Project 1: Explore and Prepare Data

CSE6242 - Data and Visual Analytics - Fall 2017

Due: Sunday, October 15, 2017 at 11:59 PM UTC-12:00 on T-Square

Vincent La (Georgia Tech ID - vla6)

October 12, 2017

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

Data

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

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

Objective

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

Instructions

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

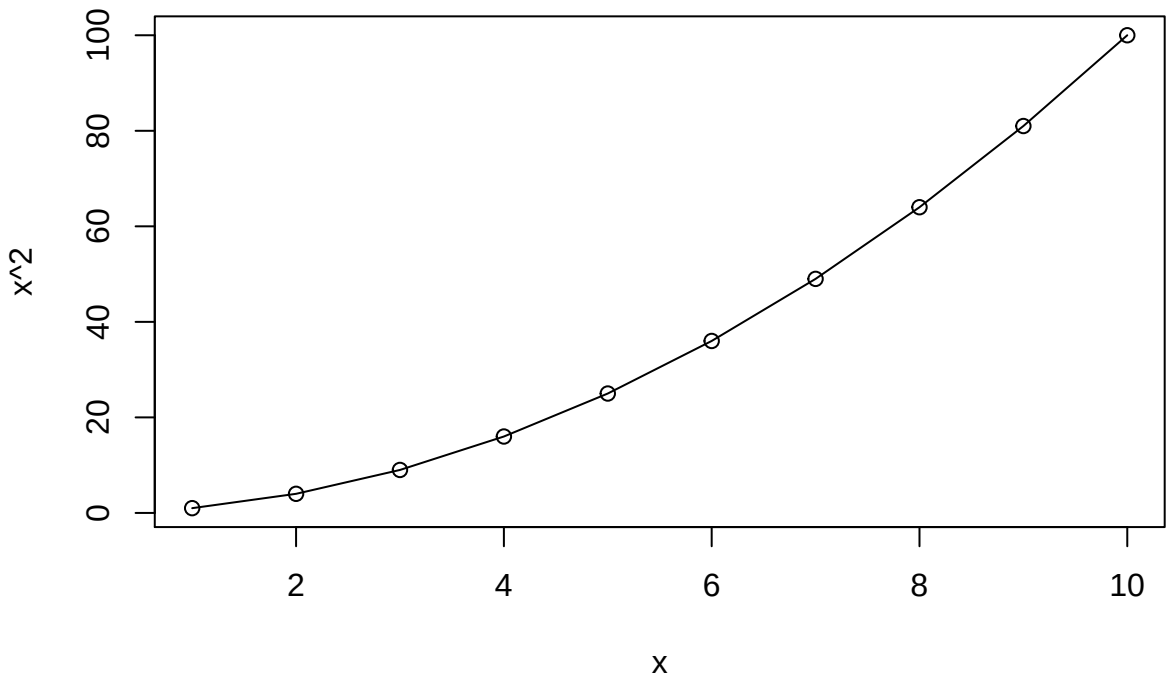
When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```
# Project 1: http://cse6242.gatech.edu/fall-2017/pr1/
# To compile R Markdown in terminal run: Rscript -e "rmarkdown::render('pr1.Rmd', clean=TRUE)"
x = 1:10
print(x^2)
```

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

Plots appear inline too:

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



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

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

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

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

Setup

Load data

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

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

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

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

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

```
## Column names:
```

```
colnames(df)
```

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

Load R packages

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

```
library(plyr)
library(ggplot2)
library(GGally)
library(qdapTools)
```

```
##
## Attaching package: 'qdapTools'
```

```
## The following object is masked from 'package:plyr':
##
##     id
```

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(reshape2)

# Note that I'm using Anaconda R, and I was having trouble installing qdapTools. I'm not sure
# if this is the same on other machines but I did have to install R XML separately
# https://anaconda.org/r/r-xml
# conda install -c r r-xml. Only after could I install qdapTools.

# To create zip file: zip pr1.zip pr1.Rmd pr1.pdf
```

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",]
original_dim = dim(df)
new_dim = dim(df2)

df = df2
# Differences in rows
print(original_dim[1] - new_dim[1])
```

```
## [1] 789
```

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

A: The number of rows left after removal are 40000.

2. Process `Runtime` column

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

```
extract_runtime = function(r){
    times = unlist(r)
    minutes = 0
    for (i in 1:length(times) - 1){
        if (times[i + 1] == 'h'){
            minutes = minutes + as.numeric(times[i]) * 60
        } else if (times[i + 1] == 'min'){
            minutes = minutes + as.numeric(times[i])
        }
    }
    if (minutes == 0){
```

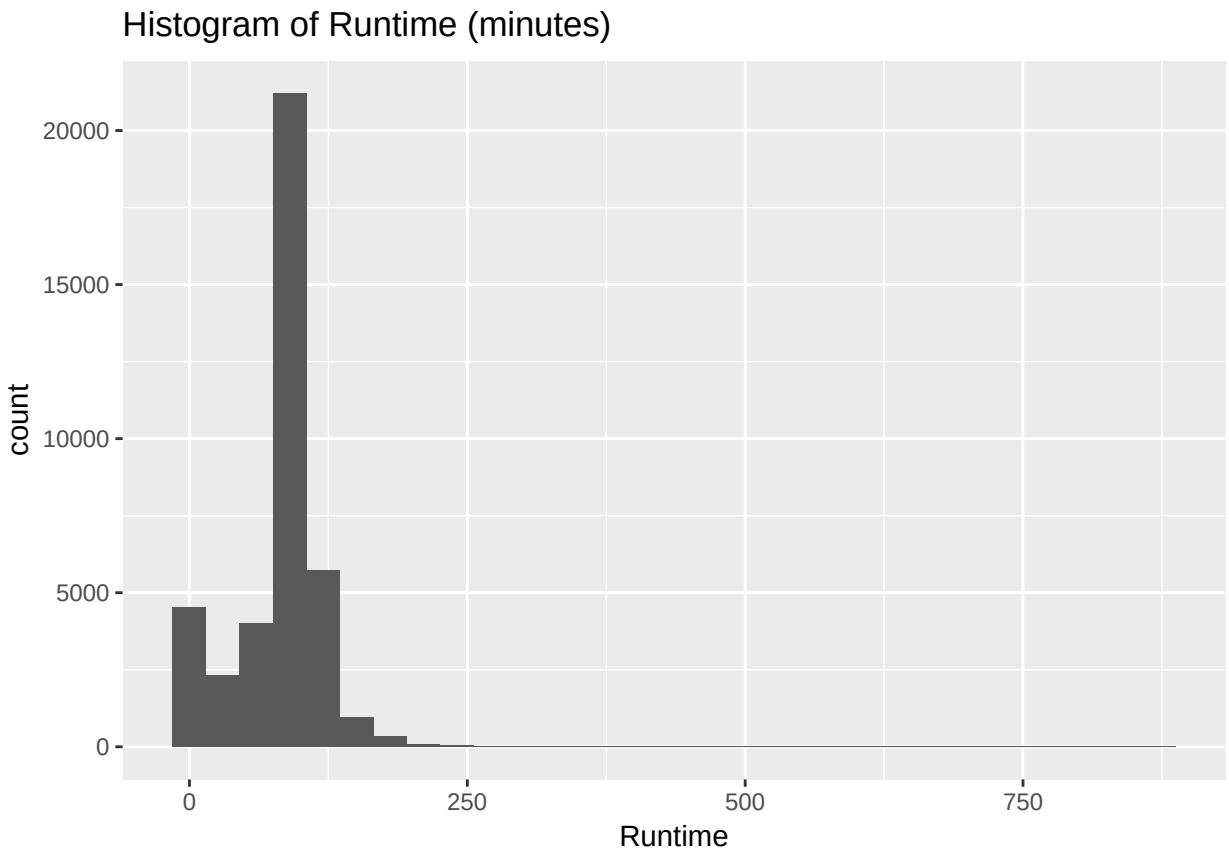
```
        return(NA)
    } else{
        return(minutes)
    }
}

y=strsplit(df$Runtime,' ')
new_runtimes = unlist(lapply(y, extract_runtime))
df$Runtime = new_runtimes

cols = c('Runtime', 'Year', 'Budget')
```

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.

```
# This first graph shows the distribution of runtimes overall and how it varies by Year and Budget
ggplot(df, aes(x=Runtime)) +
  geom_histogram() +
  ggtitle('Histogram of Runtime (minutes)')
```

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

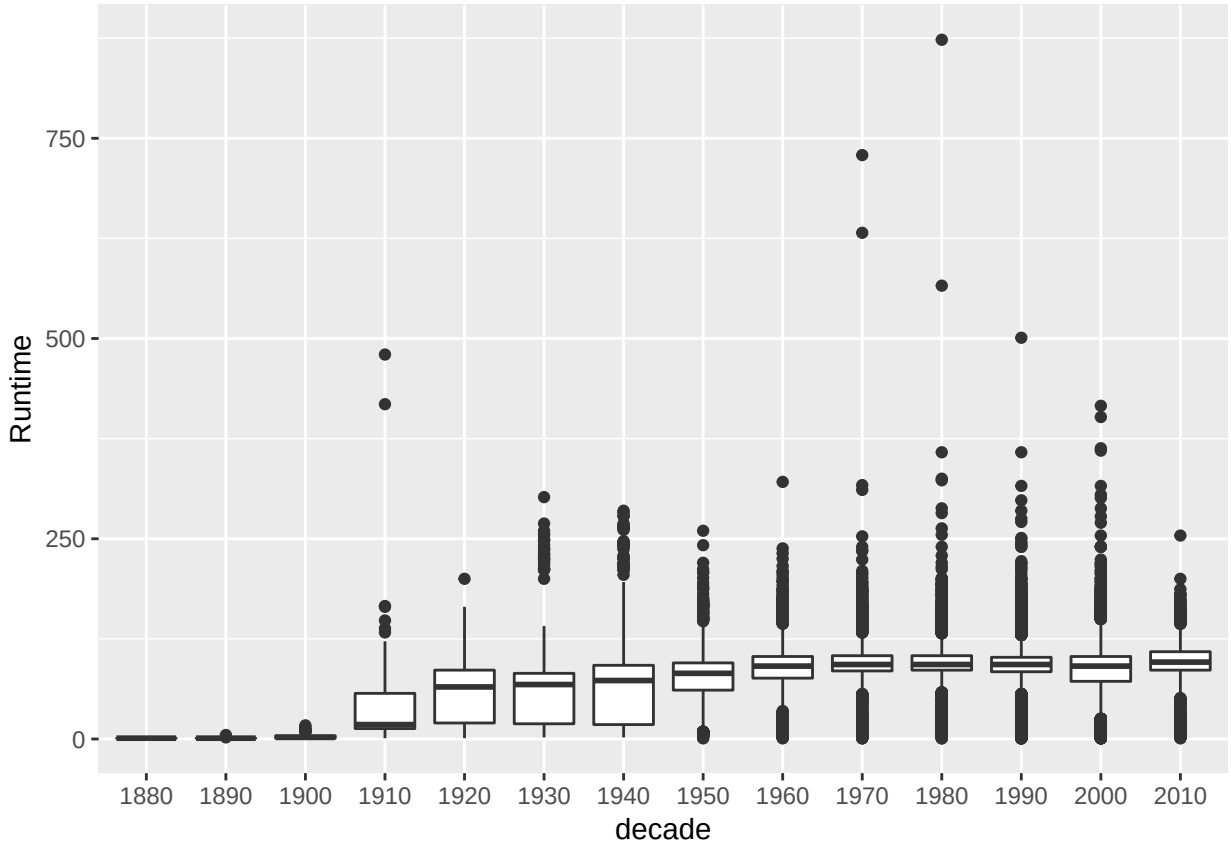
```
## Warning: Removed 751 rows containing non-finite values (stat_bin).
```




