Project 1: Explore and Prepare Data

CSE6242 - Data and Visual Analytics - Summer 2018

Due: Sunday, June 17, 2018 at 11:59 PM UTC-12:00 on T-Square

hyang390, 903320189

Note: This project involves getting data ready for analysis and doing some preliminary investigations. Project 2 will involve modeling and predictions on the same dataset, and will be released at a later date. Both projects will have equal weightage towards your grade. You may reuse some of the preprocessing/analysis steps from Project 1 in Project 2.

Data

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

> The file `movies_merged` contains a dataframe with the same name that has 40K rows and 39 columns. Each row represents a movie title and each column represents a descriptor such as `Title`, `Actors`, and `Budget`. I collected the data by querying IMDB's API (see www.omdbapi.com) and joining it with a separate dataset of movie budgets and gross earnings (unknown to you). The join key was the movie title. This data is available for personal use, but IMDB's terms of service do not allow it to be used for commercial purposes or for creating a competing repository.

Objective

Your goal is to investigate the relationship between the movie descriptors and the box office success of movies, as represented by the variable `Gross`. This task is extremely important as it can help a studio decide which titles to fund for production, how much to bid on produced movies, when to release a title, how much to invest in marketing and PR, etc. This information is most useful before a title is released, but it is still very valuable after the movie is already released to the public (for example it can affect additional marketing spend or how much a studio should negotiate with on-demand streaming companies for a second window's streaming rights).

Instructions

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

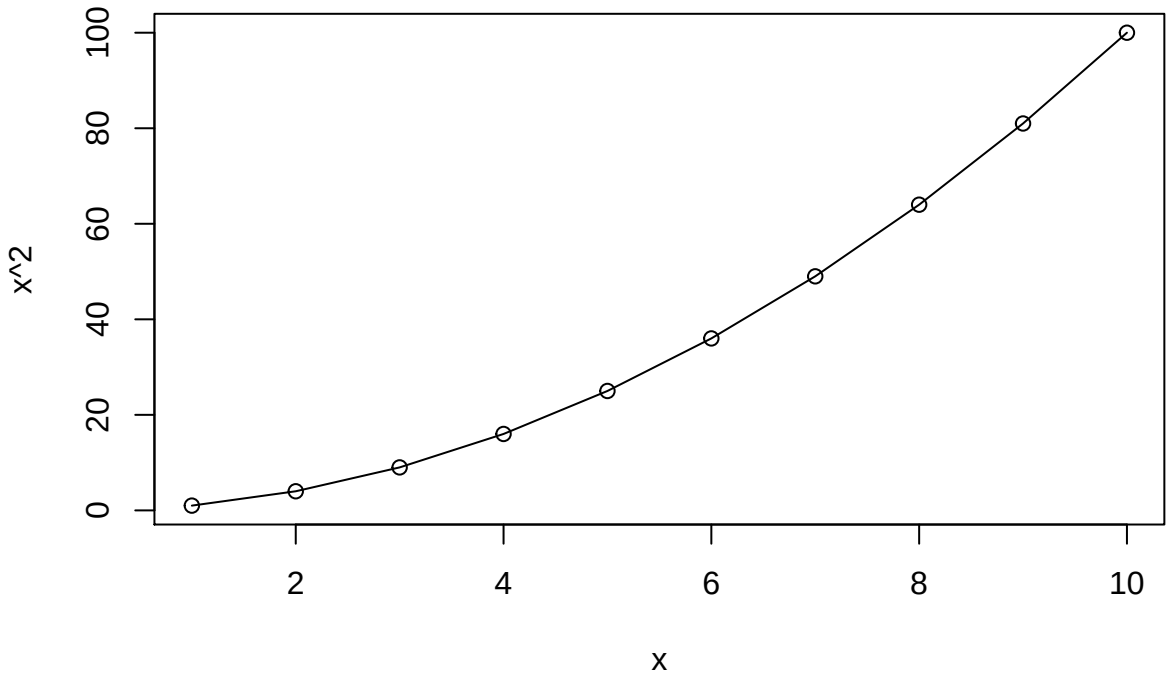
When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

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

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

Plots appear inline too:

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



Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*. Enter some R code and run it.

```r
test = seq(1, 10, 2)
print(test)
```

```
## [1] 1 3 5 7 9
```

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

Please complete all the tasks below by implementing code chunks that have a `TODO` comment in them, running all code chunks so that output and plots are displayed, and typing in answers to each question (**Q:** ...) next to/below the corresponding answer prompt (**A:**). Feel free to add code chunks/show additional output to support any of the answers.

When you are done, you will need to submit the final R markdown file (as **pr1.Rmd**) with all code chunks implemented and executed, and all text responses written in. You also need to submit a PDF export of the markdown file (as **pr1.pdf**), which should show your code, output, plots and written responses–this will be your project report. Compress these two files into a single .zip archive and upload it on T-Square.

Setup

Load data

Make sure you've downloaded the `movies_merged` file and it is in the current working directory. Now load it into memory:

```
load('movies_merged')
cat("Dataset has", dim(movies_merged)[1], "rows and", dim(movies_merged)[2], "columns", end="\n", file=
```

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

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

```
df = movies_merged
cat("Column names:", end="\n", file="")
```

```
## Column names:
```

```
colnames(df)
```

```
##  [1] "Title"             "Year"              "Rated"
##  [4] "Released"          "Runtime"           "Genre"
##  [7] "Director"          "Writer"            "Actors"
## [10] "Plot"              "Language"          "Country"
## [13] "Awards"            "Poster"            "Metascore"
## [16] "imdbRating"        "imdbVotes"         "imdbID"
## [19] "Type"              "tomatoMeter"       "tomatoImage"
## [22] "tomatoRating"      "tomatoReviews"     "tomatoFresh"
## [25] "tomatoRotten"      "tomatoConsensus"   "tomatoUserMeter"
## [28] "tomatoUserRating"  "tomatoUserReviews" "tomatoURL"
## [31] "DVD"               "BoxOffice"         "Production"
## [34] "Website"           "Response"          "Budget"
## [37] "Domestic_Gross"    "Gross"             "Date"
```

Load R packages

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

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

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

Non-standard packages used: None

Tasks

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

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

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

1. Remove non-movie rows

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

```
# TODO: Remove all rows from df that do not correspond to movies
df2 <- df[df$Type == "movie",]
dim(df2)
```

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

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

A: There are now 40,000 rows left after the removal of rows that do not correspond to movies.

2. Process `Runtime` column

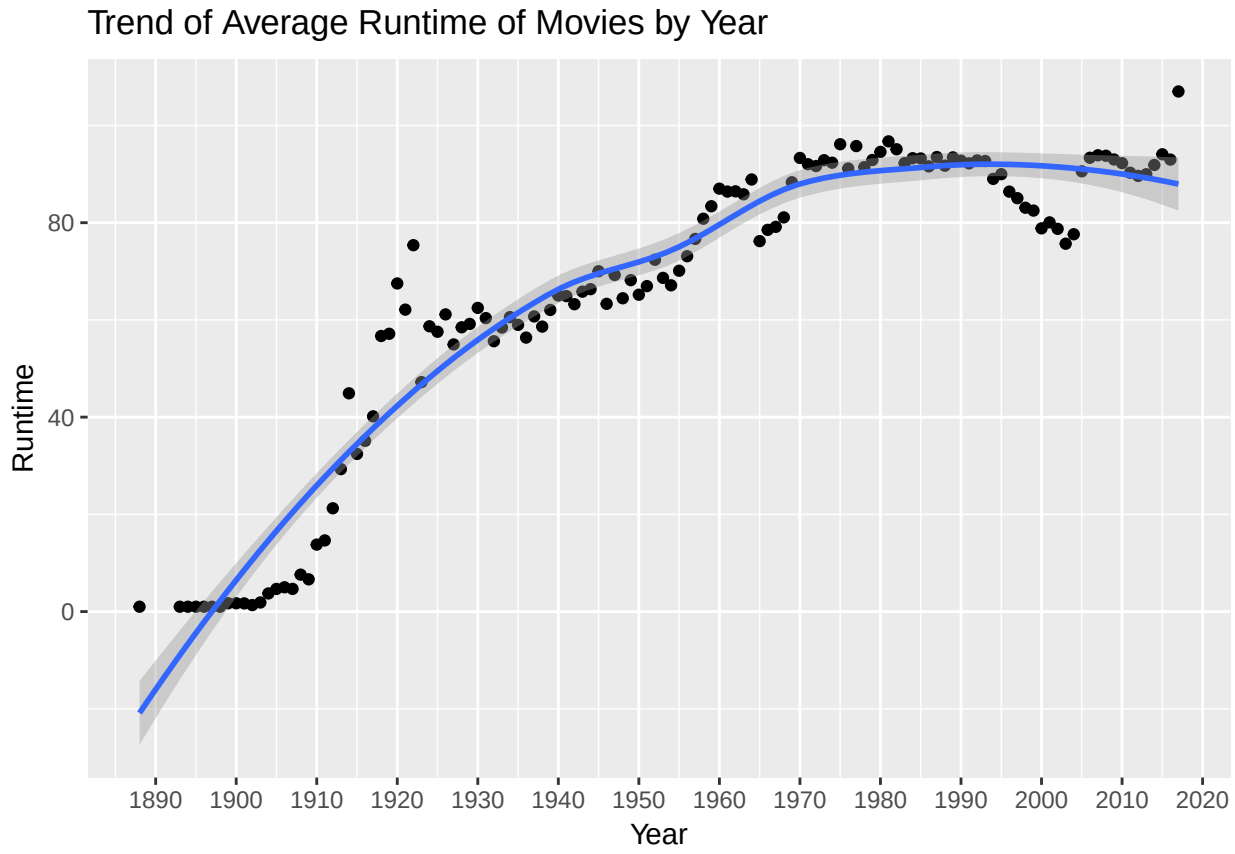
The variable `Runtime` represents the length of the title as a string. Write R code to convert it to a numeric value (in minutes) and replace `df$Runtime` with the new numeric column.

```
# TODO: Replace df$Runtime with a numeric column containing the runtime in minutes
convertRuntime = function(timestr) {
  split_strtime = strsplit(timestr, "\\s+")[[1]]
  if (length(split_strtime) == 2) {
    if (split_strtime[2] == "min") {
      time = as.numeric(split_strtime[1])
    }
    else if (split_strtime[2] == "h") {
      time = as.numeric(split_strtime[1])*60
    }
  }
  else if (length(split_strtime) == 4) {
    time = as.numeric(split_strtime[1])*60 + as.numeric(split_strtime[3])
  }
  else {
    time = suppressWarnings(as.numeric(split_strtime[1]))
  }
  return (time)
}

df2$Runtime = sapply(df2$Runtime, convertRuntime)
```

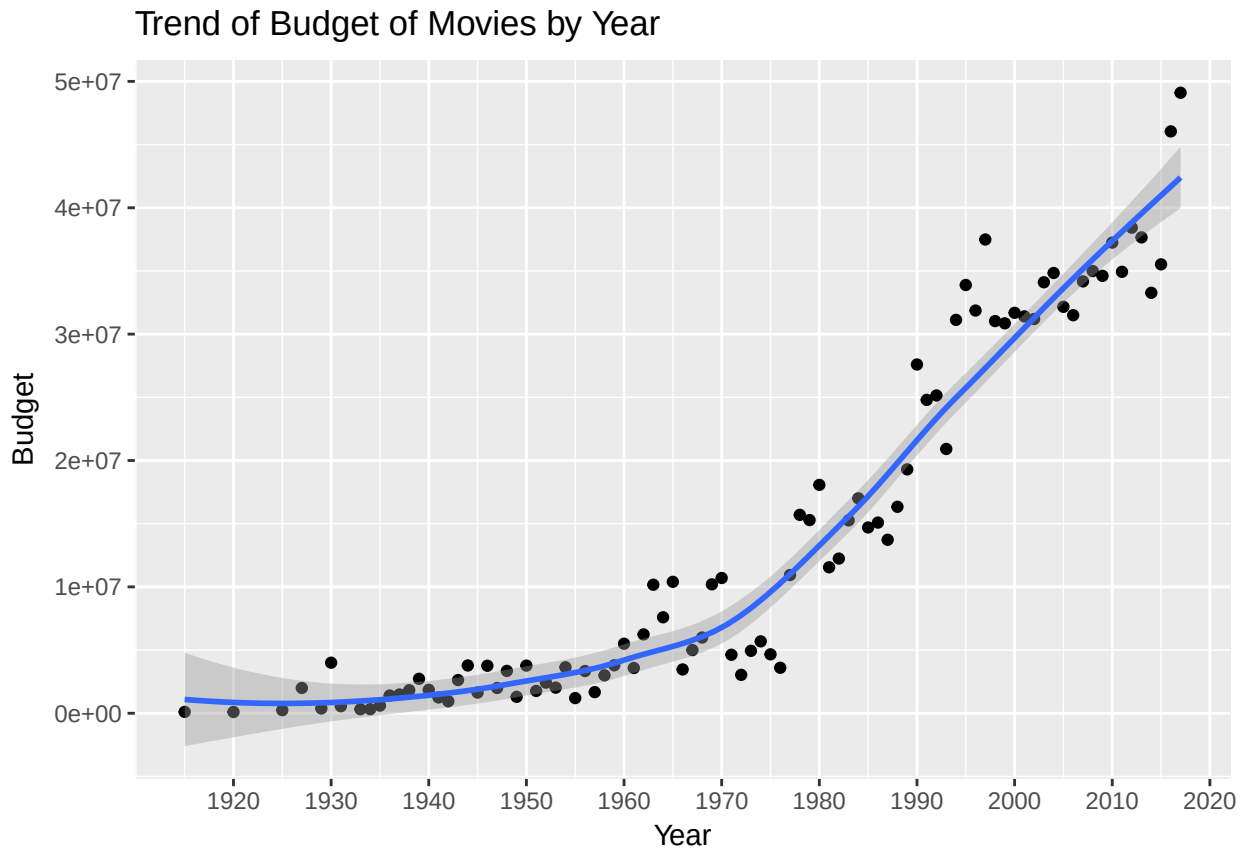
Now investigate the distribution of `Runtime` values and how it changes over years (variable `Year`, which you can bucket into decades) and in relation to the budget (variable `Budget`). Include any plots that illustrate.

```
# TODO: Investigate the distribution of Runtime values and how it varies by Year and Budget
avg_runtime_per_year = aggregate(Runtime~Year, df2, mean)
avg_budget_per_year = aggregate(Budget~Year, df2, mean)
avg_budget_per_runtime = aggregate(Budget~Runtime, df2, mean)
```

