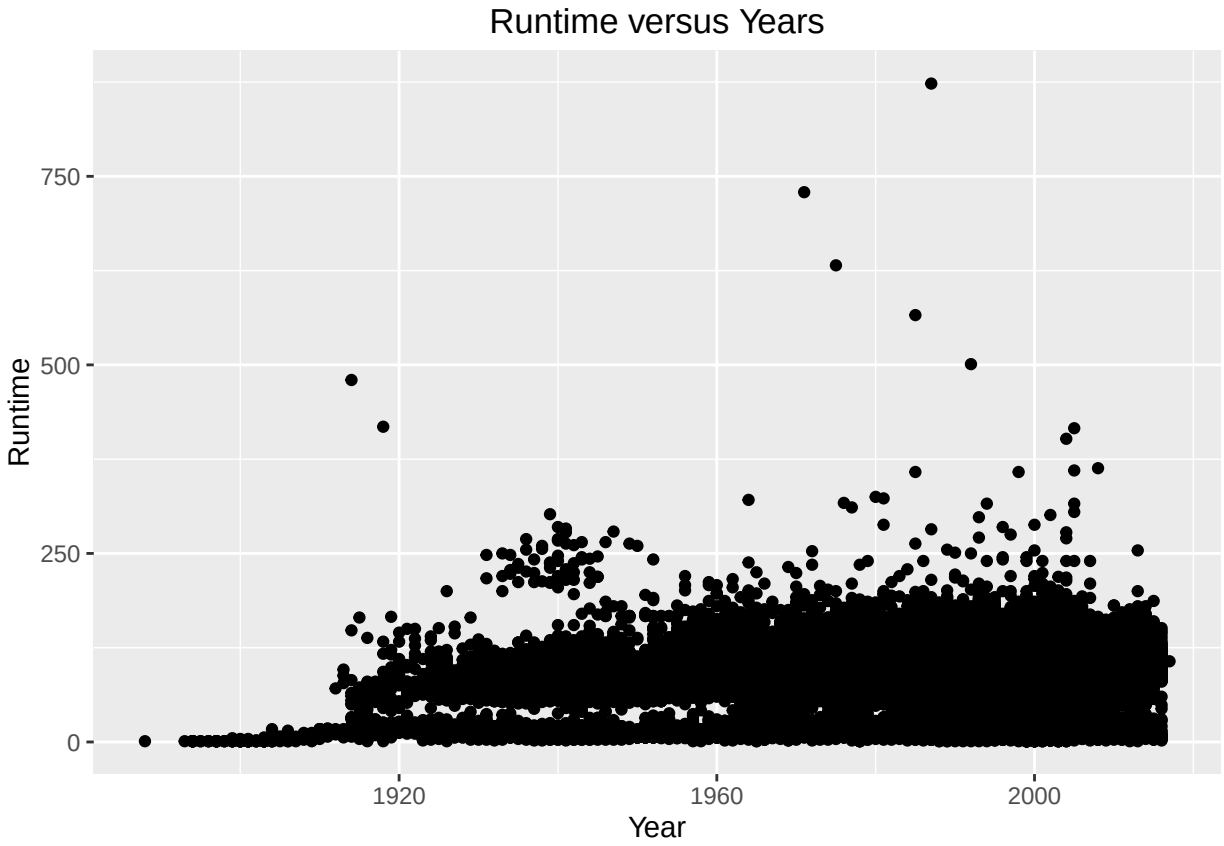
```r
# there are 4 type of Runtime, 1)"N/A", 2) "XX h xx min", 3) "xx min" 4)"xx h"

Runtime_num <- c()
for(i in seq_along(movies_merged$Runtime)){

        x <- strsplit(movies_merged$Runtime[i], ' ')[[1]]
        if (length(x) ==  2){
                if(x[2] == "min") #3) "xx min"
                {Runtime_num <- c(Runtime_num, suppressWarnings(as.numeric(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.numeric(x[1])) * 60  + suppressWarning
        }
        # 1)"N/A"
```

```
        else Runtime_num <- c(Runtime_num, NA)
}

# combine the row of Runtime with Year to a new dataframe
Runtime_Year_merged <- data.frame(Runtime = Runtime_num , movies_merged$Year)
colnames(Runtime_Year_merged) <- c("Runtime", "Year")

# remove all the NA columns of Runtime_Year_merged dataframe
Runtime_Year_merged <- Runtime_Year_merged[complete.cases(Runtime_Year_merged),]

ggplot(Runtime_Year_merged, aes(Runtime)) + geom_histogram() + ggtitle("distribution of Runtime of Movi
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

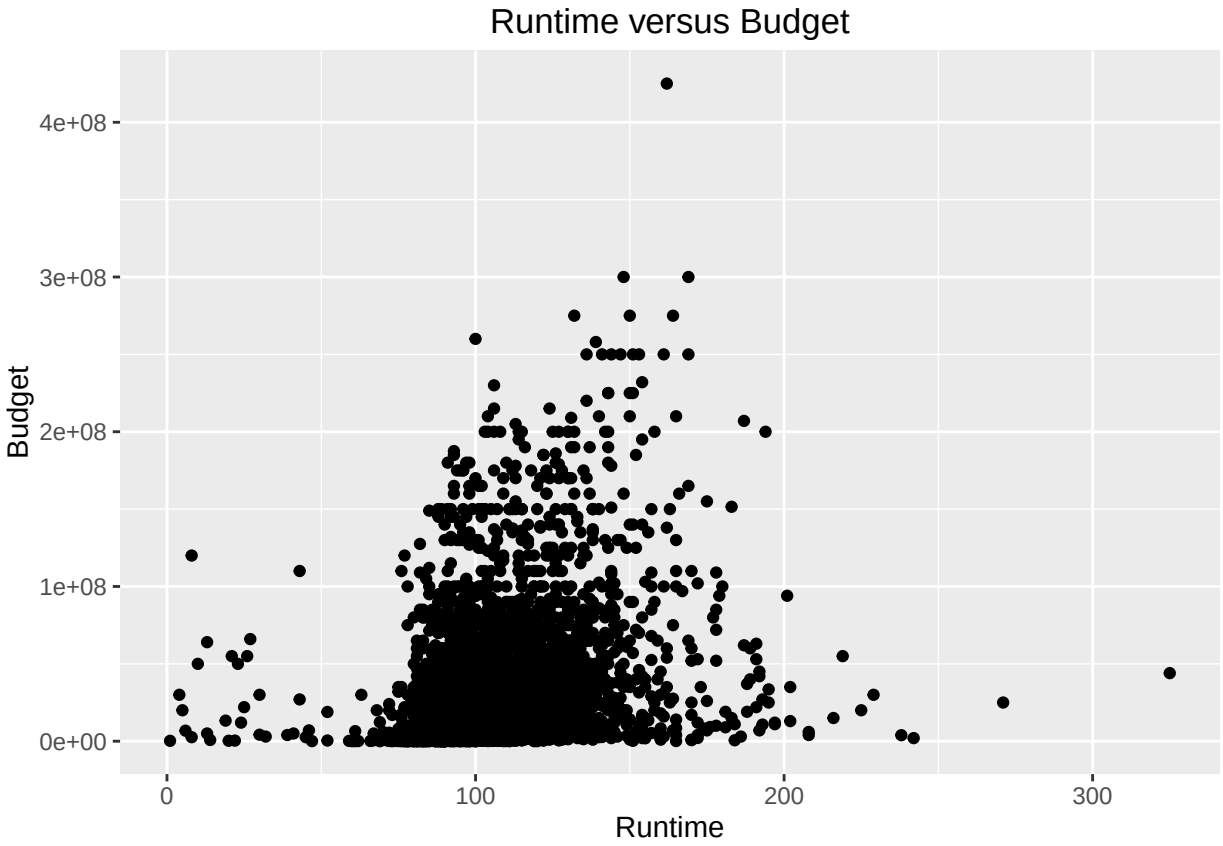



```
ggplot(data = Runtime_Year_merged, aes(Year, Runtime)) + geom_point() + ggtitle("Runtime versus Years")
```



We can see the runtime for most movies is around little less than 100 minutes. At the beginning, the runtime is very short before 1920, then it starts to increase gradually with years until around 1950s. Then it stays very stable except for some outlines.

```
# combine the row of Runtime with Budget to a new dataframe
Runtime_Budget_merged <- data.frame(Runtime = Runtime_num , movies_merged$Budget)
colnames(Runtime_Budget_merged) <- c("Runtime", "Budget")

Runtime_Budget_merged <- Runtime_Budget_merged[complete.cases(Runtime_Budget_merged),]

ggplot(data = Runtime_Budget_merged, aes(Runtime, Budget)) + geom_point() + ggtitle("Runtime versus Budg
```



We can see budget vesus runtime is clearly a normal distribution, except few skewed points at the upper right side. That means in most cases, budget does not affect runtime and only in some cases, higher budget films tend to have longer runtime.

3. 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. 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). Graph and describe the relative proportions of titles having the top 10 genres and examine how the distribution of gross revenue (variable Gross) changes across genres.

Answer:

```
# only work on the Genre and Gross columns of movies_merged

Genre_Gross_merged <- movies_merged[, c("Genre", "Gross")]

sum((Genre_Gross_merged$Genre == "N/A")) ## there are 986 rows is na for Genre
```

```
## [1] 986
```

```
sum(is.na(Genre_Gross_merged$Gross)) # but no NA rows
```

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

```
## here we can see some rows is "N/A"
##we need to change to NA and then remove this rows
# convert Genre column's "N/A" to NA
Genre_Gross_merged$Genre <- ifelse(Genre_Gross_merged$Genre == "N/A", NA, Genre_Gross_merged$Genre)
#Test to see if it is converted successfully
sum(is.na(Genre_Gross_merged$Genre))
```

```
## [1] 986
```

```
## remove the NA rows of genre column
Genre_Gross_merged <- Genre_Gross_merged[complete.cases(Genre_Gross_merged[, 1]),]
dim(Genre_Gross_merged) # 39014 X 2
```

```
## [1] 39014     2
```

```
# replace "," with " " in Genre column to prepare string for dictionary construction
Genre_Gross_merged$Genre <- gsub(",", " ", Genre_Gross_merged$Genre)
# use tm library to convert Genre columns to binary
corp <- Corpus(VectorSource(Genre_Gross_merged$Genre))


dtm <- DocumentTermMatrix(corp)
Genre_df <- data.frame(as.matrix(dtm))
# add Gross column to the new dataframe
Genre_df$Gross <- Genre_Gross_merged$Gross
head(Genre_df)
```

```
##   action adult adventure animation biography comedy crime documentary
## 1      0     0         0         0         1      0     0           1
## 2      0     0         0         0         0      0     0           0
## 3      0     0         0         0         0      0     0           1
## 4      0     0         0         0         0      0     0           0
## 5      0     0         0         0         0      0     0           0
## 6      0     0         0         0         1      0     0           1
##   drama family fantasy film.noir game.show history horror music musical
## 1     0      0       0         0         0       0      0     0       0
## 2     0      0       0         0         0       0      0     0       0
## 3     0      0       0         0         0       0      0     0       0
## 4     1      0       0         0         0       0      0     0       0
## 5     0      0       0         0         0       0      0     0       0
## 6     0      0       0         0         0       0      0     0       0
##   mystery news reality.tv romance sci.fi short sport talk.show thriller
## 1       0    0          0       1      0     0     0         0        0
## 2       0    0          0       0      0     1     0         0        1
## 3       0    0          0       0      0     0     0         0        0
## 4       0    0          0       1      0     0     0         0        0
## 5       0    0          0       0      0     1     0         0        0
## 6       0    0          0       0      0     0     0         0        0
##   war western Gross
## 1   0       0    NA
## 2   0       0    NA
## 3   0       0    NA
```