```
# Now we need to show distribution of runtime by decade. Do a boxplot grouped by decade.
df$decade = round_any(df$Year, 10, f = floor)
ggplot(df, aes(as.factor(decade), Runtime)) +
  geom_boxplot() +
```

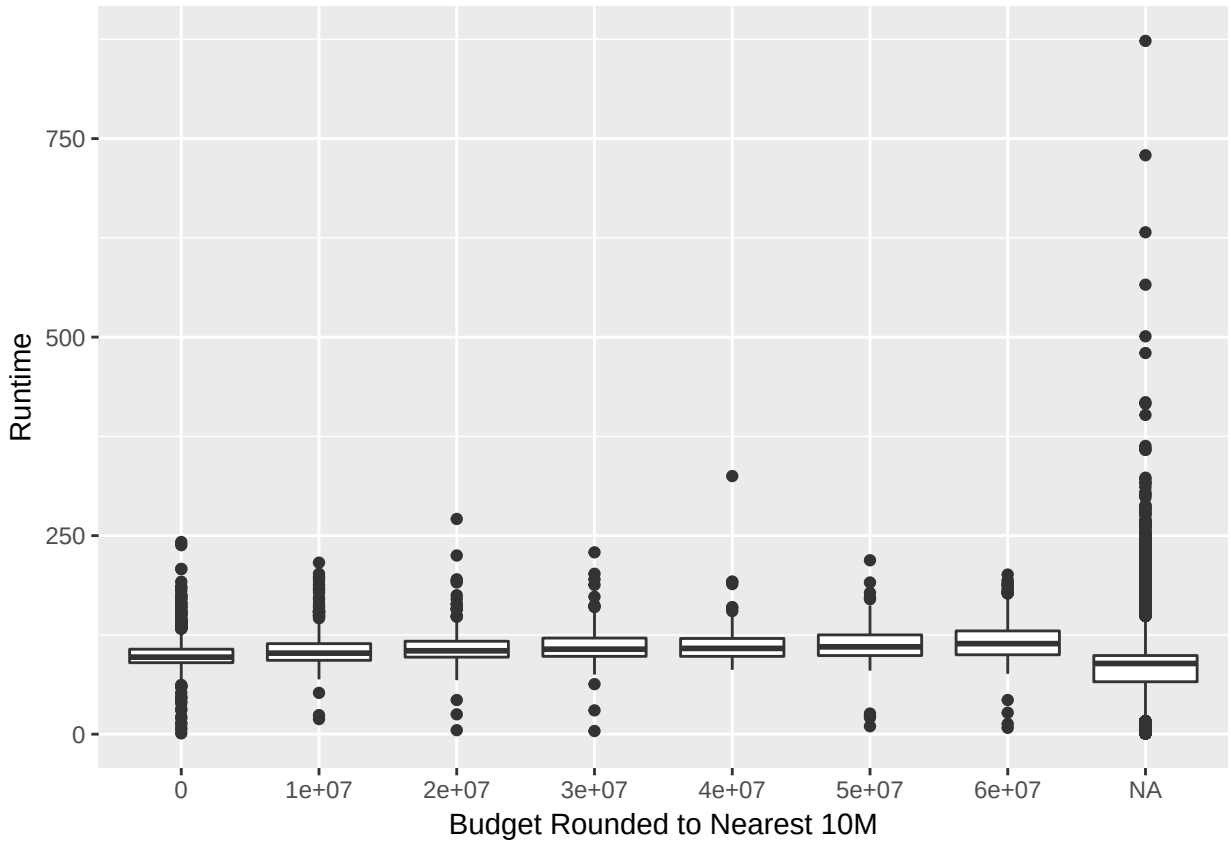
```
  scale_x_discrete("decade")
```

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




```
# Now we need to show distribution of runtime by Budget. Do a boxplot grouped by budget
df$budget_rounded = round_any(df$Budget, 10000000, f = floor)
df$budget_rounded[df$Budget >= 60000000] = 60000000
df$budget_rounded = as.character(df$budget_rounded)
df$budget_rounded[df$budget_rounded == '600000000'] = 'Over 60M'
ggplot(df, aes(as.factor(budget_rounded), Runtime)) +
  geom_boxplot() +
  scale_x_discrete("Budget Rounded to Nearest 10M")
```

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




```
ggpairs(df, columns=c('Runtime', 'Budget', 'Year'))
```

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

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 35480 rows containing missing values
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 751 rows containing missing values
```

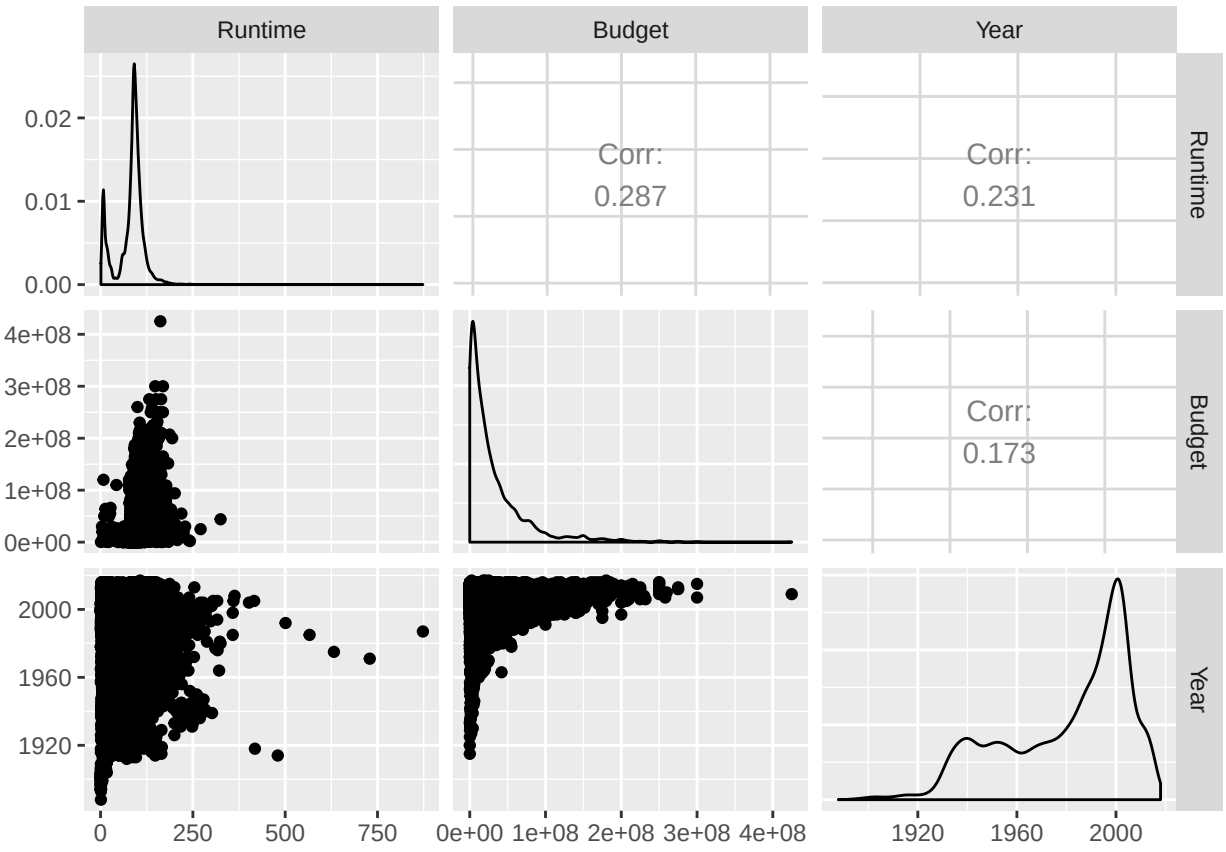
```
## Warning: Removed 35480 rows containing missing values (geom_point).
```

```
## Warning: Removed 35442 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 751 rows containing missing values (geom_point).
```

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



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:

Looking in the aggregate, it seems like the vast majority of runtimes are under 4 hours, which makes sense. There is a clear relationship between run time and year, looking a bit like a sigmoid function. This makes sense as very early years, it was probably technologically prohibitive to create long movies. As technology improved, runtimes increased, but there's only so much time you can expect to hold an audience captive. In general, it looks like higher the budget, higher the Runtime, but the relationship is not super strong. The correlation between Budget and Runtime is 0.287.

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

# All unique genres in the dataframe
print(unique(unlist(strsplit(df$Genre, ', '))))
```

```
##  [1] "Documentary" "Biography"   "Romance"     "Short"       "Thriller"
##  [6] "Drama"       "War"         "Comedy"      "Horror"      "Sci-Fi"
## [11] "Adventure"   "Family"      "History"     "Crime"       "Action"
## [16] "Music"       "Mystery"     "Fantasy"     "Sport"       "Animation"
## [21] "Musical"     "N/A"         "Talk-Show"   "Adult"       "Western"
## [26] "Film-Noir"   "Reality-TV"  "News"        "Game-Show"
```

```
# Example of how to one-hot encode: https://stackoverflow.com/questions/39778387/r-dataframe-one-hot-en

df = cbind(df, mtabulate(strsplit(df$Genre, ", ")))

cols = c('Genre', 'Action', 'Adult', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime', 'Documen

# Remove column from dataframe
df$Genre = NULL
```

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

# See https://stackoverflow.com/questions/20679702/r-find-column-with-the-largest-column-sum as referen
cols = setdiff(cols, 'Genre')
colCount = colSums(df[cols])
topTenIds = order(colCount,decreasing=TRUE)[1:10]
topTenGenres = names(df[cols][topTenIds])

# The top 10 most common genres are
print(topTenGenres)
```

```
##  [1] "Drama"       "Comedy"      "Short"       "Romance"     "Action"
##  [6] "Crime"       "Thriller"    "Documentary" "Adventure"   "Animation"
```