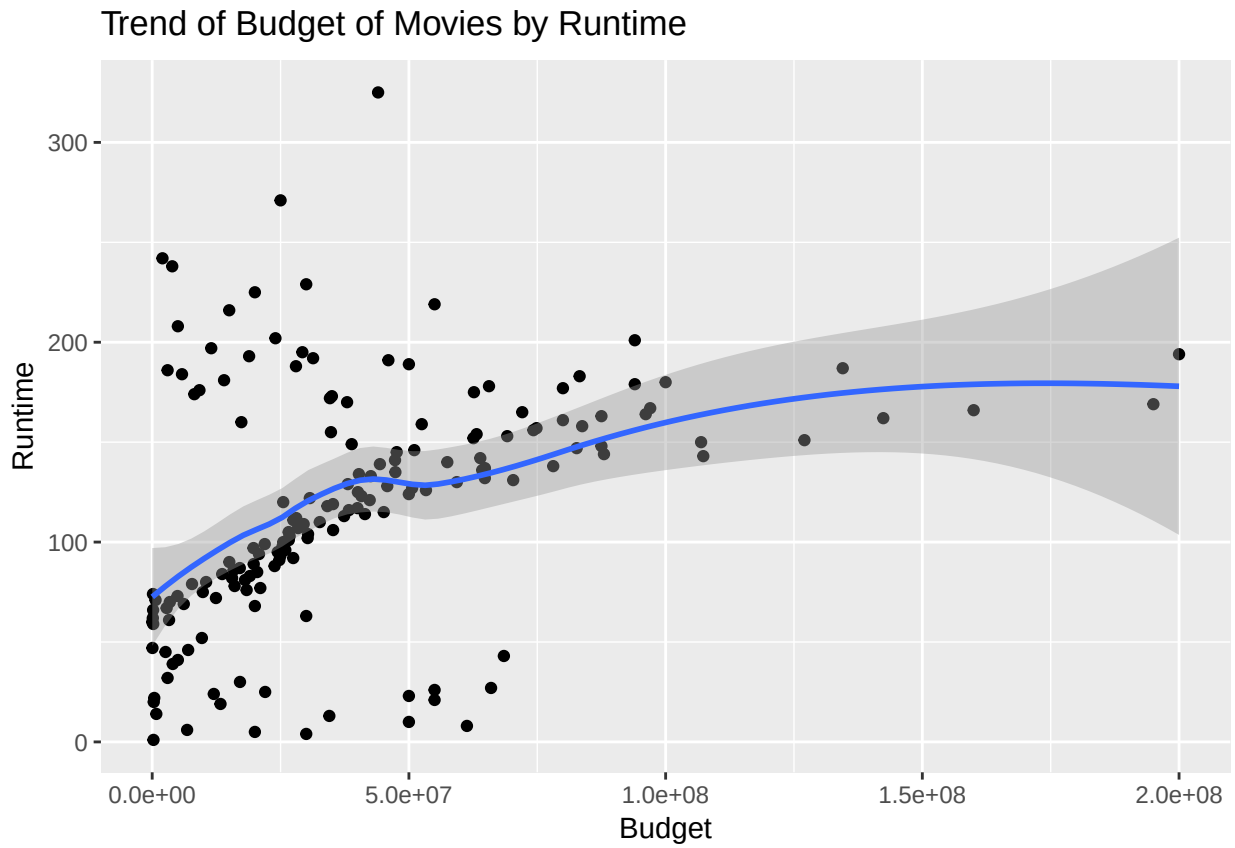
```
ggplot(avg_runtime_per_year, aes(x=Year, y=Runtime)) + geom_point() + geom_smooth(method = "loess") +
  scale_x_continuous(breaks = round(seq(min((round(avg_runtime_per_year$Year/10) * 10)), max(round(avg_
  ggtitle("Trend of Average Runtime of Movies by Year")
```




```
ggplot(avg_budget_per_year, aes(x=Year, y=Budget)) + geom_point() + geom_smooth(method = "loess") +
  scale_x_continuous(breaks = round(seq(min((round(avg_budget_per_year$Year/10) * 10)), max(round(avg_bu
  ggtitle("Trend of Budget of Movies by Year")
```

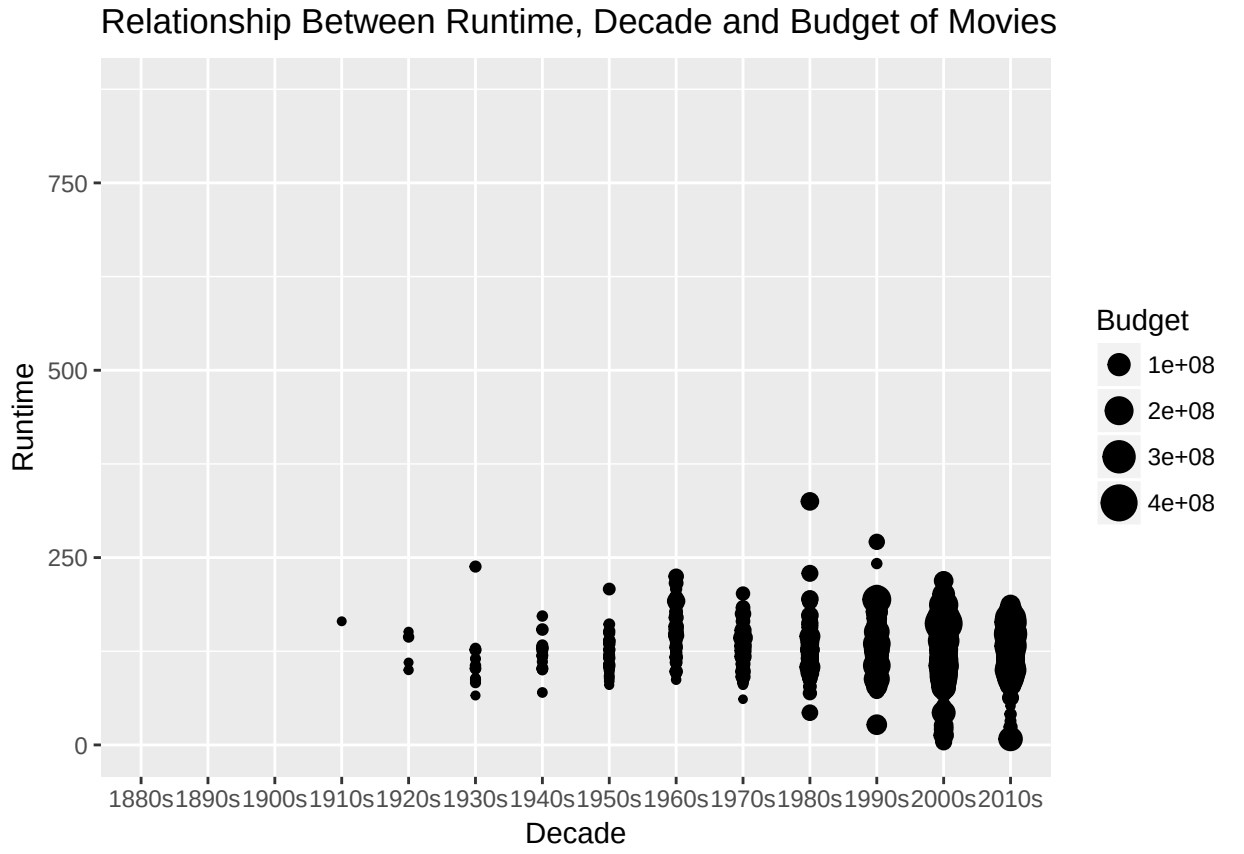



```
ggplot(avg_budget_per_runtime, aes(x=Budget, y=Runtime)) + geom_point() + geom_smooth(method = "loess")
  ggtitle("Trend of Budget of Movies by Runtime")
```



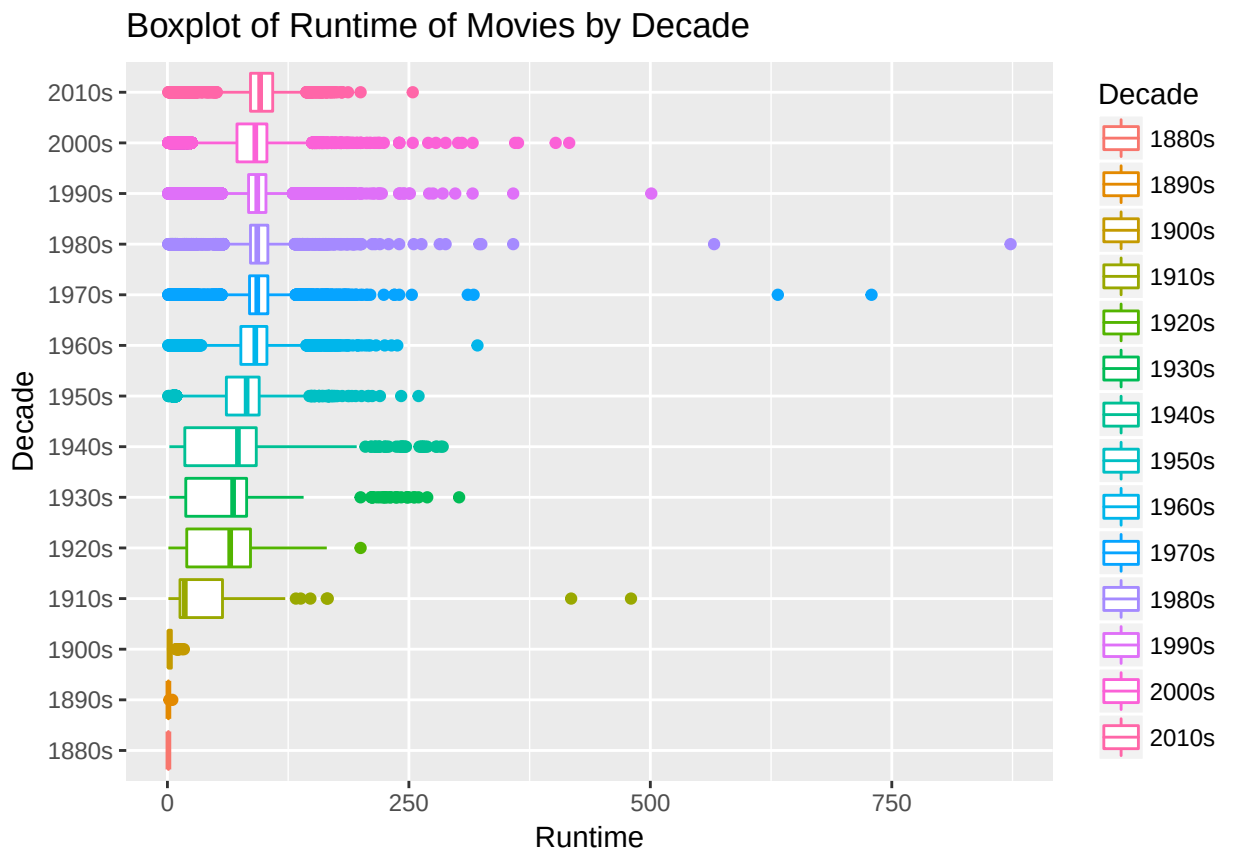

```
bucketYears = function(year){
  if (year >= 1880 & year < 1890) return ("1880s")
  if (year >= 1890 & year < 1900) return ("1890s")
  if (year >= 1900 & year < 1910) return ("1900s")
  if (year >= 1910 & year < 1920) return ("1910s")
  if (year >= 1920 & year < 1930) return ("1920s")
  if (year >= 1930 & year < 1940) return ("1930s")
  if (year >= 1940 & year < 1950) return ("1940s")
  if (year >= 1950 & year < 1960) return ("1950s")
  if (year >= 1960 & year < 1970) return ("1960s")
  if (year >= 1970 & year < 1980) return ("1970s")
  if (year >= 1980 & year < 1990) return ("1980s")
  if (year >= 1990 & year < 2000) return ("1990s")
  if (year >= 2000 & year < 2010) return ("2000s")
  if (year >= 2010 & year < 2020) return ("2010s")
}

df2$Decade = sapply(df2$Year, bucketYears)

qplot(x=Decade, y=Runtime, data=df2, size=Budget) +
  ggtitle("Relationship Between Runtime, Decade and Budget of Movies")
```

```
## Warning: Removed 35480 rows containing missing values (geom_point).
```