```
## 4   0       0    NA
## 5   1       0    NA
## 6   0       0    NA
```

```
# get the top 10 Genres
sorted_genre <- sort(colSums(Genre_df[, 1:28]), decreasing = TRUE)[1:10]
name_list <- names(sorted_genre)
# only subset the top 10 Genres and complete.cases
sub_Genre_df <- Genre_df[,c(names(sorted_genre), "Gross")]
dim(sub_Genre_df) ## 39014 X 11
```

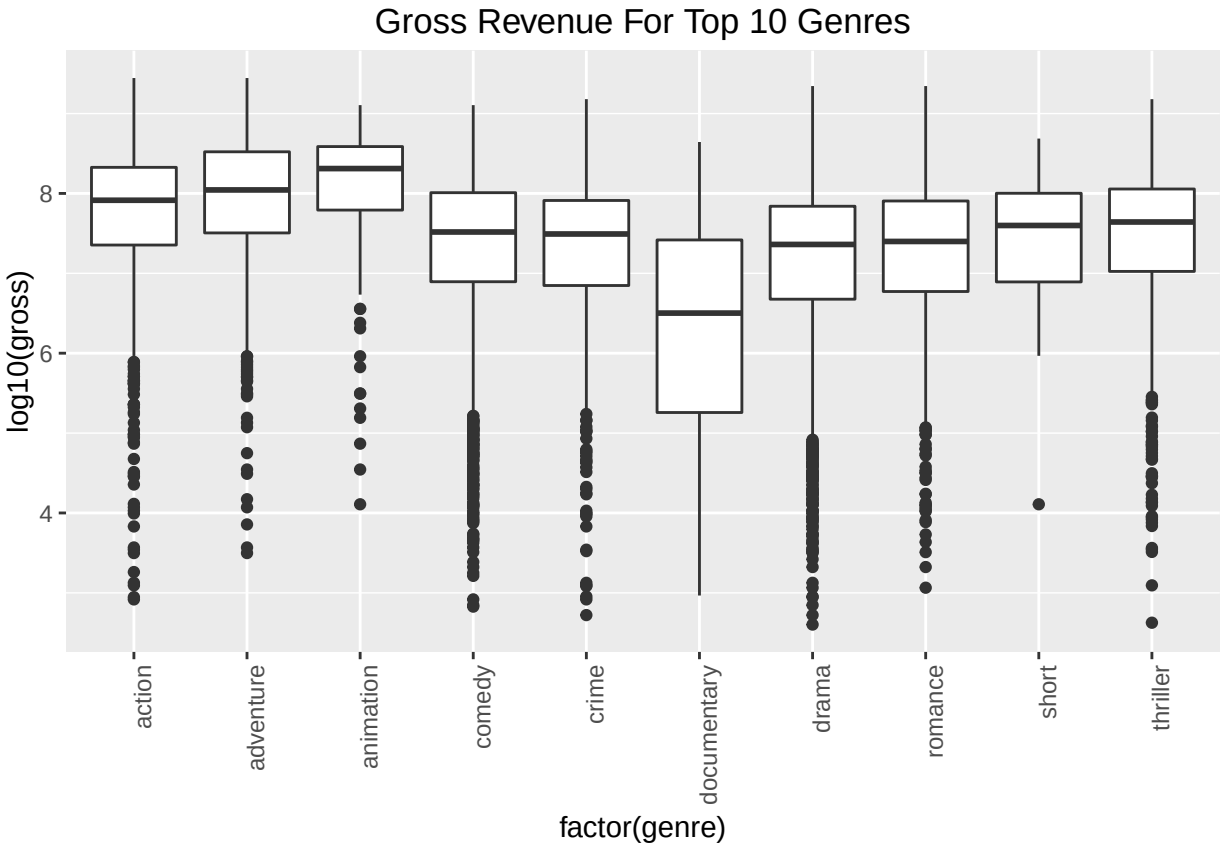
```
## [1] 39014    11
```

```
sub_Genre_df <- sub_Genre_df[complete.cases(sub_Genre_df$Gross),] #4555 X 11

# nowe we can create a data.frame with one column for the top 10 Genres and another column for correspo
Genre_df_long <- data.frame(genre = character(), gross = integer())

for(i in 1:10) {
        sub <- sub_Genre_df[sub_Genre_df[, i] == 1, c(i, 11)]
        #print(dim(sub))
        sub[, 1] <- name_list[i]
        Genre_df_long <- rbind(as.matrix(Genre_df_long), as.matrix(sub))
}

dim(Genre_df_long) # dim =  8469 X 2
```

```
## [1] 8469    2
```

```
Genre_df_long <- as.data.frame(Genre_df_long)
Genre_df_long$gross <- as.numeric(as.character(Genre_df_long$gross))
ggplot(Genre_df_long, aes(factor(genre), log10(gross))) + geom_boxplot() + ggtitle("Gross Revenue For T
theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
## Warning: Removed 421 rows containing non-finite values (stat_boxplot).
```



For the top 10 Genres, we can see animation has the highest revenue and documentary has the lowest revenue.

4. 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 source's release time was represented by the column Year (numeric representation of the year) and the second by the column Release (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 present Gross variable. What is your precise removal logic and how many rows did you end up removing?

Answer: The following strategy is used to remove the mismatch between Year and Released columns. First I checked how many NA is each column. Since there are 4949 NAs in Released column and 0 NA in Year column. If we remove all NA rows, we could remove more than 10% of the data. Furthermore, NA maybe is not caused by mergeing and it could come from original data. So I did not remove the NA rows of the Released columns. Then I extracted the Year variable from Release column and matched it with Year column. I found only 34273 rows matched. Then I also kept the rows although which do not match between Release_year and Year, but is not NA for Gross column. So finally I kept 35043 rows.

```
# first investigate the data
head(movies_merged[, c("Released", "Year", "Gross")])
```

```
##     Released Year Gross
## 1 2005-04-08 2005    NA
## 2 2005-01-25 2005    NA
## 3 2005-02-24 2005    NA
## 4 2005-01-20 2005    NA
```

```
## 5 2005-03-12 2005    NA
## 6 2005-01-23 2005    NA
```

```
sum(is.na(movies_merged$Released)) ## there are 4949 NA in Released column
```

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

```
sum(is.na(movies_merged$Year)) ## 0 NA in Year column
```

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

```
sum(is.na(movies_merged$Gross)) ## there are about 7/8 of NA in Gross column
```

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

```
# check if there are any mismatch between Year and Released
sub_movies <- movies_merged[,c("Released", "Year", "Gross")]
# extract the year information from Released column
sub_movies$Release_year <- as.numeric(substr(sub_movies$Released, 1, 4))
# then convert all the NA column of Release_year to the same value to avoid remove more rows
sub_movies$Release_year <- ifelse(is.na(sub_movies$Release_year), sub_movies$Year, sub_movies$Release_y

# first look the first 10 rows by eye
head(sub_movies, 10) # we can see row #7 has different values
```

```
##      Released Year Gross Release_year
## 1  2005-04-08 2005    NA         2005
## 2  2005-01-25 2005    NA         2005
## 3  2005-02-24 2005    NA         2005
## 4  2005-01-20 2005    NA         2005
## 5  2005-03-12 2005    NA         2005
## 6  2005-01-23 2005    NA         2005
## 7  2005-06-03 2000    NA         2005
## 8        <NA> 1999    NA         1999
## 9  2005-04-13 2005    NA         2005
## 10 2014-05-01 2014    NA         2014
```

```
# then count how many rows have different released year and Year value
matchedIndex <- sub_movies$Year == sub_movies$Release_year
sum(matchedIndex) # 5727
```

```
## [1] 34273
```

```
# further keep the rows Gross is not na
matchedIndex <- ifelse(!is.na(sub_movies$Gross), TRUE, matchedIndex)
sum(matchedIndex) # 35043
```

```
## [1] 35043
```

```
sub_movies_merged <- movies_merged[matchedIndex, ]
```

5. An important question is when to release a movie. Investigate the relationship between release date and gross revenue and comment on what times of year are most high revenue movies released in. Does your answer changes for different genres? Based on the data, can you formulate a genre-based recommendation for release date that is likely to increase the title's revenue? If you have a recommendation motivate it with the appropriate disclaimers, or otherwise explain why you are unable to produce a recommendation.

Answer: To investigate the relationship between release date and gross revenue, I extract the month information from Released column and graphed gross revenue vesus month. We can see movies released at summer (around June, July) and winter(December) tend to have higher higher revenue.

```
sub_movies <- movies_merged[, c("Released", "Genre", "Gross")]

sum(is.na(movies_merged$Gross)) #there are 35442 NAs for Gross column
```

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

```
sub_movies <- sub_movies[complete.cases(sub_movies),]
dim(sub_movies) # only 4513 rows left
```

```
## [1] 4513    3
```

```
# extract the month from released date
sub_movies$Month <- as.factor(substr(sub_movies$Released, 6, 7))
#Then further remove the NA rows for genre column

sub_movies$Genre <- ifelse(sub_movies$Genre == "N/A", NA, sub_movies$Genre)

sub_movies <- sub_movies[complete.cases(sub_movies),]
dim(sub_movies) # further remove 3 rows
```

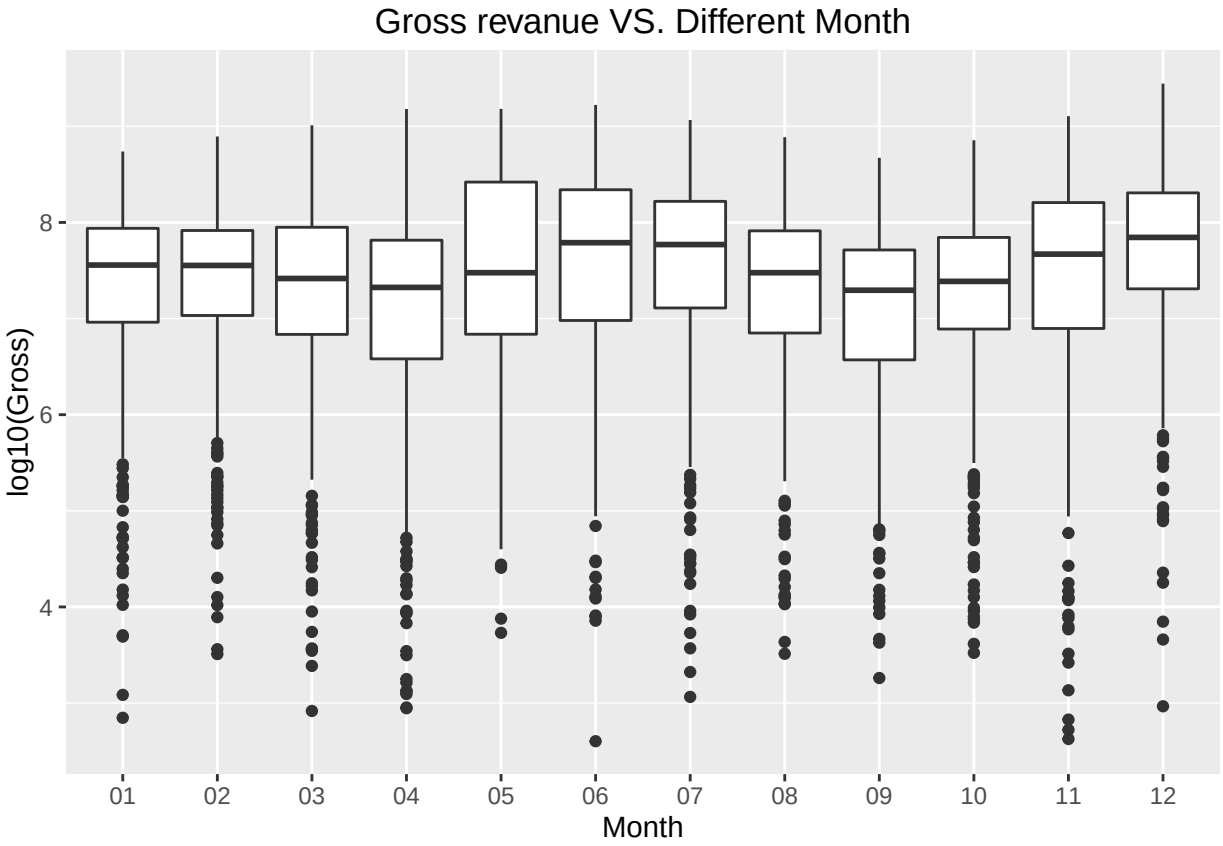
```
## [1] 4510    4
```