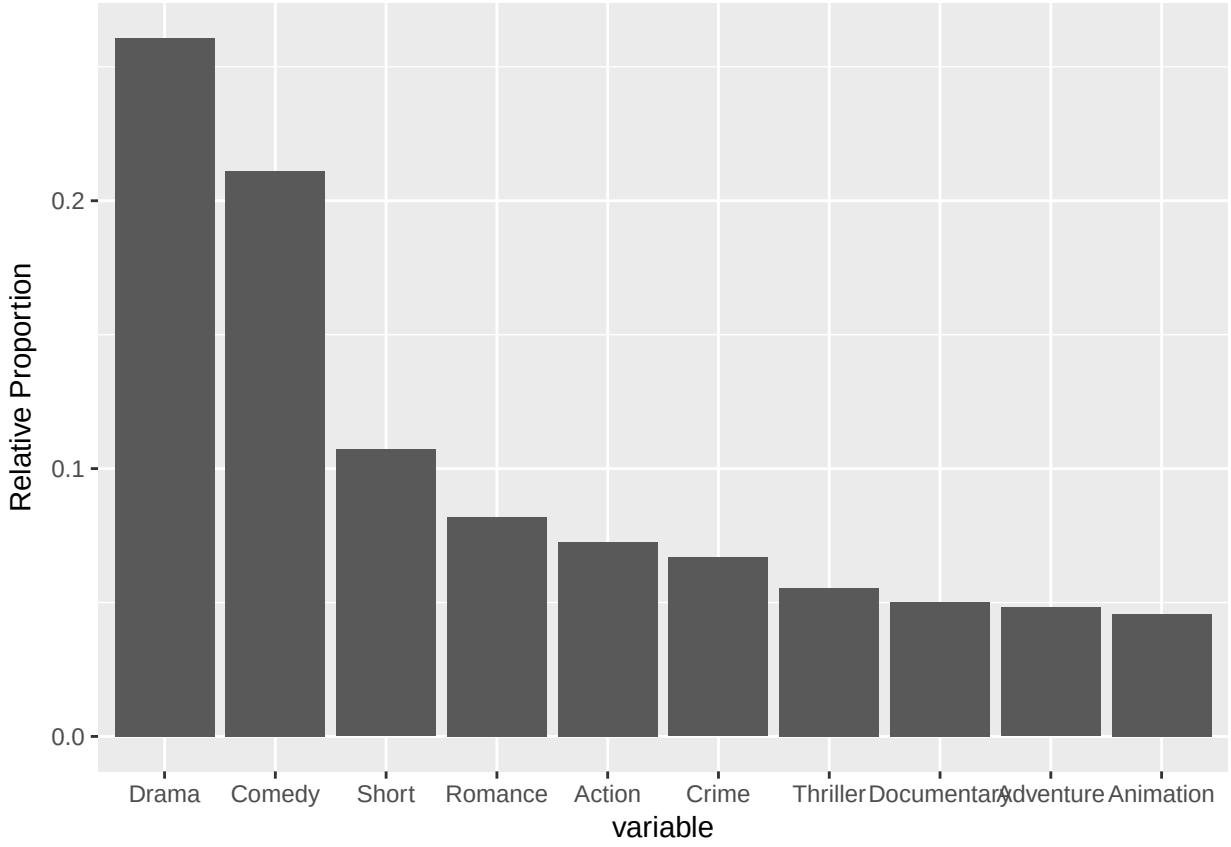
```
# Probably need to do some manual "stacking" of dataframes by each of the top 10 most common genres.
# Note that I was somewhat confused on what "relative proportions of movies" meant. I found this helpfu
# https://piazza.com/class/j6gt7ycx6nk145?cid=396
# Suppose you have ten ice creams.  Let's say that four of them are Chocolate ice cream.  What is the r
cols3 = c('Title', 'Runtime', topTenGenres)
df3 = df[cols3]
df3.long = melt(df3, id.vars=c('Title', 'Runtime'))

# Removing rows where value is 0 to answer the question
# This is because when value is 0, that means movie is not that genre.
df3.long = df3.long[apply(df3.long['value'],1,function(z) !any(z==0)),]
print(summary(df3.long))
```

```
##     Title              Runtime          variable         value
##  Length:60821       Min.   :  1.00   Drama  :15859   Min.   :1
##  Class :character   1st Qu.: 65.00   Comedy :12849   1st Qu.:1
##  Mode  :character   Median : 90.00   Short  : 6516   Median :1
##                     Mean   : 79.75   Romance: 4975   Mean   :1
##                     3rd Qu.:102.00   Action : 4413   3rd Qu.:1
##                     Max.   :873.00   Crime  : 4062   Max.   :1
##                     NA's   :809      (Other):12147
```

```
print(head(df3.long))
```

```
##                         Title Runtime variable value
## 4                   5th World      75    Drama     1
## 12 All the Stage Is a World        89    Drama     1
## 18                        Amu      102    Drama     1
## 19               And I Lived      107    Drama     1
## 20             Anthony Zimmer      89    Drama     1
## 22              Ash Wednesday      99    Drama     1
```

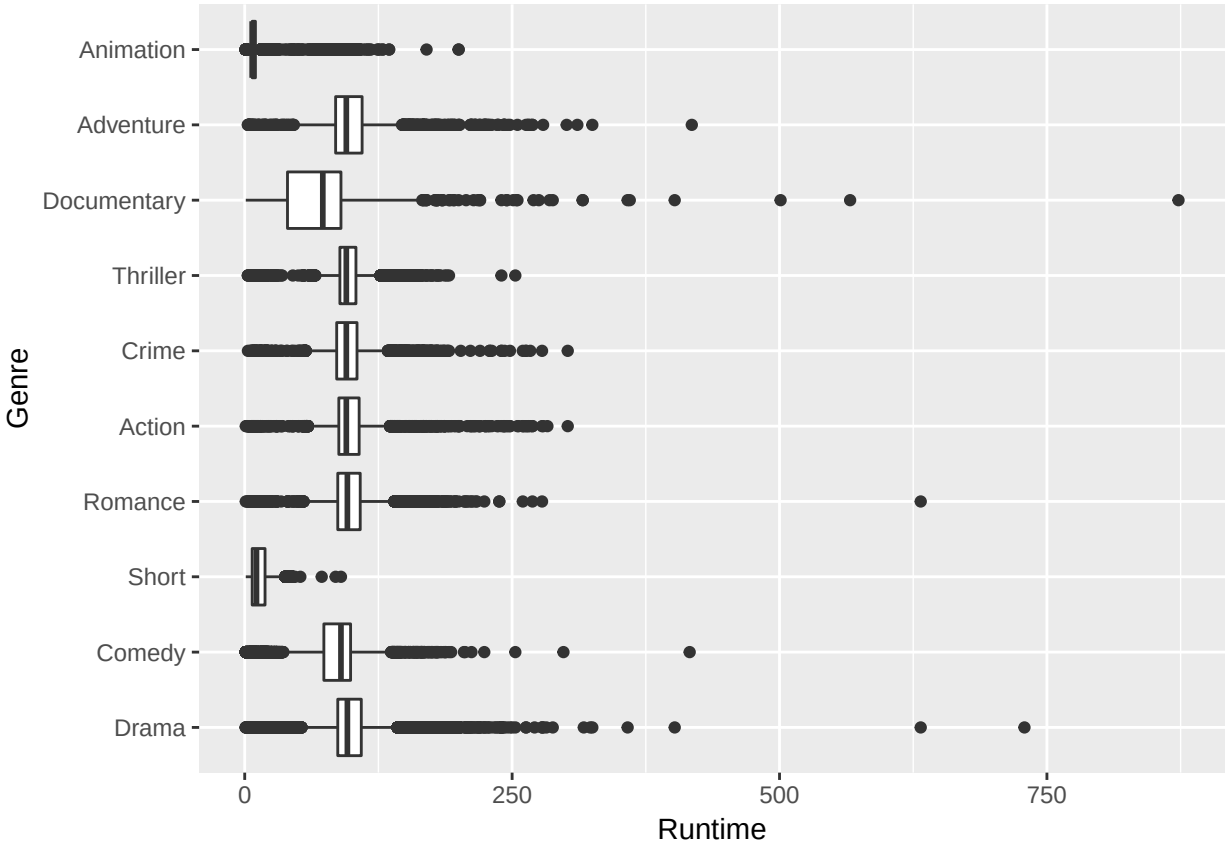
```
# Finally plot relative proportions
# https://sebastiansauer.github.io/percentage_plot_ggplot2_V2/
ggplot(df3.long, aes(x=variable)) +
  geom_bar(aes(y = (..count..)/sum(..count..))) +
  ylab('Relative Proportion')
```



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
# ggplot(df3.long, aes(x=Runtime)) +
#   geom_density(aes(color=variable, group=variable)) +
#   ggtitle('Distribution of Runtimes by Movie Genre')

ggplot(df3.long, aes(as.factor(variable), Runtime)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Genre")
```

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



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

A: One thing that immediately stood out to me was that Animation and "Short" films tend to be the shortest, which is expected as there are probably many short animation films (and "short" films are likely to be short). I was actually surprised to see that most animation films are very very short.

One that that was unexpected to me was that the distribution of runtimes for Thriller, Crime, Action, and Romance films seem to be very similar.

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
library(lubridate)  # Used to extract year from Released
```

```
## Loading required package: methods
```

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

## The following object is masked from 'package:plyr':
##
##     here

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

```
cols4 = c('Year', 'Date', 'Released', 'Gross')
df4 = df[cols4]
df4$released_year = year(df4$Released)
df4$not_matched = (df4$Year != df4$released_year)

# Number of rows with non-null Gross Value
print(nrow(df[which(!is.na(df$Gross)), ]))
```

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

```
# Printing rows in the original data set
print(nrow(df))
```

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

```
df = df[-which(df4$not_matched == TRUE & is.na(df4$Gross)), ]
print(nrow(df))
```

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

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

A: Out of the 40,000 original rows, I removed rows where `Year != released_year`. I explicitly did not remove any rows where Gross was not equal to `NA`. I did not use `Date` as a criteria since there were so many missing values of `Date`. This resulted in 4957 rows to be removed where there was a non-null `Gross` value.

There are now 35043 rows in the resulting dataset.