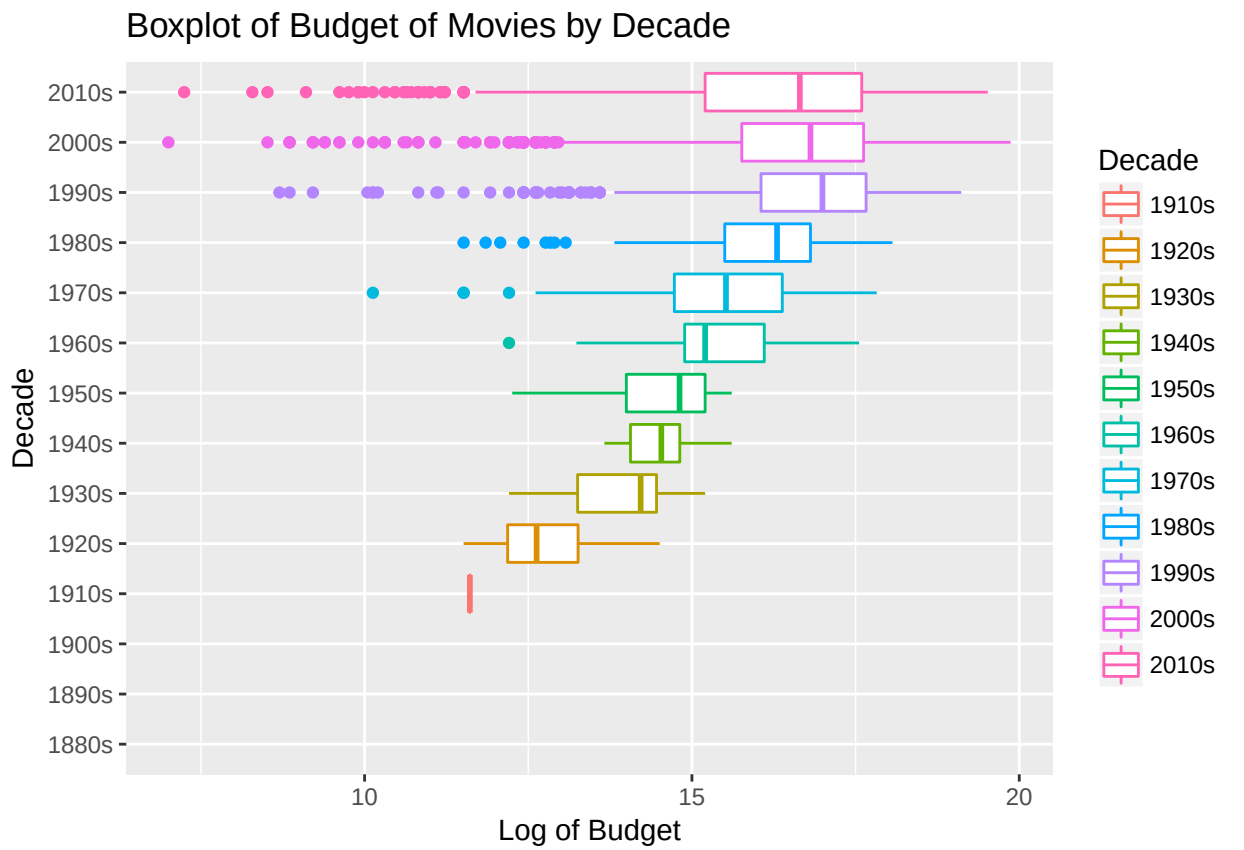

```
ggplot(df2, aes(reorder(Decade, Runtime, mean), Runtime, color=Decade)) +
  geom_boxplot() + coord_flip() +
  xlab("Decade") +
  ggtitle("Boxplot of Runtime of Movies by Decade")
```

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




```
ggplot(df2, aes(reorder(Decade, Budget, mean), log(Budget), color=Decade)) +
  geom_boxplot() + coord_flip() +
  ylab("Log of Budget") +
  xlab("Decade") +
  ggtitle("Boxplot of Budget of Movies by Decade")
```

```
## Warning: Removed 35442 rows containing non-finite values (stat_boxplot).
```



Feel free to insert additional code chunks as necessary.

Q: Comment on the distribution as well as relationships. Are there any patterns or trends that you can observe?

A: In the first chart titled "Trend of Average Runtime of Movies by Year", we can see that there is a clear relationship between Year and Runtime. As years increased, the runtime of movies also increased. This relationship continued about the 1980s when it began to stabilize around 90 minutes. We can also see two major groups in the chart, movies made before the 1930s seem to average out around 20-30 minutes. Movies that were made after seem to average around 75 minutes or so.

Analyzing Year and Budget, we can also see that as the years progressed, there has been a increase in the budget. From the early 1900s to about 1960s, this trend increase is small, as marked in the chart "Trend of Budget of Movies by Year". However, from the 1970s onwards, we have almost seen a linear increase in budget as the years also increased.

Looking at Budget and Runtime, we can see that there is indeed a positive correlation between these two variables, the trend line plotted shows there is a slight linear relationship that as budget increases, runtime tends to increase as well.

The above points are further validated by looking at the boxplot of the distributions of Runtime and Budget by decades. Looking at runtime, there is a strong trend showing runtimes have increased as years went by, as indicated by the medians in the boxplot. A similiar trend is observed for Budget, although the median seem to be the maximum during the 1990s.

3. Encode `Genre` column

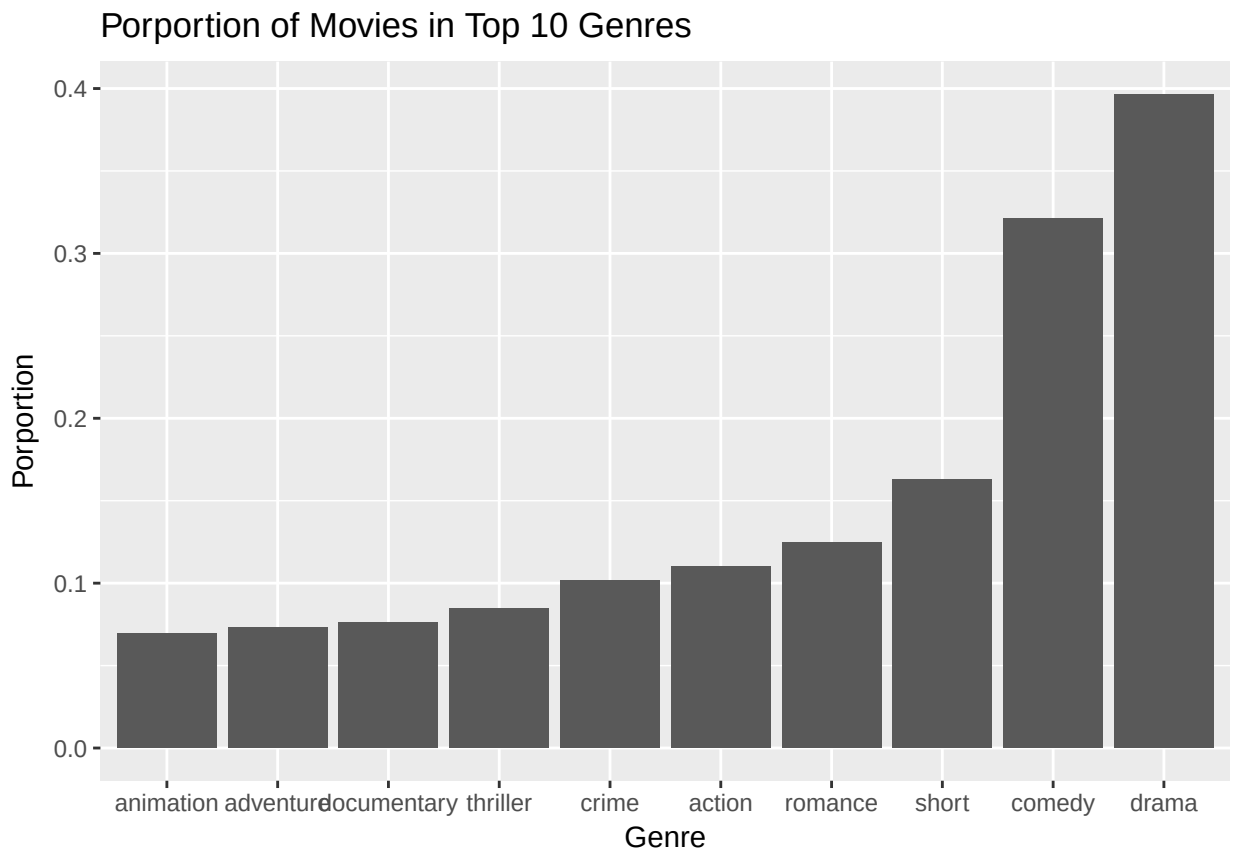
The column `Genre` represents a list of genres associated with the movie in a string format. Write code to parse each text string into a binary vector with 1s representing the presence of a genre and 0s the absence, and add it to the dataframe as additional columns. Then remove the original `Genre` column.

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

```
# TODO: Replace Genre with a collection of binary columns
genres = unlist(strsplit(df2$Genre, ','))
genres = tolower(gsub(' ', '', genres))
genres = unique(genres)

genres_df = data.frame(matrix(nrow = nrow(df2), ncol = length(genres)))
colnames(genres_df) = genres
df2$Genres_mod = tolower(gsub(' ', '', df2$Genre))
if ("Genre" %in% colnames(df2)){
  df2 = subset(df2, select=-c(Genre))
}

genres_df[,] = 0
for (item in genres) {
  genres_df[grep(item, df2[,"Genres_mod"]), item] = 1
}
df2 = cbind(df2, genres_df)
```

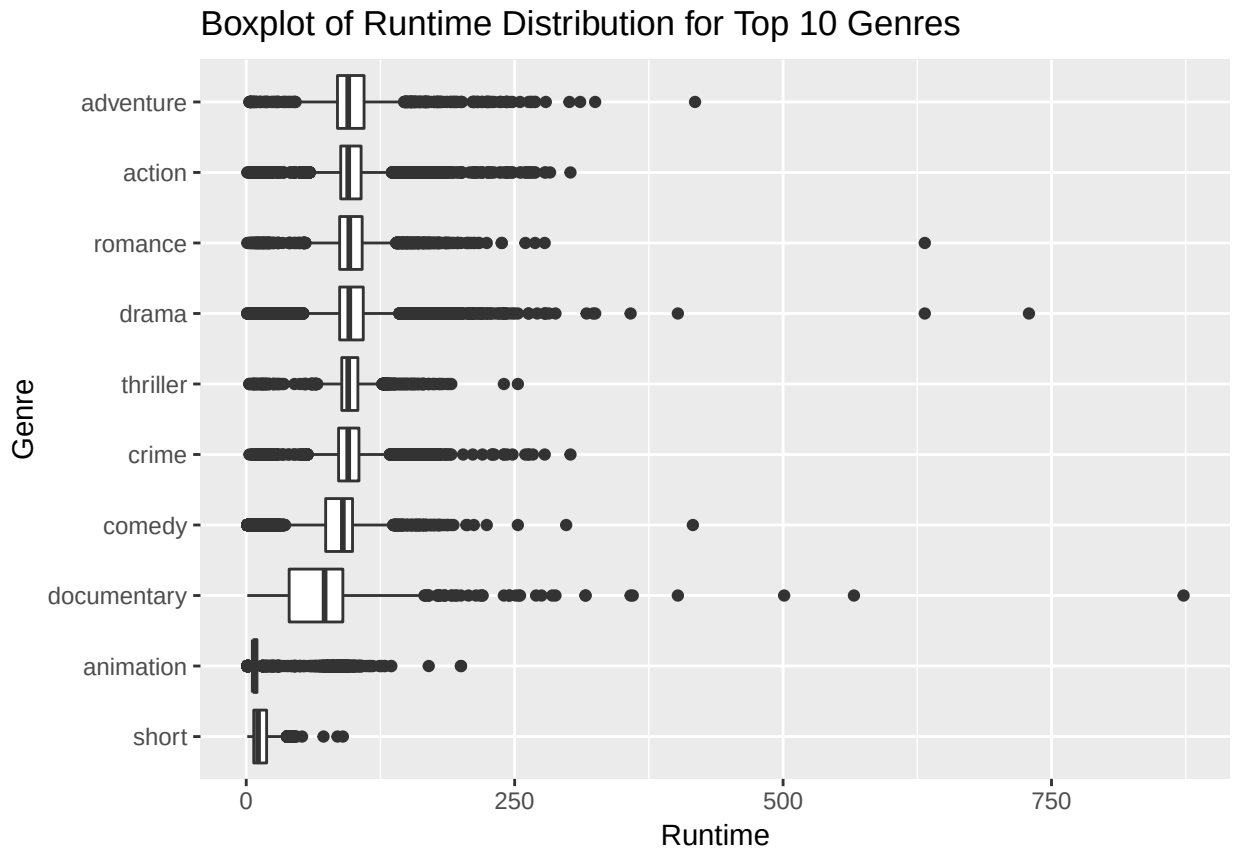
Plot the relative proportions of movies having the top 10 most common genres.

```
# TODO: Select movies from top 10 most common genres and plot their relative proportions
sum_genres = data.frame(genre = names(genres_df), count=colSums(genres_df))
sum_genres$proportion = sum_genres$count / nrow(df2)

sum_genres = sum_genres[order(-sum_genres$proportion),]
top_ten_genres = sum_genres[1:10, ]

ggplot(top_ten_genres, aes(x=reorder(genre, proportion), y=proportion)) +
  geom_bar(stat="identity") +
  xlab("Genre") + ylab("Porportion") + ggtitle("Porportion of Movies in Top 10 Genres")
```



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

```
# TODO: Plot Runtime distribution for top 10 most common genres
runtime_genres_df = NULL
for (name in top_ten_genres$genre) {
  runtime_data = df2[df2[[name]]==1,c('Title','Runtime')]
  runtime_data$genre = name
  runtime_genres_df = rbind(runtime_genres_df, runtime_data)
}

runtime_genres_df = merge(runtime_genres_df, top_ten_genres[,c("genre","proportion")], by="genre")
runtime_genres_df = runtime_genres_df[!is.na(runtime_genres_df$Runtime),]
runtime_genres_df$Runtime = as.numeric(as.character(runtime_genres_df$Runtime))

ggplot(runtime_genres_df, aes(x=reorder(genre, Runtime), y=Runtime)) + geom_boxplot() + coord_flip() +
  xlab("Genre") +
  ylab("Runtime") +
  ggtitle("Boxplot of Runtime Distribution for Top 10 Genres")
```



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

A: Looking at the box plot above, we can see that short and animation films have shorter runtimes. Documentaries tend to have a medium runtime and all other genres of films have about the same median runtimes. It is expected for short films to have short runtimes, and genres such as adventure, action, drama, triller, crime to have long runtimes as these genres tend to produce box hits and have higher production costs. One surprising trend is that animation films tend to have short runtimes, as films such as the Pixel films tend to have long runtimes. Another unexpected trend is that documentary has the largest spread, it even includes the film with the longest runtime.