```
sub_movies$Genre <- gsub(",", " ", sub_movies$Genre)
corp <- Corpus(VectorSource(sub_movies$Genre))
dtm <- DocumentTermMatrix(corp)
Genre_matrix <- as.matrix(dtm)

sub_movies_with_genre_one_hot <- data.frame(cbind(sub_movies,Genre_matrix))
dim(sub_movies_with_genre_one_hot)
```

```
## [1] 4510   29
```

```
p <- ggplot(sub_movies_with_genre_one_hot, aes(Month, log10(Gross)))
p + geom_boxplot() + ggtitle(" Gross revanue VS. Different Month ")
```

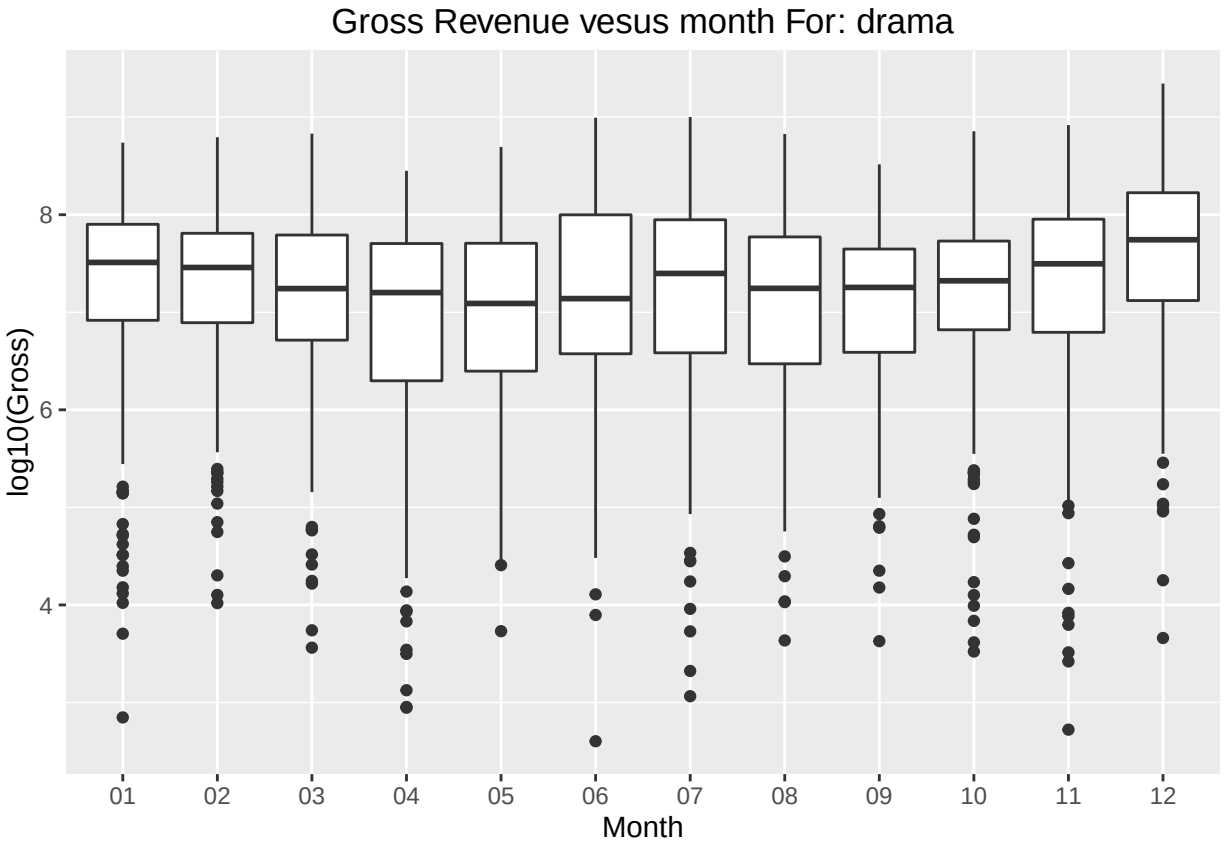
```
## Warning: Removed 258 rows containing non-finite values (stat_boxplot).
```



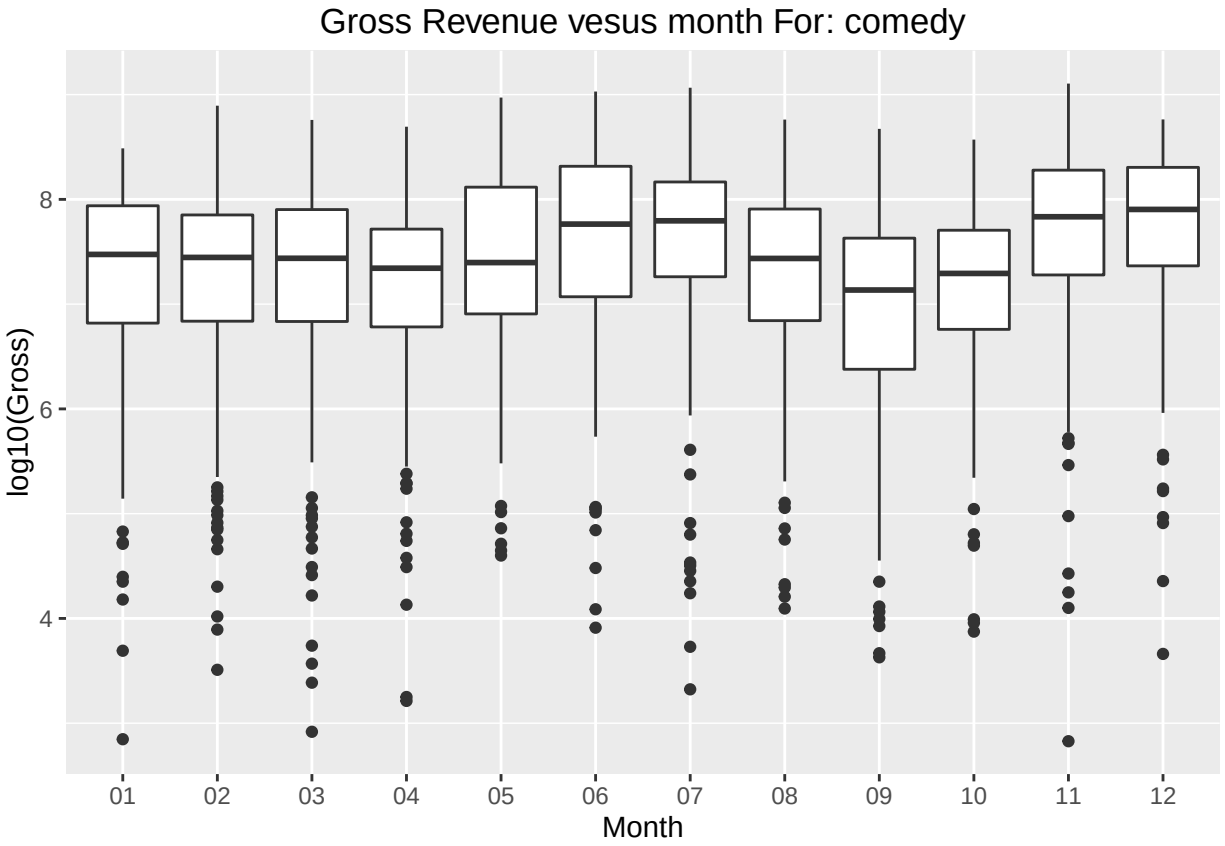
Next we try to investigate if this relationship changed for diffrent genres. Here we only analyzed top 10 genres. from the boxplot of each genre, we can see most individual genres also keep the same trend, with higher revenue around summer and winter. There also has some exceptions, such as horror movie. It tends to highest revenue at October. Since we know Hollowen is at October and people tend to watch more horror movies before hollowen.The conclusion here is we can use this system to recommend the release date for a movie based on it's genre.

```
#Find the top 10 genres
name_list <- names(sort(colSums(sub_movies_with_genre_one_hot[, 5:29]), decreasing = TRUE)[1:10])