5. Explore `Gross` revenue

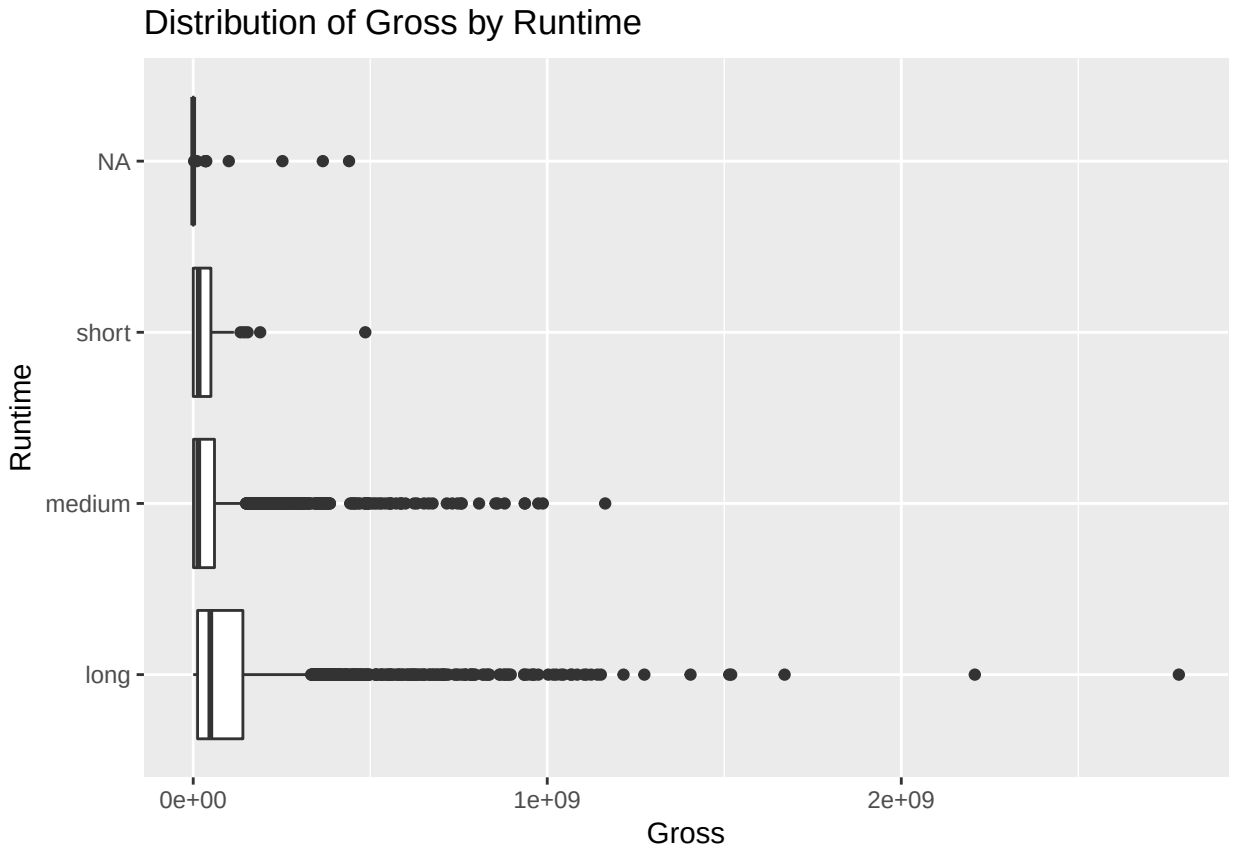
For the commercial success of a movie, production houses want to maximize Gross revenue. Investigate if Gross revenue is related to Budget, Runtime or Genre in any way.

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

```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
# Create three categorical values based on distribution of 25th, and 75 percentiles
df$runtime_categorical = ifelse(df$Runtime <= 68, 'short', ifelse(df$Runtime <= 101, 'medium', 'long'))

#cols = c('Title', 'Gross', 'Runtime', 'runtime_categorical', 'Budget', 'Released', 'Action', 'Adult',

# Just look at top ten most common Genres
cols = c('Title', 'Gross', 'Runtime', 'runtime_categorical', 'Budget', 'budget_rounded')
cols = c(cols, topTenGenres)
df5 = df[cols]

ggplot(df5, aes(as.factor(runtime_categorical), Gross)) +
```

```
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Runtime") +
  ggtitle('Distribution of Gross by Runtime')
```

```
## Warning: Removed 30485 rows containing non-finite values (stat_boxplot).
```

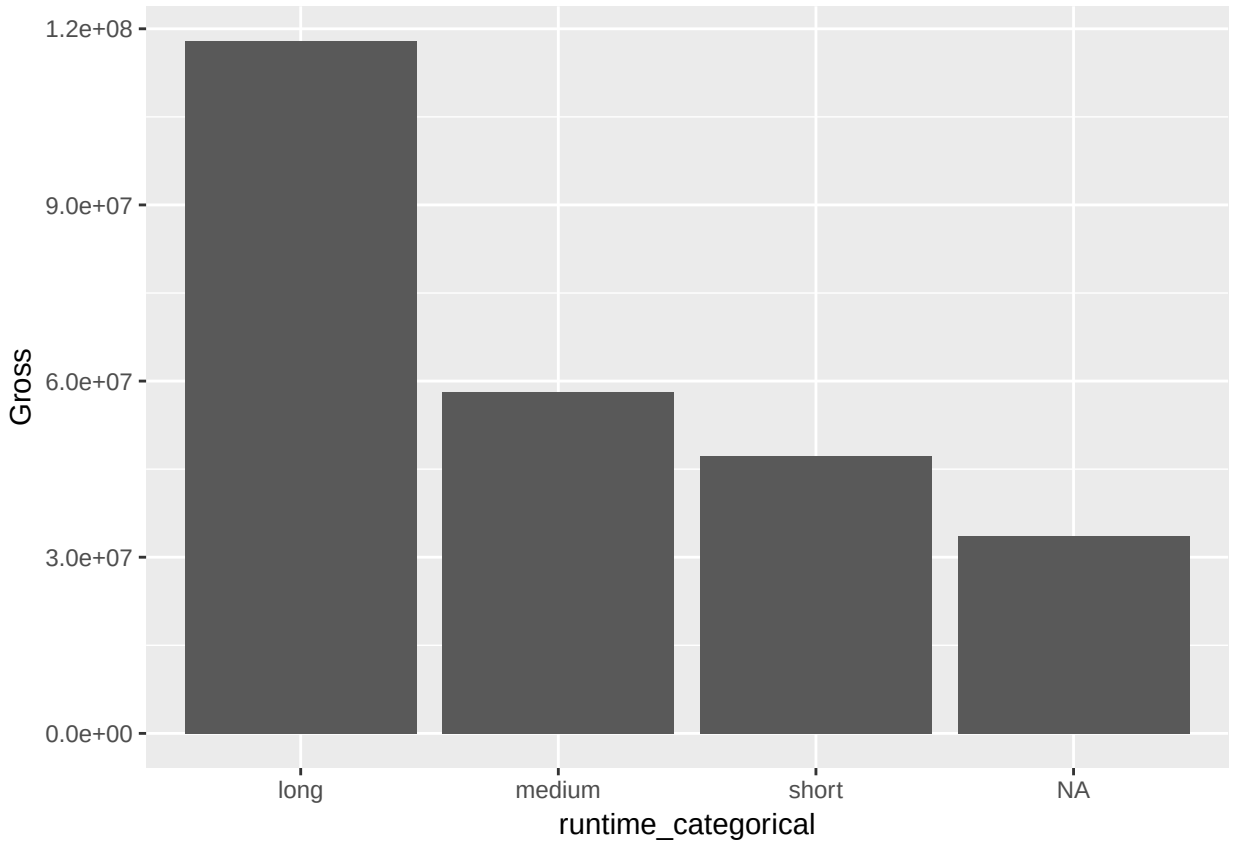



```
ggplot(df5, aes(runtime_categorical, Gross)) +
  geom_bar(stat='summary', fill.y='mean')
```

```
## Warning: Ignoring unknown parameters: fill.y
```

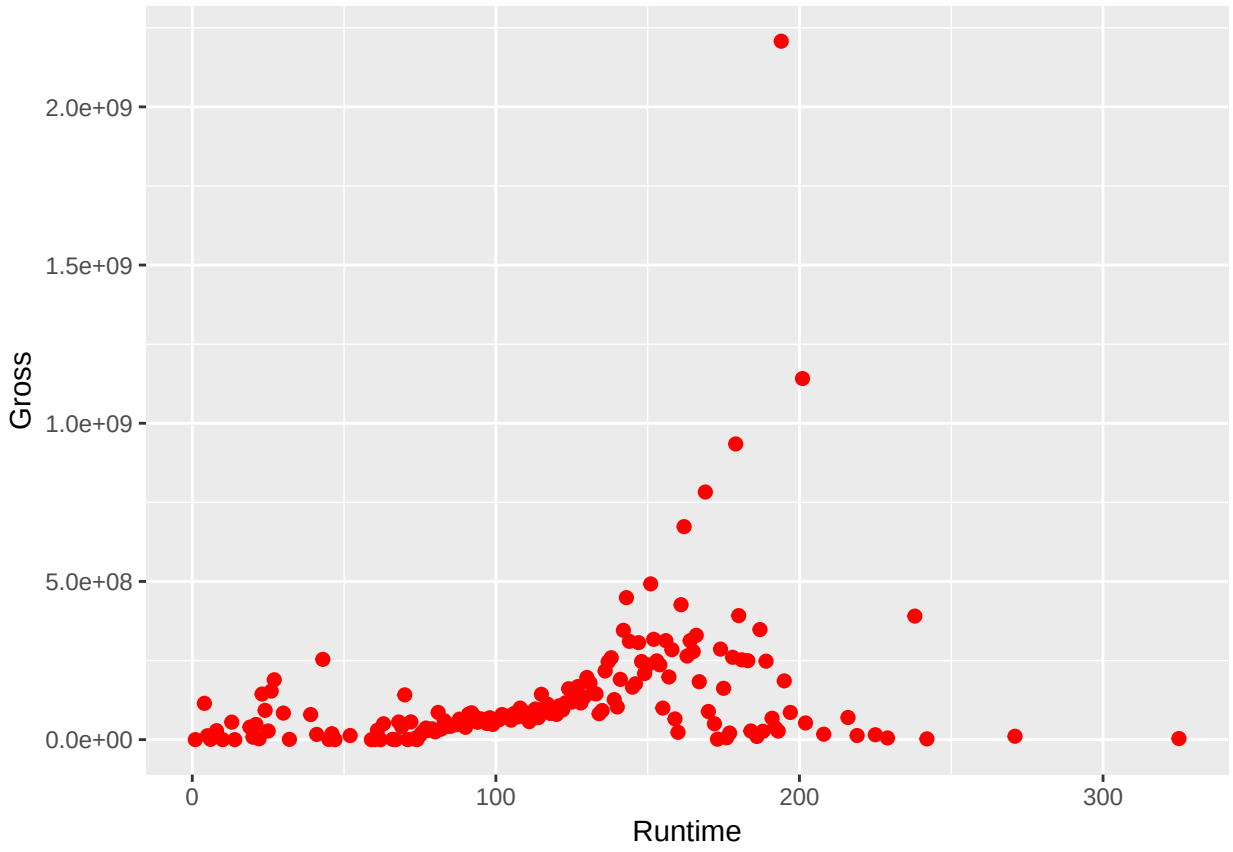
```
## Warning: Removed 30485 rows containing non-finite values (stat_summary).
```

```
## No summary function supplied, defaulting to `mean_se()
```



