Code

Project 1: Explore and Prepare Data

CSE6242 - Data and Visual Analytics - Spring 2017 Due: Sunday, March 5, 2017 at 11:59 PM UTC-12:00 on T-Square

GT account name: syan62; GT ID: 903292896

Note: This project involves getting data ready for analysis and doing some preliminary investigations. Project 2 will involve modeling and predictions, and will be released at a later date. Both projects will have equal weightage towards your grade.

Data

In this project, you will explore a dataset that contains information about movies, including ratings, budget, gross revenue and other attributes. It was prepared by Dr. Guy Lebanon, and here is his description of the dataset:

> The file `movies_merged` (https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6242/project/movies_merged) contains a dataframe with the same name that has 40K rows and 39 columns. Each row represents a movie title and each column represents a descriptor such as `Title`, `Actors`, and `Budget`. I collected the data by querying IMDb's API (see www.omdbapi.com (http://www.omdbapi.com/)) and joining it with a separate dataset of movie budgets and gross earnings (unknown to you). The join key was the movie title. This data is available for personal use, but 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).

Instructions

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. Open this file in RStudio to get started.

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

Hide

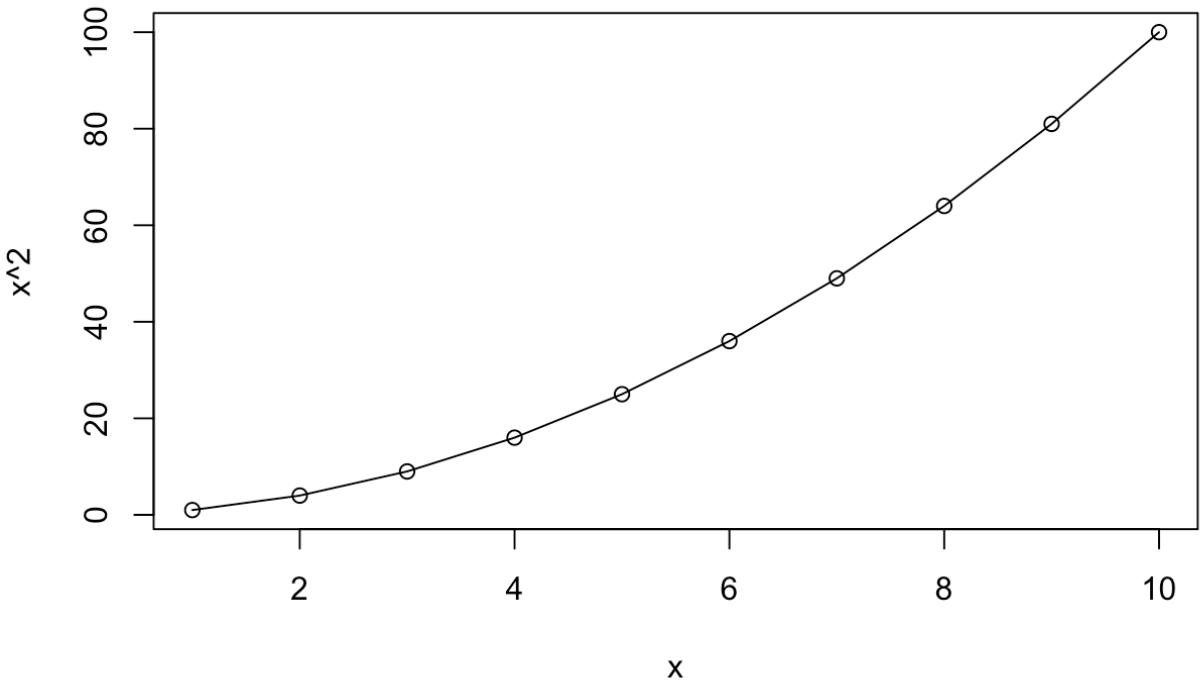
```
x = 1:10
print(x^2)
```

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

Plots appear inline too:

Hide

```
plot(x, x^2, 'o')
```



Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

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

Setup

Load data

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

Hide

```
load('movies_merged')
```

This creates an object of the same name (`movies_merged`). For convenience, you can copy it to `df` and start using it:

Hide

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

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

Hide

```
colnames(df)
```

```
 [1] "Title"             "Year"              "Rated"             "Released"
 [5] "Runtime"           "Genre"             "Director"          "Writer"
 [9] "Actors"            "Plot"              "Language"          "Country"
[13] "Awards"            "Poster"            "Metascore"         "imdbRating"
[17] "imdbVotes"         "imdbID"            "Type"              "tomatoMeter"
[21] "tomatoImage"       "tomatoRating"      "tomatoReviews"     "tomatoFresh"
[25] "tomatoRotten"      "tomatoConsensus"   "tomatoUserMeter"   "tomatoUserRat
ing"
[29] "tomatoUserReviews" "tomatoURL"         "DVD"               "BoxOffice"
[33] "Production"        "Website"           "Response"          "Budget"
[37] "Domestic_Gross"    "Gross"             "Date"
```

Load R packages

Load any R packages that you will need to use. You can come back to this chunk, edit it and re-run to load any additional packages later.

Hide

```
library(ggplot2)
library(GGally)
```

If you are loading any non-standard packages (ones that have not been discussed in class or explicitly allowed for this project), please mention them below. Include any special instructions if they cannot be installed using the regular `install.packages('<pkg name>')` command.

Non-standard packages used: None

Tasks

Each task below is worth **10** points, and is meant to be performed sequentially, i.e. do step 2 after you have processed the data as described in step 1. Total points: **100**

Complete each task by implementing code chunks as described by `TODO` comments, and by responding to questions ("**Q**:") with written answers ("**A**:"). If you are unable to find a meaningful or strong relationship in any of the cases when requested, explain why not by referring to appropriate plots/statistics.

It is OK to handle missing values below by omission, but please omit as little as possible. It is worthwhile to invest in reusable and clear code as you may need to use it or modify it in project 2.

1. Remove non-movie rows

The variable `Type` captures whether the row is a movie, a TV series, or a game. Remove all rows from `df` that do not correspond to movies.

Hide

```
# TODO: Remove all rows from df that do not correspond to movies
df <- df[(df$Type == "movie"),]
```

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

A: 40000 rows left.

2. Process `Runtime` column

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

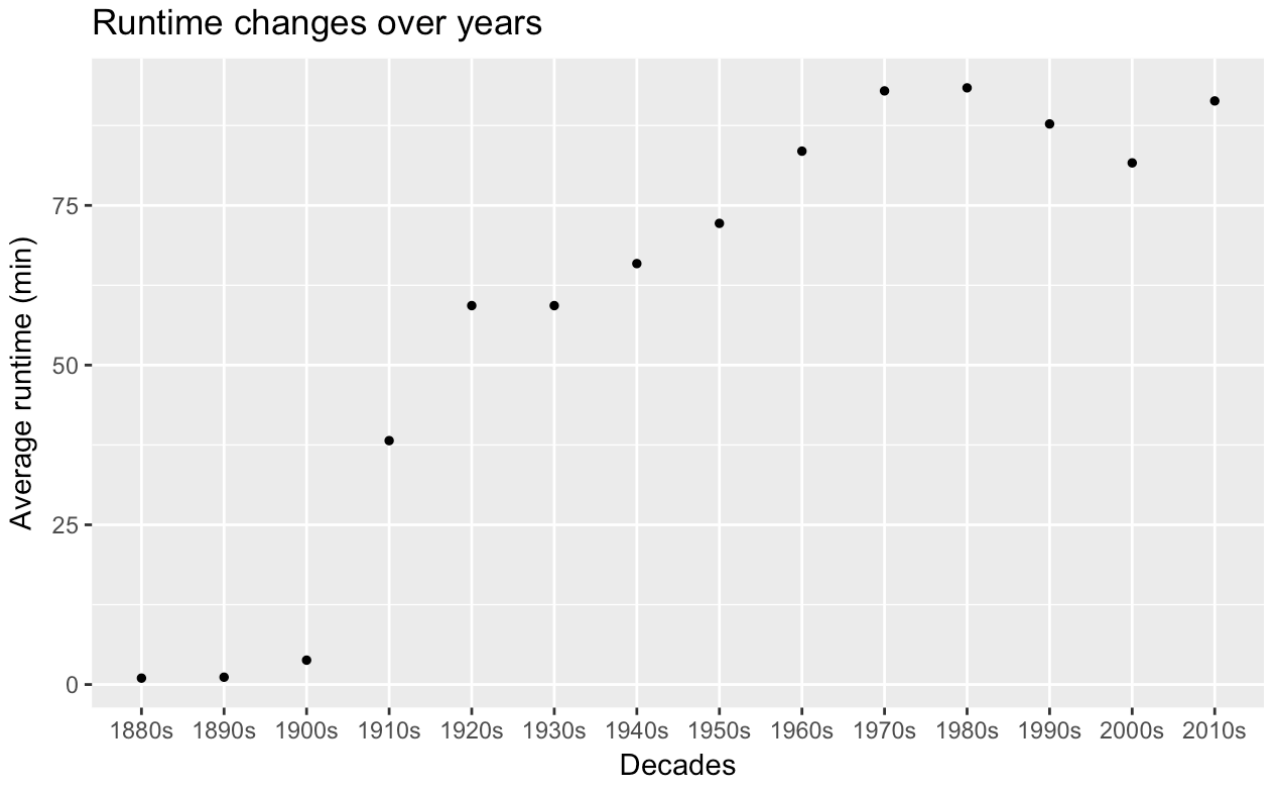
Hide

```
# TODO: Replace df$Runtime with a numeric column containing the runtime in minu
tes
# Create an empty vector to store the processed runtime values
dfRows = nrow(df)
num_Runtime = vector(length = dfRows, mode = "numeric")
# Convert the runtime to numeric value in minutes.
for (i in seq(1, dfRows, by =1)) {
  # Convert character "N/A" to NA
  if (grepl("N/A", df$Runtime[i], ignore.case = T)) {
    num_Runtime[i] = NA
  }
  # Convert the runtime originally in hour+ minutes expression to correct value
s with minutes as unit
  if (grepl("[0-9] h", df$Runtime[i], ignore.case = T)) {
    if (grepl("[0-9] h [0-9]+ min", df$Runtime[i])) {
      runtimeSplit <- strsplit(x = df$Runtime[i], split = " ", fixed = T)
      num_Runtime[i] = as.numeric(runtimeSplit[[1]][1]) * 60 + as.numeric(runti
meSplit[[1]][3])
    }
    else if (grepl("[0-9] h", df$Runtime[i])) {
      runtimeSplit <- strsplit(x = df$Runtime[i], split = " ", fixed = T)
      num_Runtime[i] = as.numeric(runtimeSplit[[1]][1]) * 60
    }
  }
  # Convert the runtime originally in minutes to numeric values
  if (grepl("^[0-9]+ min", df$Runtime[i], ignore.case = T)) {
     runtimeSplit <- strsplit(x = df$Runtime[i], split = " ", fixed = T)
     num_Runtime[i] = as.numeric(runtimeSplit[[1]][1])
  }
}
# replace the column df$Runtime with the new numeric column
df$Runtime = num_Runtime
#df_noNA_runTime = na.omit(df)
#transform(df_noNA_runTime, Runtime = as.numeric(Runtime))
```

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

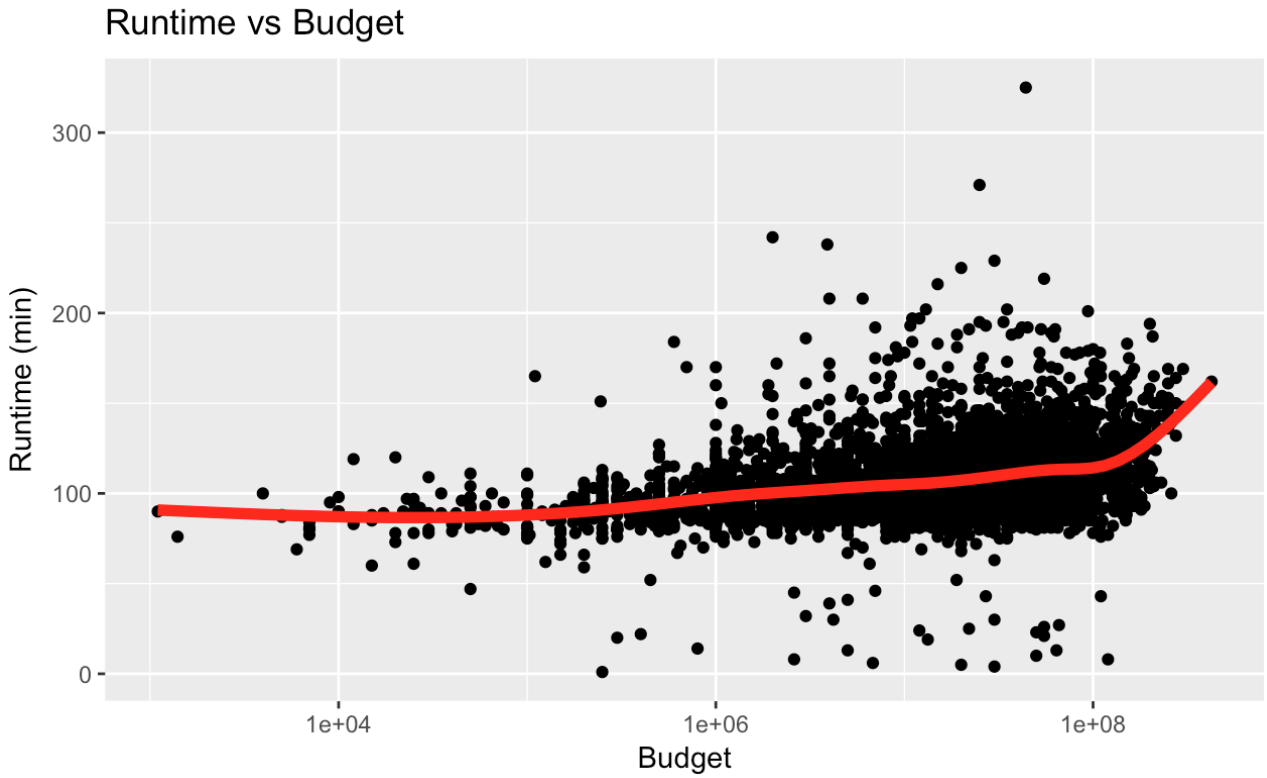
Hide

```
# TODO: Investigate the distribution of Runtime values and how it varies by Yea
r and Budget
# Use the plyr package to manipulate the dataframe
library(plyr)
# Compute mean of Runtime for all movies produced in the same year.
Runtime_year_mean = ddply(df, "Year", summarise, mean.Runtime = mean(Runtime, n
a.rm = TRUE), median.Runtime = median(Runtime, na.rm = TRUE))
# Count number of movies produced in each year
Runtime_year_count = ddply(df, "Year", "nrow")
# Inner join dataframe Runtime_year_mean and Runtime_year_count, omit NA and Na
N
Runtime_year = na.omit(merge(Runtime_year_mean, Runtime_year_count))
names(Runtime_year)[4] = "Num_Movies"
# Discard year 2017 in the analysis, because 2017 just started, the data from 2
017 is not complete yet
Runtime_year = Runtime_year[Runtime_year$Year < 2017,]
# Bin years to decades in a new column "Decades"
Runtime_year$Decades = floor(Runtime_year$Year / 10)
# Compute the sum of all movies' runtime in the same year
Runtime_year$Sum = Runtime_year$mean.Runtime * Runtime_year$Num_Movies
# compute the total runtime of all movies and total number of movies in every d
ecade
Runtime_decades = ddply(Runtime_year, "Decades", summarise, sum.Runtime = sum(S
um), sum.NumMovies = sum(Num_Movies))
# Compute the mean of runtime for each decade
Runtime_decades$mean.Runtime = Runtime_decades$sum.Runtime / Runtime_decades$su
m.NumMovies
# plot the mean of runtime against decades; the decade 1880s represents year [1
880,1889], decade 1920s represents year [1920,1929], etc.
qplot(x = paste0(Decades*10,"s"), y = mean.Runtime, size = I(1), data = Runtime
_decades, main = "Runtime changes over years", ylab = "Average runtime (min)",
xlab = "Decades")
```



Hide

```
# Drop the rows whose Budget is NA
df_Budget_noNA = df[!is.na(df$Budget),]
# Drop the rows in df_Budget_noNAwhose Runtime in NA
df_RT_Budget_noNA = df_Budget_noNA[!is.na(df_Budget_noNA$Runtime),]
# plot the relation of Runtime and Budget (Budget in log-scale)
qplot(x = Budget, y = Runtime, log = "x", data = df_RT_Budget_noNA, main = "Run
time vs Budget", ylab = "Runtime (min)")+ stat_smooth(color = "red", size = I(2
), se = F)
```



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: The Runtime changes over decades: The averaged runtime of all movies in every decades There are three stages of runtime over decades. For the decades 1880s, 1890s, 1900s, the runtime of movies are very short, the mean is less than 10 minutes. From 1910s to 1970s, the average runtime increases almost linearly, from 38.2 minutes (1910s) to 92.7 minutes (1970s). After 1970s, the runtime enters a stable stage. There is a little decrease in 1990s and 2000s, but the variation is not large. This trend is not suprising. From 1880s to 1900s, the movie industry just initiated. The technology and production have not become mature, so only short movies were made. From 1910s to 1970s, new techniques were being developed and production of movies became more and more mature, so the runtime of movies kept increasing. After 1970s, the movie industry and movie market are very mature. Runtime around 90 minutes is an optimized result that is long enough to tell a full story and not too long to make audience tierd.

The relationships between runtime and budget: From the scatter points of runtime at different budgets, we can see the spread of runtimes increases with budget. For movies with budget around 1e+04, all their runtimes are in range 50 - 130 minutes, but for movies with large budget (>1e+06), we can see runtimes as short as a few minutes and also as long as 250 minutes. The red line in the graph is a smoothing curve showing the average tendency of runtime with different budget. The red line shows a slowly increasing trend of runtime as the budget increases.

3. Encode `Genre` column

The column `Genre` represents a list of genres associated with the movie in a string format. Write code to parse each text string into a binary vector with 1s representing the presence of a genre and 0s the absence, and add it to the dataframe as additional columns. Then remove the original `Genre` column.