# only subset the corresponding columns
sub_movies_with_genre_one_hot <- sub_movies_with_genre_one_hot[, c(name_list, "Month", "Gross")]
# for each top 10 genre, graph its gross revenue for each month
for(i in 1:10) {
        sub <- sub_movies_with_genre_one_hot[sub_movies_with_genre_one_hot[, i] == 1, c(i, 11, 12)]
        ttl <- paste("Gross Revenue vesus month For:", name_list[i], sep = " ")
        p <- ggplot(sub, aes(Month,log10(Gross)))
        print(p + geom_boxplot() + ggtitle(ttl))
}
```

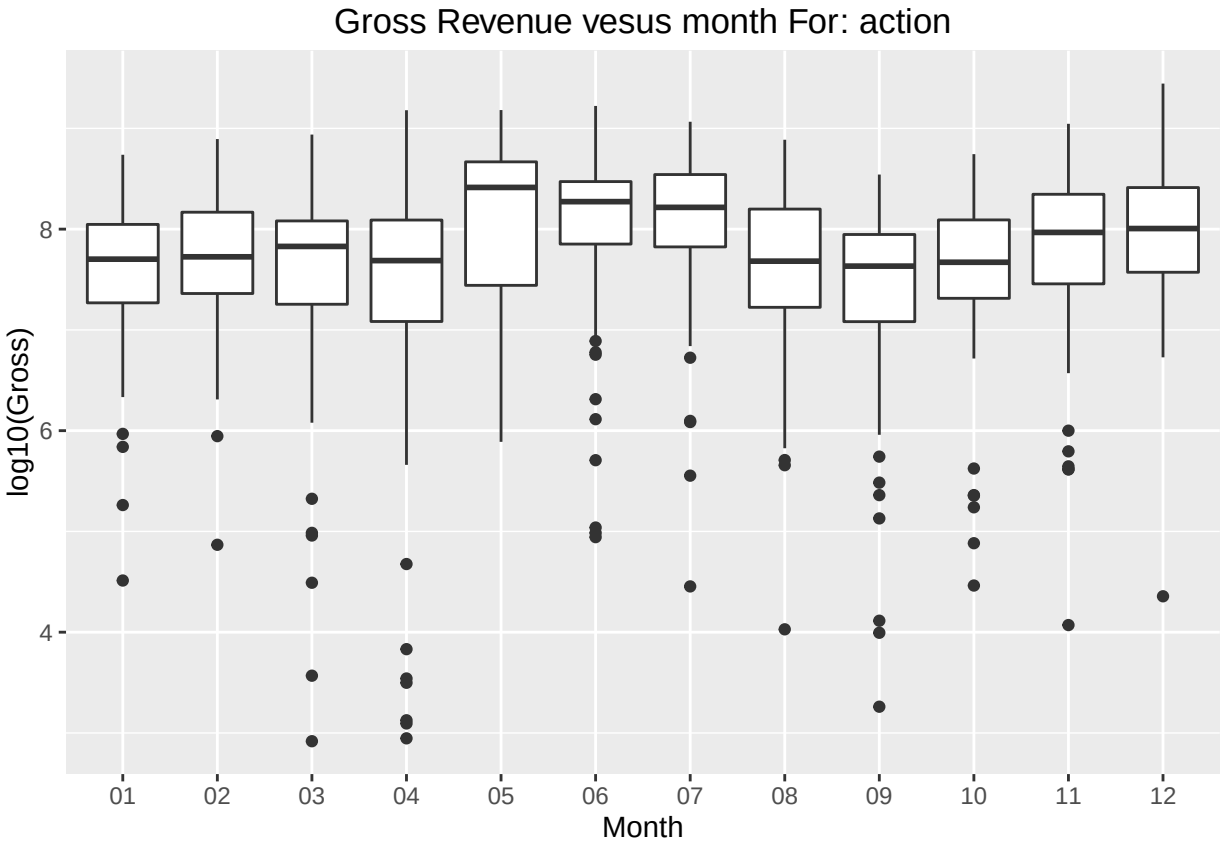
```
## Warning: Removed 113 rows containing non-finite values (stat_boxplot).
```



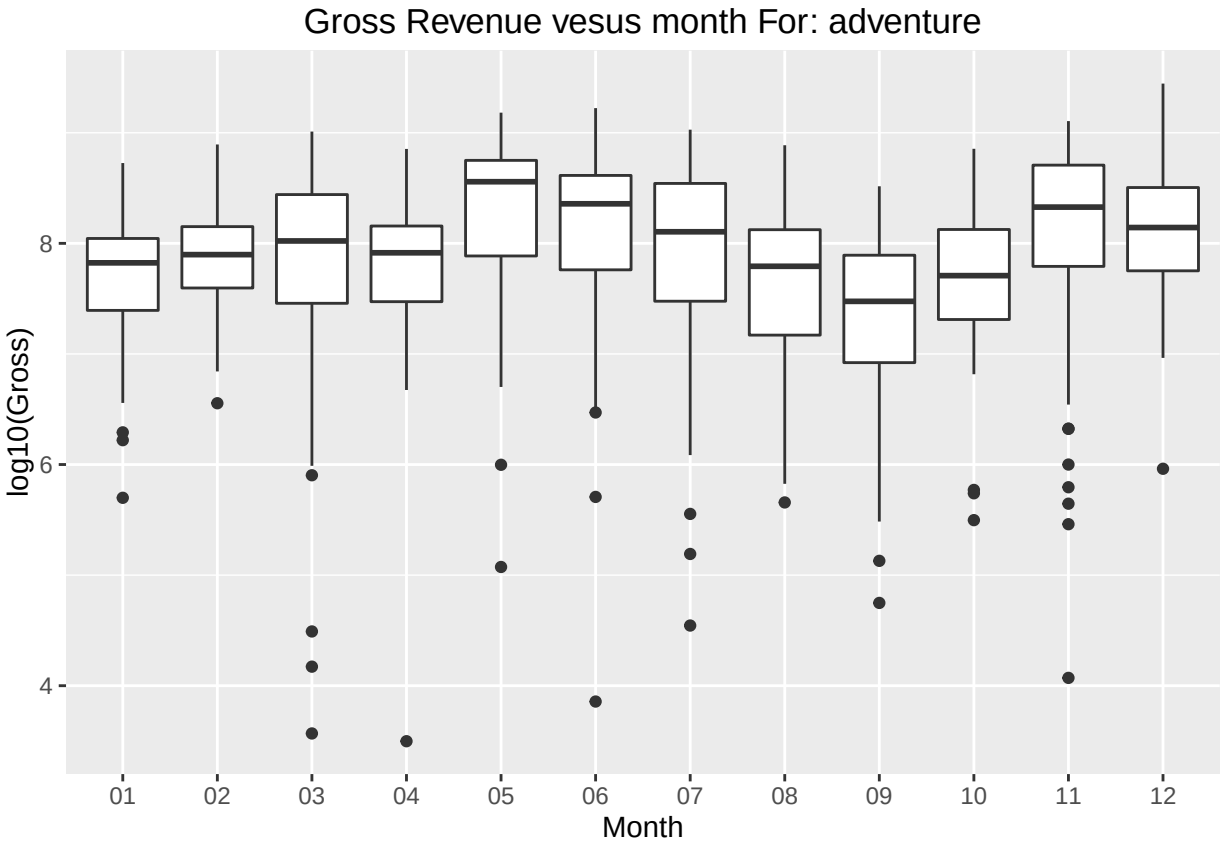

```
## Warning: Removed 72 rows containing non-finite values (stat_boxplot).
```



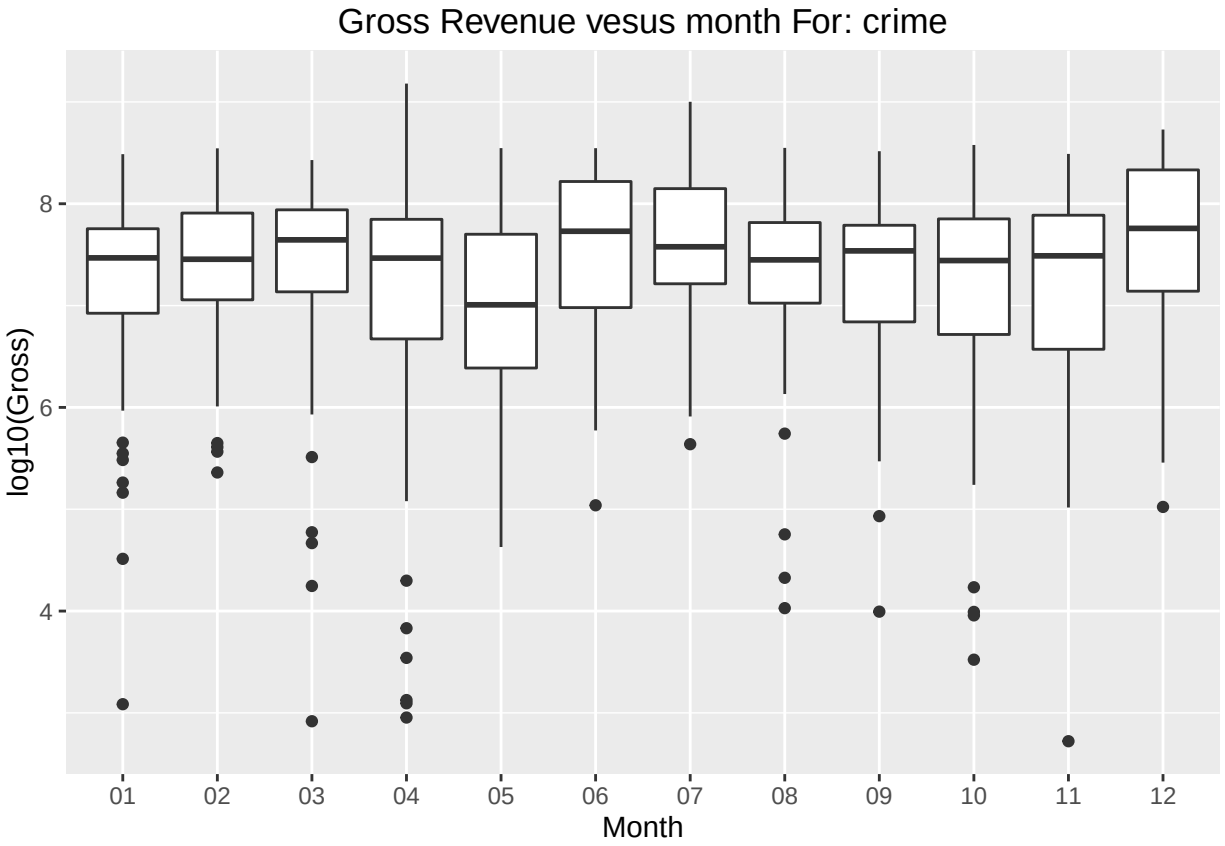

```
## Warning: Removed 52 rows containing non-finite values (stat_boxplot).
```



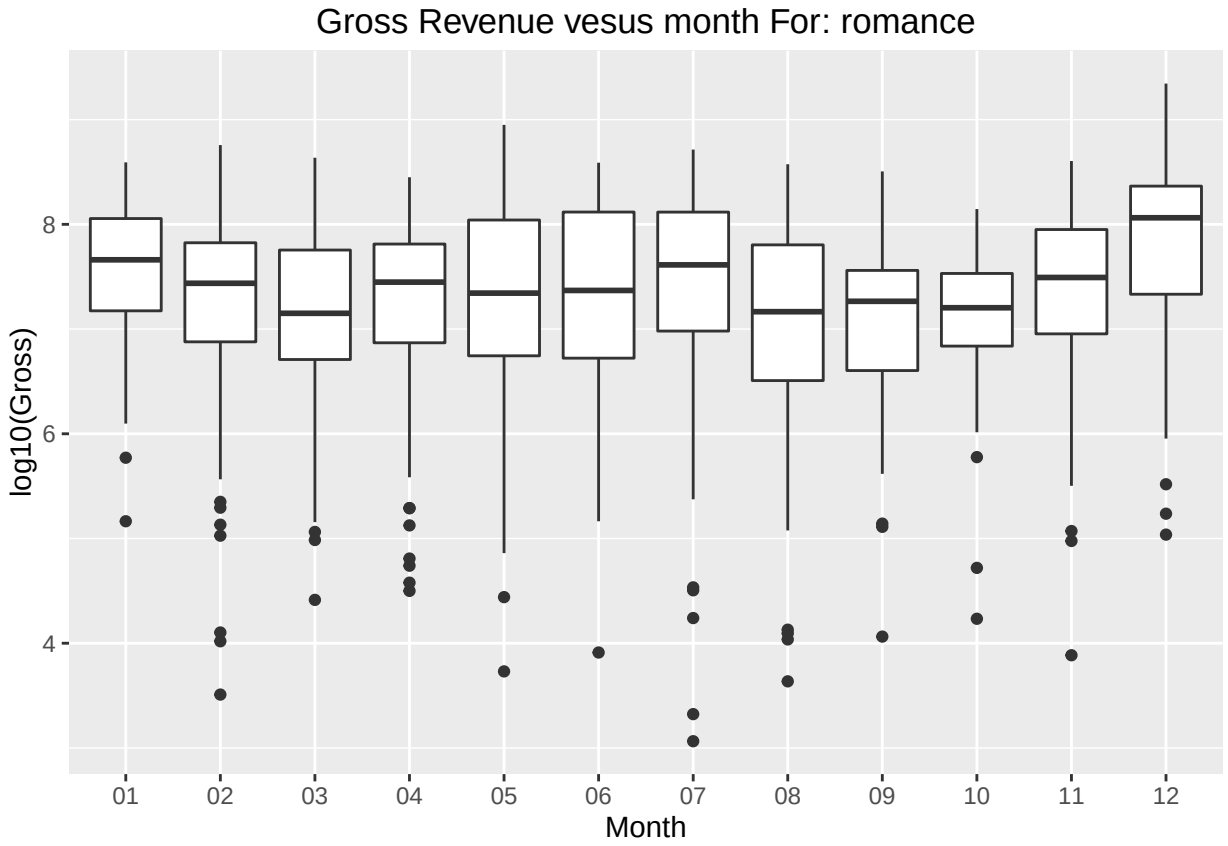