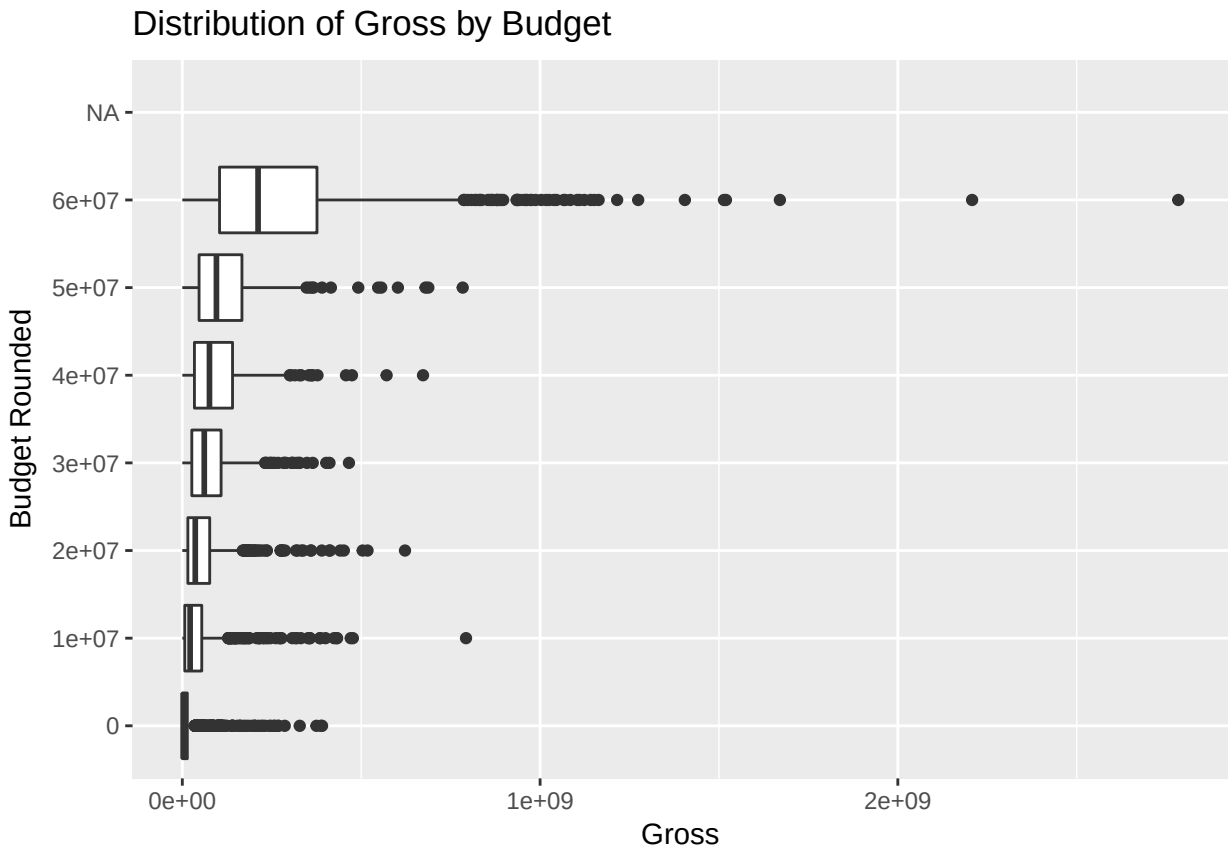

```
ggplot(df, aes(Runtime, Gross)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 30523 rows containing non-finite values (stat_summary).
```



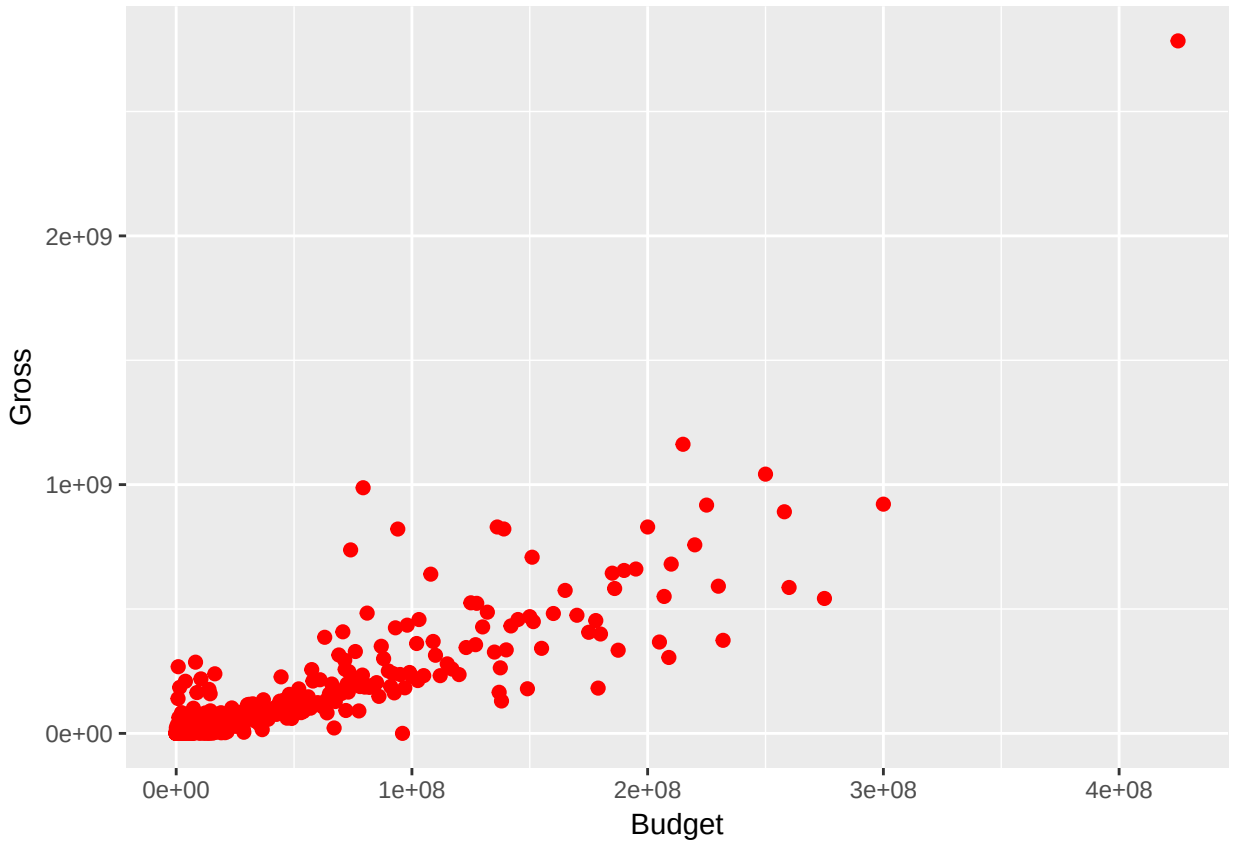

```
ggplot(df5, aes(as.factor(budget_rounded), Gross)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Budget Rounded") +
  ggtitle('Distribution of Gross by Budget')
```

```
## Warning: Removed 30485 rows containing non-finite values (stat_boxplot).
```



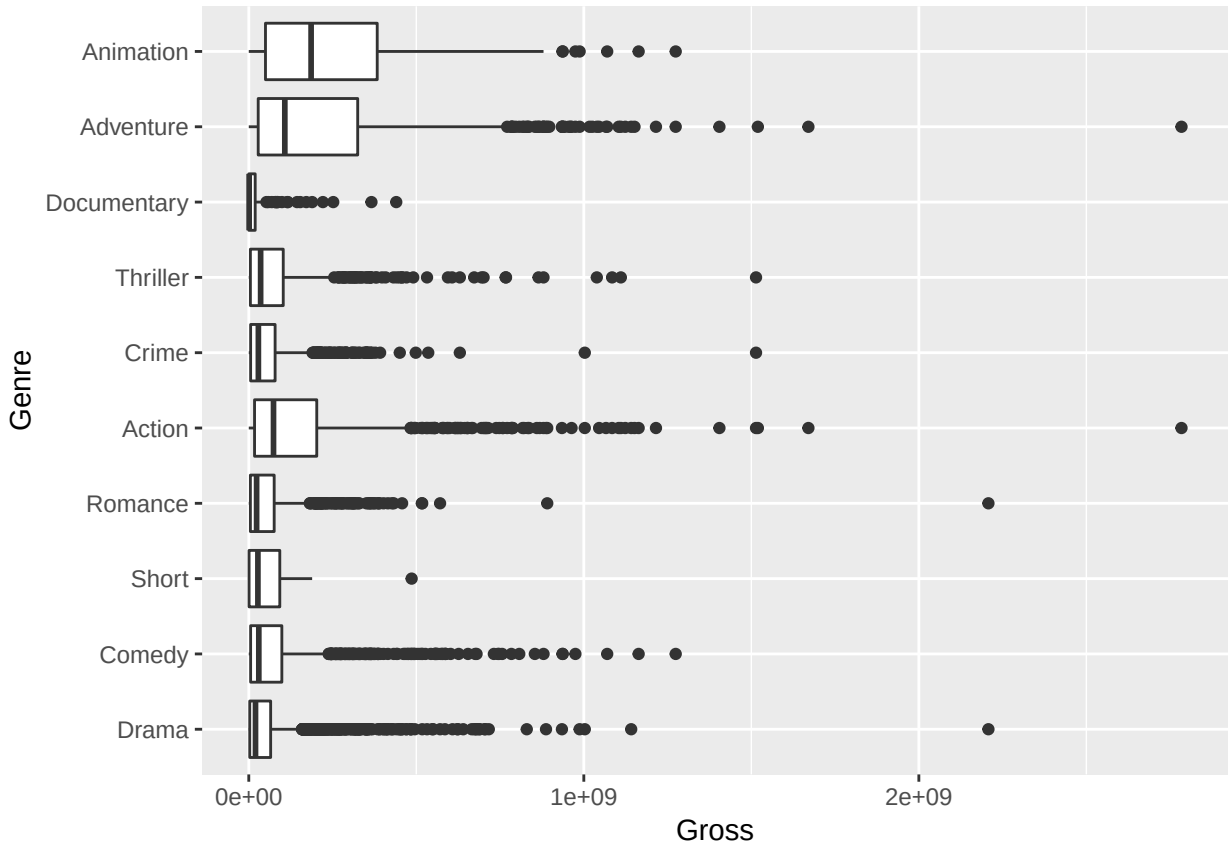

```
ggplot(df, aes(Budget, Gross)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 30485 rows containing non-finite values (stat_summary).
```




```
# Next melt the df to get genre as one column (like in question 3) and plot grouped by genre
df5.long = melt(df5, id.vars=c('Title', 'Gross', 'Runtime', 'runtime_categorical', 'Budget', 'budget_rou

# Removing rows where value is 0 to answer the question
# This is because when value is 0, that means movie is not that genre.
df5.long = df5.long[apply(df5.long['value'], 1, function(z) !any(z==0)),]

ggplot(df5.long, aes(as.factor(genre), Gross)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Genre")
```

```
## Warning: Removed 45172 rows containing non-finite values (stat_boxplot).
```