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

Hide

```
# TODO: Replace Genre with a collection of binary columns
# Use the 'tm' package to create the collection of all genre types.
library(tm)
```

```
Loading required package: NLP

Attaching package: ‘NLP’

The following object is masked from ‘package:ggplot2’:

    annotate
```

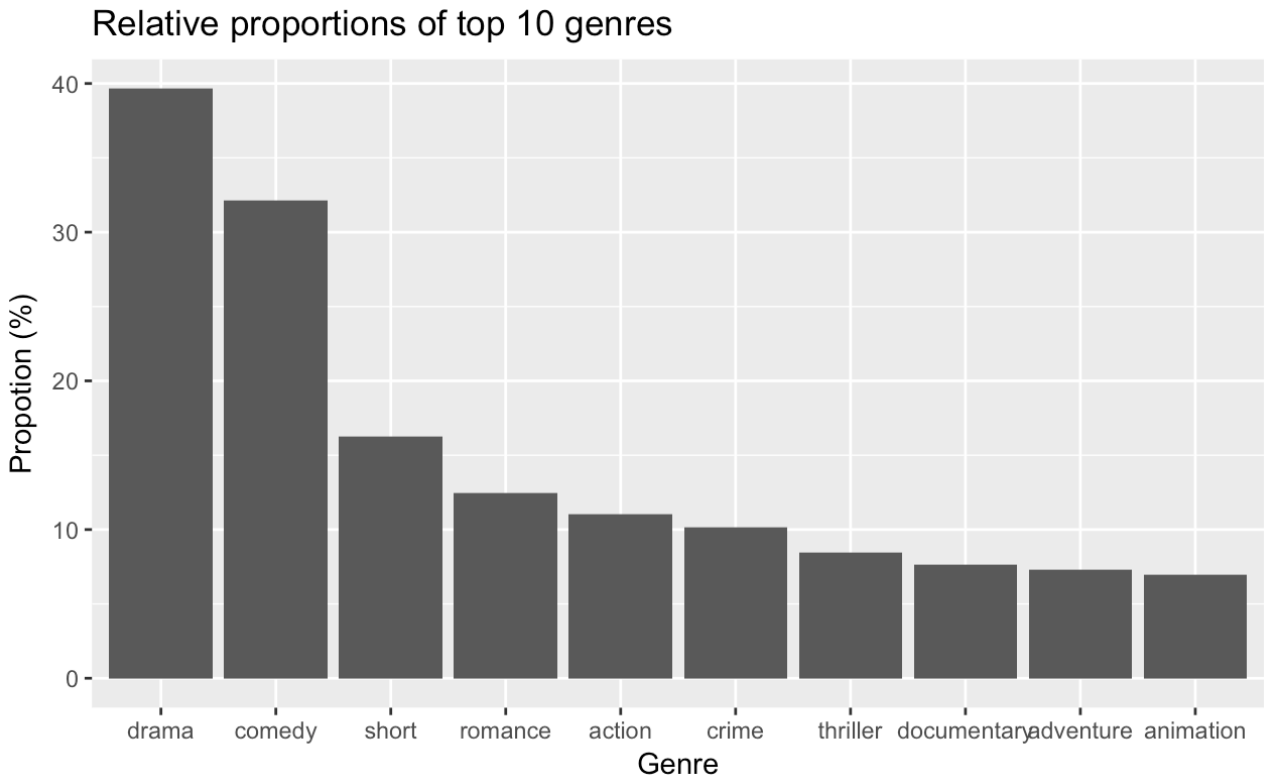
Hide

```
genre_noComma = sapply(df$Genre, function(x) strsplit(x, split = ",", fixed = T
))
myCorpus = Corpus(VectorSource(genre_noComma))
myDTM = DocumentTermMatrix(myCorpus, control = list(minWordLength = 1))
genre_dict = findFreqTerms(myDTM, 1)
# Add the separated genres as individual column to the dataframe, with proper n
ames.
variableNames = c(names(df),genre_dict)
df_columns = length(df)
for (i in seq(1, length(genre_dict), by=1)) {
  df[,i+df_columns] <- NA
}
names(df) = variableNames
# Loop through the genre dictionary and the rows of df, and compare the text st
ring in genre dictionary and the content in df$Genre using grepl() function. If
the result is TRUE, set 1 to the cell on that row in that genre column. If the
result is FALSE, set 0 to the cell.
dfRows = nrow(df)
for (j in seq(1, length(genre_dict), by=1)) {
  for (i in seq(1, dfRows, by=1)) {
    if (grepl(names(df)[j+df_columns], df$Genre[i], ignore.case = TRUE)) {
      df[i,j+df_columns] = 1
    }
    else {
      df[i,j+df_columns] = 0
    }
  }
}
# Delete the original `Genre` column, and exclude the binary genre column of N/
A
drops <- c("n/a", "Genre")
df <- df[,!(names(df) %in% drops)]
```

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

Hide

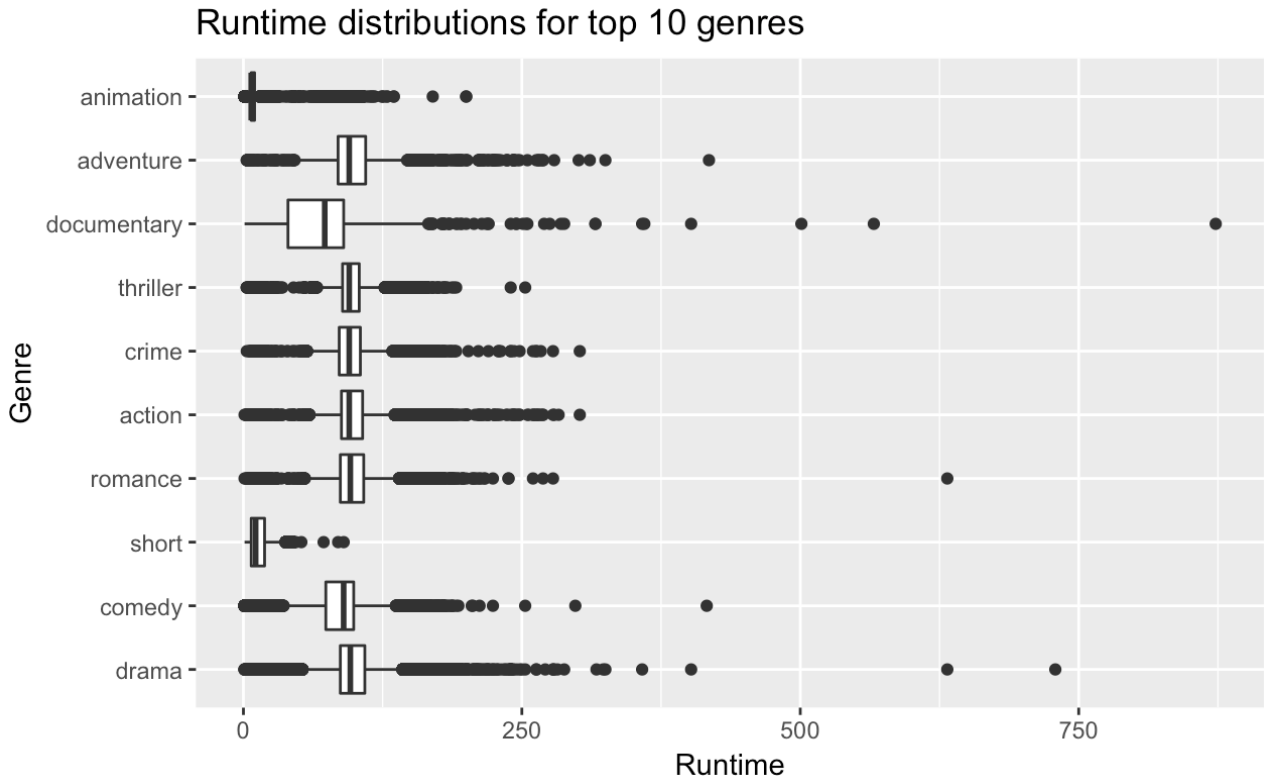
```
# TODO: Select movies from top 10 most common genres and plot their relative pr
oportions

library(ggplot2)
genre_name = genre_dict[!(genre_dict %in% drops)]
# Count number of movies in each genre
last_column = length(df)
first_genreCol = last_column - (length(genre_dict) -1) + 1
genre_count= apply(df[,first_genreCol:last_column], 2, sum)
# Create a dataframe genre_summary for binary genres to store the number of mov
ies and the proportion of each genre
genre_summary = data.frame(Genre = as.factor(genre_name), count = genre_count,
proportion = genre_count/nrow(df))
# Prepare the genre_summary for ploting. Set the factor levels of genre_summary
$Genre according to the decreasing order of proportion
genre_summary$Genre <- factor(genre_summary$Genre, levels = genre_summary$Genre
[order(genre_summary$proportion, decreasing = T)])
# Only plot the top 10 most common genres
top10_genre = levels(genre_summary$Genre)[1:10]
top10_genreSummary = genre_summary[row.names(genre_summary) %in% top10_genre,]
ggplot(top10_genreSummary, aes(x = Genre, y = proportion*100)) + geom_bar(stat
= "identity")+ ylab("Propotion (%)") + ggtitle("Relative proportions of top 10
genres")
```



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

Hide

```
# TODO: Plot Runtime distribution for top 10 most common genres
# Create a new dataframe genre_runtime to store the genre and runtime, which is
easy for boxplot
genre = vector(mode = "character", length = 0)
runtime = vector(mode = "numeric", length = 0)
for (i in 1:10) {
  genre = c(genre, rep(row.names(top10_genreSummary)[i], top10_genreSummary$cou
nt[i]))
  name = row.names(top10_genreSummary)[i]
  runtime = c(runtime, df[df[[name]] == 1,]$Runtime)
}
genre_runtime = data.frame(Genre = genre, Runtime = runtime)
genre_runtime <- na.omit(genre_runtime)
# Prepare the genre_summary for ploting. Set the factor levels of genre_runtime
$Genre according to the decreasing order of their relative proportion
genre_runtime$Genre <- factor(genre_runtime$Genre, levels = genre_summary$Genre
[order(genre_summary$proportion, decreasing = T)])
ggplot(genre_runtime, aes(x= Genre, y =Runtime)) + geom_boxplot() + coord_flip(
) + scale_x_discrete() + ggtitle("Runtime distributions for top 10 genres")
```



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

A: Relative proportion of 10 most common genres: I calculate the percentage proportion of each genre over total movie number. The three most common genres are drama, comedy and short. They are all general type of movie rather than specific types. Drama is the most common movie genre, which takes about 40% of all movies. This is an expected results, because many types of movie (e.g. romance, action, comedy,etc) can also be classified as Drama as long as it tells a story. Comedy is the second most common movie genre (about 32 %), because movies tell different type of stories can also be a comedy movie. Similarly situation for short movies, which ranks the third common. For other types movies, romance (~12 %), action (~11 %), crime (~ 10%), thriller (~8%), documentry (~ 8%), adventure (~7%) and animation (~7%), they may also have overlap with other movie types, but the chances of overlapping are much less than the top three commone genres drama, comedy and short. The proportion values of these seven types are not varying too much, ranges from 12% to 7%.

There is no strong correlation between movie proportion and runtime for the 10 most common genres. Except for short, documentary and animation movies, all the other seven genres of movies that are telling a story have similar average runtime (longer than 75 mins but shorter than 100 mins). This is an expected result, because this length of movie is long enough to tell a good story but not too long to make audience tierd. Short movies are expected to be short with a narrow distribution. Animation movies have broader distribution than short movies, even though their median runtime is similar. Documentary has the broadest distribution of runtime, ranging from a few minutes to 800+ minutes.

4. Eliminate mismatched rows

The dataframe was put together by merging two different sources of data and it is possible that the merging process was inaccurate in some cases (the merge was done based on movie title, but there are cases of different movies with the same title). The first source's release time was represented by the column `Year` (numeric representation of the year) and the second by the column `Released` (string representation of release date).

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

Hide

```
# TODO: Remove rows with Released-Year mismatch
# calculate the number of rows with non NA gross information
Gross_noNA = sum(!is.na(df$Gross))
mismatch = vector("numeric", length = 0) # Create a vector to store the mismatc
hs
counter = 0
# loop through the rows of df, split the strings in df$Released and exact the r
eleased year from it. Then, compare the exacted released year with the value in
$Year column.
for (i in seq(1, nrow(df), by = 1)) {
  release_split = strsplit(as.character(df$Released[i]), split = "-", fixed = T
)
  releasedYear = as.numeric(release_split[[1]][1])
  # Rows with released year as NA won't be removed. The Released year is smalle
r than the value in $Year column or the Released year is bigger than 1 year aft
er the $Year value are removed.
  if (!is.na(df$Released[i]) && (releasedYear - df$Year[i] > 1 || releasedYear
- df$Year[i] < 0)) {
    counter = counter + 1
    mismatch[counter] = i
  }
}
# Remove the mismatched rows
df <- df[-mismatch,]
# Calculate the number of rows are removed and the percentage of removed rows t
hat have a `Gross` value present
rowRemoved = length(mismatch)
Gross_noNA_noMismatch = sum(!is.na(df$Gross))
removedGross = (Gross_noNA - Gross_noNA_noMismatch) / Gross_noNA
rowRemoved
```

```
[1] 1831
```

Hide

```
removedGross
```

```
[1] 0.0199649
```

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

A: The removal logic: 1) if the Released year of one movie is shown as NA in df, I do not remove it, because it is not a real mismatch. 2) Since some movies their released date is a little later than the year they were pictured, so if the Release year is one year later than the Year column I do not remove them. So for a movie whose released year is more than one year after the value in “Year” column, I remove it.

Also, if the released year of a movie is earlier than the year it produced, I remove it.

Finally, I removed 1831 rows in df. About 2 % of the rows that have a `Gross` value present was removed.

5. Explore `Gross` revenue

For the commercial success of a movie, production houses want to maximize Gross revenue. Investigate if Gross revenue is related to Budget, Runtime or Genre in any way.

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

Hide

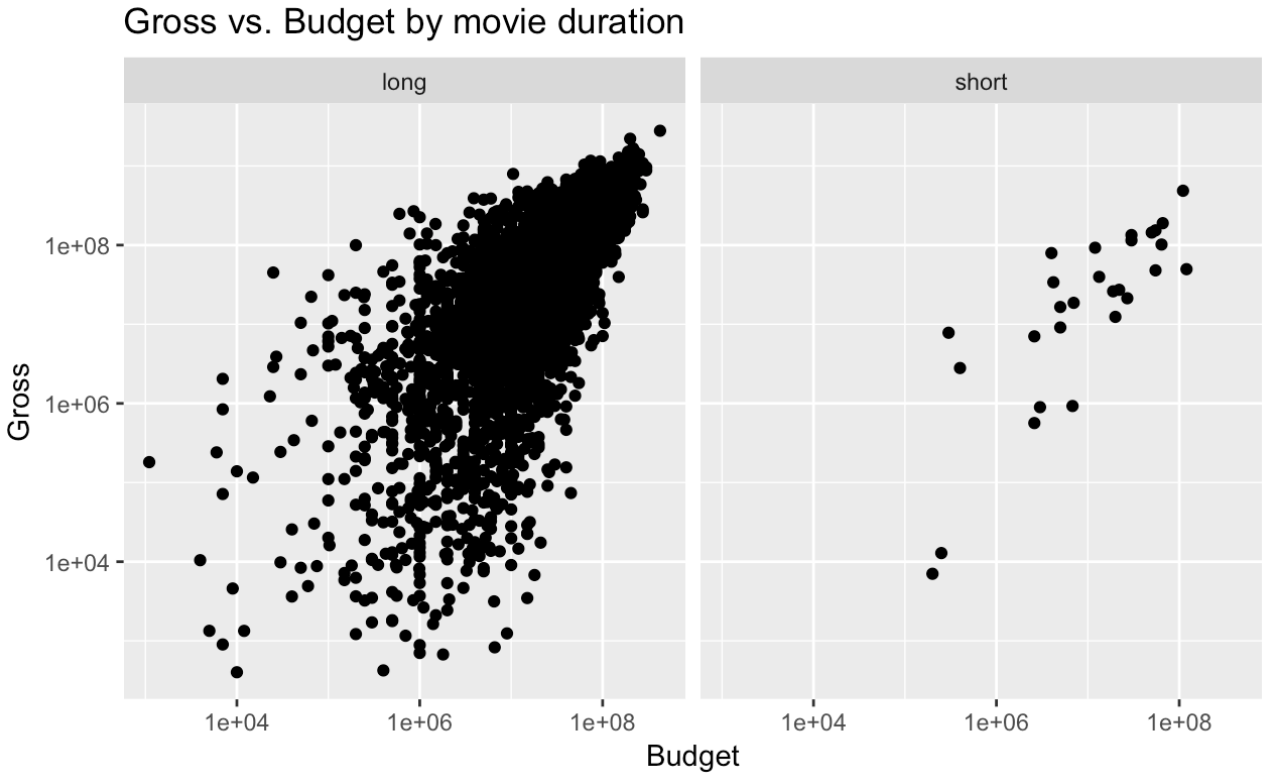
```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
# Exclude the NAs in 'Gross' column. Gross = 0 is also excluded, according to t
he instructor anwser to question "Q4 & Q5: Is Gross = 0 considered missing Gros
s value or not?""
df_grossNoNA = df[!is.na(df$Gross),]
df_grossClean = df_grossNoNA[df_grossNoNA$Gross != 0, ]
# Remove NA in 'Runtime' column
df_grossRuntimeNoNA = df_grossClean[!is.na(df_grossClean$Runtime),]
# Partition movies to short movies (Runtime < 60 minutes) and long movies (Runt
ime >60 min)
df_grossRuntimeNoNA$duration[df_grossRuntimeNoNA$Runtime< 60] = "short"
df_grossRuntimeNoNA$duration[df_grossRuntimeNoNA$Runtime >=60] = "long"
# Calculate some statistics of Budget, Gross for short and long movies
BudgetShort = df_grossRuntimeNoNA[df_grossRuntimeNoNA$duration == "short",]$Bud
get
GrossShort = df_grossRuntimeNoNA[df_grossRuntimeNoNA$duration == "short",]$Gros
s
medianBudget_short = median(BudgetShort)
medianGross_short = median(GrossShort)
GrossBudgeRatio_short = median(GrossShort/BudgetShort)
BudgetLong = df_grossRuntimeNoNA[df_grossRuntimeNoNA$duration == "long",]$Budge
t
GrossLong = df_grossRuntimeNoNA[df_grossRuntimeNoNA$duration == "long",]$Gross
medianBudget_long = median(BudgetLong)
medianGross_long = median(GrossLong)
GrossBudgeRatio_long = median(GrossLong/BudgetLong)
corShort = cor(log(BudgetShort), log(GrossShort))
corLong = cor(log(BudgetLong), log(GrossLong))
stat_duration_BG = data.frame(median_Budget = c(medianBudget_short, medianBudge
t_long), median_Gross = c(medianGross_short, medianGross_long), Correlation_Bud
getGross = c(corShort, corLong), GrossBudgetRatio = c(GrossBudgeRatio_short, Gr
ossBudgeRatio_long))
row.names(stat_duration_BG) = c("Short", "Long")
stat_duration_BG
```

	median_Budget <dbl>	median_Gross <dbl>	Correlation_BudgetGross <dbl>	GrossBudgetRatio <dbl>
Short	12650000	26650963	0.8068725	2.684273
Long	20000000	35538668	0.6877232	1.904697