```
## Warning: Removed 14 rows containing non-finite values (stat_boxplot).
```



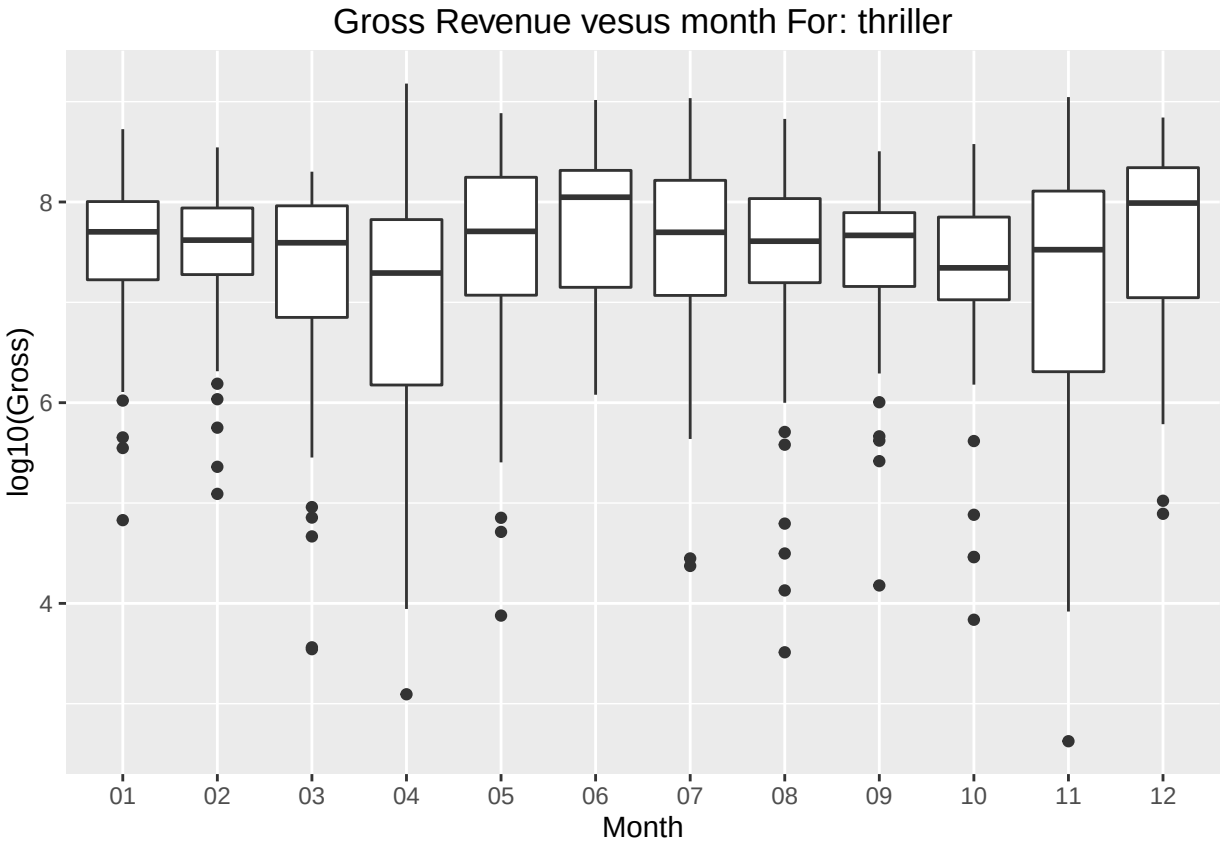

```
## Warning: Removed 30 rows containing non-finite values (stat_boxplot).
```



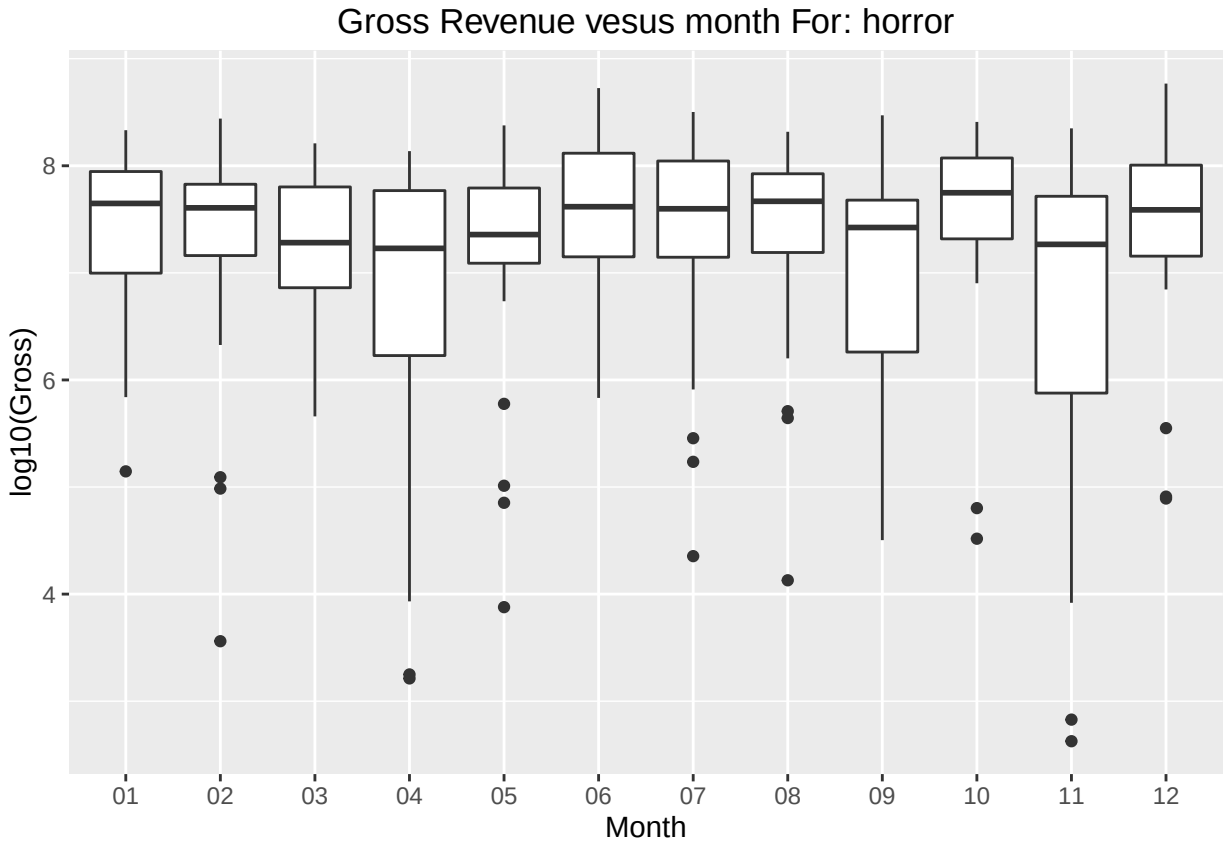

```
## Warning: Removed 21 rows containing non-finite values (stat_boxplot).
```



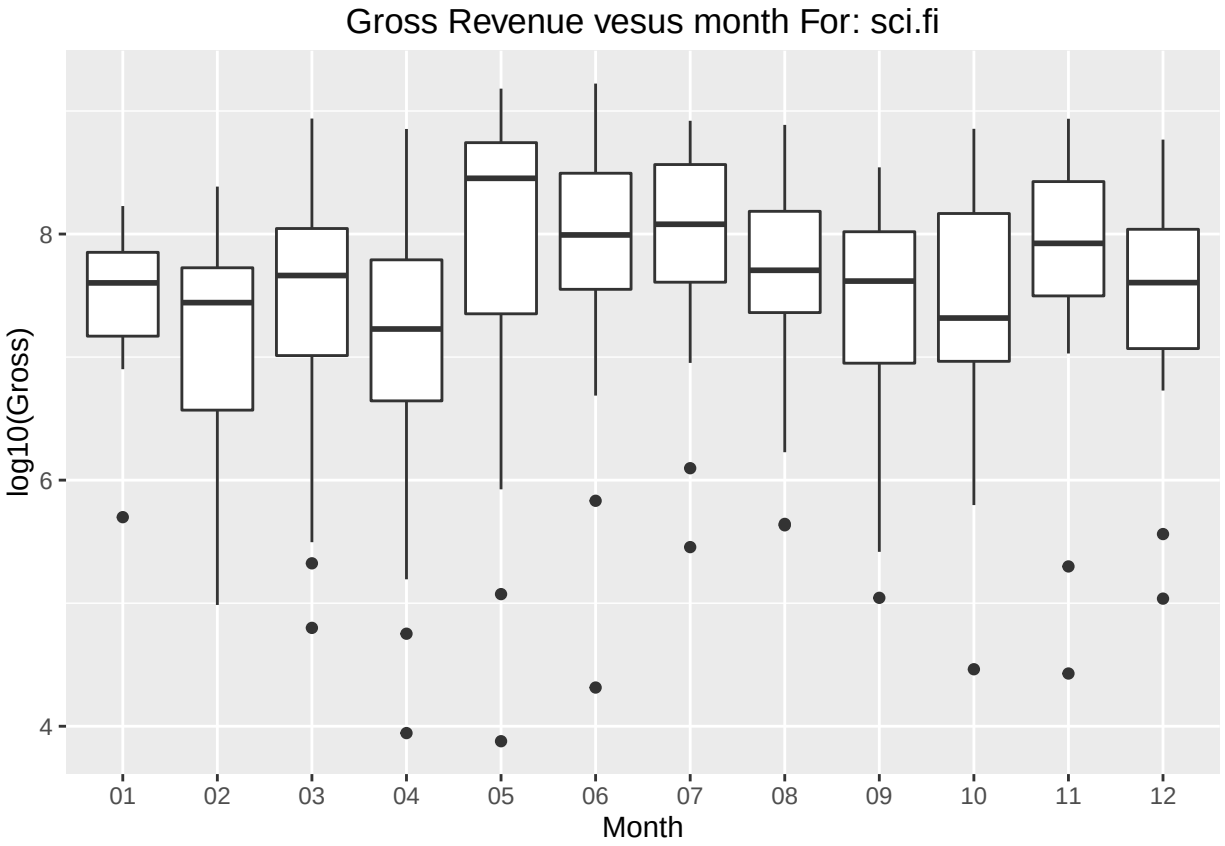