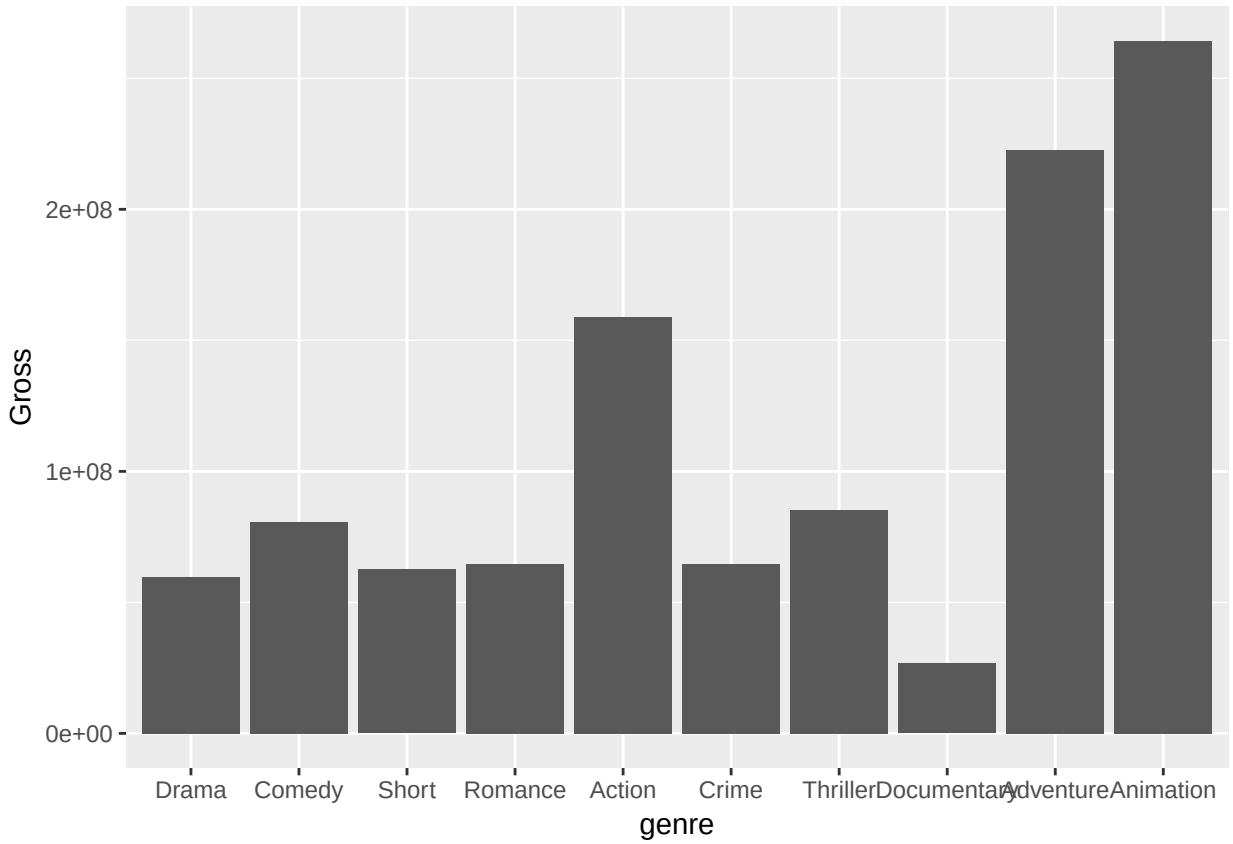



```
ggplot(df5.long, aes(genre, Gross)) +
  geom_bar(stat='summary', fill.y='mean')
```

```
## Warning: Ignoring unknown parameters: fill.y
```

```
## Warning: Removed 45172 rows containing non-finite values (stat_summary).
```

```
## No summary function supplied, defaulting to `mean_se()
```

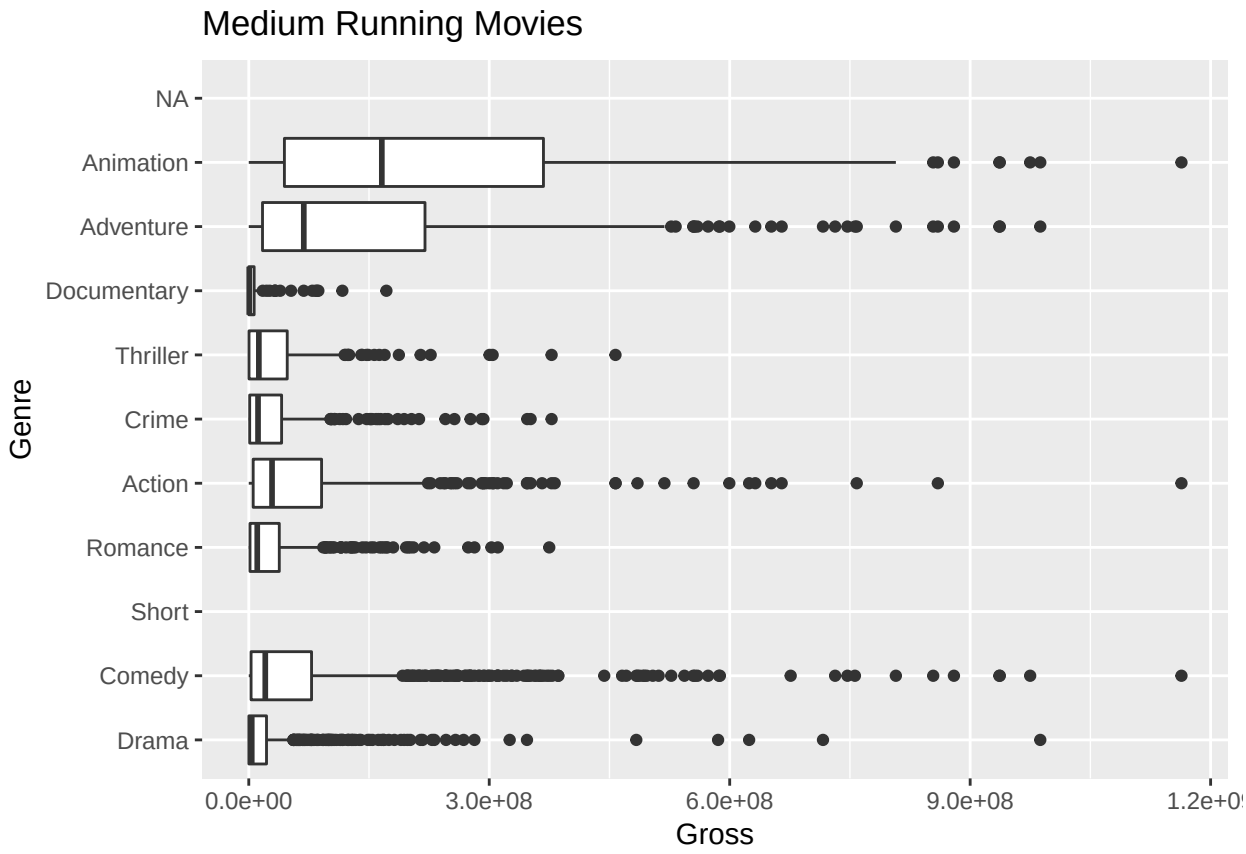



```
ggplot(df5.long[df5.long$runtime_categorical=='short', ], aes(as.factor(genre), Gross)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Genre") +
  ggtitle('Short Running Movies')
```

```
## Warning: Removed 15605 rows containing non-finite values (stat_boxplot).
```

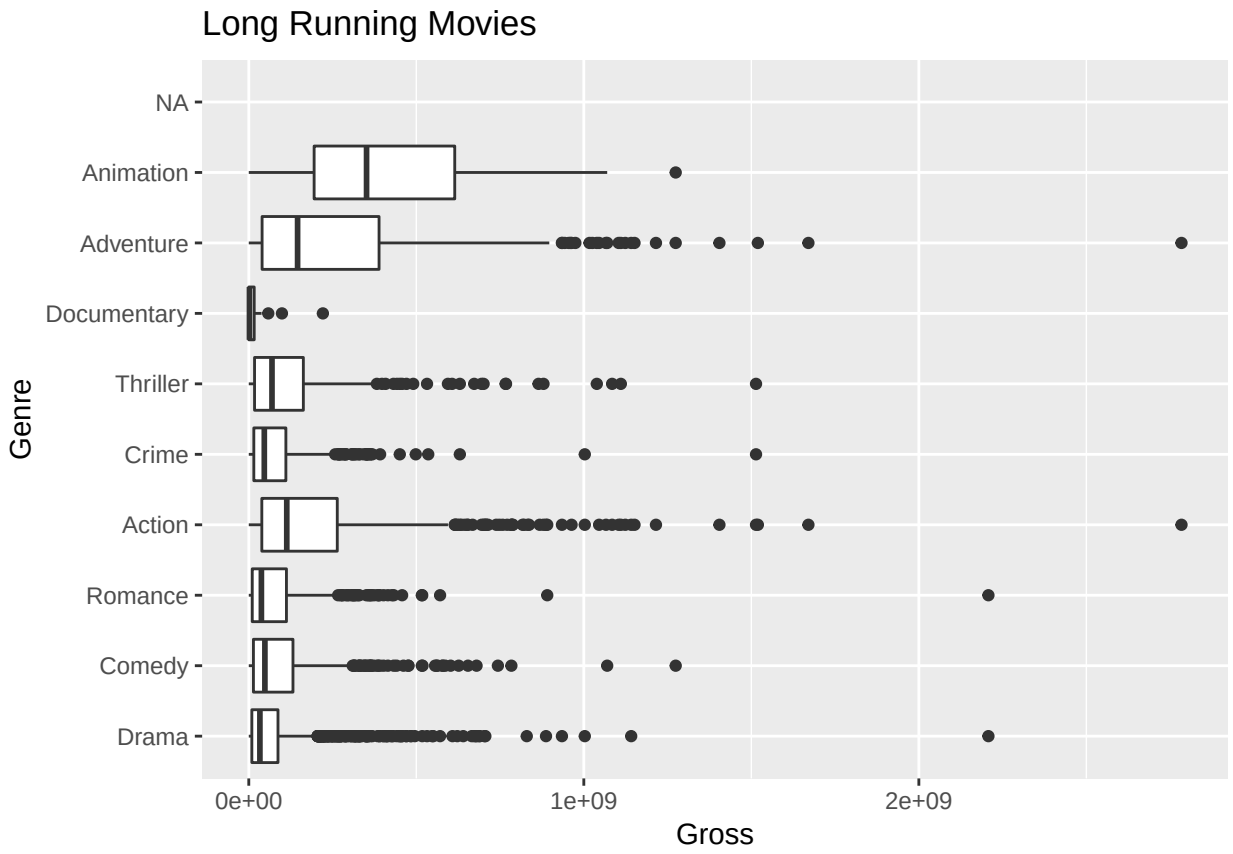

```
ggplot(df5.long[df5.long$runtime_categorical=='medium', ], aes(as.factor(genre), Gross)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Genre") +
  ggtitle('Medium Running Movies')
```

```
## Warning: Removed 21462 rows containing non-finite values (stat_boxplot).
```