2 rows

Hide

```
# Plot the Gross against Budget at log-scale for short and long movies
qplot(x=Budget, y=Gross, log="xy", facets = .~duration, data = df_grossRuntimeNoNA, main = "Gross vs. Budget by movie duration")
```

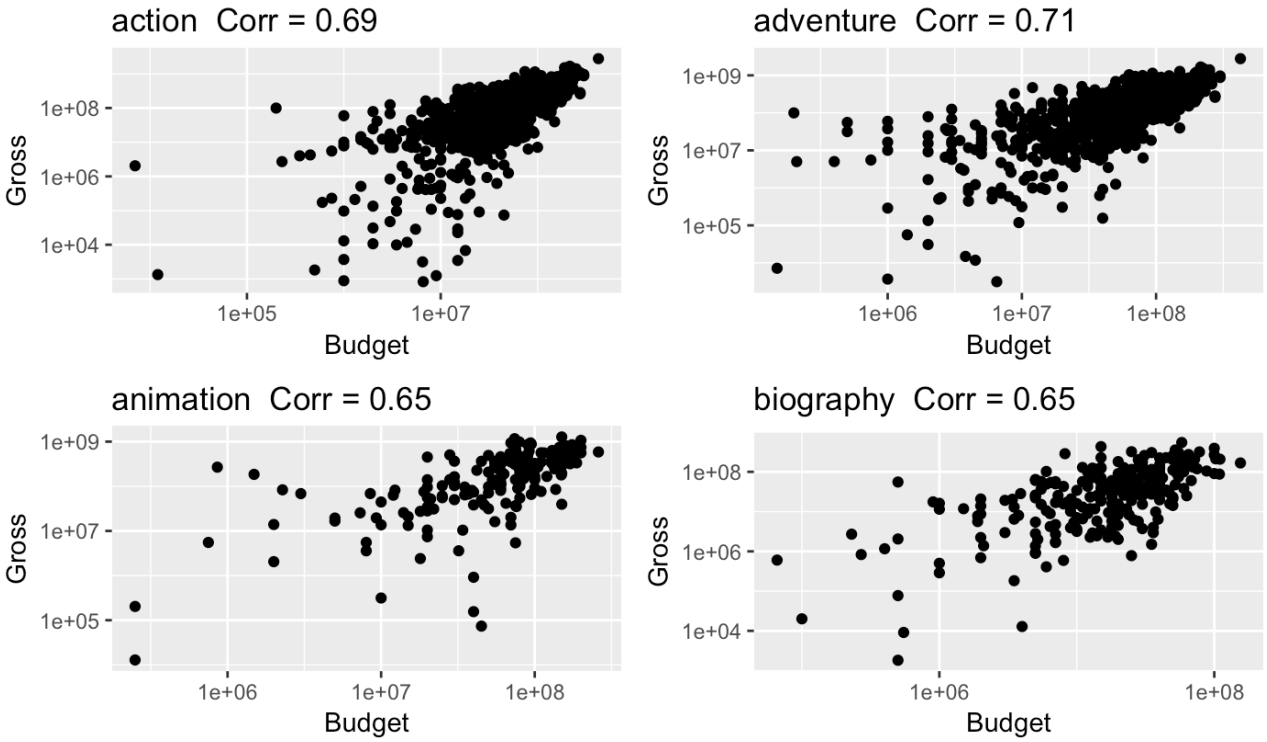


Hide

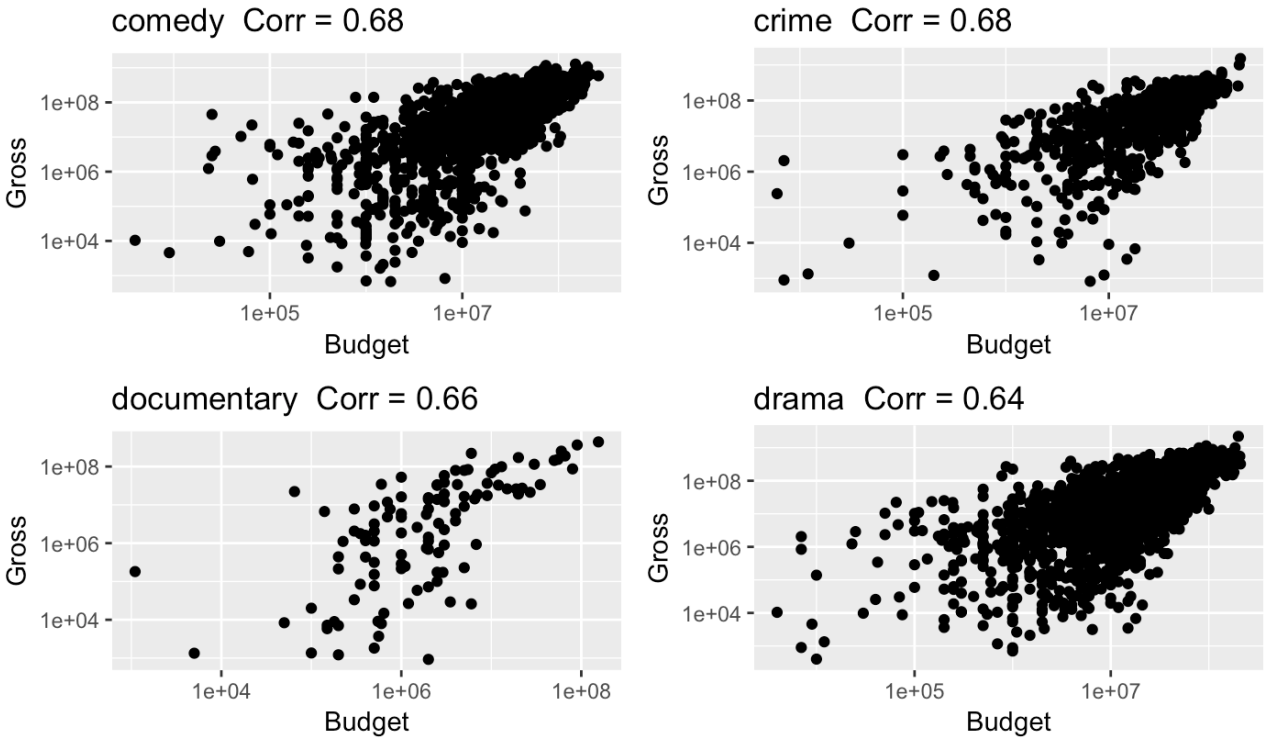
```
# Study the relationships between Gross, Budget and Genres
library(ggplot2)
library(grid)
library(gridExtra)
# Calculate the ratio of Gross and Budget, which is a good standard to characte
rize movies' success
df_grossNoNA = df[!is.na(df$Gross),]
df_grossClean = df_grossNoNA[df_grossNoNA$Gross != 0, ]
df_grossClean$GBratio = df_grossClean$Gross/df_grossClean$Budget
# Rename column "sci-fi" as "scifi", easier for data processing
colnames(df_grossClean)[60] <- "scifi"
# Create a new dataframe to summarize Number of movies, median and mean of Gros
s/Budget ratio for all the genres
last_genreCol = length(df_grossClean)-1
first_genreCol = last_genreCol - (length(genre_dict) -2)
GBratio_genre_sum = data.frame(Genre = colnames(df_grossClean)[first_genreCol:l
ast_genreCol], NumOfMovies=NA, GBratio_median = NA, GBratio_mean = NA, Budget_m
edian = NA, Gross_median = NA)
# Obtain the number of movies for each genre
NumOfMovies= apply(df_grossClean[,first_genreCol:last_genreCol], 2, sum)
GBratio_genre_sum$NumOfMovies = NumOfMovies
# Exclude the genres with too few movies, which may not reliable for relationsh
ip study
GBratio_genre_sum = GBratio_genre_sum[GBratio_genre_sum$NumOfMovies > 10,]
# Calculate the median of Budget, Gross and Gross/Budget ratio for each genre
for (i in seq(1, nrow(GBratio_genre_sum), by =1)) {
   budget_median = ddply(df_grossClean, GBratio_genre_sum$Genre[i], summarise,
median.Budget = median(Budget, na.rm = TRUE))
   gross_median = ddply(df_grossClean, GBratio_genre_sum$Genre[i], summarise, m
edian.Gross = median(Gross, na.rm = TRUE))
   GBratio_genre_sum[i,5] = budget_median[2, 2]
   GBratio_genre_sum[i,6] = gross_median[2, 2]
   GBratio_median = ddply(df_grossClean, GBratio_genre_sum$Genre[i], summarise,
median.GBratio = median(GBratio, na.rm = TRUE), mean.GBratio=mean(GBratio, na.r
m = TRUE))
   GBratio_genre_sum[i,3] = GBratio_median[2, 2]
   GBratio_genre_sum[i,4] = GBratio_median[2, 3]
}
# Create a new dataframe genre_BudgetGross to store the budget, gross in variou
s genres, which is easy for plot
genre = vector(mode = "character", length = 0)
budget = vector(mode = "numeric", length = 0)
gross = vector(mode = "numeric", length = 0)
for (i in seq(1, nrow(GBratio_genre_sum), by=1)) {
  genre = c(genre, rep(as.character(GBratio_genre_sum$Genre[i]), GBratio_genre_
sum$NumOfMovies[i]))
```

```
  name = as.character(GBratio_genre_sum$Genre[i])
  budget = c(budget, df_grossClean[df_grossClean[[name]] == 1,]$Budget)
  gross = c(gross, df_grossClean[df_grossClean[[name]] == 1,]$Gross)
}
genre_BudgetGross = data.frame(Genre = genre, Budget = budget, Gross = gross)
for (i in seq(1, nrow(GBratio_genre_sum), by =1)) {
  BudgeGross_1genre = genre_BudgetGross[genre_BudgetGross$Genre == as.character
(GBratio_genre_sum$Genre[i]),]
  GBratio_genre_sum$GBcorrelation[i] = cor(log(BudgeGross_1genre$Budget), log(B
udgeGross_1genre$Gross))
}
# Plot the Relationship between Gross and Budget for each genre
pl <- lapply(1:22, function(i)
             qplot(Budget, Gross, log = "xy", data = genre_BudgetGross[genre_Bu
dgetGross$Genre == as.character(GBratio_genre_sum$Genre[i]),], main = paste(GBr
atio_genre_sum$Genre[i], " Corr =", format(round(GBratio_genre_sum$GBcorrelatio
n[i], 2), nsmall = 2))))
ml <- marrangeGrob(pl, nrow=2, ncol=2)
ml
```