```
## Warning: Removed 56 rows containing non-finite values (stat_boxplot).
```



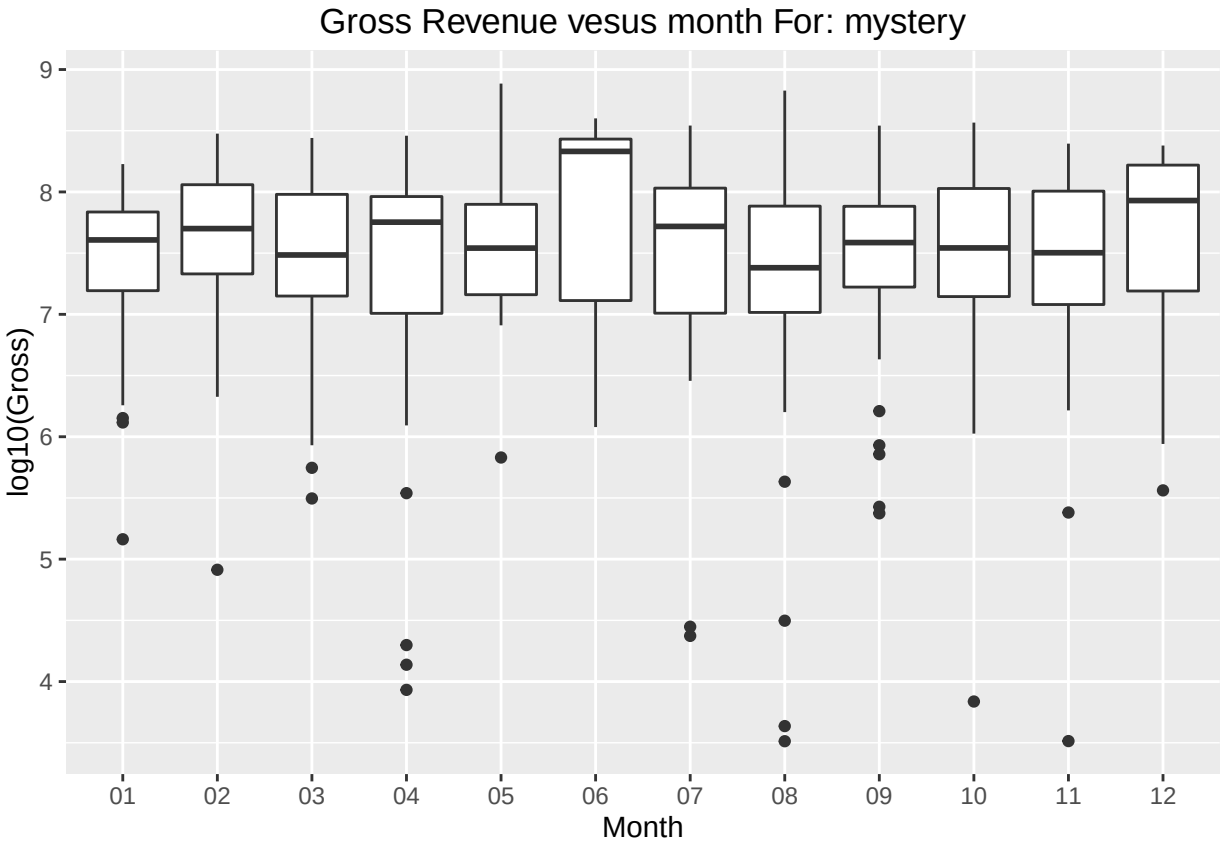

```
## Warning: Removed 57 rows containing non-finite values (stat_boxplot).
```




```
## Warning: Removed 22 rows containing non-finite values (stat_boxplot).
```




```
## Warning: Removed 21 rows containing non-finite values (stat_boxplot).
```



6. 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 http:// www.imdb.com/help/show_leaf?votestopfaq) and investigate the pairwise relationships between these different descriptors using graphs. Comment on similarities and differences between the user ratings of IMDb and the critics ratings of Rotten Tomatoes. Comment on the relationships between these variables and the gross revenue. Which of these ratings are the most highly correlated with gross revenue (use the R function cor and remove rows with missing values)?

Answer: there are 2 IMDb related ratings: imdbRating and imdbVotes.

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.

```
# use ggpairs cor
cor(movies_merged[,c(16:17, 20, 22:25, 27:29)], use = "complete.obs")
```

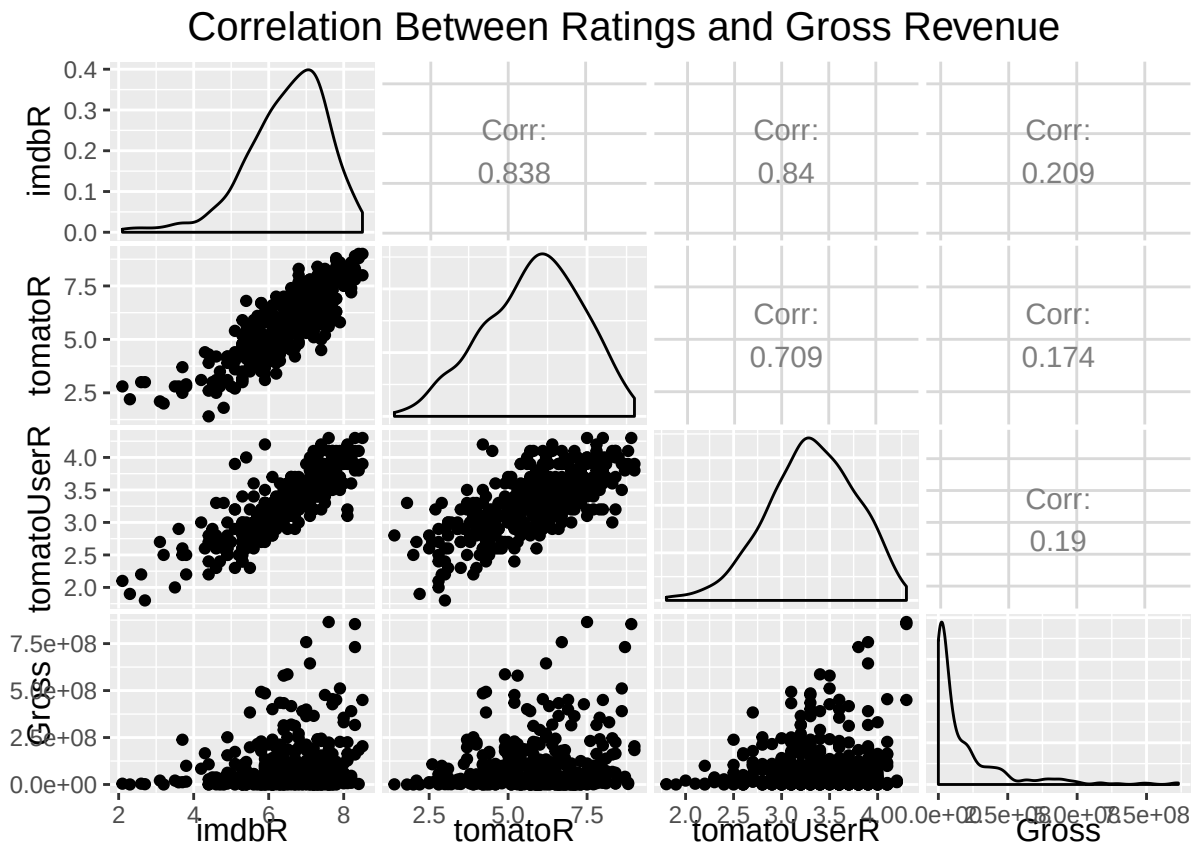
```
##                    imdbRating imdbVotes tomatoMeter tomatoRating
## imdbRating         1.00000000 0.3005994   0.7486719   0.79695408
```