4. Eliminate mismatched rows

The dataframe was put together by merging two different sources of data and it is possible that the merging process was inaccurate in some cases (the merge was done based on movie title, but there are cases of different movies with the same title). There are 3 columns that contain date information: `Year` (numeric year), `Date` (numeric year), and `Released` (string representation of the release date).

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

Note: Do not remove the rows with `Gross == NA` at this point, just use this a guideline.

```
# TODO: Remove rows with Year/Date/Released mismatch
orig_rowcount = nrow(df2)
df2$Released_Year = as.numeric(format(as.Date(df2$Released, format = "%Y-%m-%d"), "%Y"))
```

```r
df2$removal = 1
for (i in seq(1:nrow(df2))) {
  if (is.na(df2$Gross[i])) {
    df2$removal[i] = 0
  }
  else if (is.na(df2$Year[i]) & is.na(df2$Date[i]) & is.na(df2$Date[i])) {
    df2$removal[i] = 0
  }
  else if (is.na(df2$Year[i])) {
    if (is.na(df2$Date[i]) | is.na(df2$Released_Year[i])) {
      df2$removal[i] = 0
    }
    else if (df2$Date[i] == df2$Released_Year[i]) {
      df2$removal[i] = 0
    }
  }
  else if (is.na(df2$Date[i])) {
    if (is.na(df2$Released_Year[i])) {
      df2$removal[i] = 0
    }
    else if (df2$Year[i] == df2$Released_Year[i]) {
      df2$removal[i] = 0
    }
  }
  else if (is.na(df2$Released_Year[i])) {
    if (df2$Date[i] == df2$Year[i]) {
      df2$removal[i] = 0
    }
  }
  else if (df2$Date[i] == df2$Year[i] & df2$Year[i] == df2$Released_Year[i]) {
    df2$removal[i] = 0
  }
}

df2 = subset(df2, removal == 0)
```

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

A: I used the following logic to denote if rows should be kept, note that these are checked in order, so some redundant logic are removed, such as not checking if Released and Date are NA in 4, as they would have passed in the above steps.

1. Keep if Gross is NA
2. Keep if all three of Year, Date and Released are NA
3. If Year is NA, and either Released is NA or Date is NA
4. If Year is NA, and Date is equal to Released
5. If Date is NA, keep if Released is also NA
6. If Date is NA, and Released is equal to Year
7. If Released is NA, check if Date and Year are equal
8. Check if all three are equal

This ensures that we only keep all the valid rows. Prior to the removal logic, there were 40,000 rows, after the removal, there remains 38,982 rows.

5. Explore `Gross` revenue

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

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

```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
df5 = df2
df5$Runtime_Binary = ifelse(df5$Runtime > 40, "Long", "Short")
df5 = df5[!is.na(df5$Runtime_Binary),]

df_q5 = NULL
for (name in top_ten_genres$genre) {
  q5_data = df5[df5[[name]]==1,c('Runtime_Binary', 'Gross', 'Budget', 'Runtime')]
  q5_data$Genre = name
  df_q5 = rbind(df_q5, q5_data)
}

qplot(x=log(Budget), y=log(Gross), data=df_q5, color=Genre, facets=Genre~Runtime_Binary) +
  stat_smooth(se = FALSE,colour="black") + ggtitle("Budget vs Gross Revenue by Genre")
```

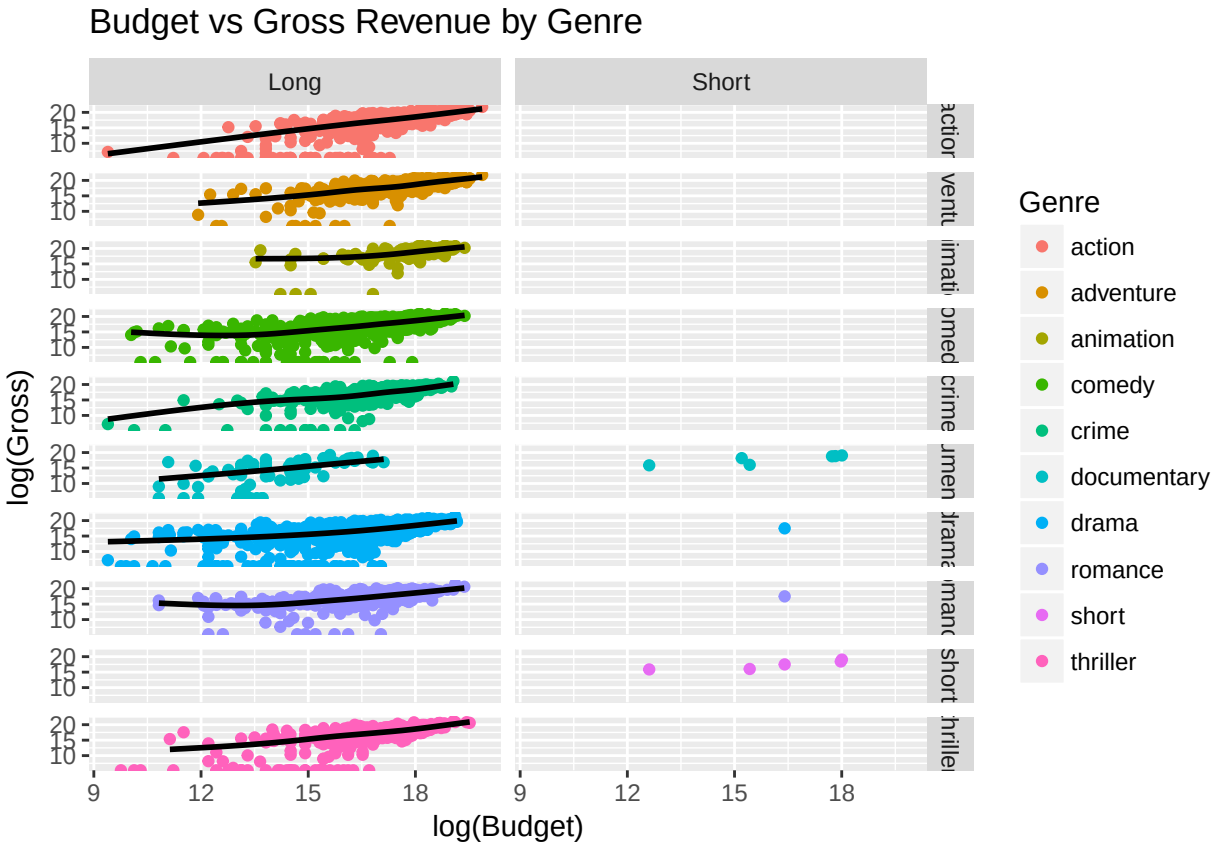
```
## `geom_smooth()` using method = 'gam'
```

```
## Warning: Removed 51797 rows containing non-finite values (stat_smooth).
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```

```
## Warning: Removed 51596 rows containing missing values (geom_point).
```




```
qplot(x=log(Runtime), y=log(Gross), data=df_q5, color=Genre, facets=Runtime_Binary~Genre) +
  stat_smooth(se = FALSE,colour="black") + ggtitle("Runtime vs Gross Revenue by Genre")
```

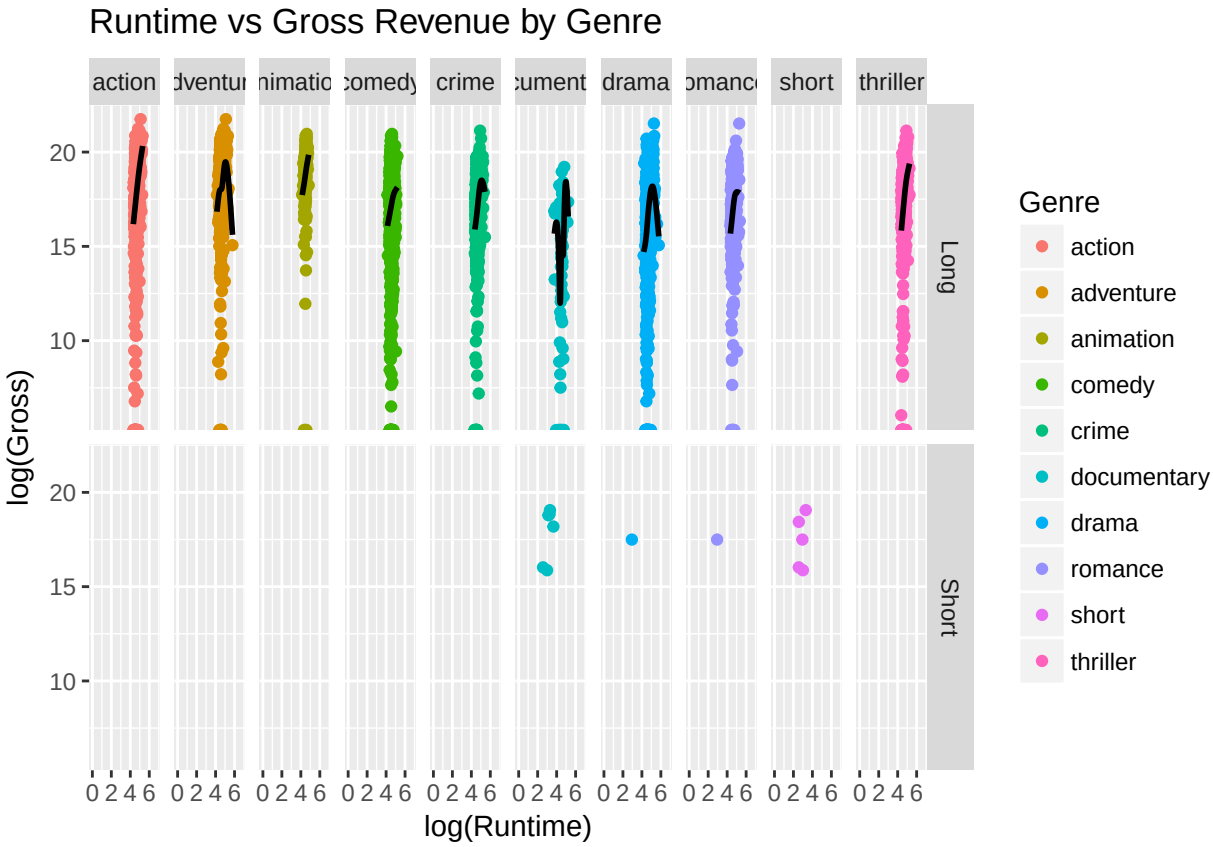
```
## `geom_smooth()` using method = 'gam'
```

```
## Warning: Removed 51797 rows containing non-finite values (stat_smooth).
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```