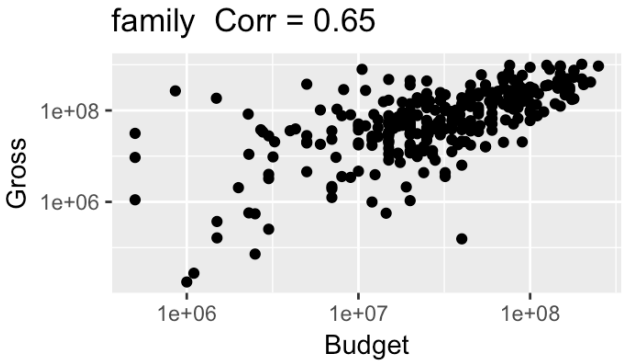
page 1 of 6

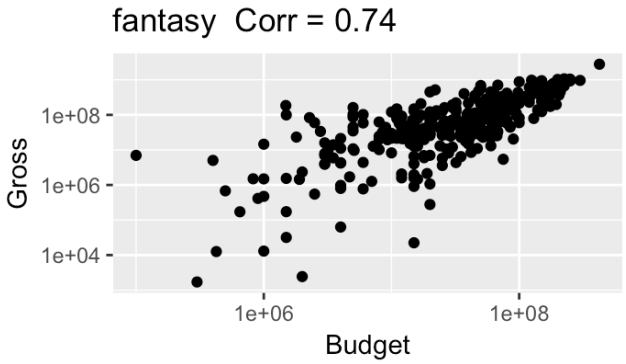


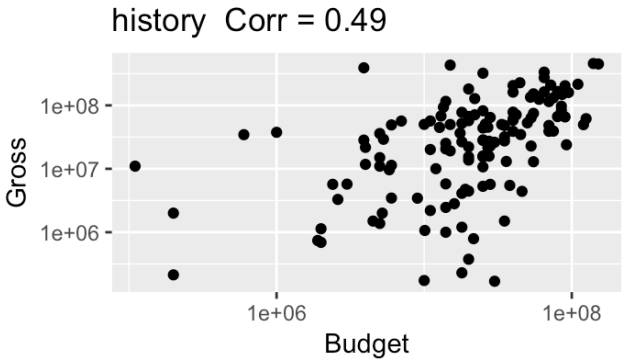
page 2 of 6

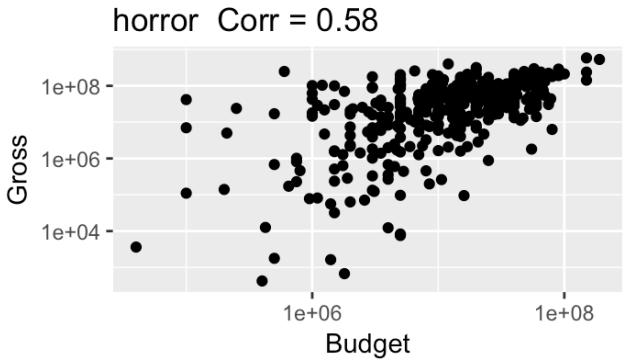


page 3 of 6

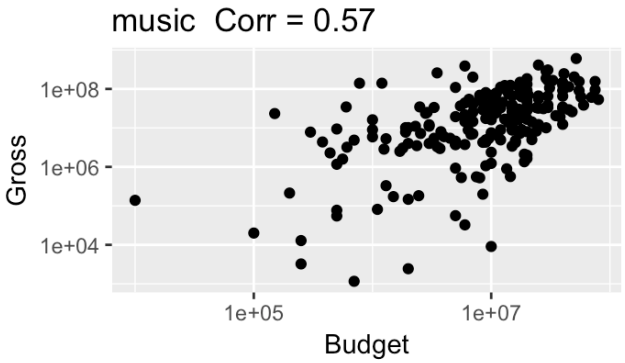


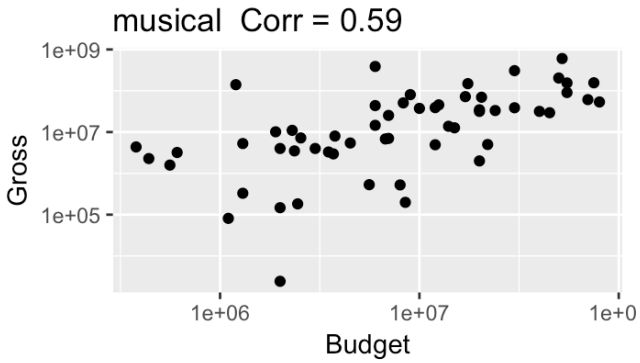


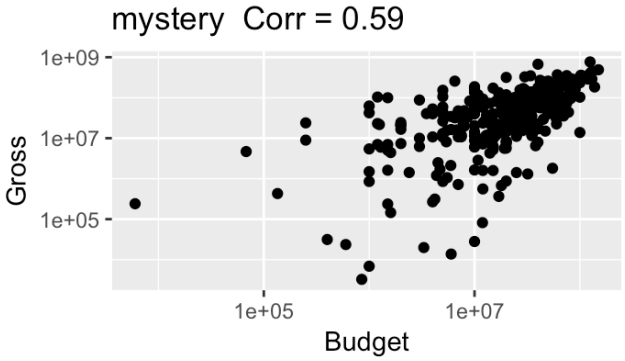


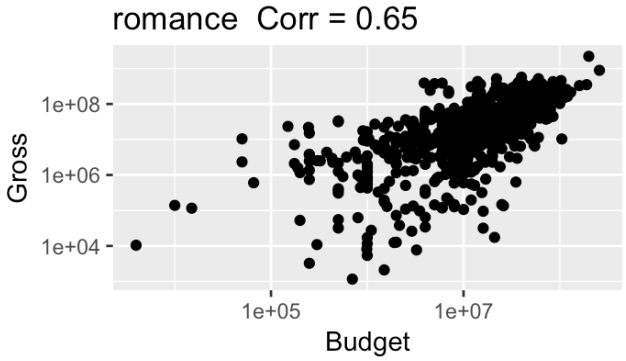


page 4 of 6

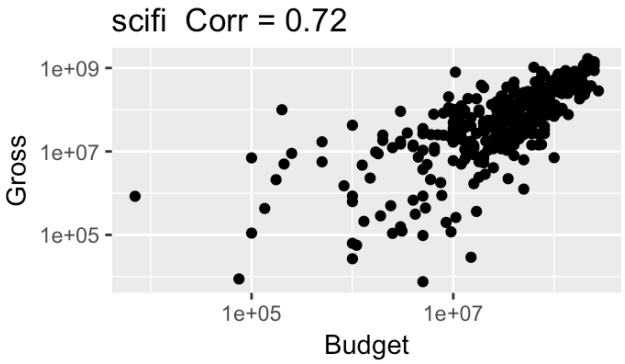


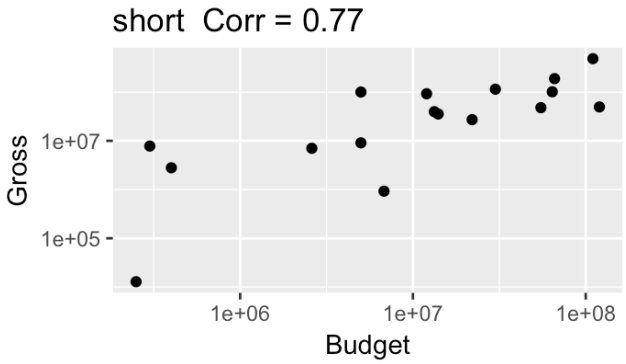


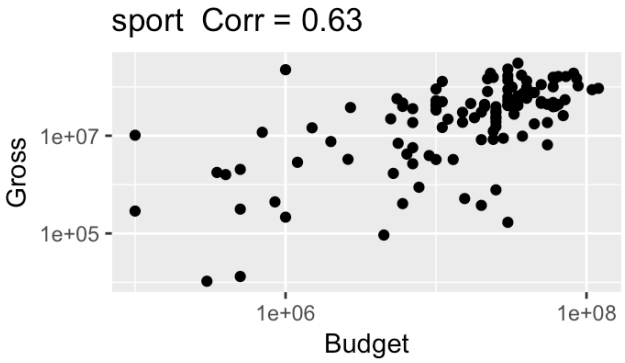


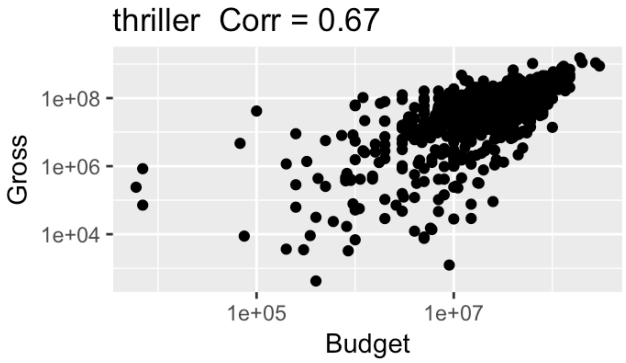


page 5 of 6

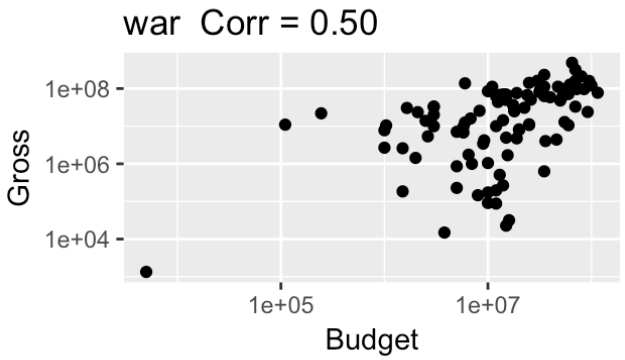


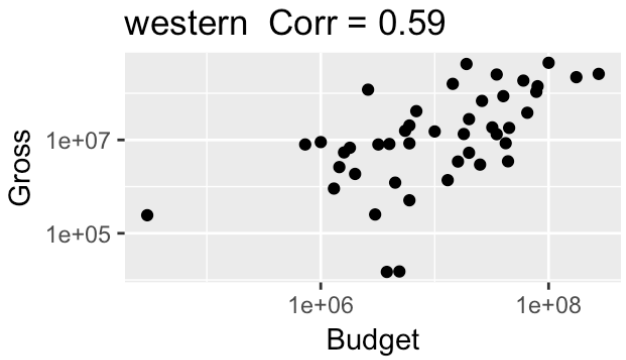






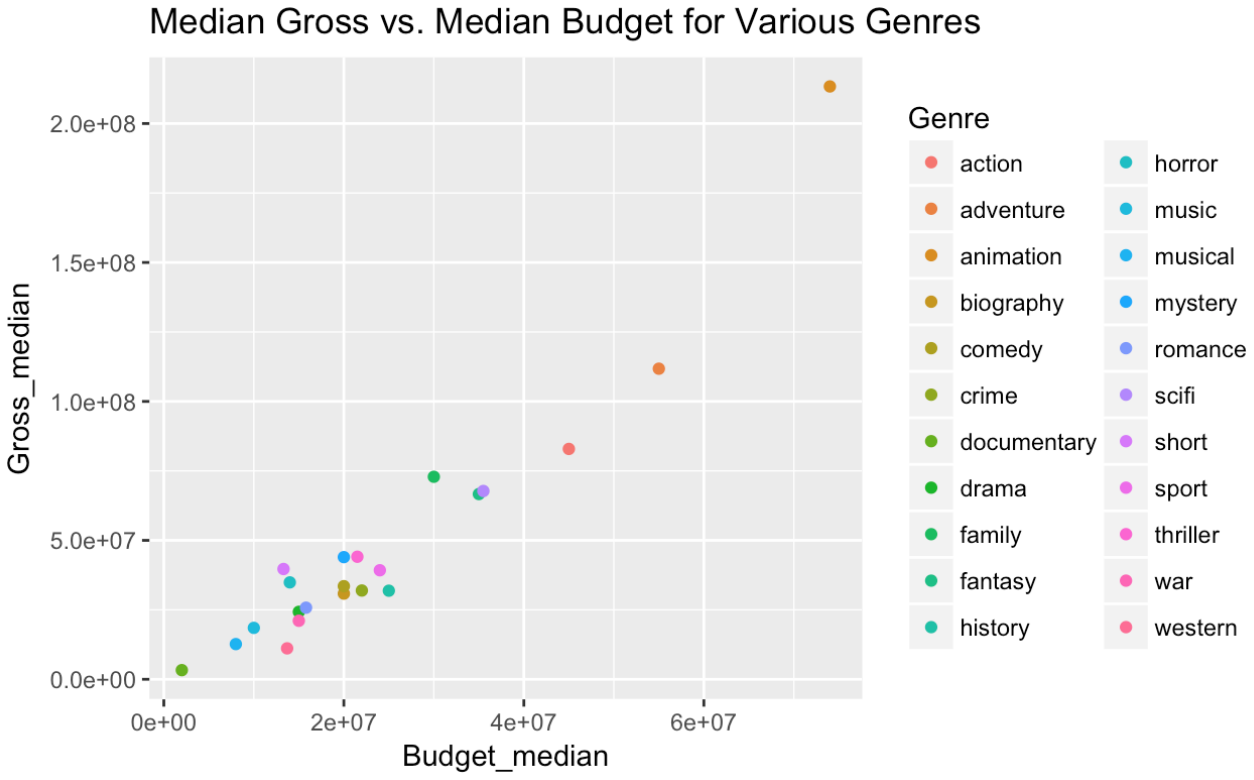
page 6 of 6





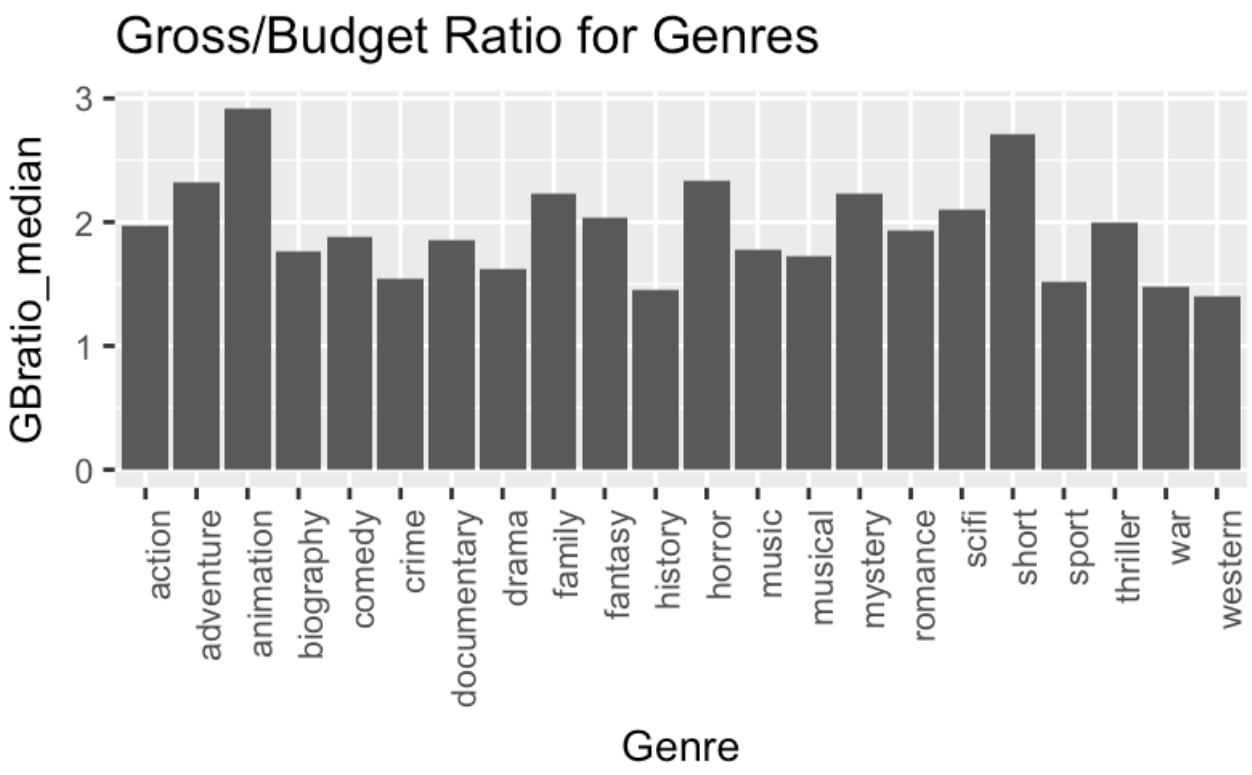
Hide

```
# Plot the Relationship between median Gross and median Budget for each genre
qplot(x= Budget_median, y=Gross_median, data = GBratio_genre_sum, colour = Genr
e, main = "Median Gross vs. Median Budget for Various Genres")
```



Hide

```
# Plot the Gross/Budget ratio for each genre
ggplot(GBratio_genre_sum, aes(x=Genre, y=GBratio_median)) + geom_bar(stat = "id
entity") +
    theme(text = element_text(size=10), axis.text.x = element_text(angle=90, hj
ust=1)) + ggtitle("Gross/Budget Ratio for Genres")
```



Hide

```
#p2 = ggplot(GBratio_genre_sum, aes(x=Genre, y=GBratio_mean)) + geom_bar(stat =
"identity") +
#    theme(text = element_text(size=10), axis.text.x = element_text(angle=90, h
just=1))
#}
 # GBratio_median = ddply(df_grossNoNA, , summarise, median.GBratio = median(GB
ratio, na.rm = TRUE))
#  GBratio_genre_sum[i,3] = GBratio_median[2, 2]
```

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: Firstly, I studied the Budget/Gross relation with different movie durations. Movies with runtime shorter than 60 mins are labeled as short duration and movies longer than 60 are classified as long duration movies. Then, I plot the Gross again Budget for ‘short’ movies and ‘long’ movies in a facet graph. It is obvious that for both short and long movies the ‘Gross’ and ‘Budget’ are positively correlated. The long movies have broader range of ‘Budget’ and ‘Gross’, but we should note that the there are much more movies in the ‘long’ duration class than movies in ‘short’ class, so the broader range may result from less number of data points. From the graph, I see a tighter correlation of ‘Gross’ and ‘Budget’ for the short movies than long movies. In addition to the graph, I calculated several statistical parameters for the ‘Gross’ and ‘Budget’ for short and long movies, as shown in the table

before the facet graph. On average, short movies has less Budget and less Gross than long movies. The correlation of log(Gross) and log(Budget) for short movies is about 0.81, but that correlation for long movies is only 0.69, indicating the gross revenue of short movies is more predictable than long movies. I also calculated the averaged ratio of Gross and Budget, which is a direct parameter to describe the profitability of two types of movies. The Gross/Budget ratio of short movies and long movies are 2.68 and 1.90, respectively, suggesting long movies have a little better profitability. As a conclusion, the gross of short movies are more predictable and profitable than long movies.

Secondly, I studied the Budget/Gross correlation with different genres. For genres who have less than 10 movies with valid 'Gross' values, I excluded them from analysis, because such little data would not be reliable for analysis. (I note that I treat 'music' and 'musical' movies as two different type of genres, because I don't know the standard that the person used to class each movie. I assume 'music' movies are telling stories related with music or talking about music, but 'musical' movies are movies with a lot of singing and dancing in it but telling stories inrelavent to music.) Then, I analyzed the Budget/Gross relationship for the remaining 22 genres in following steps: a) I plot the all pairs of 'Gross' against 'Budget' for every genre. As shown in the 22 scatter point plots, all of them show positive correlation between 'Gross' and 'Budget'. However, some genres have better correlation than others. The correlation value is denoted on each graph. History and War genres have the worst correlation, both of them have correlation less than 0.5. Genres short, fantasy, sci-fi and adventure have best Gross-Budget correlation (>0.7). Other genres have the correlations between 0.5 to 0.7. These observations indicate the gross revenue of short, fantasy, sci-fi and adventure movies are more predictable by their budget. b) Noticing the quite different x- and y- axises scale of the 22 scatter-points plots of different genres, I found the budget to make different types of movies is quite different. So I investigate the median of budget and median of gross for each type of movie in one plot, using colorful points representing different genres. From this graph, I found the animation movies have very high budget and gross than all other movies. The adventure movies and action movies rank the second and third for Budget and Gross. Not surprisingly, the documentary movies have lest budget and gross. From this graph, I clearly see that it is unfair to directly compare the Gross between different types of movies, because their budget vary too much. So similarly as for movie duration part, I calculated the ratio of Gross/Budget, which is a direct parameter to describe the profitability. c) The median of Gross/Budget ratio is plotted against its genre in a histogram. The animation movie shows best Gross/Budget ratio with median 2.92, and 'short' movies ranks second (the Gross/Budget ratio median is 2.71). Genres adventure, family, fantasy, horror, mystery, sci-fi, short and thriller are all have Gross/Budget ratio bigger or equal 2.00. The ratio is 1.97 for action movie also close to 2.00.

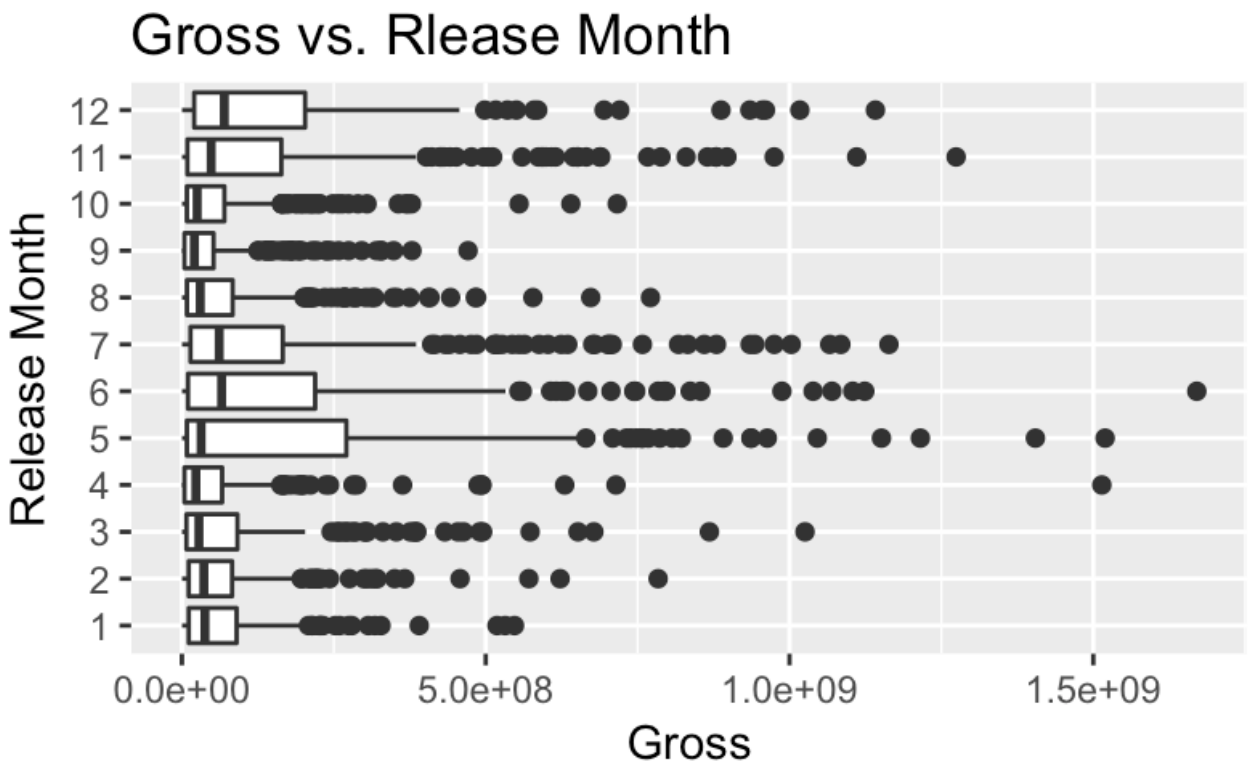
As a conclusion, genres short, fantasy, sci-fi and adventure are most predictable for their gross revenue and they all show good Gross/Budget ratio. The amination movies can have very good gross revenue and with decent predictability.

Investigation for Gross Revenue related to Release Month a) I plot the Gross revenue in box plot for different months. May shows broadest box, followed by June, December, July and November. The June July and December have highest gross revenue median, and November ranks the fourth highest gross median. This result is expected, because schools stop classes and many companies have flexible summer hours in June and July. November and December are holiday season. These four months are all vacation months, so people have more time to watch movies in movie theaters. From

my knowledge, many movie companies aim at these vacation seasons and schedule to show their best movies. b) I plot the Gross/Budget ratio in box plot for different months. June, July and December still have highest Gross/Budget ratio medians and May shows comparable Gross/Budget ratio median to them. Generally, schools stop classes at the last a couple of weeks in May, so it is reasonble for May to show broad distribution of Gross and high Gross/Budget ratio (students have to prepare for finals in the first a couple of weeks in May and then start summer vacation from late May).