```
## imdbVotes          0.30059944 1.0000000   0.1533657   0.21704090
## tomatoMeter        0.74867193 0.1533657   1.0000000   0.94100190
## tomatoRating       0.79695408 0.2170409   0.9410019   1.00000000
## tomatoReviews      0.14619256 0.5947674   0.0366415   0.10705546
## tomatoFresh        0.36507265 0.6329374   0.3463722   0.39946498
## tomatoRotten      -0.25233748 0.2565540  -0.4469521  -0.38369875
## tomatoUserMeter    0.83772664 0.2494005   0.6843610   0.70919710
## tomatoUserRating   0.82790126 0.2343559   0.6592432   0.69462722
## tomatoUserReviews  0.05750358 0.2729423   0.0244896   0.03737935
##                   tomatoReviews tomatoFresh tomatoRotten tomatoUserMeter
## imdbRating           0.14619256   0.3650727  -0.25233748      0.83772664
## imdbVotes            0.59476741   0.6329374   0.25655401      0.24940054
## tomatoMeter          0.03664150   0.3463722  -0.44695207      0.68436099
## tomatoRating         0.10705546   0.3994650  -0.38369875      0.70919710
## tomatoReviews        1.00000000   0.8822014   0.70531392      0.09992758
## tomatoFresh          0.88220145   1.0000000   0.28842998      0.31407331
## tomatoRotten         0.70531392   0.2884300   1.00000000     -0.26963641
## tomatoUserMeter      0.09992758   0.3140733  -0.26963641      1.00000000
## tomatoUserRating     0.10492395   0.3062981  -0.24777090      0.90093077
## tomatoUserReviews    0.16235863   0.1626114   0.08533976      0.02169800
##                   tomatoUserRating tomatoUserReviews
## imdbRating             0.827901265       0.057503576
## imdbVotes              0.234355940       0.272942253
## tomatoMeter            0.659243211       0.024489596
## tomatoRating           0.694627218       0.037379351
## tomatoReviews          0.104923953       0.162358632
## tomatoFresh            0.306298080       0.162611406
## tomatoRotten          -0.247770903       0.085339765
## tomatoUserMeter        0.900930772       0.021698000
## tomatoUserRating       1.000000000      -0.003614453
## tomatoUserReviews     -0.003614453       1.000000000
```

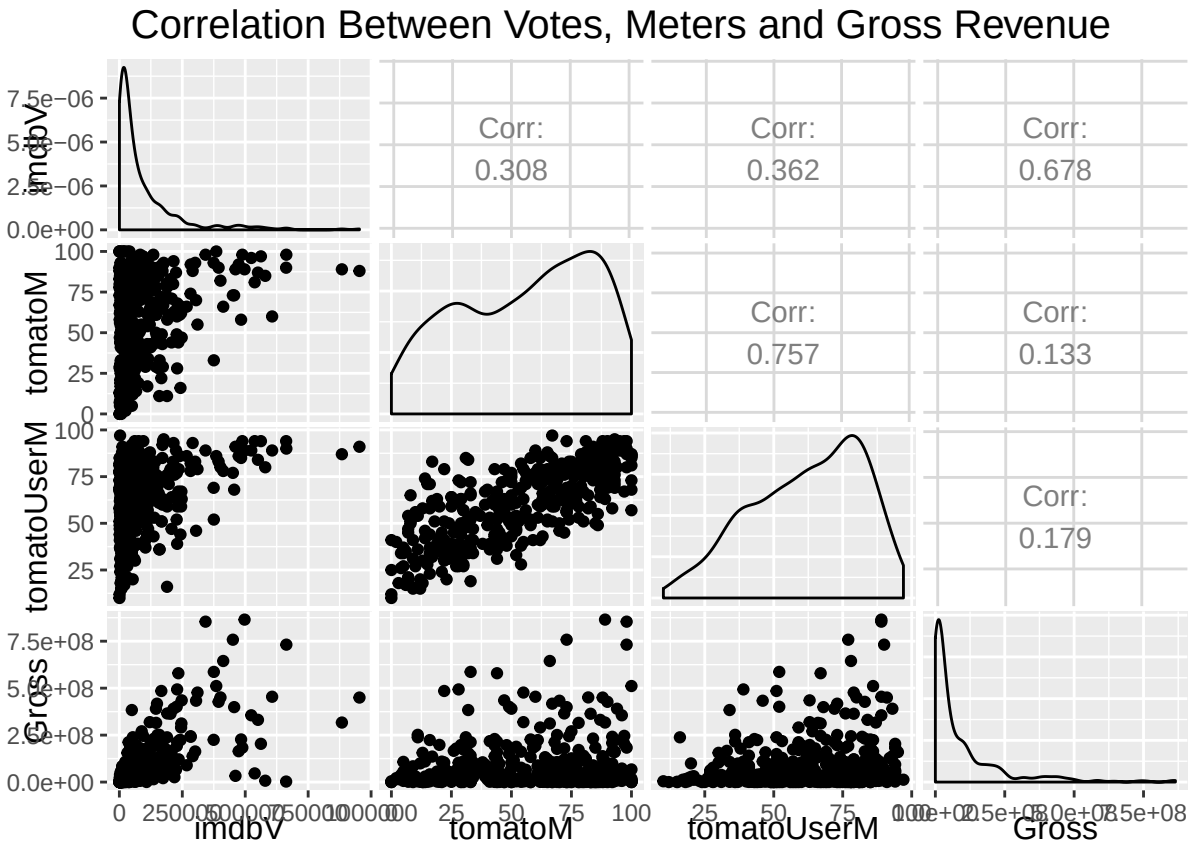
We can see there are very high correlation between imdbRating, tomatoMeter, tomatoRating, tomatoUserMeter, tomatoUserRating. As we expected, if a movie has higher rating, it tends to have higher value for Meter(both cirtics'and Users'). Also imdbVotes has high correlation with tomotoReviews, that means if a movie got more votes from imdb, it also tends to get more votes from rotten tomato.

Since there are two many variables,it will become too crawd if we put them all in one graph. Therefore I split them into two graphs. First we graphed all the ratings (imdbRating, tomatoRating, tomatouserRating) and Gross. Then we graphed all votes (imdbVotes, tomatoMeter, tomatoUserMeter) and Gross. We can see imdbRating is highly correlated with tomatoRating and tomatoUserRating, Also with the increase of all three, the Gross revenue is also increasing in some movies, but most of films still have low revenue even with high rating in all three categories. From the second graph, we can see tomotaMeter is highly correlated with tomatoUserMeter. With the increase of TomatoMeter and TomatoUserMeter, the Gross revenue has the same trend as last tree variables. Only imdbVotes has the clearing positive correlation with Gross revenue.

```
rowIndex <-  complete.cases(movies_merged[,c(16:17, 20, 22, 27:28, 38)])
sub_rating <- movies_merged[rowIndex, c(16:17, 20, 22, 27:28, 38)]
# sample the data
set.seed(1)
sampleIndex <- sample(seq_along(sub_rating$Gross), size = 400, replace = FALSE)
sub_rating_sample <- sub_rating[sampleIndex,]
ggpairs(sub_rating_sample, columns = c(1, 4, 6 ,7),
        columnLabels = c("imdbR", "tomatoR", "tomatoUserR", "Gross"),
        title = "Correlation Between Ratings and Gross Revenue")
```




```
ggpairs(sub_rating_sample, columns = c(2, 3, 5 ,7),
        columnLabels = c("imdbV", "tomatoM", "tomatoUserM", "Gross"),
        title = "Correlation Between Votes, Meters and Gross Revenue")
```



7. The variable Awards describes nominations and awards in text format. Convert it to a three dimensional binary vector whose first component represents no nomination or awards, the second component represents some nominations/awards, and the third component represents many nominations or awards. The relationship between the second and the third categories should be close to 5:1 (not precisely - this is a broad guideline to help you avoid creating a third category that is useless due to being extremely small and to encourage consistency). How did you construct your conversion mechanism? How does the gross revenue distribution changes across these three categories.

Answer: First we can extract the number of awards and norminations from Awards column, Then sum the awards and norminations to get the total for each movies. We found 6 awards and normation can split "some awards" and "many awards" to a 5:1 ratio. Then we can convert awards to a 3 dimensional binary vector. And find the distribution of Gross for each awards category.

```
awards_normination <- c()

for( i in seq_along(movies_merged$Awards)){
        if(movies_merged$Awards[i] == "N/A"){
                awards_normination <- c(awards_normination, 0)
        }else{
                x <- movies_merged$Awards[i]
                temp <- gregexpr("[0-9]+", x) # find the numbers with any number of digits
                # extract the awards and normination and sum them
              total_awards_normination  <- sum(as.numeric(unlist(regmatches(x, temp))))

         awards_normination <- c(awards_normination, total_awards_normination)
        }
```

```
}

length(awards_normination[awards_normination > 6])
```

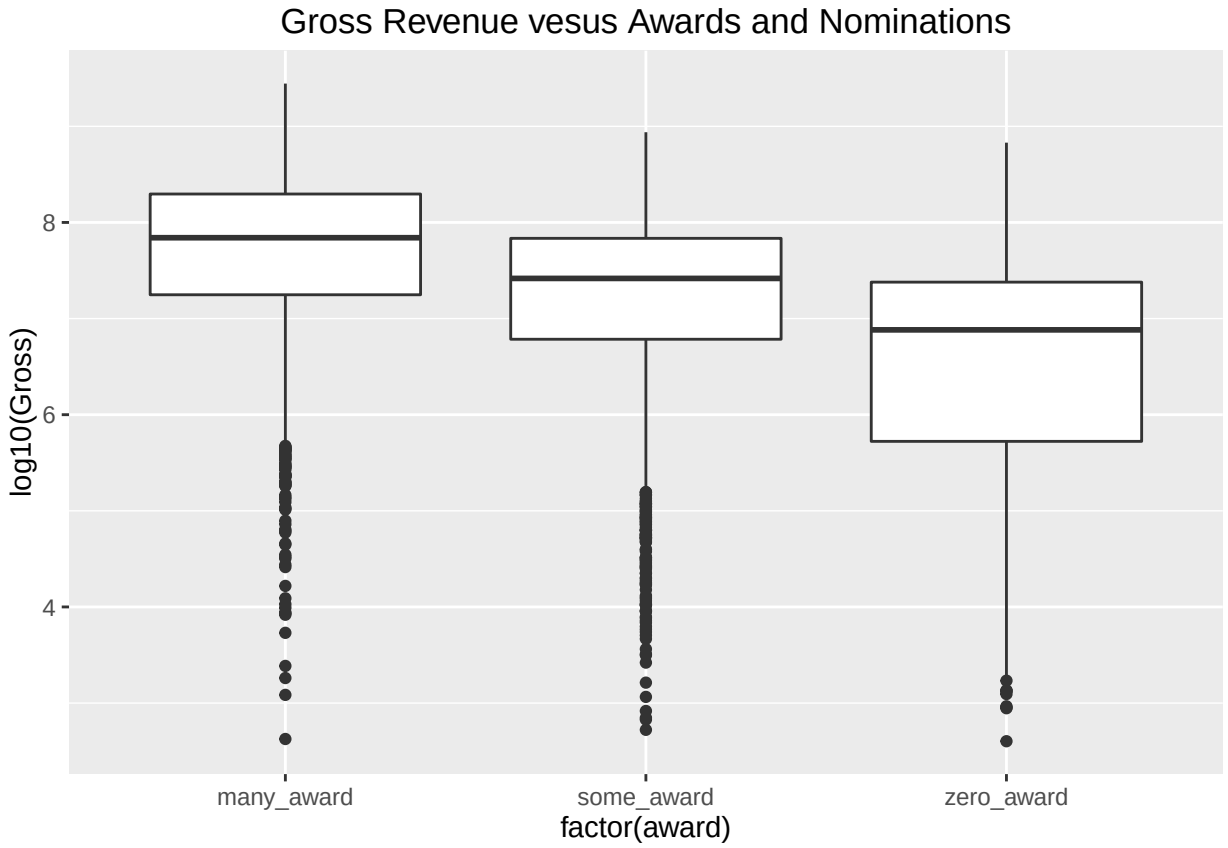
```
## [1] 4257
```

```
# by a simple test we can split the movie with awards and normination to 1:5 ratio
movies_merged$zero_award <- 0
movies_merged$some_award <- 0
movies_merged$many_award <- 0
for( i in seq_along(awards_normination)){
        if(awards_normination[i] > 6){
                movies_merged$many_award[i] <- 1
        }else if(awards_normination[i] > 0 ){
               movies_merged$some_award[i] <- 1
        }else{
                movies_merged$zero_award[i] <- 1
        }
}

sub_award <- movies_merged[,c("many_award", "some_award", "zero_award", "Gross")]
sub_award <- sub_award[complete.cases(sub_award),]
dim(sub_award)
```

```
## [1] 4558    4
```

```
sub_award$award <- NA
for( i in seq_along(sub_award$many_award)){
        if(sub_award$many_award[i] == 1){
                sub_award$award[i] <- "many_award"
        }else if(sub_award$some_award[i] == 1){
               sub_award$award[i] <- "some_award"
        }else{
                sub_award$award[i] <- "zero_award"
        }
}
ggplot(sub_award, aes(factor(award), log10(Gross))) +
        geom_boxplot() +
        ggtitle("Gross Revenue vesus Awards and Nominations")
```