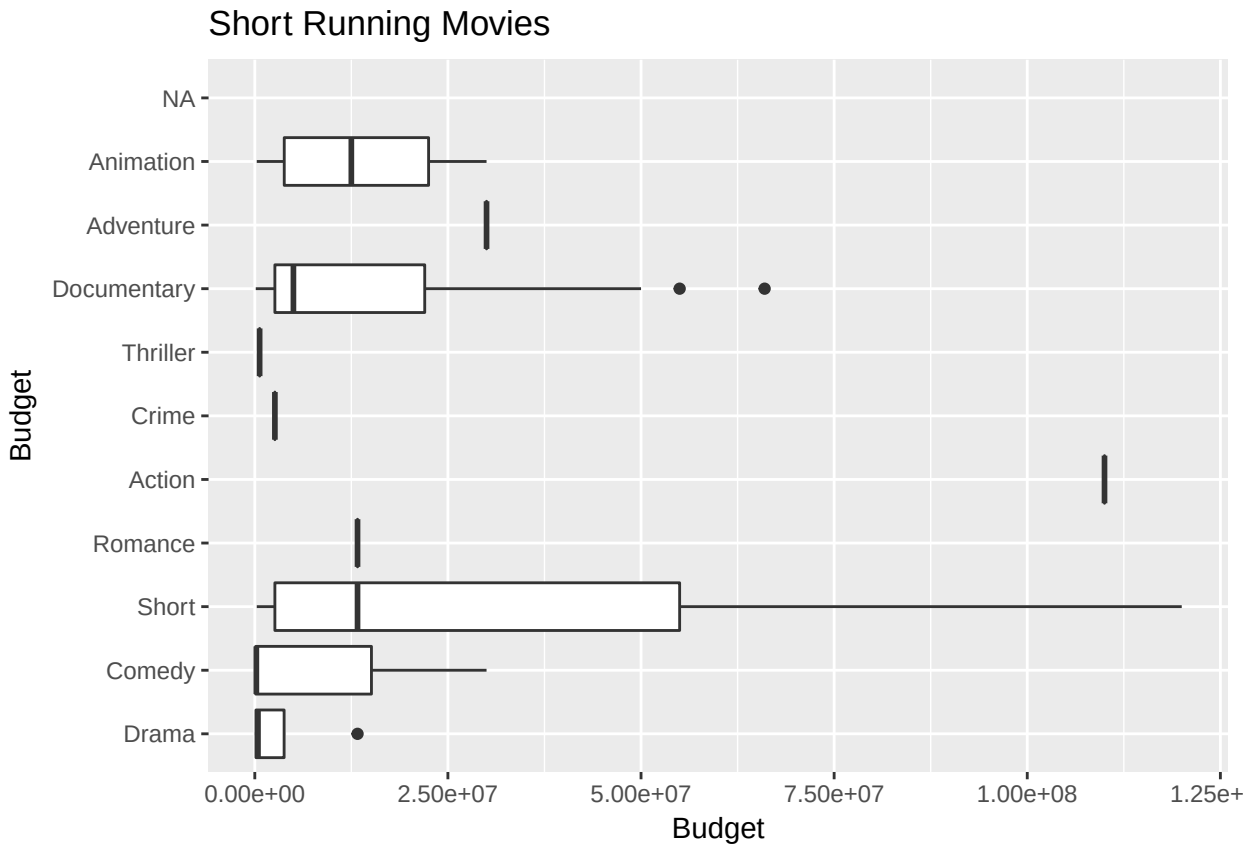

```
ggplot(df5.long[df5.long$runtime_categorical=='long', ], aes(as.factor(genre), Gross)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Genre") +
  ggtitle('Long Running Movies')
```

```
## Warning: Removed 9740 rows containing non-finite values (stat_boxplot).
```



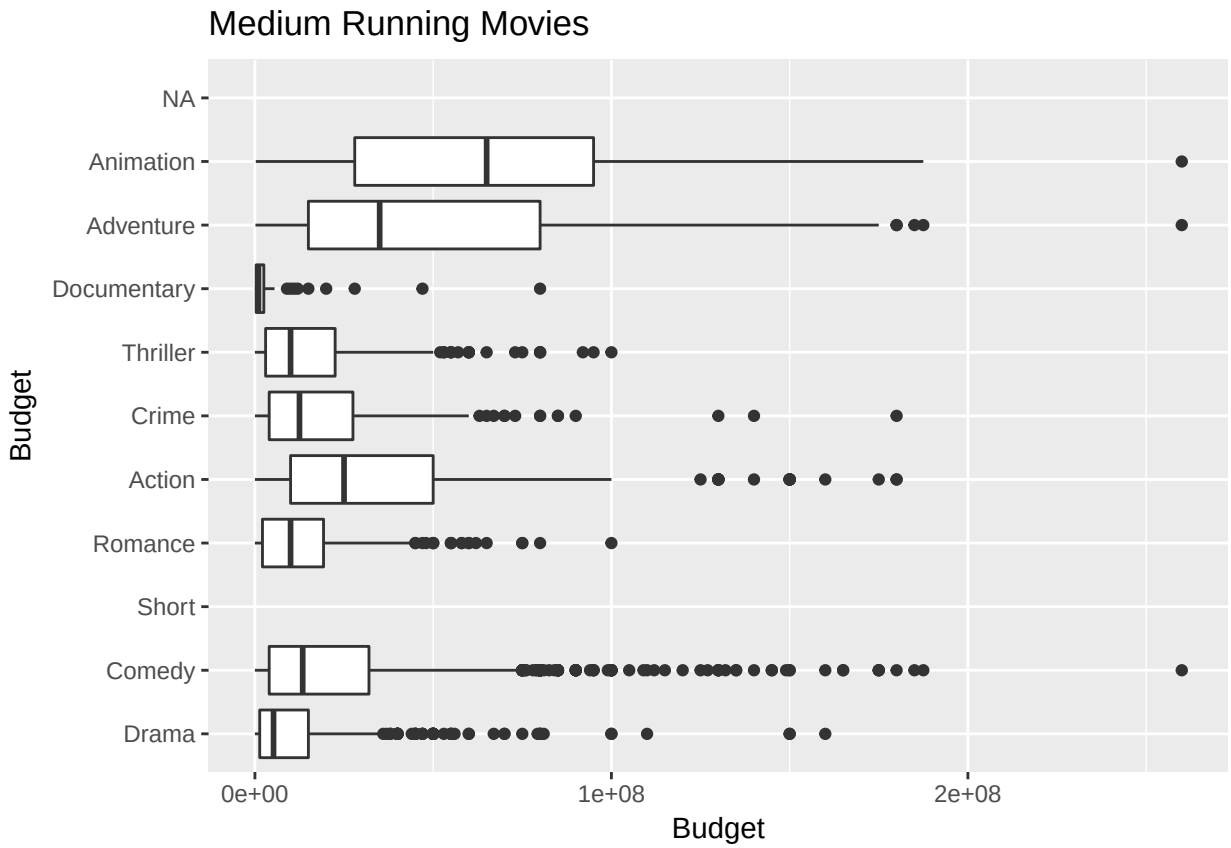

```
ggplot(df5.long[df5.long$runtime_categorical=='short', ], aes(as.factor(genre), Budget)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Budget") +
  ggtitle('Short Running Movies')
```

```
## Warning: Removed 15605 rows containing non-finite values (stat_boxplot).
```



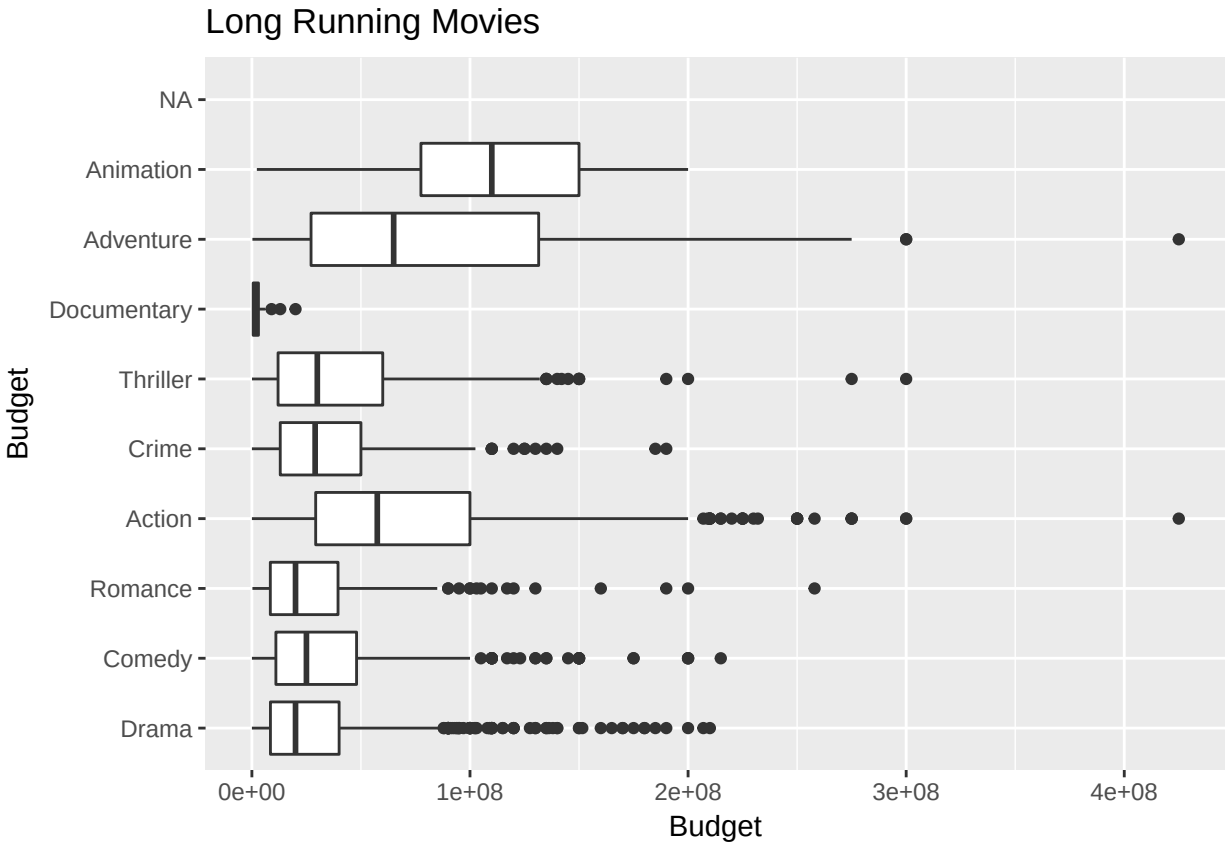

```
ggplot(df5.long[df5.long$runtime_categorical=='medium', ], aes(as.factor(genre), Budget)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Budget") +
  ggtitle('Medium Running Movies')
```

```
## Warning: Removed 21462 rows containing non-finite values (stat_boxplot).
```




```
ggplot(df5.long[df5.long$runtime_categorical=='long', ], aes(as.factor(genre), Budget)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Budget") +
  ggtitle('Long Running Movies')
```

```
## Warning: Removed 9740 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:

Yes, looking at the simple bivariate correlations, Gross Revenue and Runtime are positively correlated. Long movies (greater than 75th percentile) are associated with the highest revenue. It also seems that Budget and Gross Revenue are positively correlated. In terms of Genre, when we look at the top ten Genres, Animation and Adventure Films seem to on average gross the most. I also created boxplots for short running, medium running, and long running movies, by Genre, by mostly Animation and Adventure Films seemed to still Gross the most.