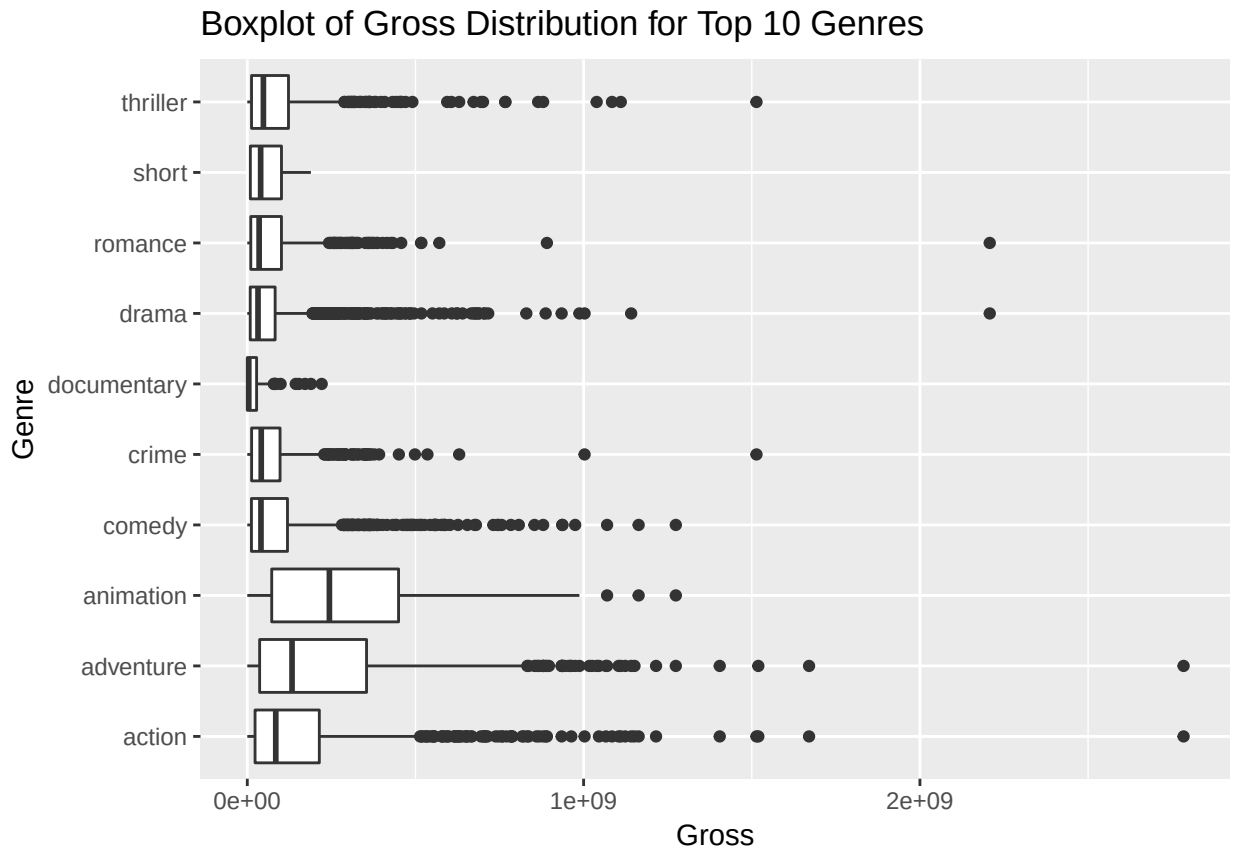
```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```

```
## Warning: Removed 51596 rows containing missing values (geom_point).
```




```
ggplot(df_q5, aes(x=reorder(Genre, Gross, mean), y=Gross)) + geom_boxplot() + coord_flip() +
  scale_x_discrete("Genre") +
  xlab("Genre") +
  ylab("Gross") +
  ggtitle("Boxplot of Gross Distribution for Top 10 Genres")
```

```
## Warning: Removed 51596 rows containing non-finite values (stat_boxplot).
```



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: I first divided movies into long and short with those longer than 40 minutes being long and shorter being short. Then, I looked at the relationships between the factors for the movies in the top ten genres to avoid clutter.

1. Budget and Gross Revenue. There is a strong trend between Budget and Gross Revenue in almost all the categories for long films. In the box plot, the positive linear relationship is quite obvious. So we can conclude high budget results in high gross revenue. This is also noticeable in shorter films, but not enough data is available.
2. Runtime and Gross Revenue. Ther is some correlation between runtime and gross revenue. In genres such as action, thrillers and documentaries, we can see there is an upward trend as runtime increases. Even in shorter films, there are some data to support this. However, many genres, such as adventure and dramas show very little correlation.
3. Genre and Gross Revenue. Looking at the boxplot above, we can see that genres such as Animation, Adventure, and Actions have higher median gross revenues. Documentaries have the lowest gross revenue in the top ten, as expected by common sense.This means there is a relationship between genres and gross revenue, with some genres raking in more revenue than other genres.

```
# TODO: Investigate if Gross Revenue is related to Release Month
df5$Released_Month = as.numeric(format(as.Date(df5$Released, format = "%Y-%m-%d"), "%m"))
df5 = df5[!is.na(df5$Released_Month),]
df5 = df5[!is.na(df5$Gross),]

df5$Released_Month = as.numeric(as.character(df5$Released_Month))
```

```
df5$Gross = as.numeric(as.character(df5$Gross))

ggplot(df5, aes(x=reorder(Released_Month, Gross, median), y=log(Gross))) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Released_Month") +
  xlab("Released Month") +
  ylab("Gross Revenue") +
  ggtitle("Gross Revenue by Movie Release Month")
```

```
## Warning: Removed 126 rows containing non-finite values (stat_boxplot).
```

A: Gross Revenue is indeed related to release month, we can see that June, December, July, May and November are the top months for gross revenue, these are indeed the holiday months or the months when kids are on vacation. Fall months (September, October) tend to have the lowest gross revenue.

6. Process `Awards` column

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

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

```r
# TODO: Convert Awards to 2 numeric columns: wins and nominations
df2$Awards_mod = tolower(df2$Awards)
df2$Awards_Split = strsplit(df2$Awards_mod, "\\,|\\ |\\.|\\&")

different_win_nominations=vector()
for (i in seq(1:nrow(df2))) {
  awards_vector = unlist(df2$Awards_Split[i])
  wins= 0
  noms = 0
  if ("win" %in% awards_vector) {
    index = which(awards_vector == "win")
    wins= wins + as.integer(awards_vector[index-1])
  }
  if ("wins" %in% awards_vector) {
    index = which(awards_vector == "wins")
    wins= wins + as.integer(awards_vector[index-1])
  }
  if ("won" %in% awards_vector) {
    index = which(awards_vector == "won")
    wins= wins + as.integer(awards_vector[index+1])
  }
  if ("nominated" %in% awards_vector) {
    index = which(awards_vector == "nominated")
    noms= noms + as.integer(awards_vector[index+2])
  }
  if ("nomination" %in% awards_vector) {
    index = which(awards_vector == "nomination")
    noms= noms + as.integer(awards_vector[index-1])
  }
  if ("nominations" %in% awards_vector) {
    index = which(awards_vector == "nominations")
    noms= noms + as.integer(awards_vector[index-1])
  }
  df2$Awards_Won[i] = wins
  df2$Awards_Nom[i] = noms
}

valid_wins=nrow(subset(df2,Awards_Won != 0))
valid_noms=nrow(subset(df2,Awards_Nom != 0))
valid_wins_noms_or = nrow(subset(df2, Awards_Won != 0 | Awards_Nom != 0))
valid_wins_noms_and = nrow(subset(df2, Awards_Won != 0 & Awards_Nom != 0))
```

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

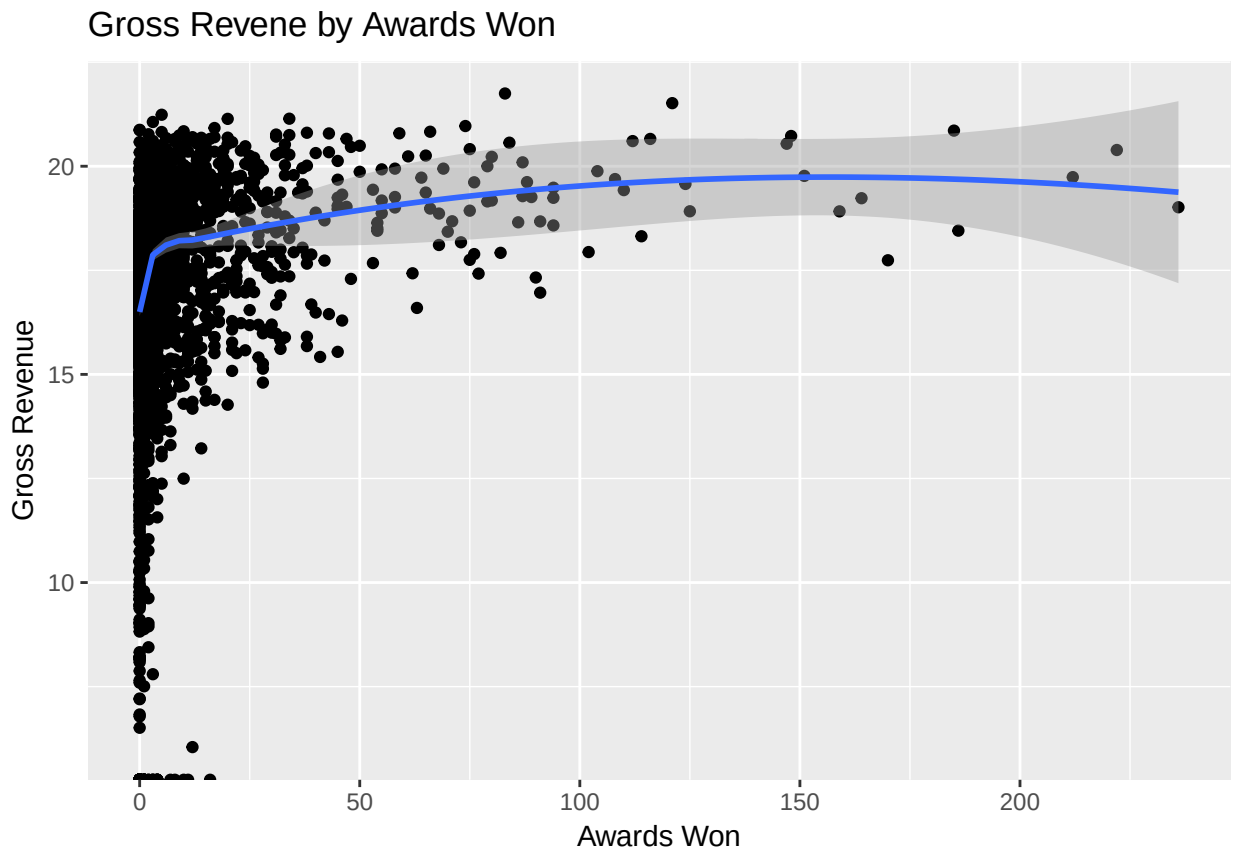
A: I first split the string for Awards in df2 into into string vectors. Then, I iterated all the rows in df2 one by one, initializing the number of wins and nominations to 0. Then unlisted the vector into strings. Then, I looked for keywords in the unlisted strings. I know from looking at the Awards column that the number precedes keywords "win" and "wins", and succeeds "won". So I looked for these keywords then found the number index position accordingly. These are then accumulated to the current wins. A similiar technique was used for nominations with the keywords "nomination", "nominations" and "nominated". These values are then set to two new columns in df2 for the current row.

Looking at statistics, there are 10,264 rows with valid wins, 10,684 with valid nominations. In addition, there are 7,024 with both valid wins and nominations, and 13,922 rows with either a valid nomination or win.

```
# TODO: Plot Gross revenue against wins and nominations
ggplot(df2, aes(x=Awards_Won, y=log(Gross))) +
  geom_point() + geom_smooth(method = "loess") +
  xlab("Awards Won") +
  ylab("Gross Revenue") +
  ggtitle("Gross Revene by Awards Won")
```

```
## Warning: Removed 35593 rows containing non-finite values (stat_smooth).
```

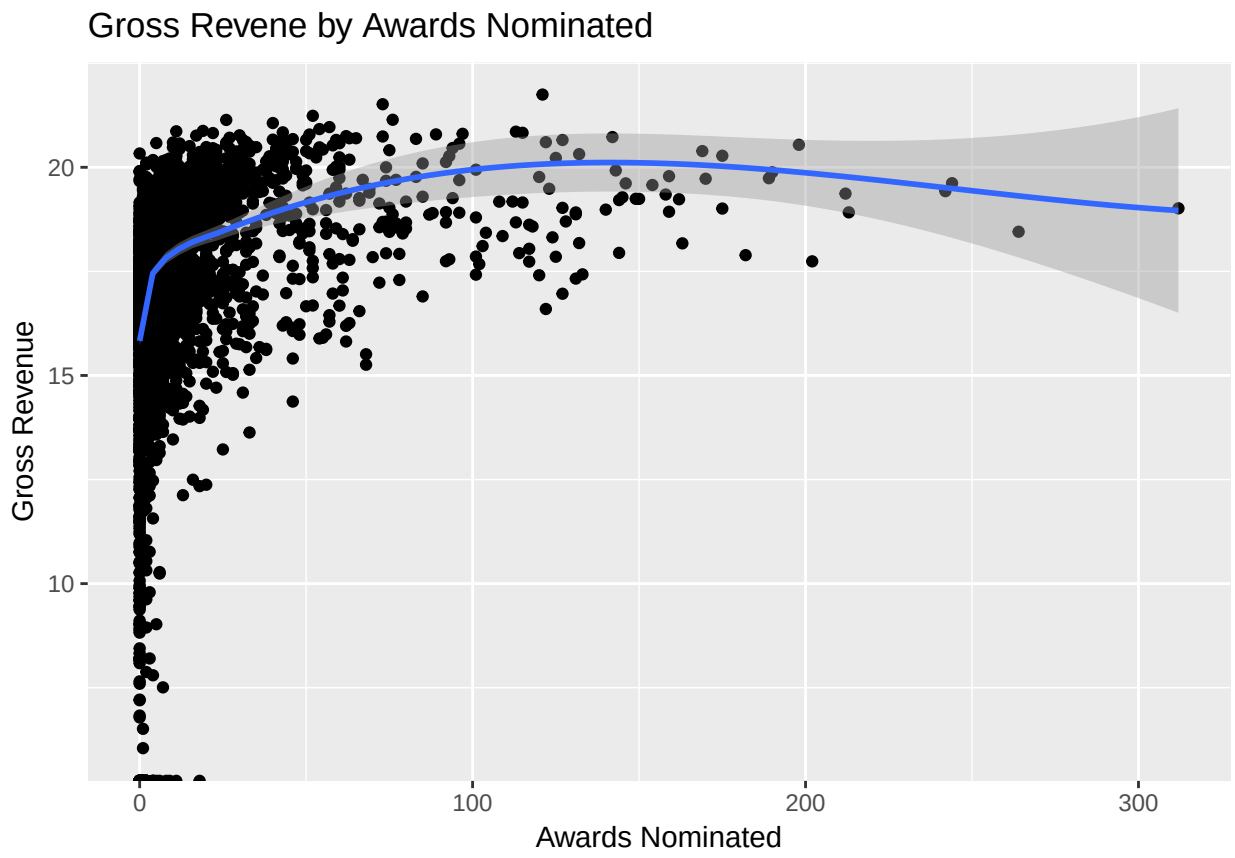
```
## Warning: Removed 35442 rows containing missing values (geom_point).
```