```
## Warning: Removed 285 rows containing non-finite values (stat_boxplot).
```

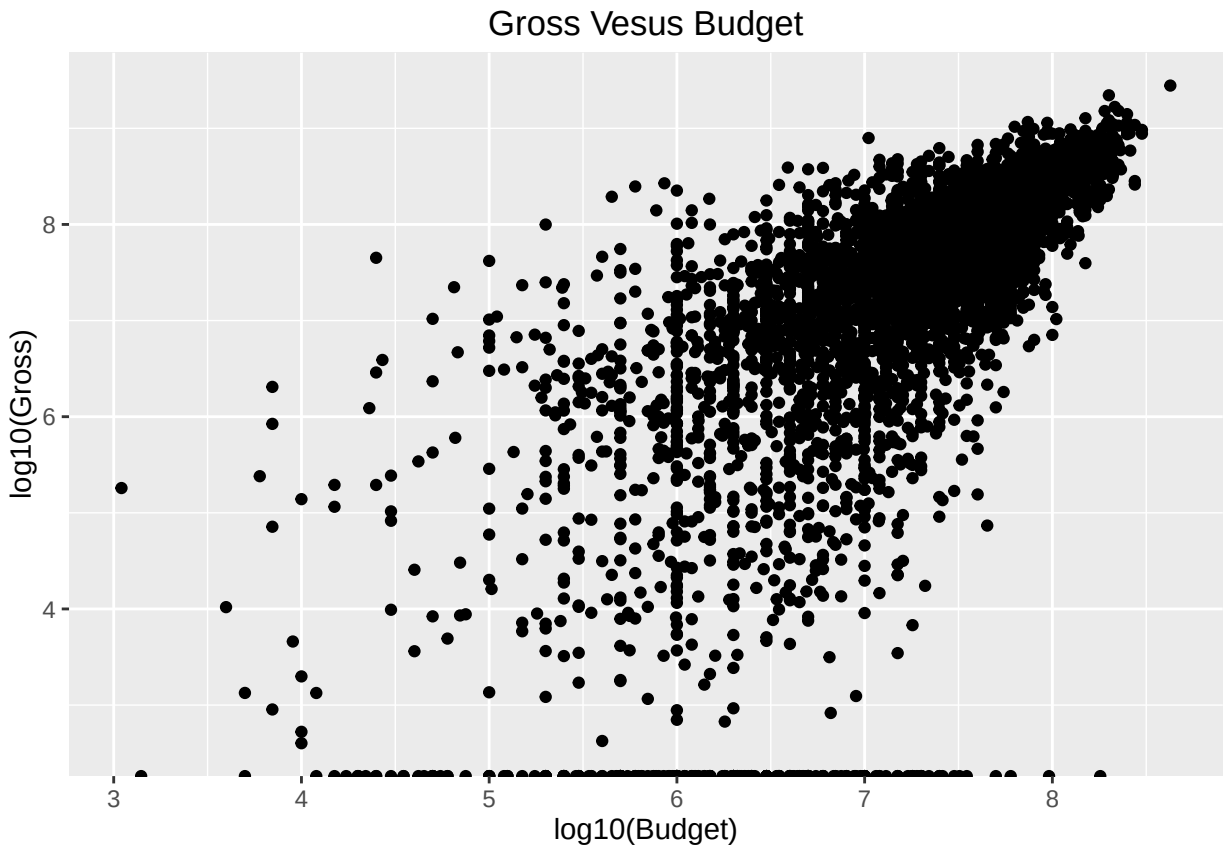


From the graph, We can see clearly with more rewards, the movie tends to have higher revenues.

8. Come up with two new insights (backed up by the data and graphs) that are expected, and one new insight (backed up by data and graphs) that is unexpected at first glance and do your best to motivate it. By "new" here I mean insights that are not an immediate consequence of one of the above assignments.

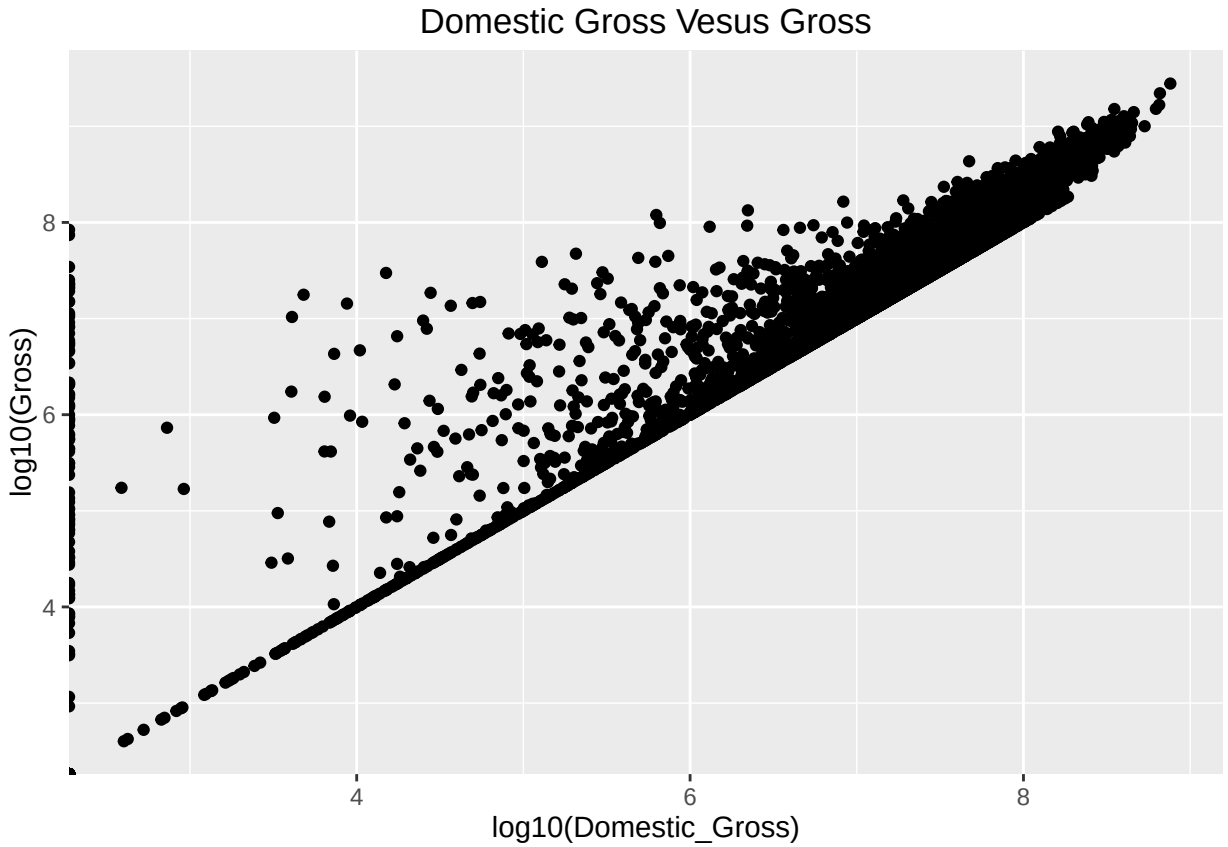
Answer: Expected Insight 1: if a movies has higher budget, normally we can expect it also can gain higher gross revenue. Therefore we extracted gross and budget and graphed it. We can see they are really positively coorelated. But there are also some points along the budget line, that means this movies did not get the gross revenue as expected.

```
gross_budget <- movies_merged[,c("Budget", "Gross")]
gross_budget <- gross_budget[complete.cases(gross_budget),]
ggplot(gross_budget, aes(log10(Budget), log10(Gross))) + geom_point() + ggtitle("Gross Vesus Budget")
```



Expected Insight 2: if a movies has higher domestic gross revenue, normally we can expect it also can gain higher gross revenue. Therefore we extracted domestic gross and gross and graphed it. We can see most movies on the straight line of slope as we expected. But there are also lots of points above the slope line, even some on the vertical line along the gross axis, that means these movies gained much higher gross revenue compared with domestic gross revenue.

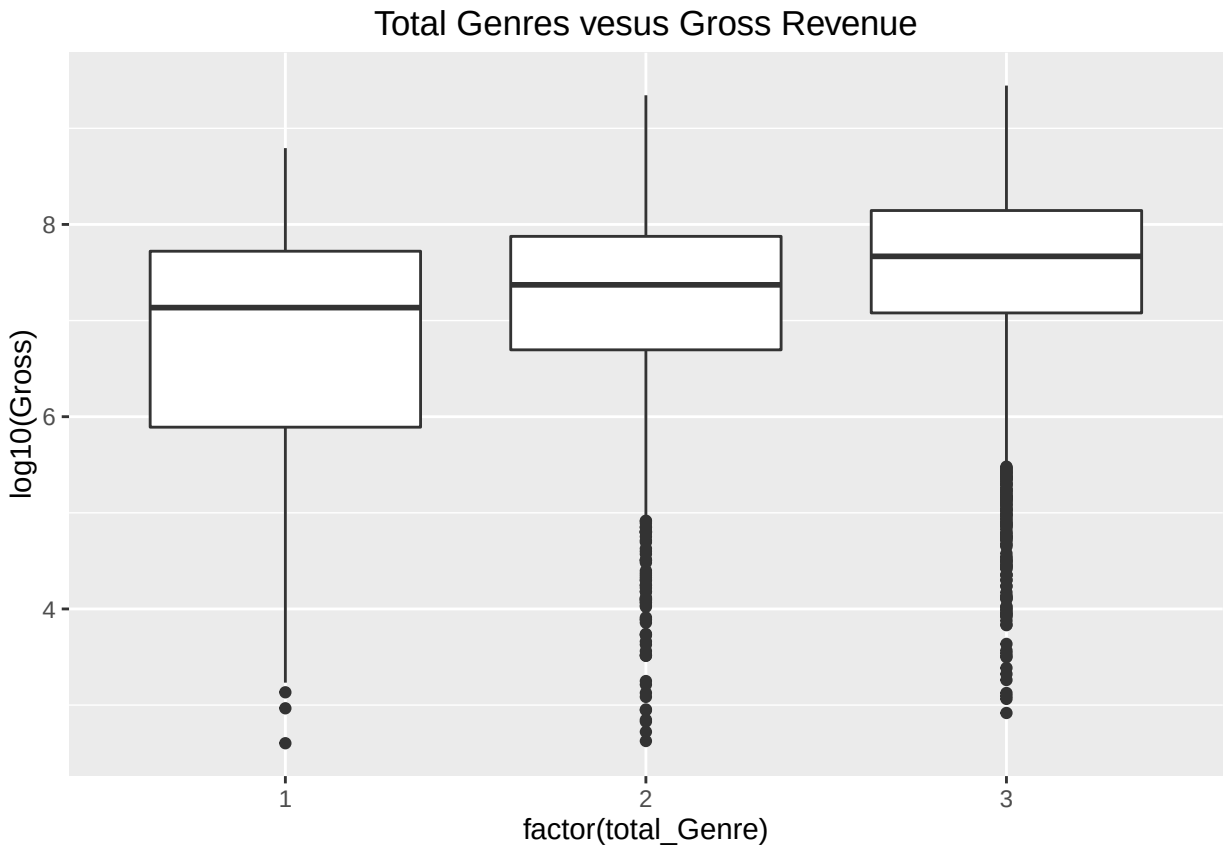
```
gross_domestic_gross <- movies_merged[,c("Domestic_Gross", "Gross")]
gross_domestic_gross <-gross_domestic_gross[complete.cases(gross_domestic_gross),]
ggplot(gross_domestic_gross, aes(log10(Domestic_Gross), log10(Gross))) + geom_point() + ggtitle("Domest
```



Unexpected Insight: Since normally how many genres a movie was assigned to should not be related with its gross revenue. So here I want to check if there are any relationship between them. I count how many genres of each movie has. Then I graphed the gross revenue according to its total_genre. We can see with more genres a movie belongs to, it tends to gain more gross revenue.

```
Genre_df <- Genre_df[complete.cases(Genre_df),]

Genre_df$total_Genre <-rowSums(Genre_df[, 1:28])
ggplot(Genre_df, aes(factor(total_Genre), log10(Gross))) + geom_boxplot() + ggtitle("Total Genres vesus
```

```
## Warning: Removed 284 rows containing non-finite values (stat_boxplot).
```


Total Genres vesus Gross Revenue
log10(Gross)
8
6
4
1
2
3
factor(total_Genre)