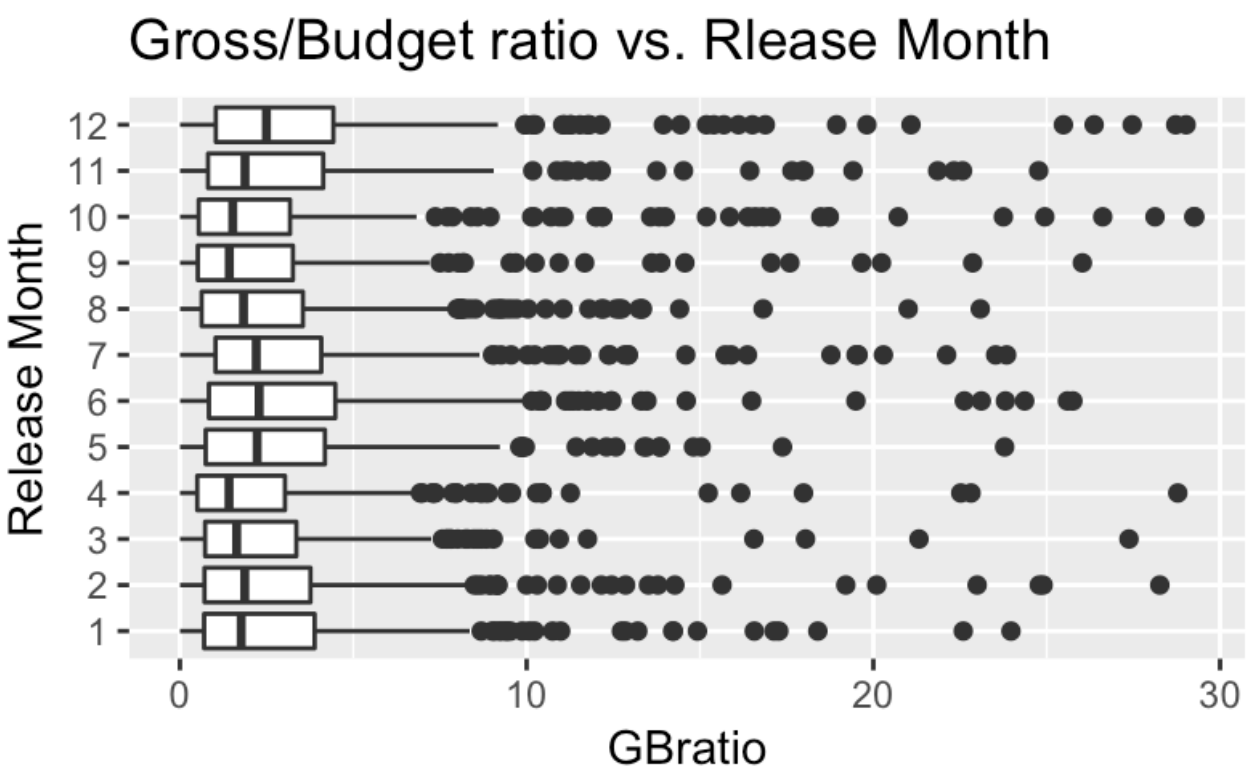
Hide

```
# TODO: Investigate if Gross Revenue is related to Release Month
# loop through the rows of df_grossClean, split the strings in df_grossClean$Re
leased and exact the released month from it.
for (i in seq(1, nrow(df_grossClean), by = 1)) {
  if (is.na(df$Released[i])) {
    df_grossClean$ReleasedMonth[i] = NA
  }
  release_split = strsplit(as.character(df_grossClean$Released[i]), split = "-"
, fixed = T)
  df_grossClean$ReleasedMonth[i] = as.numeric(release_split[[1]][2])
}
# Remove rows with release month = NA. Plot the Gross against release month usi
ng boxplot, to view the main data better omit the Gross values larger than 2e+0
9.
df_grossMonthClean = df_grossClean[!is.na(df_grossClean$ReleasedMonth),]
ggplot(df_grossMonthClean[df_grossMonthClean$Gross<2e+09,], aes(x= as.factor(Re
leasedMonth), y = Gross)) + geom_boxplot() + coord_flip() + scale_x_discrete()
+ ggtitle("Gross vs. Rlease Month") + xlab("Release Month")
```



Hide

```
# Plot the Gross/Budget ratio against release month using boxplot, to view the
main data better omit the Gross/Budget ratio larger than 50.
ggplot(df_grossMonthClean[df_grossMonthClean$GBratio < 30,], aes(x= as.factor(R
eleasedMonth), y = GBratio)) + geom_boxplot() + coord_flip() + scale_x_discrete
() + ggtitle("Gross/Budget ratio vs. Rlease Month") + xlab("Release Month")
```



Hide

```
# Remove the mismatched rows
#df <- df[-mismatch,]
```

6. Process `Awards` column

The variable `Awards` describes nominations and awards in text format. Convert it to 2 numeric columns, the first capturing the number of wins, and the second capturing nominations. Replace the `Awards` column with these new columns, and then study the relationship of `Gross` revenue with respect to them.

Note that the format of the `Awards` column is not standard; you may have to use regular expressions to find the relevant values. Try your best to process them, and you may leave the ones that don't have enough information as NAs or set them to 0s.

Hide

```
# TODO: Convert Awards to 2 numeric columns: wins and nominations
df$Wins <- NA
df$Nominations <- NA
# Counter for movies without any wins or nominations
nonAward_count = 0
# Loop through every row in df
for (i in seq(1, nrow(df), by = 1)) {
  # Convert movies do not have any awards
  if (df$Awards[i] == "N/A") {
    df$Wins[i] = NA
    df$Nominations[i] = NA
    nonAward_count = nonAward_count + 1
  }
  # Convert df$Awards contents start with "Won [0-9]+".
  if (grepl("Won [0-9]+", df$Awards[i], ignore.case = T)) {
    won.regexpr = regexpr(pattern= "Won [0-9]+", text= df$Awards[i])
    won = regmatches(m=won.regexpr, x = df$Awards[i])
    won_split = unlist(strsplit(won, " "))
    won_num = as.numeric(won_split[2])
    # Convert movies df$Awards contents matching the regular expression "Won [0
-9]+.*Another"
    if (grepl("Won [0-9]+.*Another", df$Awards[i], ignore.case = T)) {
      if (grepl("Won [0-9]+.*Another [0-9]+ win.*& [0-9]+ nomination", df$Award
s[i], ignore.case = T)) {
        win.regexpr = regexpr(pattern= "[0-9]+ win", text= df$Awards[i])
        win = regmatches(m = win.regexpr, x = df$Awards[i])
        win_split = unlist(strsplit(win, " "))
        win_num = as.numeric(win_split[1])

        nomination.regexpr = regexpr(pattern= "[0-9]+ nomination", text= df$Awa
rds[i])
        nomination = regmatches(m = nomination.regexpr, x = df$Awards[i])
        nomination_split = unlist(strsplit(nomination, " "))
        nomination_num = as.numeric(nomination_split[1])

        df$Wins[i] = won_num + win_num
        df$Nominations[i] = nomination_num
      }

      else if (grepl("Won [0-9]+.*Another [0-9]+ win", df$Awards[i], ignore.cas
e = T)) {
        win.regexpr = regexpr(pattern= "[0-9]+ win", text= df$Awards[i])
        win = regmatches(m = win.regexpr, x = df$Awards[i])
        win_split = unlist(strsplit(win, " "))
        win_num = as.numeric(win_split[1])
```

```
        df$Wins[i] = won_num + win_num
      }
      # Convert movies df$Awards contents matching the regular expression "Won
[0-9]+.*Another [0-9]+ nomination"
      else {
        nomination.regexpr = regexpr(pattern= "[0-9]+ nomination", text= df$Awa
rds[i])
        nomination = regmatches(m = nomination.regexpr, x = df$Awards[i])
        nomination_split = unlist(strsplit(nomination, " "))
        nomination_num = as.numeric(nomination_split[1])

        df$Wins[i] = won_num
        df$Nominations[i] = nomination_num
      }
    }
    # Convert df$Awards contents start with "Won [0-9]+" and no other wins and
nominations.
    else {
      df$Wins[i] = won_num
    }
  }
  # Convert df$Awards contents start with "Nominated for [0-9]+".
  if (grepl("Nominated for [0-9]+", df$Awards[i], ignore.case = T)) {
    nominated.regexpr = regexpr(pattern= "Nominated for [0-9]+", text= df$Award
s[i])
    nominated = regmatches(m = nominated.regexpr, x = df$Awards[i])
    nominated_split = unlist(strsplit(nominated, " "))
    nominated_num = as.numeric(nominated_split[3])

    # Convert movies df$Awards contents matching the regular expression "Nomina
ted for     [0-9]+.*Another [0-9]+ win.*& [0-9]+ nomination"
    if (grepl("Nominated for [0-9]+.*Another [0-9]+ win.*& [0-9]+ nomination",
df$Awards[i], ignore.case = T)) {
      win.regexpr = regexpr(pattern= "[0-9]+ win", text= df$Awards[i])
      win = regmatches(m = win.regexpr, x = df$Awards[i])
      win_split = unlist(strsplit(win, " "))
      win_num = as.numeric(win_split[1])

      nomination.regexpr = regexpr(pattern= "[0-9]+ nomination", text= df$Award
s[i])
      nomination = regmatches(m = nomination.regexpr, x = df$Awards[i])
      nomination_split = unlist(strsplit(nomination, " "))
      nomination_num = as.numeric(nomination_split[1])

      df$Wins[i] = win_num
      df$Nominations[i] = nomination_num + nominated_num
```

```
    }
    # Convert movies df$Awards contents matching the regular expression "Nomina
ted for     [0-9]+.*Another [0-9]+ win"
    else if (grepl("Nominated for [0-9]+.*Another [0-9]+ win", df$Awards[i], ig
nore.case = T)) {
      win.regexpr = regexpr(pattern= "[0-9]+ win", text= df$Awards[i])
      win = regmatches(m = win.regexpr, x = df$Awards[i])
      win_split = unlist(strsplit(win, " "))
      win_num = as.numeric(win_split[1])

      df$Wins[i] = win_num
      df$Nominations[i] = nominated_num
    }
    # Convert movies df$Awards contents matching the regular expression "Nomina
ted for     [0-9]+.*Another [0-9]+ nomination"
    else if (grepl("Nominated for [0-9]+.*Another [0-9]+ nomination", df$Awards
[i], ignore.case = T)) {
      nomination.regexpr = regexpr(pattern= "[0-9]+ nomination", text= df$Award
s[i])
      nomination = regmatches(m = nomination.regexpr, x = df$Awards[i])
      nomination_split = unlist(strsplit(nomination, " "))
      nomination_num = as.numeric(nomination_split[1])

      df$Nominations[i] = nomination_num + nominated_num
    }
    # Convert df$Awards contents have "Nominated for [0-9]+" only without other
awards.
    else {
      df$Nominations[i] = nominated_num
    }
  }

  # Convert df$Awards contents start with "^[0-9]+ win".
  if (grepl("^[0-9]+ win", df$Awards[i], ignore.case = T)) {
    win.regexpr = regexpr(pattern= "[0-9]+ win", text= df$Awards[i])
    win = regmatches(m = win.regexpr, x = df$Awards[i])
    win_split = unlist(strsplit(win, " "))
    win_num = as.numeric(win_split[1])

    if (grepl("^[0-9]+ win.*& [0-9]+ nomination", df$Awards[i], ignore.case = T
)) {
      nomination.regexpr = regexpr(pattern= "[0-9]+ nomination", text= df$Award
s[i])
      nomination = regmatches(m = nomination.regexpr, x = df$Awards[i])
      nomination_split = unlist(strsplit(nomination, " "))
      nomination_num = as.numeric(nomination_split[1])
```

```
      df$Wins[i] = win_num
      df$Nominations[i] = nomination_num
    }
    else {
      df$Wins[i] = win_num
    }
  }

  # Convert df$Awards contents start with "^[0-9]+ nomination".
  if (grepl("^[0-9]+ nomination", df$Awards[i], ignore.case = T)) {
    nomination.regexpr = regexpr(pattern= "^[0-9]+ nomination", text= df$Awards
[i])
    nomination = regmatches(m = nomination.regexpr, x = df$Awards[i])
    nomination_split = unlist(strsplit(nomination, " "))
    nomination_num = as.numeric(nomination_split[1])
    df$Nominations[i] = nomination_num
  }
}
# Calculate how many movies have wins and nominations
wins_count = nrow(df) - sum(is.na(df$Wins))
nominations_count = nrow(df) - sum(is.na(df$Nominations))
winOrNomination = nrow(df) - nonAward_count
wins_count
```

```
[1] 10327
```

Hide

```
nominations_count
```

```
[1] 10893
```

Hide

```
winOrNomination
```

```
[1] 13954
```

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

A: By observing the 'Award' column, I found expressions are in the format "number win/wins", "number nomination/nominations", "number win/wins & number nomination/nominations", "Won number awards. Another number win/wins and number nomination/nomiations", "Nominated for

number awards. Another number win/wins and number nomination/nomiations" and so on.

I construct my conversion using multiple flow of controls with if, else if and else clauses. First, I use one if clause to check and convert the "N/A" in the 'Award' column to NA in the new 'Wins' column and the new 'Nominations' column.

Secondly, I deal with the cells starts with "Won [0-9]+". In this part, four types of expressions are treated and converted in four if/else if/else clauses, respectively, their regular expressions are "Won [0-9]+*.Another [0-9]+ win.*& [0-9]+ nomination", "Won [0-9]+*.Another [0-9]+ win","Nominated for [0-9]+.*Another [0-9]+ nomination" and simply "Won [0-9]+" no "Another" wins and nominations. To check whether the content in each cell matches the regular expression in the condiction, I use the grepl() function. To convert the text in each cell to numeric value for the new "Wins" and "Nominations" column, I use the regexpr(), regmatches() and strsplit() functions.

Thirdly, I deal with the cells starts with "Nominated for [0-9]+". Similarly as the above situation, in this part, there are also four types of expressions need to be converted accordingly. Their regular expressions are "Nominated for [0-9]+*.Another [0-9]+ win.*& [0-9]+ nomination", "Nominated for [0-9]+*.Another [0-9]+ win","Nominated for [0-9]+.*Another [0-9]+ nomination" and simply "Nominated for [0-9]+".

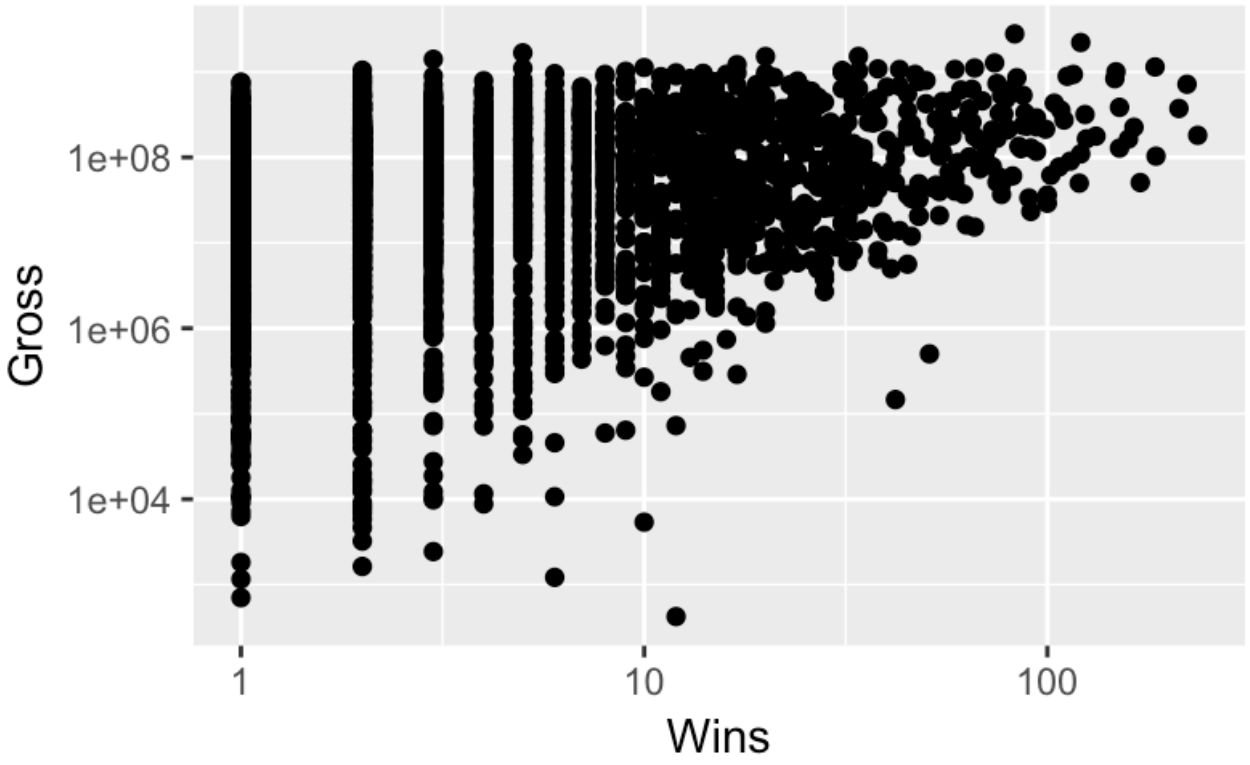
Forthly, I deal with the cells starts with "[0-9]+ win/wins". There are two possible situatons, one is "^{1}+ win.*& [0-9]+ nomination" and the other one is simply "^{2}+ win" without nominations. The '^' in these regular expressions are necessary to match these expressions from the beginning of the cell content, because these expressons also match the "Another" sentence after "Won [0-9]+" and "Nominated for [0-9]+" sentences in the second and third conditions.

At last, I deal with the cells starts with "^{3}+ nomination". Only one type of expression match this regular expression.

From my calculation, there are 13954 rows having valid win or nomiations. There are 10327 rows having valid wins, there are 10893 rows having valid nominations.

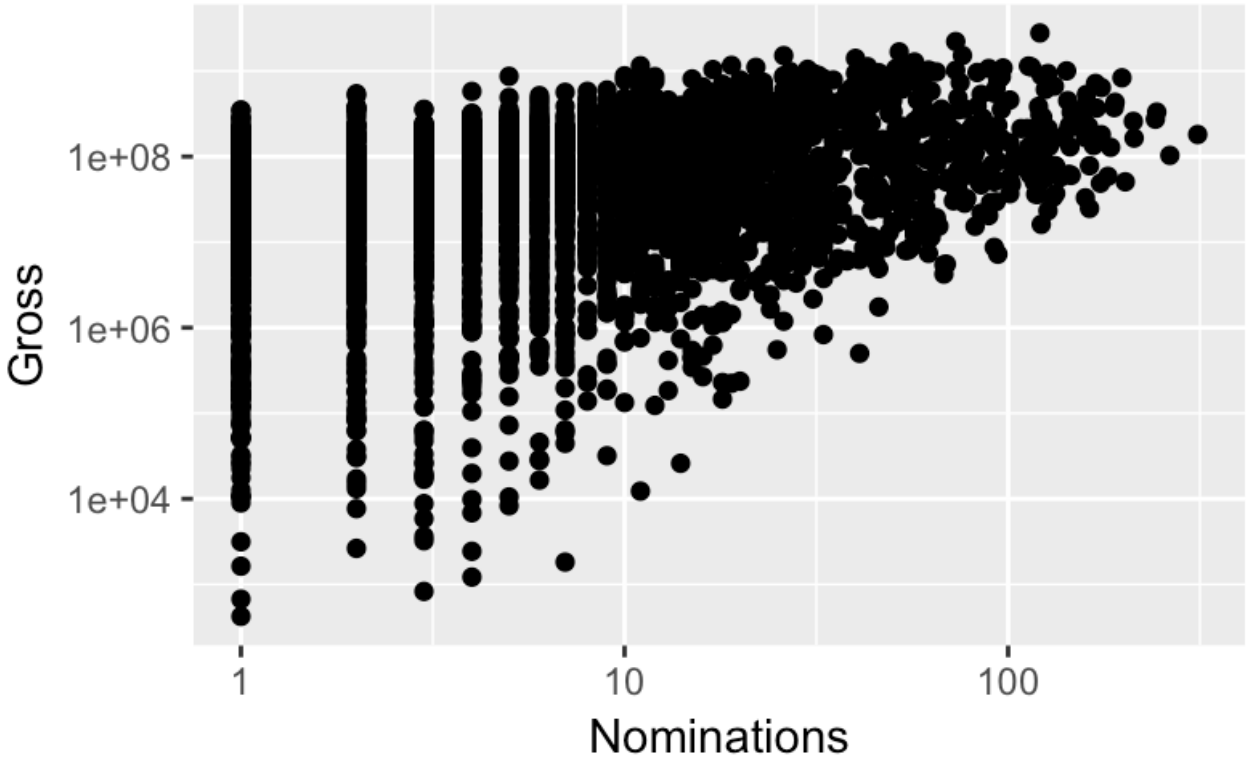
Hide

```
# TODO: Plot Gross revenue against wins and nominations
df_grossNoNA = df[!is.na(df$Gross),]
df_grossClean = df_grossNoNA[df_grossNoNA$Gross != 0, ]
df_GrossWinNoNA = df_grossClean[!is.na(df_grossClean$Wins),]
df_GrossNomnNoNA = df_grossClean[!is.na(df_grossClean$Nominations),]
library(ggplot2)
qplot(x=Wins, y=Gross, log="xy", data = df_GrossWinNoNA)
```



Hide

```
qplot(x=Nominations, y=Gross, log="xy", data = df_GrossNomnNoNA)
```



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

A: Using the log-scale for both Gross and Wins axis, I can see a upper-left triangle distribution of the Gross revenue against the number of wins. The movies won a lot of awards always have very high gross revenue. For example, the movies won 100 awards always have gross revenue larger than 1e+07 and the highest one has revenue larger than 1e+09. For movies with low number of wins, the gross revenue spreads in a broad range. They can have very little gross revenue close to zero but also can be very successful in the gross revenue around 1e+09, which is very close to movies won many awards. Using the log-scale plot for Gross against Nominations, I found similar distribution as the Gross vs. Wins graph.