```
ggplot(df2, aes(x=Awards_Nom, y=log(Gross))) +
  geom_point() + geom_smooth(method = "loess") +
  xlab("Awards Nominated") +
  ylab("Gross Revenue") +
  ggtitle("Gross Revene by Awards Nominated")
```

```
## Warning: Removed 35593 rows containing non-finite values (stat_smooth).
```

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



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

A: There is a weak positive relationship between number of awards won and gross revenue as well as number of nominations and gross revenue. For both awards and nominations, the gross revenue tends to increase dramatically from 0 to 25 awards/nominations. Then onwards ther is no clear positive relationship and plateaus.

7. Movie ratings from IMDb and Rotten Tomatoes

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

Investigate the pairwise relationships between these different descriptors using graphs.

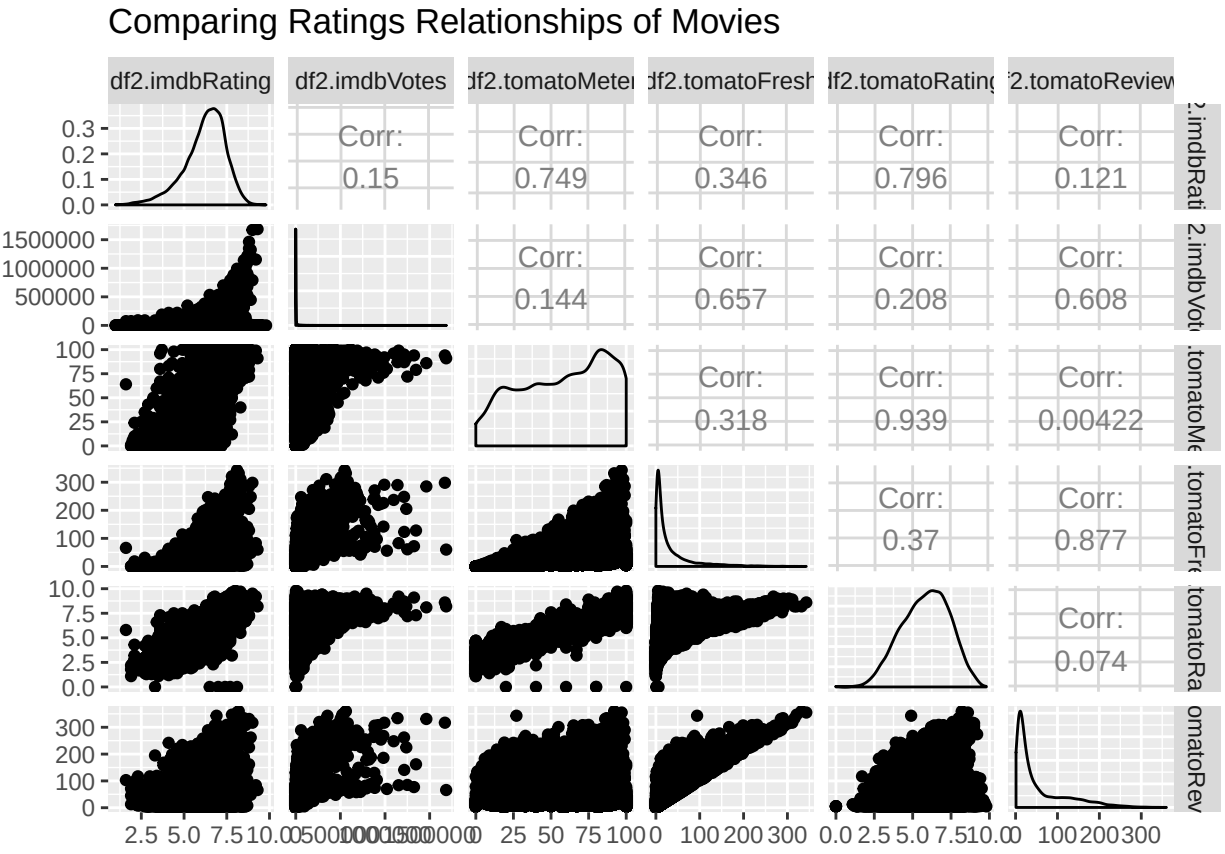
```
# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
ratings_df = data.frame(df2$imdbRating, df2$imdbVotes, df2$tomatoMeter, df2$tomatoFresh, df2$tomatoRati
ratings_ggp = ggpairs(ratings_df, title="Comparing Ratings Relationships of Movies")
print(ratings_ggp, progress = F)
```

```
## Warning in ggmatrix_gtable(x, ...): Please use the 'progress' parameter
## in your ggmatrix-like function call. See ?ggmatrix_progress for a few
## examples. ggmatrix_gtable 'progress' and 'progress_format' will soon be
## deprecated.TRUE
```

```
## Warning: Removed 1114 rows containing non-finite values (stat_density).
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1152 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30371 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30327 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30380 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30327 rows containing missing values

## Warning: Removed 1152 rows containing missing values (geom_point).

## Warning: Removed 1152 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30373 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30330 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30382 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30330 rows containing missing values

## Warning: Removed 30371 rows containing missing values (geom_point).

## Warning: Removed 30373 rows containing missing values (geom_point).

## Warning: Removed 30369 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30377 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30378 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30377 rows containing missing values

## Warning: Removed 30327 rows containing missing values (geom_point).

## Warning: Removed 30330 rows containing missing values (geom_point).

## Warning: Removed 30377 rows containing missing values (geom_point).

## Warning: Removed 30325 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30378 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30325 rows containing missing values

## Warning: Removed 30380 rows containing missing values (geom_point).

## Warning: Removed 30382 rows containing missing values (geom_point).
```

```
## Warning: Removed 30378 rows containing missing values (geom_point).

## Warning: Removed 30378 rows containing missing values (geom_point).

## Warning: Removed 30378 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30378 rows containing missing values

## Warning: Removed 30327 rows containing missing values (geom_point).

## Warning: Removed 30330 rows containing missing values (geom_point).

## Warning: Removed 30377 rows containing missing values (geom_point).

## Warning: Removed 30325 rows containing missing values (geom_point).

## Warning: Removed 30378 rows containing missing values (geom_point).

## Warning: Removed 30325 rows containing non-finite values (stat_density).
```



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

A: First of all, we can see there is a strong linear relationship between imdbRating and tomatoRating, with a strong correlation of 0.796. This says that the generic public generally agrees with the critics when it comes to rating movies.

There is also a strong correlation between tomatoFresh as well as imdbVotes as well as tomatoReviews and imdbVotes. This might suggest that Rotten Tomatoes reviews tend to influence more votes on IMDB.

It is also interesting to note that imdbVotes have almost no relationship to imdbRating, so the number of votes do not influence higher ratings. However, there is more correlation among the Rotten Tomatoes categories, we see a very strong linear relationship between tomatoReviews and tomatoFresh, as well as tomatoRating and tomatoMeter. Some other categories tend to have little correlation, such as between tomatoRating and tomatoReviews, or between tomatoFresh and tomatoMeter.

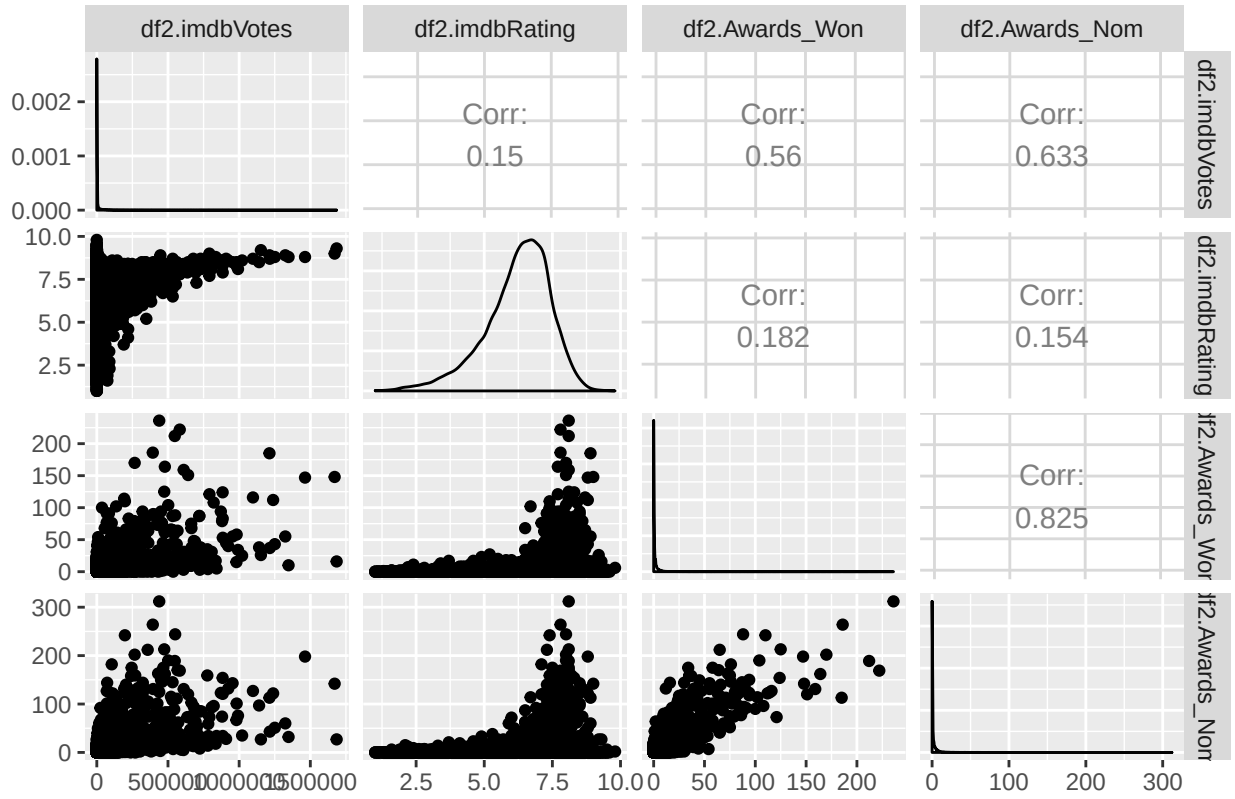
8. Ratings and awards

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

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

```
# TODO: Show how ratings and awards are related
ratings_awards_imdb_df = data.frame(df2$imdbVotes, df2$imdbRating, df2$Awards_Won, df2$Awards_Nom)
ratings_awards_tomato1_df = data.frame(df2$tomatoRating, df2$tomatoMeter, df2$Awards_Won, df2$Awards_Nom
ratings_awards_tomato2_df = data.frame(df2$tomatoFresh, df2$tomatoReviews, df2$Awards_Won, df2$Awards_N
ratings_imdb_ggp = ggpairs(ratings_awards_imdb_df, title="Comparing Awards against IMDB Factors")
ratings_toma1_ggp = ggpairs(ratings_awards_tomato1_df, title="Comparing Awards against Rotten Tomatoes
ratings_toma2_ggp = ggpairs(ratings_awards_tomato2_df, title="Comparing Awards against Rotten Tomatoes

print(ratings_imdb_ggp, progress = F)
```

```
## Warning in ggmatrix_gtable(x, ...): Please use the 'progress' parameter
## in your ggmatrix-like function call. See ?ggmatrix_progress for a few
## examples. ggmatrix_gtable 'progress' and 'progress_format' will soon be
## deprecated.TRUE

## Warning: Removed 1152 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1152 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1152 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1152 rows containing missing values

## Warning: Removed 1152 rows containing missing values (geom_point).

## Warning: Removed 1114 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1114 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1114 rows containing missing values

## Warning: Removed 1152 rows containing missing values (geom_point).

## Warning: Removed 1114 rows containing missing values (geom_point).

## Warning: Removed 1152 rows containing missing values (geom_point).

## Warning: Removed 1114 rows containing missing values (geom_point).
```

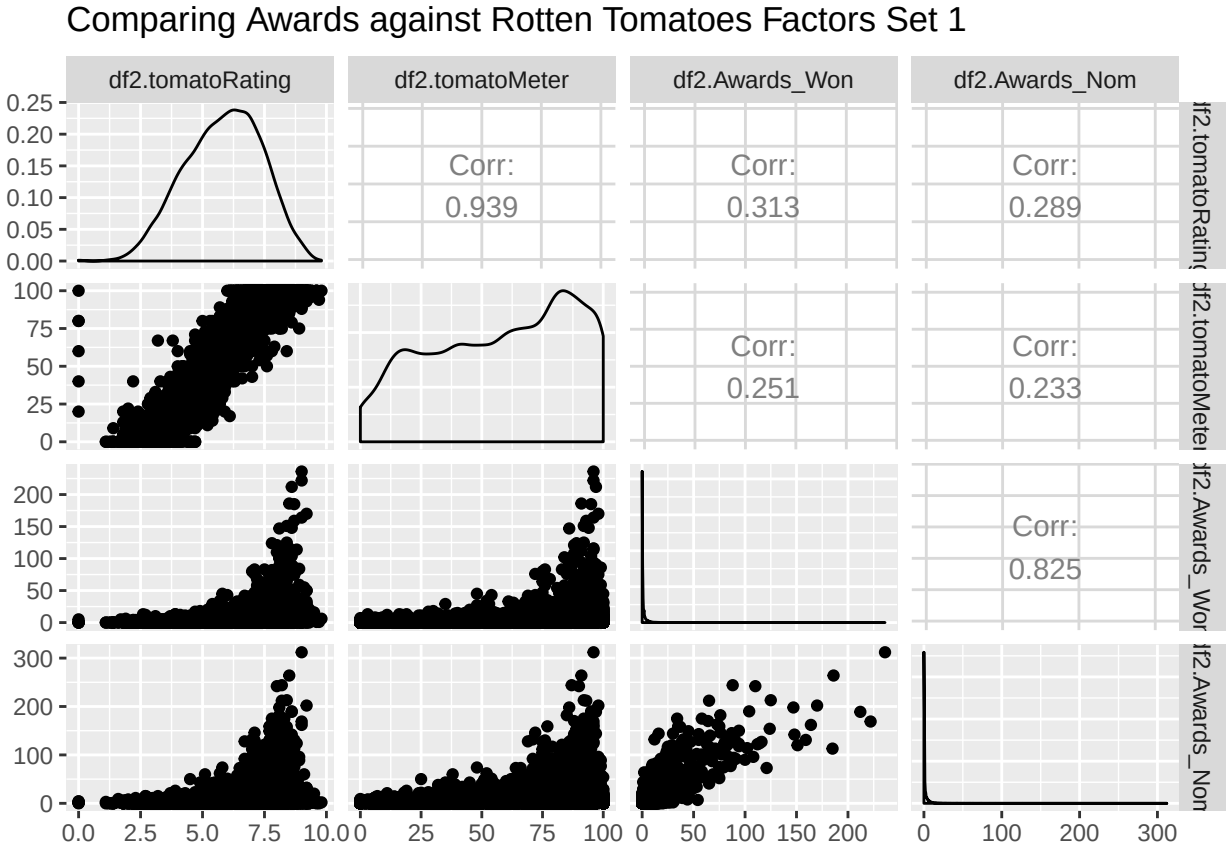



```
print(ratings_toma1_ggp, progress = F)
```

```
## Warning in ggmatrix_gtable(x, ...): Please use the 'progress' parameter
## in your ggmatrix-like function call. See ?ggmatrix_progress for a few
## examples. ggmatrix_gtable 'progress' and 'progress_format' will soon be
## deprecated.TRUE

## Warning: Removed 30378 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30378 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30378 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30378 rows containing missing values

## Warning: Removed 30378 rows containing missing values (geom_point).

## Warning: Removed 30369 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30369 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30369 rows containing missing values

## Warning: Removed 30378 rows containing missing values (geom_point).

## Warning: Removed 30369 rows containing missing values (geom_point).
```

```
## Warning: Removed 30378 rows containing missing values (geom_point).
```

```
## Warning: Removed 30369 rows containing missing values (geom_point).
```




```
print(ratings_toma2_ggp, progress = F)
```

```
## Warning in ggmatrix_gtable(x, ...): Please use the 'progress' parameter
## in your ggmatrix-like function call. See ?ggmatrix_progress for a few
## examples. ggmatrix_gtable 'progress' and 'progress_format' will soon be
## deprecated.TRUE
```

```
## Warning: Removed 30325 rows containing non-finite values (stat_density).
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30325 rows containing missing values
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30325 rows containing missing values
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30325 rows containing missing values
```

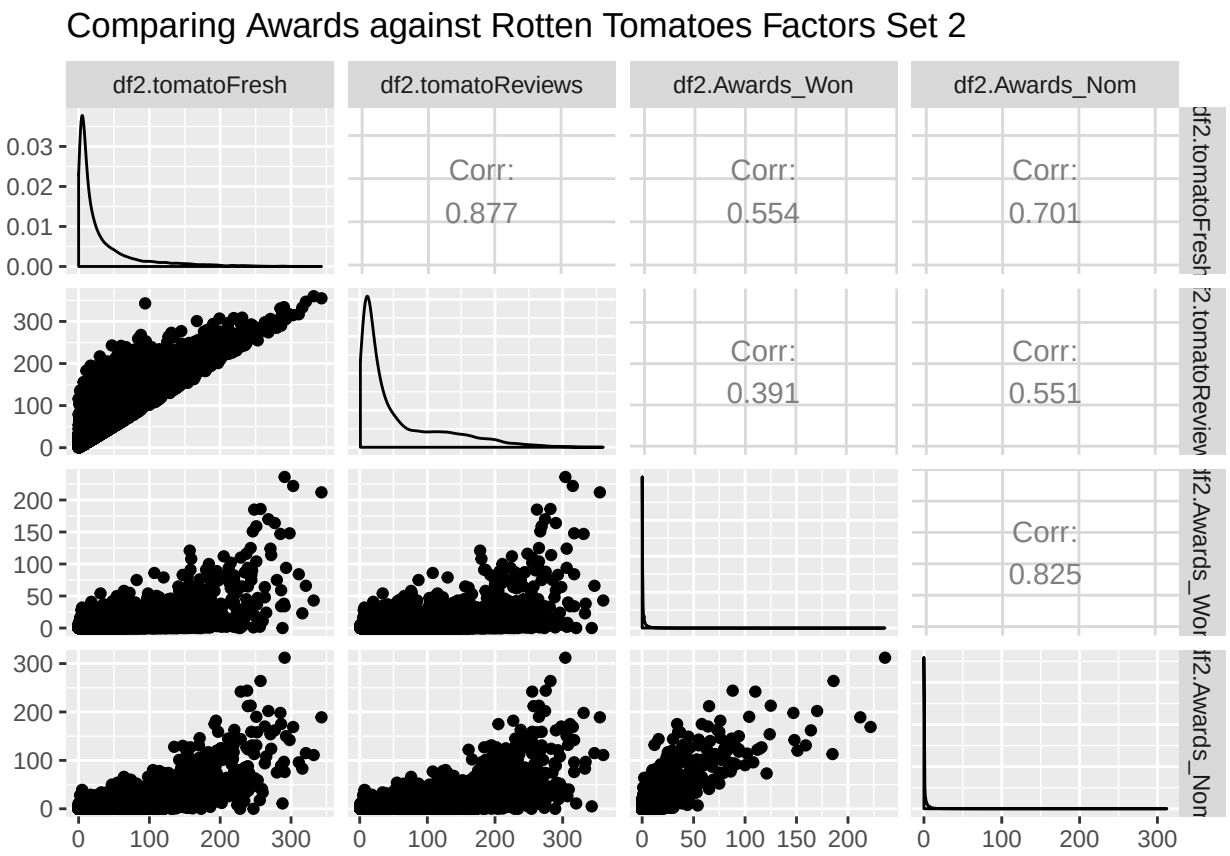
```
## Warning: Removed 30325 rows containing missing values (geom_point).
```

```
## Warning: Removed 30325 rows containing non-finite values (stat_density).
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30325 rows containing missing values
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30325 rows containing missing values
```

```
## Warning: Removed 30325 rows containing missing values (geom_point).

## Warning: Removed 30325 rows containing missing values (geom_point).

## Warning: Removed 30325 rows containing missing values (geom_point).

## Warning: Removed 30325 rows containing missing values (geom_point).
```



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: First looking at IMDB ratings, there does seem to be a correlation between imdbRatings and the awards nominated and won. We can see a strong increase in the number of awards as the ratings approach 7-7.5, the relationship then tends to invert as ratings inch higher, as rewards with really high ratings (e.g 9.5 - 10) do not seem to have more rewards won or nominations. The same relationship can be observed with tomatoRating, as the ratings approach 7.5-9, the number of rewards won and nominations are at the highest, this then tends to decrease as ratings get higher.

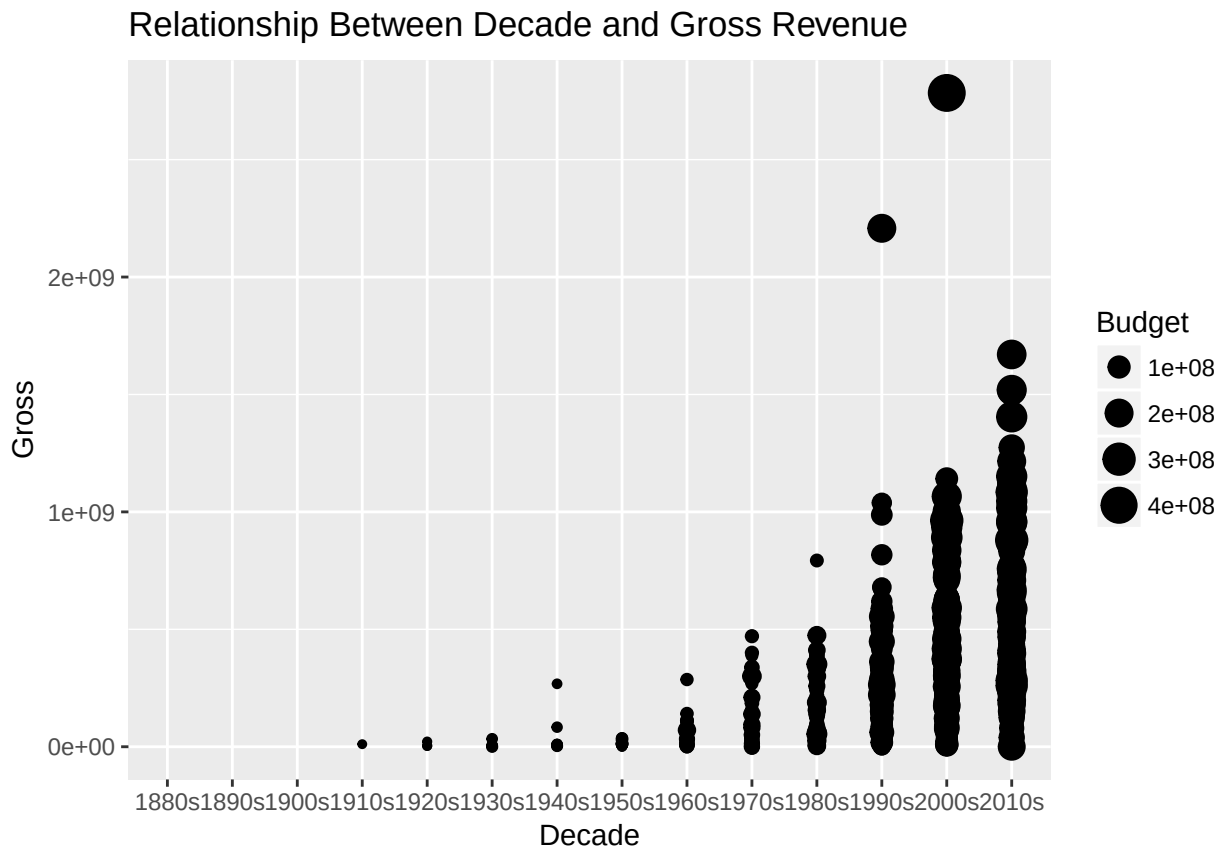
We can also observe similiar positive relationships with tomatoMeter, tomatoReviews and tomatoFresh as related to awards won and nominated. These actually do not show the decrease in awards won and nominated as these factors increase beyond a certain value.

9. Expected insights

Come up with two new insights (backed up by data and graphs) that is expected. Here "new" means insights that are not an immediate consequence of one of the above tasks. You may use any of the columns already explored above or a different one in the dataset, such as `Title`, `Actors`, etc.

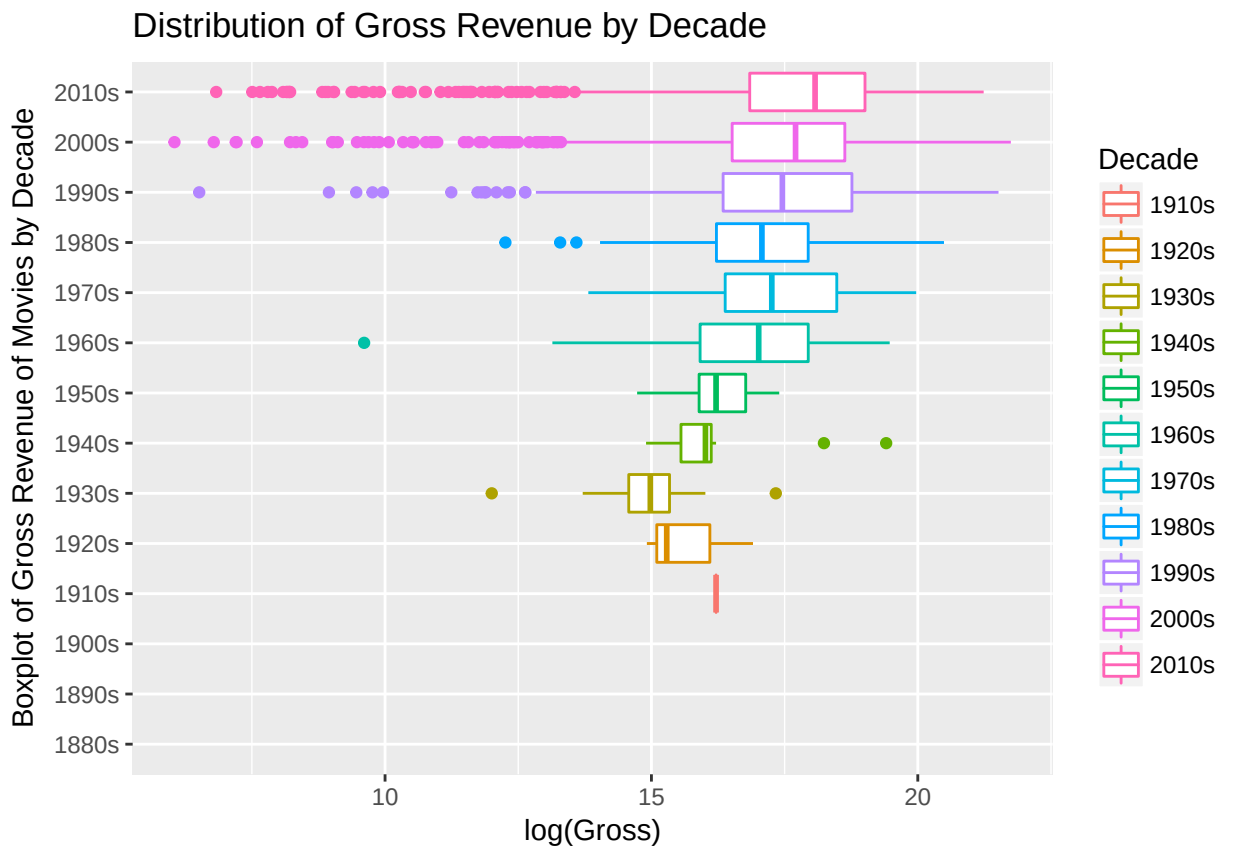
```
# TODO: Find and illustrate two expected insights
qplot(x=Decade, y=Gross, data=df2, size=Budget) + ggtitle("Relationship Between Decade and Gross Revenue
```

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