These results are reasonable. Because movies won many awards must be very good movies, they should have many audiences and make high profit. For the movies won fewer awards, they may be not so good in all aspects, but they could still be fun for audiences to watch or be successful at marketing. These movies won few awards may also be very unattrative for audience but good at some point, so they can win a few awards but not much gross revenue.

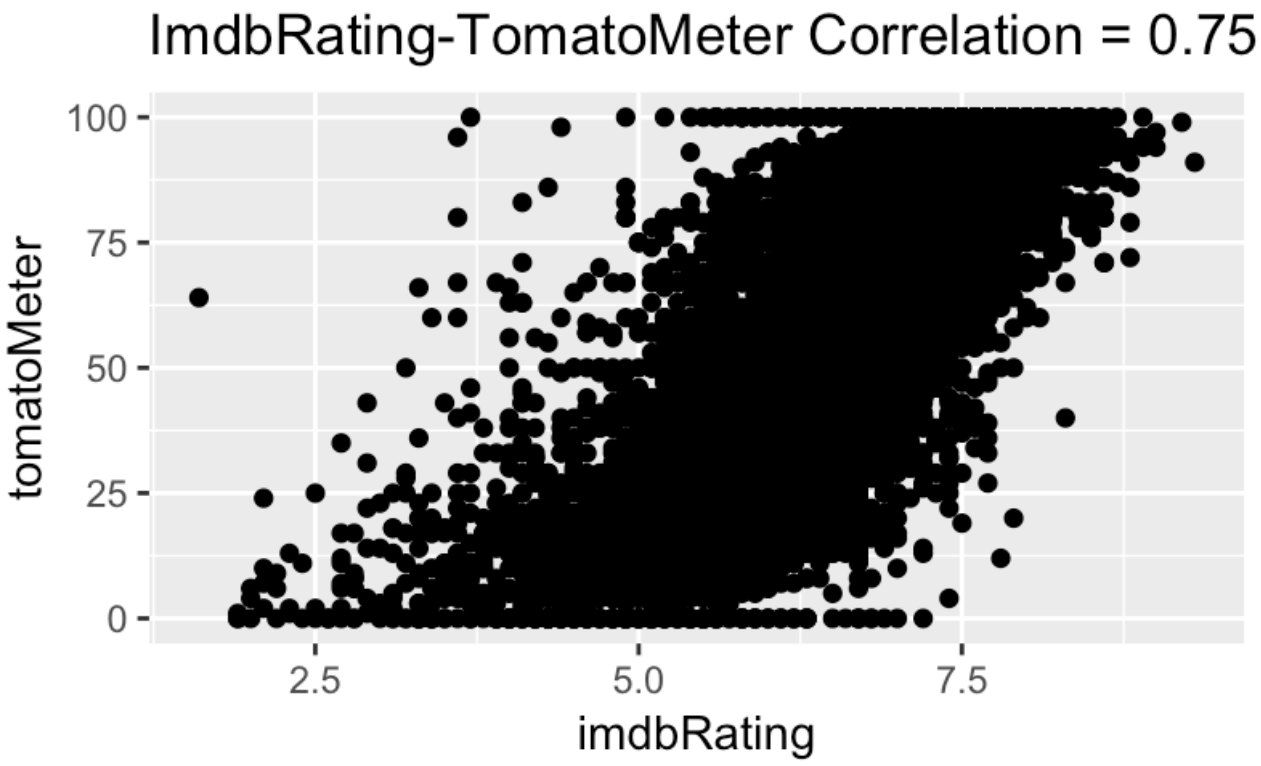
7. Movie ratings from IMDb and Rotten Tomatoes

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 (https://www.rottentomatoes.com/about) and www.imdb.com/help/show_leaf?votestopfaq (http://%20www.imdb.com/help/show_leaf?votestopfaq)).

Investigate the pairwise relationships between these different descriptors using graphs.

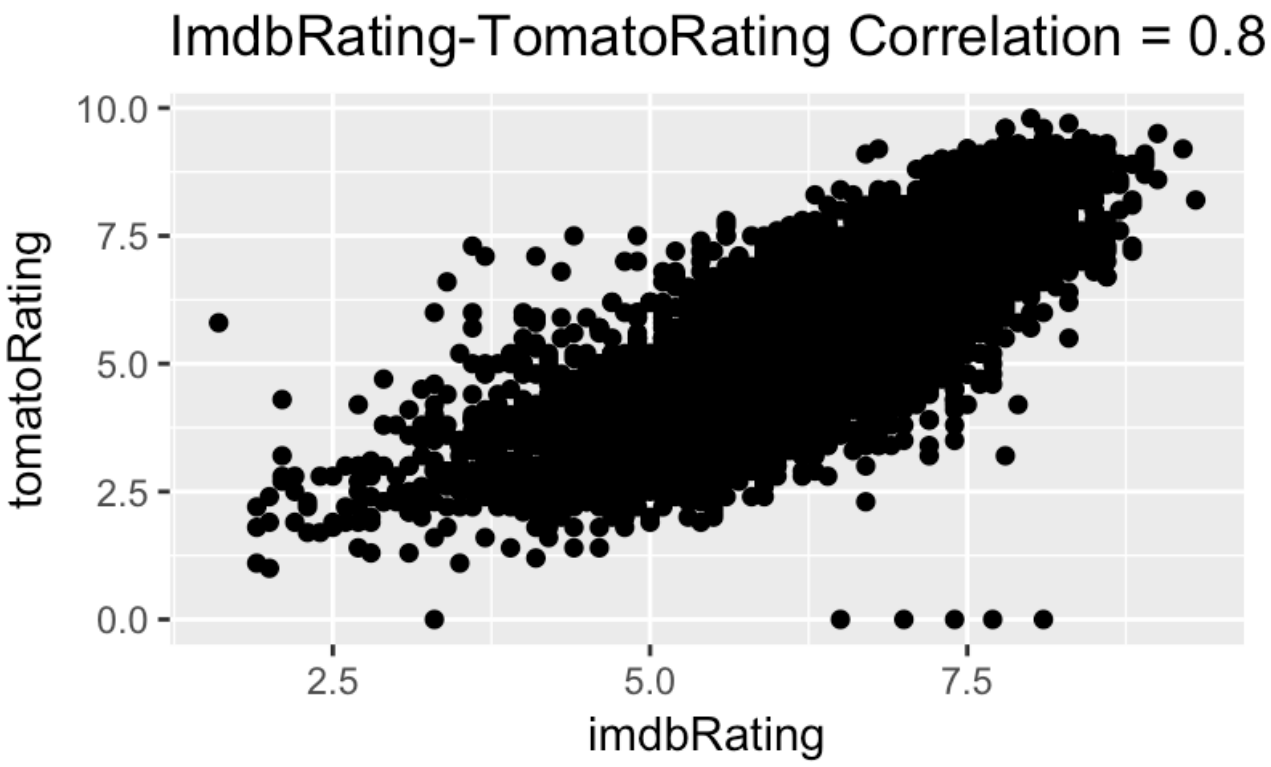
Hide

```
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
# Create proper clean dataframe to study the correlations.
df_imdbRatingNoNA = df[!is.na(df$imdbRating),]
df_imdbTomatoMeterNoNA = df_imdbRatingNoNA[!is.na(df_imdbRatingNoNA$tomatoMeter
),]
df_imdbTomatoUMNoNA = df_imdbRatingNoNA[!is.na(df_imdbRatingNoNA$tomatoUserMete
r),]
df_imdbTomatoRateNoNA = df_imdbRatingNoNA[!is.na(df_imdbRatingNoNA$tomatoRating
),]
df_tomatoMeterNoNA = df[!is.na(df$tomatoMeter),]
df_tomatoMeterRateNoNA = df_tomatoMeterNoNA[!is.na(df_tomatoMeterNoNA$tomatoRat
ing),]
df_tomatoMeter_UserMeterNoNA = df_tomatoMeterNoNA[!is.na(df_tomatoMeterNoNA$tom
atoUserMeter),]
# Plot imdbRating vs tomatoMeter
corr_imdbRating_tomatoMeter = cor(df_imdbTomatoMeterNoNA$imdbRating, df_imdbTom
atoMeterNoNA$tomatoMeter)
qplot(x=imdbRating, y=tomatoMeter, data = df_imdbTomatoMeterNoNA, main = paste(
"ImdbRating-TomatoMeter Correlation =", format(round(corr_imdbRating_tomatoMete
r, 2), nsmall = 2)))
```



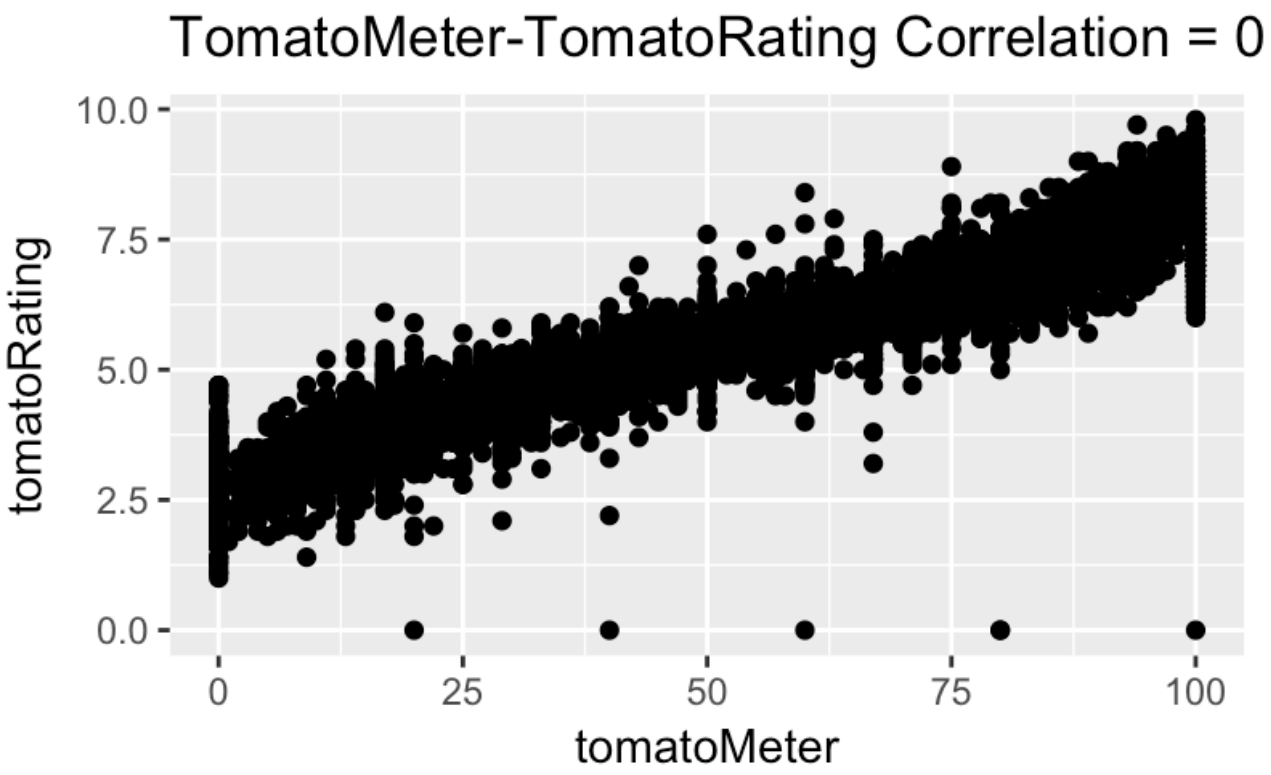
Hide

```
# Plot imdbRating vs tomatoRating
corr_imdbRating_tomatoRating = cor(df_imdbTomatoRateNoNA$imdbRating, df_imdbTom
atoRateNoNA$tomatoRating)
qplot(x=imdbRating, y=tomatoRating, data = df_imdbTomatoRateNoNA, main = paste(
"ImdbRating-TomatoRating Correlation =", format(round(corr_imdbRating_tomatoRat
ing, 2), nsmall = 2)))
```



Hide

```
# Plot tomatoMeter vs tomatoRating
corr_tomatoMeter_tomatoRating = cor(df_tomatoMeterRateNoNA$tomatoMeter, df_toma
toMeterRateNoNA$tomatoRating)
qplot(x=tomatoMeter, y=tomatoRating, data = df_tomatoMeterRateNoNA, main = past
e("TomatoMeter-TomatoRating Correlation =", format(round(corr_tomatoMeter_tomat
oRating, 2), nsmall = 2)))
```



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

A: Firstly, I compared the user rating of IMDb (column 'imdbRating') with the critics rating TomatoMeter (column 'tomatoMeter'). They are positively correlated with correlation 0.75. The range of imdbRating is between 0-10, and in the dataset we have very few imdbRatings smaller than 1.2 and very few higher than 9.0. The tomatoMeter ranges from 0 to 100, and our dataset covers the full range. For one movies with similar tomatoMeter values they may have different imdbRatings. For example, movies with tomatoMeter = 50, their imdbRatings may range from 3.0-8.0, but mainly in between 5.0 and 7.5.

Secondly, I compared imdb rating with another critics rating TomatoRating (column 'tomatoRating'). They have better positive correlation (corr = 0.80) than imdbRating-TomatoMeter. TomatoRatings range from 0-10.0. Similar as tomatoMeter, I can see movies with tomatoRating equals to 1.0 all the way up to 10.0. Several points with tomatoRating value 0.0 may be incorrect values or corrupted values. The better correlation between imdbRating and tomatoRating may be due to they are in similar rating scale, so it is more difficult to have larger inconsistency in imdbRating dimension and tomatoRating.

Thirdly, I compared the critics rating tomatoRating vs. tomatoMeter. Not surprisingly, they have very good correlation at 0.94. There are five points with tomatoRating = 0, but have substantial tomatoMeter values. This may indicate these five data points are corrupted. On the other hand, for those points with tomatoMeter values equal to 0, we need to be careful to interpret them. We cannot simply say they are all corrupted, because their tomatoRating is also quite low (all of them are lower than 5). Some of them may be corrupted data, but most of them should be simply bad movies. We need more data to understand these points better.

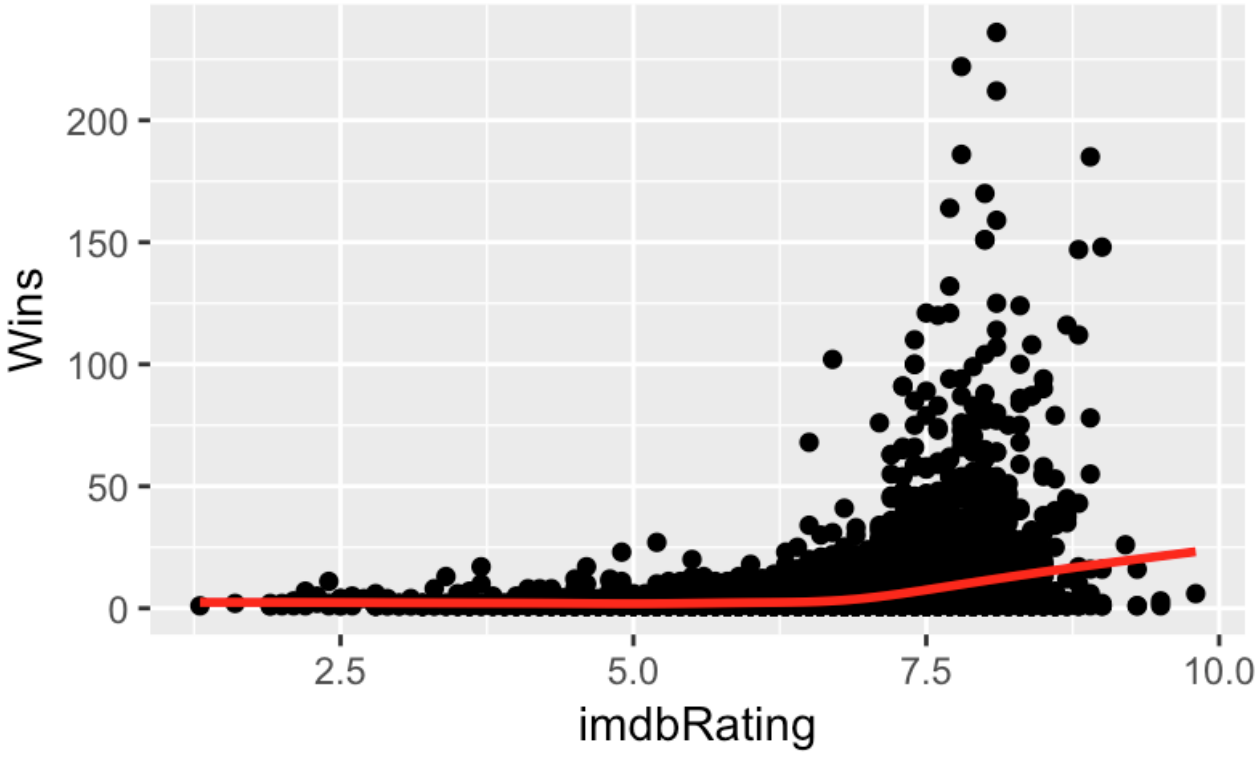
8. Ratings and awards

These ratings typically reflect the general appeal of the movie to the public or gather opinions from a larger body of critics. Whereas awards are given by professional societies that may evaluate a movie on specific attributes, such as artistic performance, screenplay, sound design, etc.