```
ggplot(df2, aes(reorder(Decade, Gross, median), log(Gross), color=Decade)) +
  geom_boxplot() + coord_flip() +
  scale_x_discrete("Boxplot of Gross Revenue of Movies by Decade") +
  ggtitle("Distribution of Gross Revenue by Decade")
```

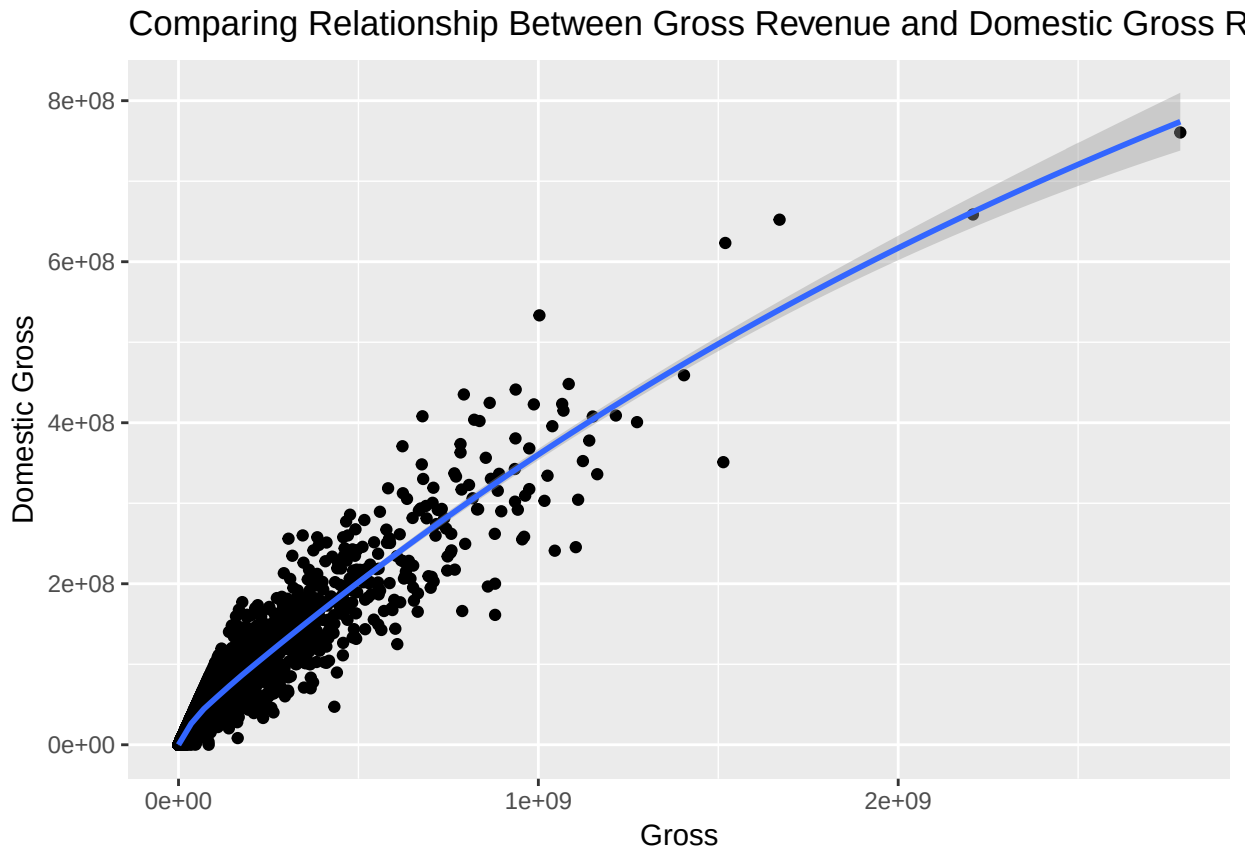
```
## Warning: Removed 35593 rows containing non-finite values (stat_boxplot).
```




```
ggplot(df2, aes(x=Gross, y=Domestic_Gross)) + geom_point() + geom_smooth(method = "loess") +
  ylab("Domestic Gross") +
  ggtitle("Comparing Relationship Between Gross Revenue and Domestic Gross Revenue")
```

```
## Warning: Removed 35442 rows containing non-finite values (stat_smooth).
```

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




```
corr_gross = (cor(df2$Domestic_Gross, df2$Gross, use="complete.obs", method = "pearson"))
```

Q: Expected insight #1.

A: For my first expected insight, I looked at the relationship between gross revenue as related to the year of the film. I expected the gross revenue to steadily increase as the years increased, as we know from earlier that budget increased with increasing years. As expected, the box plot above shows that the median does indeed line up by decade, with the most recent decades with the highest gross revenues.

Q: Expected insight #2.

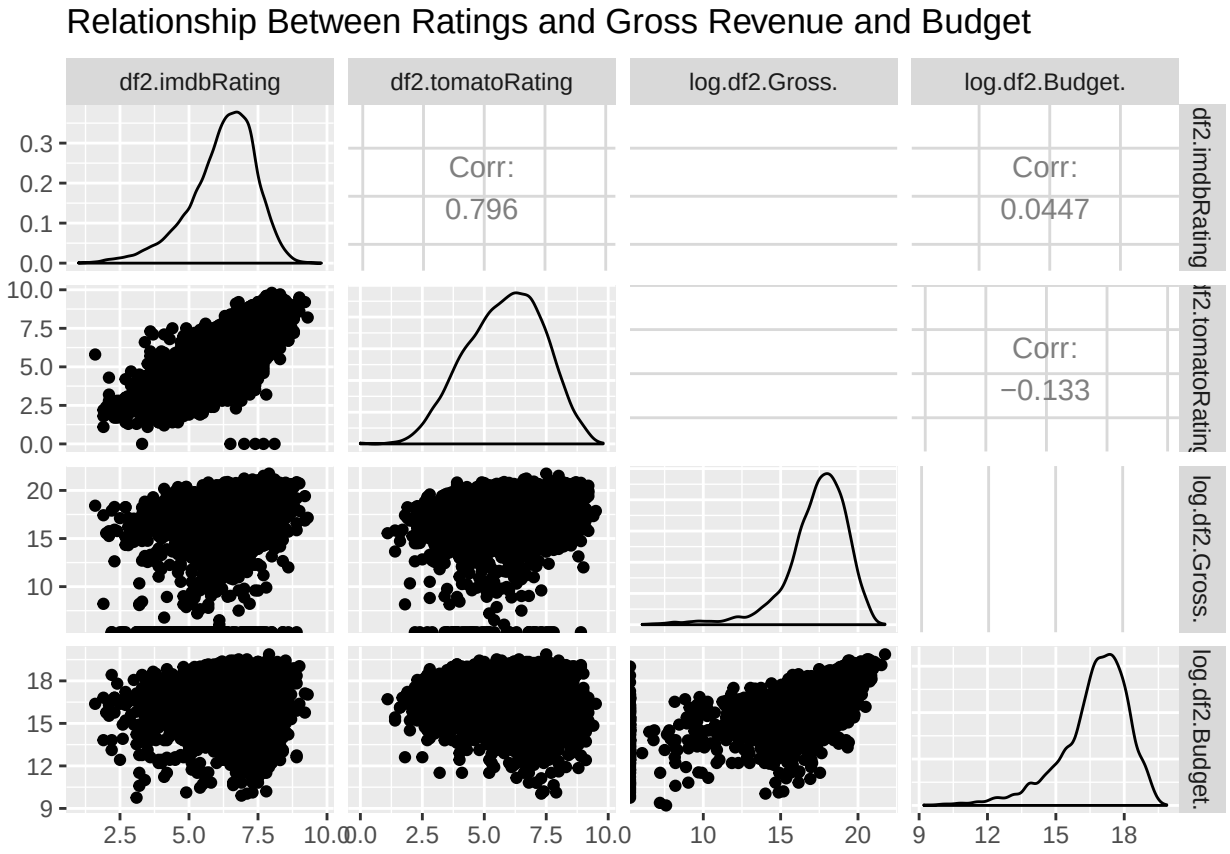
A: For my second insight, I looked at the relationship between gross revenue and domestic gross revenue. I expected there to be a strong relationship between these two variables, as films tend to do well globally if they also do well domestically. As expected, there is a strong linear relationship between gross revenue and domestic gross revenue. Calculating the correlation with the Pearson method, the value came out to approximately 0.9397, which is very strong.

10. Unexpected insight

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

```
# TODO: Find and illustrate one unexpected insight
df_insight1 = data.frame(df2$imdbRating, df2$tomatoRating, log(df2$Gross), log(df2$Budget))
corr_imdb_gross = (cor(df2$imdbRating, df2$Gross, use="complete.obs", method = "pearson"))
corr_tomato_gross = (cor(df2$tomatoRating, df2$Gross, use="complete.obs", method = "pearson"))
q10_ggp = ggpairs(df_insight1, title="Relationship Between Ratings and Gross Revenue and Budget")
print(q10_ggp, progress = F)
```

```
## Warning in ggmatrix_gtable(x, ...): Please use the 'progress' parameter
## in your ggmatrix-like function call. See ?ggmatrix_progress for a few
## examples. ggmatrix_gtable 'progress' and 'progress_format' will soon be
## deprecated.TRUE

## Warning: Removed 1114 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30380 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 35466 rows containing missing values

## Warning: Removed 1 rows containing missing values (geom_text).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 35466 rows containing missing values

## Warning: Removed 30380 rows containing missing values (geom_point).

## Warning: Removed 30378 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 35726 rows containing missing values

## Warning: Removed 1 rows containing missing values (geom_text).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 35726 rows containing missing values

## Warning: Removed 35466 rows containing missing values (geom_point).

## Warning: Removed 35726 rows containing missing values (geom_point).

## Warning: Removed 35593 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 35442 rows containing missing values

## Warning: Removed 1 rows containing missing values (geom_text).

## Warning: Removed 35466 rows containing missing values (geom_point).

## Warning: Removed 35726 rows containing missing values (geom_point).

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

## Warning: Removed 35442 rows containing non-finite values (stat_density).
```



Q: Unexpected insight.

A: I compared the ratings from IMDB and Rotten Tomatoes with the gross revenue as well as budget to see if there is a relationship between. I originally thought that as budget increases, the ratings might be higher. As well as for films with higher ratings, the gross revenues should be higher as well. However, looking at the pairwise graphs above, there is not strong correlation between these factors. In addition, I calculated the Pearson correlations between IMDB ratings and gross revenue as well as Rotten Tomtaoes ratings with gross revenues, these values are 0.2446 and 0.2123 respectively, which show very weak correlation.