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

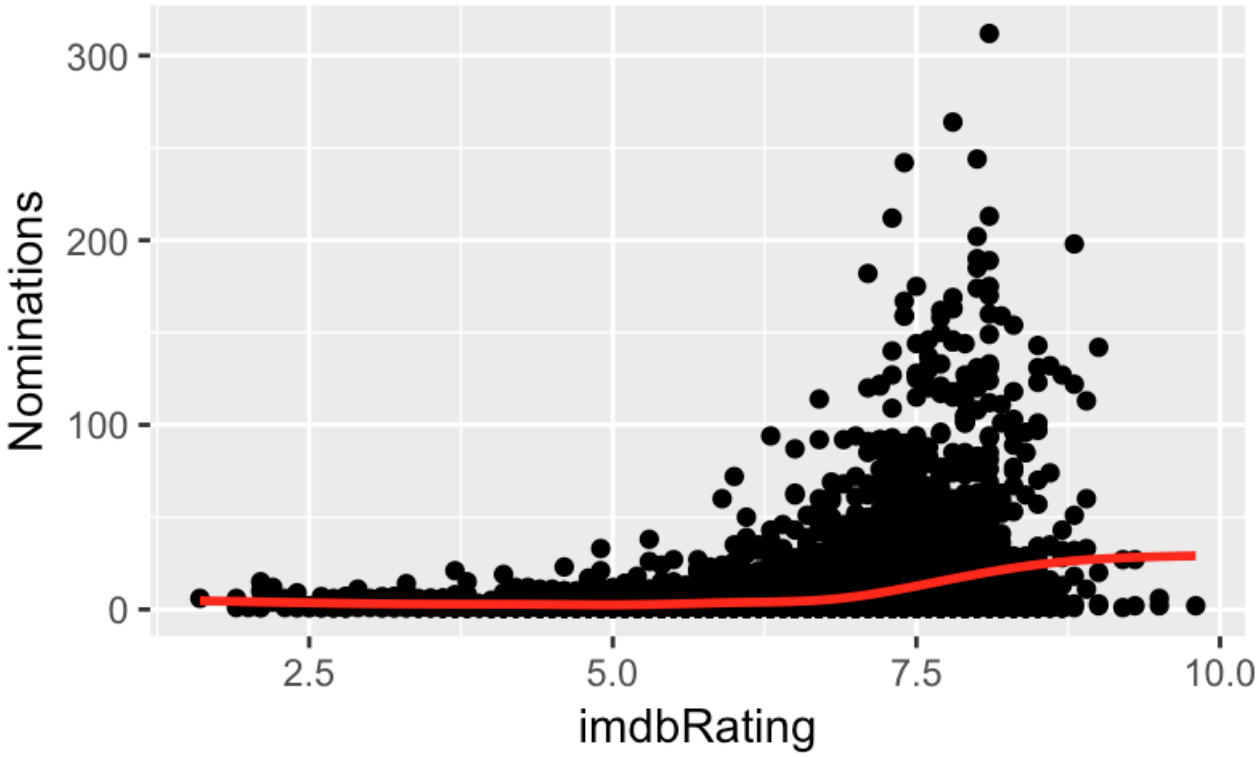
Hide

```
# TODO: Show how ratings and awards are related
library(ggplot2)
# Cleaning the data for plot.
df_imdbRatingNoNA = df[!is.na(df$imdbRating),]
df_imdbRating_Wins_NoNA = df_imdbRatingNoNA[!is.na(df_imdbRatingNoNA$Wins),]
df_imdbRating_Nomn_NoNA = df_imdbRatingNoNA[!is.na(df_imdbRatingNoNA$Nomination
s),]
# Plot imdbRating against Wins and Nominations.
qplot(x = imdbRating, y = Wins, data = df_imdbRating_Wins_NoNA) + stat_smooth(c
olor = "red", size = I(1), se = F)
```



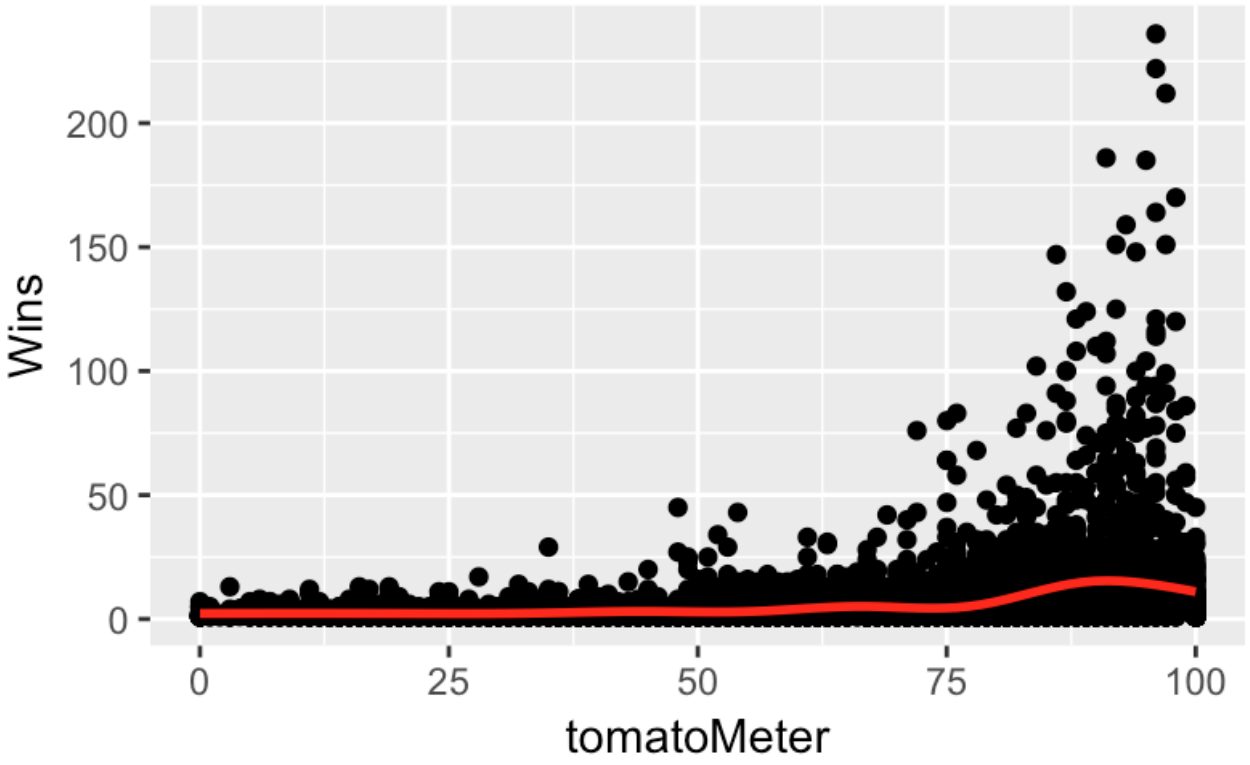
Hide

```
qplot(x = imdbRating, y = Nominations, data = df_imdbRating_Nomn_NoNA) + stat_s
mooth(color = "red", size = I(1), se = F)
```



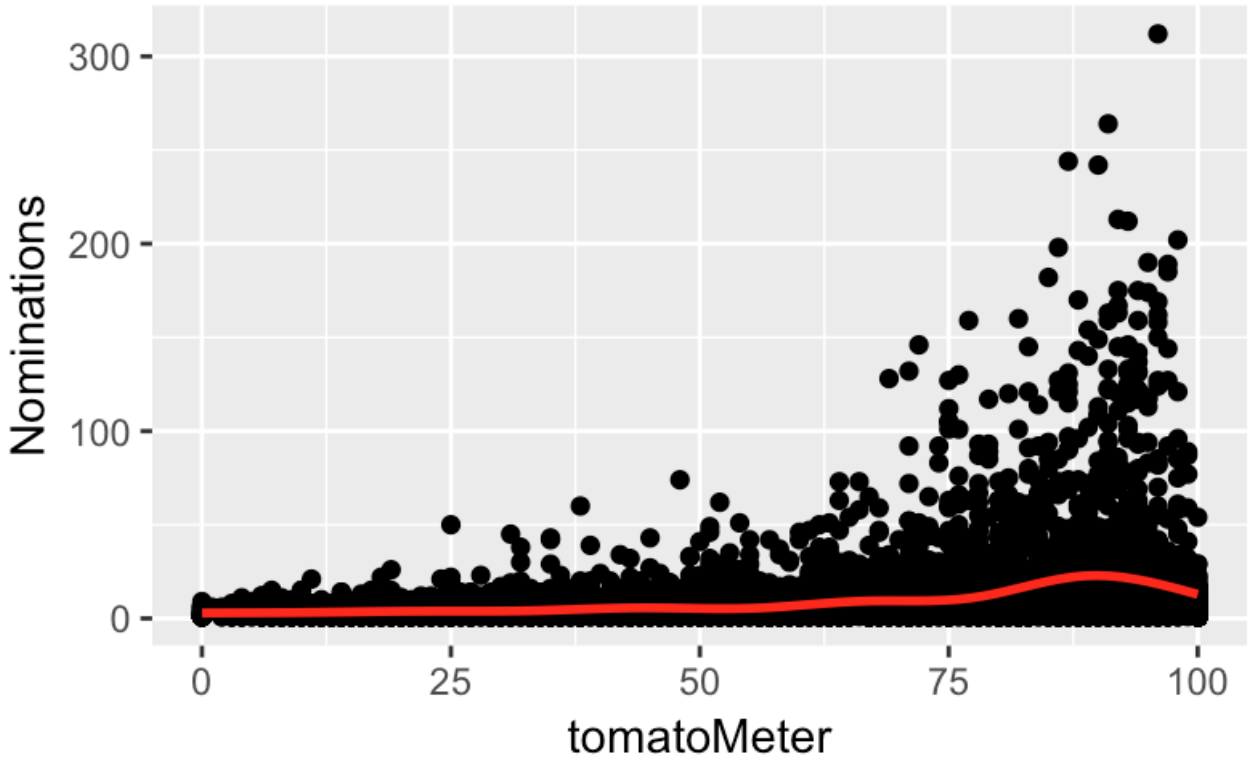
Hide

```
# Cleaning the data for plot.
df_tomatoMeterNoNA = df[!is.na(df$tomatoMeter),]
df_tomatoMeter_Wins_NoNA = df_tomatoMeterNoNA[!is.na(df_tomatoMeterNoNA$Wins),]
df_tomatoMeter_Nomn_NoNA = df_tomatoMeterNoNA[!is.na(df_tomatoMeterNoNA$Nominat
ions),]
# Plot tomatoMeter against Wins and Nominations.
qplot(x = tomatoMeter, y = Wins, data = df_tomatoMeter_Wins_NoNA) + stat_smooth
(color = "red", size = I(1), se = F)
```



Hide

```
qplot(x = tomatoMeter, y = Nominations, data = df_tomatoMeter_Nomn_NoNA) + stat
_smooth(color = "red", size = I(1), se = F)
```



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: By plotting Wins/Nominations against imdbRating, I found the movies winning most awards have imdbRating 7.0- 9.0, so as for nominations. The general relation between imdbRating and number of awards are expected: movies with few awards have low imdbRatings. But for some movies whole imdbRatings are higher than 9.0, their awards number is not very high. There are two possible reasons in my opinion. One is that some movies are very appealing for audience but they are only good at that appealing part, rather than all the aspects of a movie, so they only win a few awards. The second reason may be related with the number of votes for imdbRating. Similar as the example in our reading material (Chapter 11. Data processing, Page 356), a minor movie may be given votes by a small circle of fans about this type movie. So the total number of votes only limits in these fans, and ratings with a few votes are easier to reach very high values, but not necessarily means they are good at all aspects of a movie.

Then, I studied the relation between Wins/Nominations and a critics rating TomatoMeter. Since tomatoMeter is calculated by professional critics maybe overlapping with the awards committee members, the correlation is much better than Wins/Nomination vs. imdbRatings. Movies won many awards show very high tomatoMeter values. However, we should notice that many movies have high tomatoMeter scores but still won few awards.

As a conclusion, for awards vs. imdbRatings, there is a positive correlation, but it is not very consistent. From the imdbRating, we can say, if this rating of a movie is higher than 7.0, the movie has better chance to win many awards, but not guaranteed. If the imdbRating is quite low for a movie, it is few chances to win many awards. The tomatoMeter is better in predicting awards numbers, but still

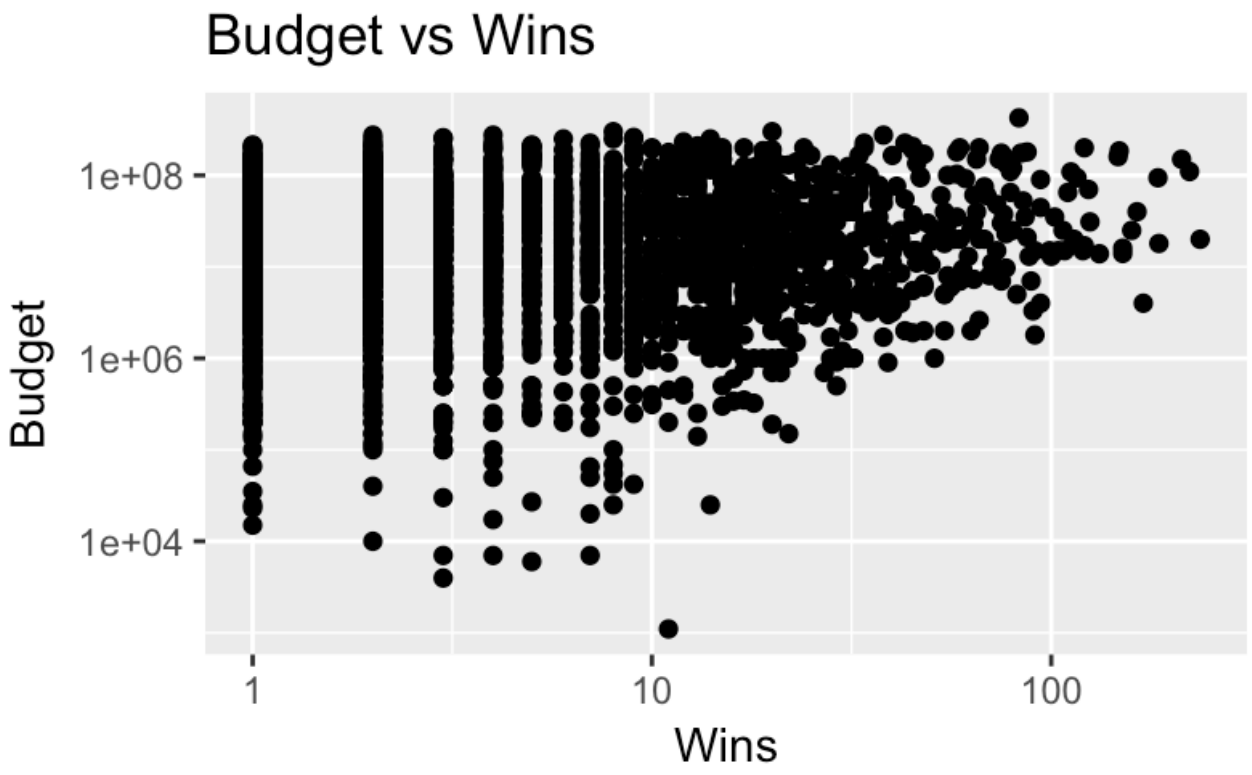
not guaranteed.

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.

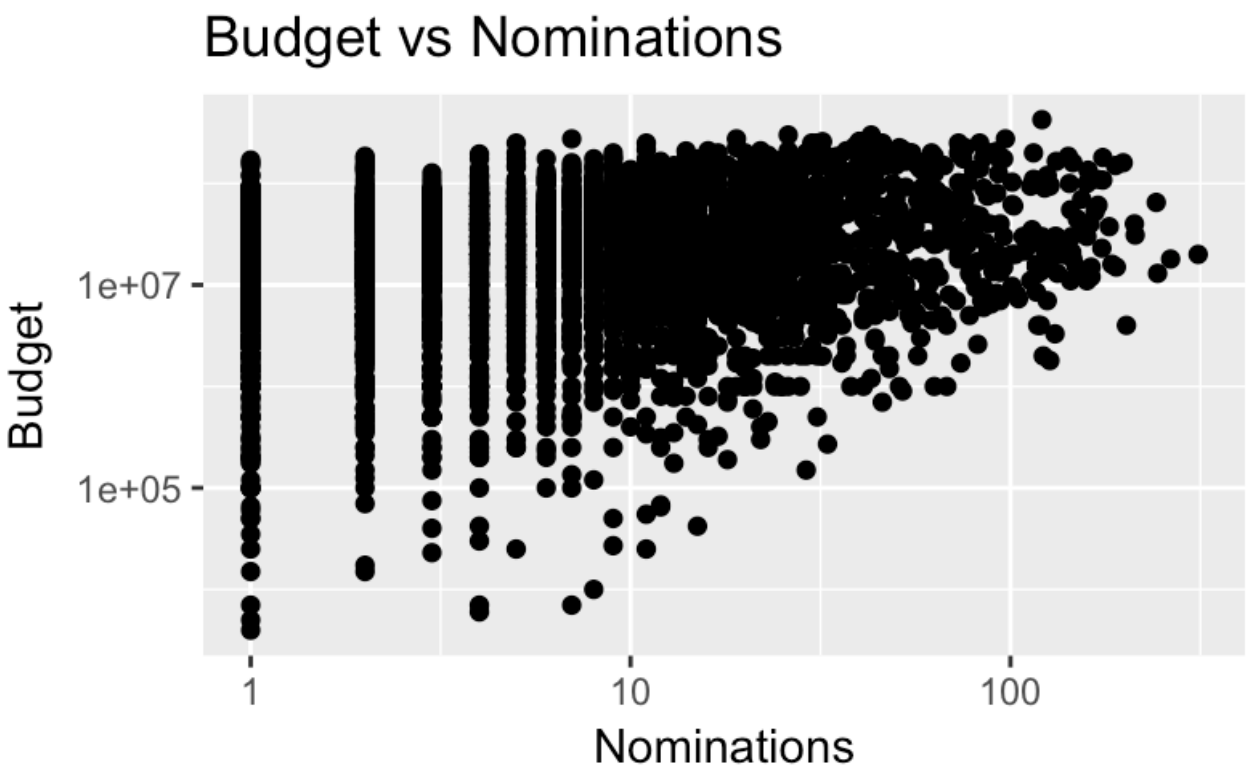
Hide

```
# TODO: Find and illustrate two expected insights
# Expected insights #1 Relationship between budget and awards
library(ggplot2)
df_BudgetNoNA = df[!is.na(df$Budget),]
df_BudgetWinNoNA = df_BudgetNoNA[!is.na(df_BudgetNoNA$Wins),]
df_BudgetNomnNoNA = df_BudgetNoNA[!is.na(df_BudgetNoNA$Nominations),]
qplot(x=Wins, y= Budget, log="xy", data = df_BudgetWinNoNA, main = "Budget vs W
ins")
```



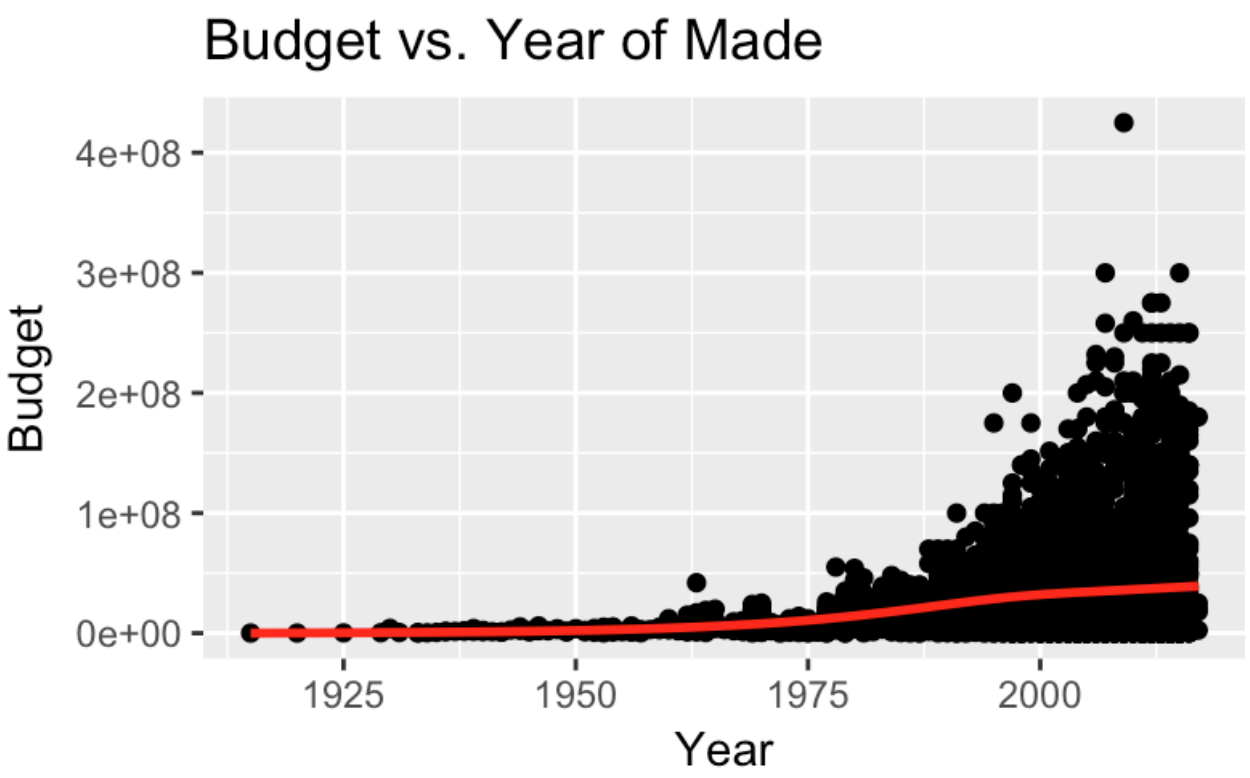
Hide

```
qplot(x=Nominations, y= Budget, log="xy", data = df_BudgetNomnNoNA, main = "Bud
get vs Nominations")
```



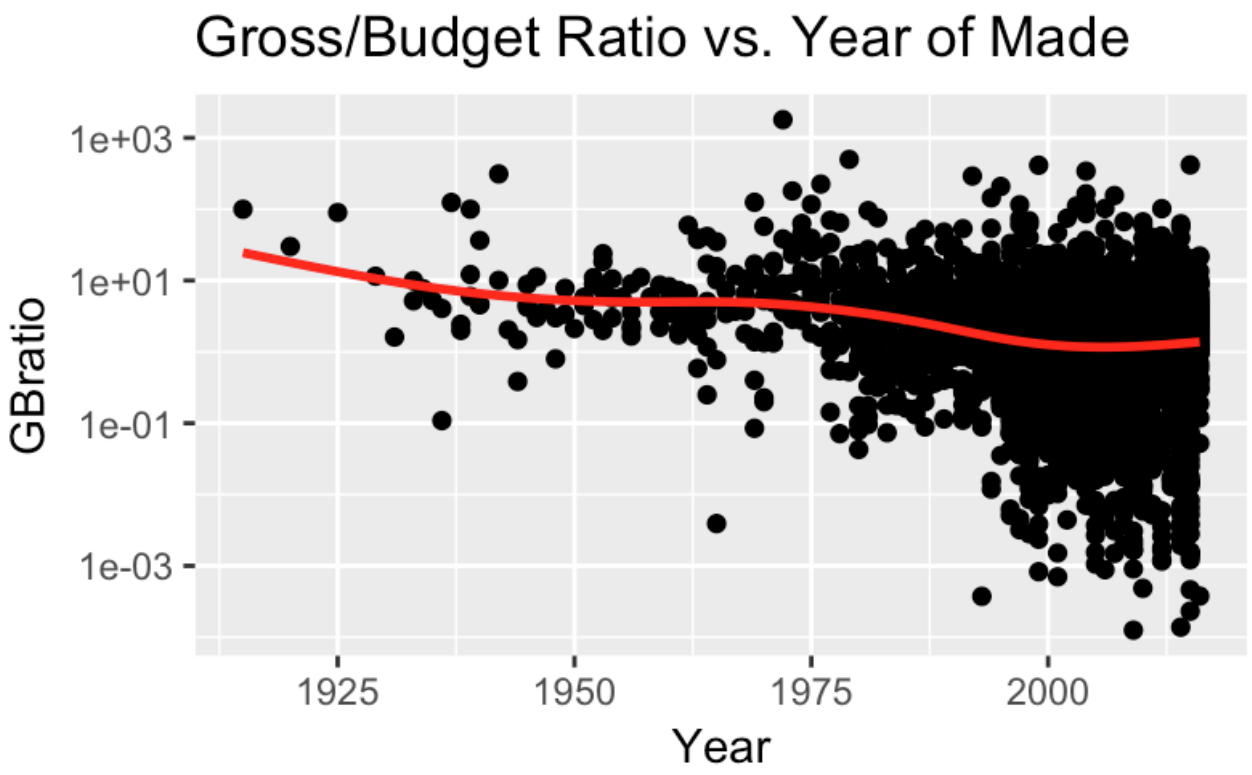
Hide

```
# Expected insights #2 Relationship of production year and budget, Gross/Budget
ratio vs. Year.
qplot(x=Year, y= Budget, data = df_BudgetNoNA, main = "Budget vs. Year of Made"
) + stat_smooth(color = "red", size = I(1), se = F)
```



Hide

```
df_grossNoNA$GBratio = df_grossNoNA$Gross/df_grossNoNA$Budget
df_grossClean = df_grossNoNA[df_grossNoNA$Gross!= 0, ]
qplot(x=Year, y= GBratio, log= "y", data = df_grossClean, main = "Gross/Budget
Ratio vs. Year of Made") + stat_smooth(color = "red", size = I(1), se = F)
```



Q: Expected insight #1.

A: I studied the relation between Budget and number of awards (Wins/Nominations). Before plotting them, I want to know whether a high budget for a movie will ensure the movie win more awards. From the Budget vs. Wins graph in log-scale, I get a upper-left triangle distribution, which is similar as the one of Gross vs. Wins. The Budget-Wins graph indicates that movies won many awards all have decent amount of budget, but a movie with high budget not necessarily won many awards. Actually, the chance of a high budget movie won a few awards is still higher than the high budget movie to win many awards. This is an expected insight. Good movies which can win awards need to have enough budget input to ensure its quality. However, a lot of money input won't guarantee a good movie, because movies are not only a result of money but many other things else, like a good story and artistic quality.

Q: Expected insight #2.

A: I studied the profitability of movies against the year movie was made. Firstly, I plot the budget and year of movies to show that the average budget of movies are increasing as year increases. Then, I plotte the Gross/Budget ratio of movies with the year of made. Opposite to the trend of budget increasing with year increasing, the average Gross/Budget ratio decreases as year increases. From the graph, we can see more and more movies were made in each year after 1975, so the decreasing Gross/Budget ratio may be a result of greater competition. From 1990s to 2010s, there are many movies still have high Gross/Budget ratio (bigger than 10), and the absolute movies numbers with high Gross/Budget ratio are bigger than years before. However, since too many movies have been made each year after 1990s, audience have enough choices of good movie, so those movies are not that appealing to audience won't attract large number of audience to make high gross revenue compared

to its input budget.

10. Unexpected insight

Come up with one new insight (backed up by data and graphs) that is unexpected at first glance and do your best to motivate it. Same instructions apply as the previous task.

Hide

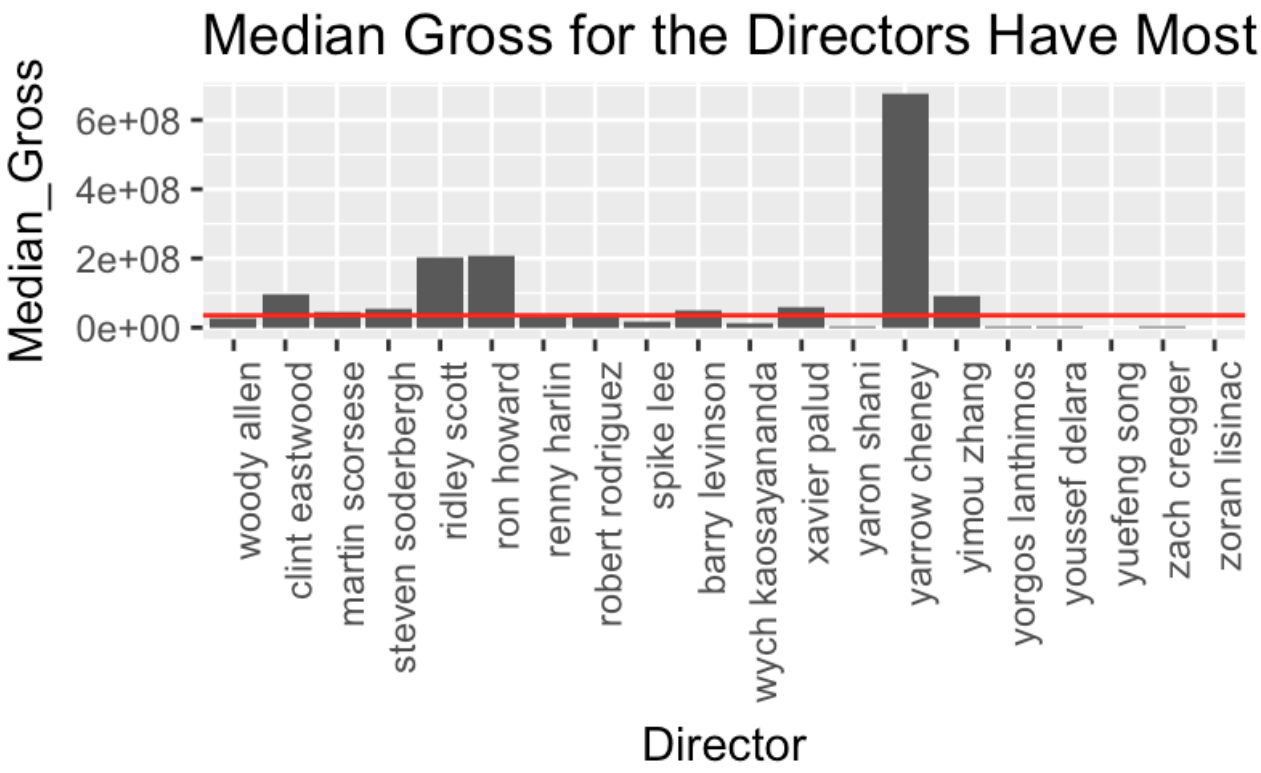
```
# TODO: Find and illustrate one unexpected insight
# Investigate on how number of movies one director directed related with the gr
oss revenue.
library(tm)
# Remove rows with NA and zeros in 'Gross' column
df_grossNoNA = df[!is.na(df$Gross),]
df_grossClean = df_grossNoNA[df_grossNoNA$Gross!= 0, ]
# Prepare the 'Director column' to a format easier for freqency analysis.
director_noComma1 = vector("character", length = nrow(df_grossClean))
director_noComma2 = vector("character", length = nrow(df_grossClean))
# Remove commas, combine firstname and lastname of every direct as one word.
for (i in seq(1, nrow(df_grossClean), by = 1)) {
  director_noComma1[i] = gsub(", ", ",", df_grossClean$Director[i])
  text = director_noComma1[i]
  director_noComma1[i] = gsub(" ", "_", text)
}
director_noComma2 = sapply(director_noComma1, function(x) strsplit(x, split = "
,", fixed = T))
# Use the text analysis functions in 'tm' package to obtain the name frequency
(number of movies) of each director.
myCorpusDirector = Corpus(VectorSource(director_noComma2))
myDTM_Director = DocumentTermMatrix(myCorpusDirector, control = list(minWordLen
gth = 1))
freq = sort(colSums(as.matrix(myDTM_Director)), decreasing = T)
director_nameFreq = data.frame(name = names(freq), MoviesNum = freq)
# Exclude NAs.
director_nameFreq = director_nameFreq[2:nrow(director_nameFreq),]
# Get the 10 directors with most movies with valid gross value in the dataframe
directorTop10Freq = as.character(director_nameFreq$name[1:10])
# Get the 10 directors with least movies with valid gross value in the datafram
e
last10th = nrow(director_nameFreq) - 9
directorLeast10Freq = as.character(director_nameFreq$name[last10th:nrow(directo
r_nameFreq)])
# Combine the top 10 and last 10 productive directors
directorFreq = c(directorTop10Freq, directorLeast10Freq)
grossNoNA_names = c(names(df_grossClean), directorFreq)
columns = length(df_grossClean)
for (i in seq(1, 20, by=1)) {
  df_grossClean[,i+columns] <- NA
}
# Add these 20 director names to dataframe df_grossClean as binary columns
names(df_grossClean) = grossNoNA_names
# Back format the directors' names
directorFreq_name = sapply(directorFreq, function(x) gsub("_", " ", x))
# Assign 1 or 0 to the director name binary columns according to wether the mov
```

```
ie is directed by that director
for (j in seq(1, 20, by=1)) {
  for (i in seq(1, nrow(df_grossClean), by=1)) {
    if (grepl(directorFreq_name[j], df_grossClean$Director[i], ignore.case = TR
UE)) {
      df_grossClean[i,j+columns] = 1
    }
    else {
      df_grossClean[i,j+columns] = 0
    }
  }
}
# Create a dataframe to summarize productive director names, Median_Gross and n
umber of movies each director directed for the top 10 and last 10 productive di
rectors
library(plyr)
director_gross = data.frame(Director = directorFreq_name, Median_Gross = NA, Mo
vieNum = NA)
for (i in seq(1, 20, by = 1)) {
  director_gross_median = ddply(df_grossClean, names(df_grossClean)[columns + i
], summarise, median.Gross = median(Gross, na.rm = TRUE), sum.Gross = sum(Gross
, na.rm = TRUE))
  director_gross[i, 2] = director_gross_median[2,2]
  director_gross$MovieNum[i] = sum(df_grossClean[columns + i])
}
director_gross$Director <- factor(director_gross$Director, levels = director_gr
oss$Director[order(director_gross$MovieNum, decreasing = T)])
library(ggplot2)
Gross_median_allmovies = median(df_grossClean$Gross)
Gross_median_allmovies
```

```
[1] 35319689
```

Hide

```
ggplot(director_gross, aes(x = Director, y = Median_Gross)) + geom_bar(stat = "
identity") + theme(axis.text.x = element_text(angle = 90, hjust = 1)) + ggtitle
("Median Gross for the Directors Have Most and Least Number of Movies") + geom_
hline(yintercept = Gross_median_allmovies, colour = "Red")
```



Q: Unexpected insight.

A: I studied the Gross revenue of the 10 directors with highest number of movies with valid gross values, and comparing with the gross of 10 directors only have one movie with valid gross value in our dataframe. The order of directors' name in the bar graph (from left to right) is the order of number of movies the director have in the dataframe. Woody Allen to Barry Levinson are the directors with most movies in df, and from Wych Kaosayananda to Zoran Lisinac are directors with one movie in the df. I think the result is unexpected, because I thought the 10 directors with highest number of movies should be famous directors and they should be able to direct good movies. So commonly, we should think these directors can produce movies with high gross revenue as well. If they cannot produce with movies with high gross revenue, film companies and investors won't invite them to direct so many movies. However, my analysis results are different from what I thought. I calculated the median of the gross revenue of the all the movies in our dataframe, which is 35319689, the red horizontal line in the graph. The 2 out of 10 most productive director have median gross of their movies less than 35319689. One is Woody Allen, whose gross median is 25029110 (almost one-third less than the median gross of all movies) and has 23 movies with valid gross revenue in our dataframe; the other one is Spike Lee, who has 15 movies in our dataframe, median gross is 16153000 (less than a half of the median of all movies). Moreover, Renny Harlin and Robert Rodriguez have median gross at 38164784 and 40283321, respectively, just barely higher than the median gross of all movies. Martin Scorsese, Steven Soderbergh and Barry Levinson their gross medians are only a little higher than the red line. This result is surprising to me, but after carefully thinking I can understand it. One reason for that is the type of movies one director is good at. Even for a famous director, if he only direct on small budget movies, he may not have all movies with very high gross revenue. Secondly, the data we have is incomplete, many movies' gross and budget information is missing, so we cannot completely trust

the results.

Interestingly, but not surprisingly, the directors who direct less movies may have movies with pretty good gross revenue. Yarrow Cheney only has one movie with valid gross value in our dataframe which is “The Secret Life of Pets” and this movie has very high gross revene 676075380. Zhang yimou’s House of Flying Daggers has gross revenue 92863945 which is also a good gross revenue. From my knowledge, I know Yarrow Cheney and Zhang yimou have many other popular movies but they are not included in this dataframe with a gross value.

---

1. 0-9↩
2. 0-9↩
3. 0-9↩