```
# TODO: Investigate if Gross Revenue is related to Release Month
print(summary(df5.long))
```

```
##     Title               Gross              Runtime
##  Length:53641       Min.   :0.000e+00   Min.   :  1.00
##  Class :character   1st Qu.:6.149e+06   1st Qu.: 60.00
##  Mode  :character   Median :3.346e+07   Median : 90.00
##                     Mean   :9.927e+07   Mean   : 78.43
##                     3rd Qu.:1.114e+08   3rd Qu.:102.00
##                     Max.   :2.784e+09   Max.   :632.00
##                     NA's   :45172       NA's   :791
##  runtime_categorical     Budget          budget_rounded
##  Length:53641        Min.   :     1100   Length:53641
##  Class :character    1st Qu.:  7000000   Class :character
##  Mode  :character    Median : 20000000   Mode  :character
##                      Mean   : 35213799
```

```
##                       3rd Qu.: 48000000
##                       Max.   :425000000
##                       NA's   :45172
##      genre            value
##  Drama  :13366   Min.   :1
##  Comedy :11553   1st Qu.:1
##  Short  : 6144   Median :1
##  Romance: 4307   Mean   :1
##  Action : 3919   3rd Qu.:1
##  Crime  : 3599   Max.   :1
##  (Other):10753
```

```
print(head(df5.long))
```

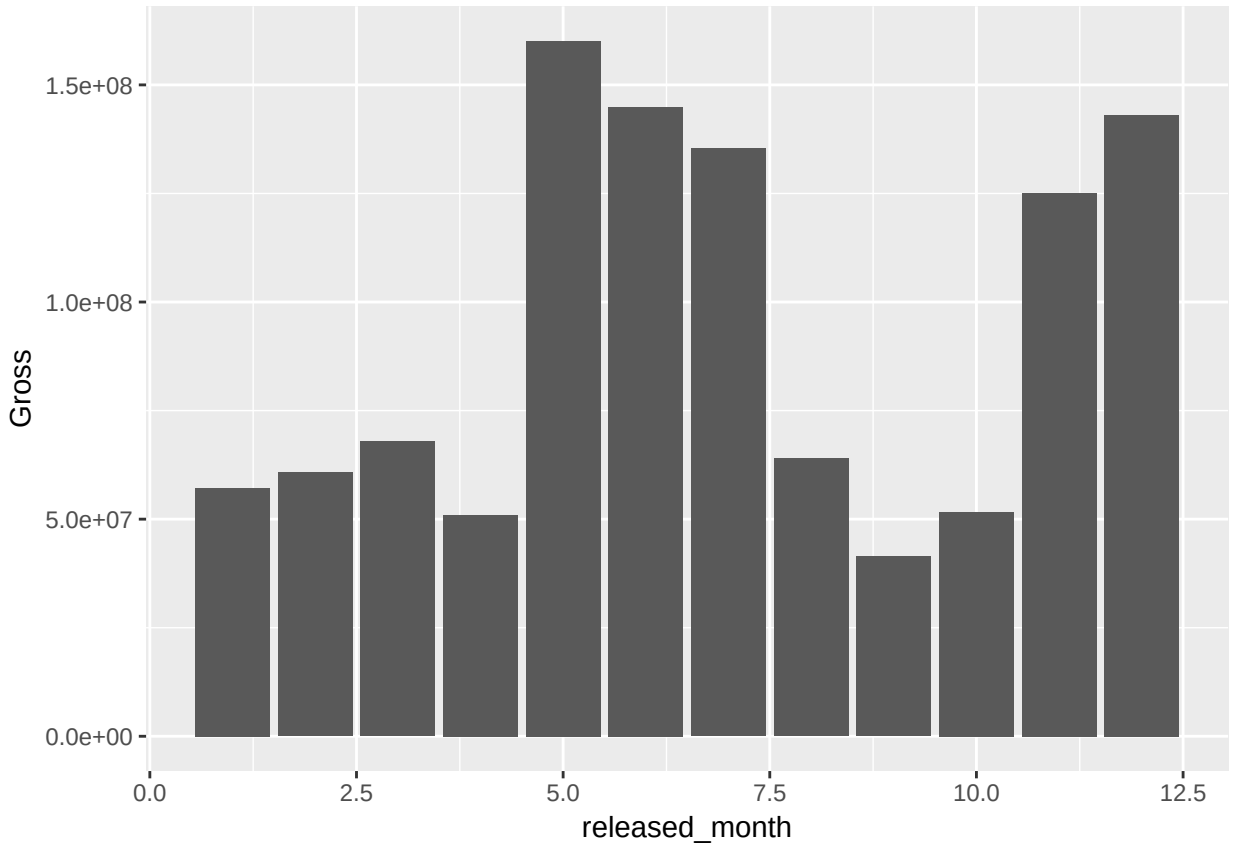
```
##                       Title Gross Runtime runtime_categorical Budget
## 4                 5th World    NA      75              medium     NA
## 11 All the Stage Is a World    NA      89              medium     NA
## 17                      Amu    NA     102                long     NA
## 18              And I Lived    NA     107                long     NA
## 19           Anthony Zimmer    NA      89              medium     NA
## 21   Assault on Precinct 13    NA     109                long     NA
##    budget_rounded genre value
## 4             <NA> Drama     1
## 11            <NA> Drama     1
## 17            <NA> Drama     1
## 18            <NA> Drama     1
## 19            <NA> Drama     1
## 21            <NA> Drama     1
```

```
df$released_month = month(df$Released)

ggplot(df, aes(released_month, Gross)) +
  geom_bar(stat='summary', fill.y='mean')
```

```
## Warning: Ignoring unknown parameters: fill.y
```

```
## Warning: Removed 30530 rows containing non-finite values (stat_summary).
```

```
## No summary function supplied, defaulting to `mean_se()
```



There is definitely a relationship of Gross Revenue and release month. Summer month movies and holiday movies gross significantly more revenue than movies released in other months.

6. Process `Awards` column

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

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

```
# TODO: Convert Awards to 2 numeric columns: wins and nominations
library(stringi)
cols = c('Title', 'Awards', 'Gross')
df6 = df[cols]

awards = stri_extract_all(df6$Awards, regex="\\d+")
award_labels = stri_extract_all(tolower(df6$Awards), regex="win|won|wins|nomin", ignore.case=TRUE)
print(head(awards))
```

```
## [[1]]
## [1] "3"
##
## [[2]]
## [1] NA
```

```
##
## [[3]]
## [1] "2"
##
## [[4]]
## [1] "1"
##
## [[5]]
## [1] NA
##
## [[6]]
## [1] "5"
```

```
print(head(award_labels))
```

```
## [[1]]
## [1] "win"
##
## [[2]]
## [1] NA
##
## [[3]]
## [1] "win"
##
## [[4]]
## [1] "win"
##
## [[5]]
## [1] NA
##
## [[6]]
## [1] "win"
```

```
wins_list = c()
nominations_list = c()

for (i in 1:length(awards)){
    wins = 0
    nominations = 0
    for(j in 1:length(awards[[i]])){
        if(!is.na(award_labels[[i]][j]) & (award_labels[[i]][j] == 'wins' | award_labels[[i]][j] == 'wi
            wins = wins + as.numeric(awards[[i]][j])
        } else if (!is.na(award_labels[[i]][j]) & award_labels[[i]][j] == 'nomin'){
            nominations = nominations + as.numeric(awards[[i]][j])
        }
    }
    wins_list = c(wins_list, wins)
    nominations_list = c(nominations_list, nominations)
}

df$wins = wins_list
df$nominations = nominations_list

# Counting number of rows with at least 1 win or 1 nomination.
sum(ifelse(df$wins==1, 1, 0))
```

```
## [1] 3587
```

```
sum(ifelse(df$nominations==1, 1, 0))
```

```
## [1] 3235
```

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

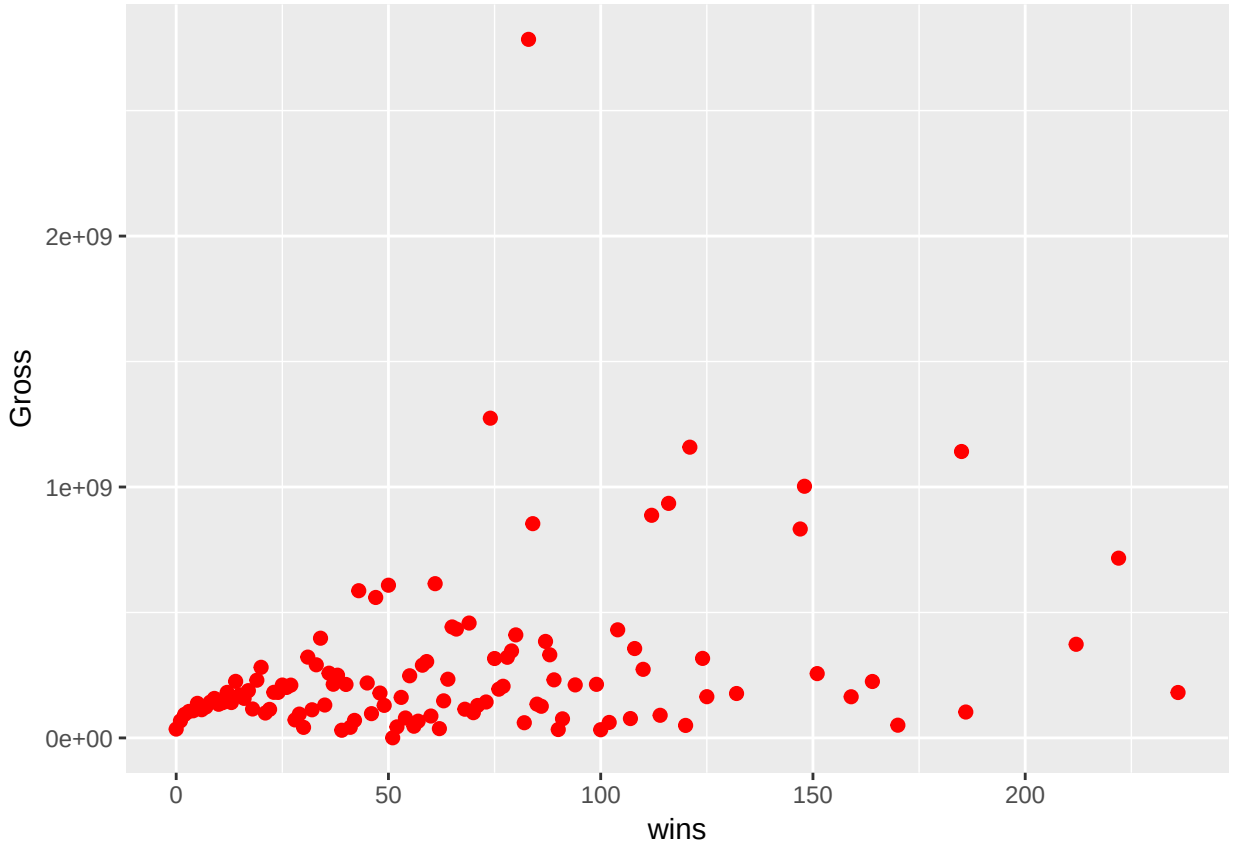
A: To construct the conversion mechanism, I first used `stri_extract_all` to extract all numerical substrings. I then extracted all instances of "win, won, or nomin", after altering the column to be all lower case so I didn't have to worry about that in the regex. I then assigned the numerical to however it corresponded to the position of "win, won, or nomin".

For example, for the movie "12 Years a Slave", the Awards entry reads "Won 3 Oscars. Another 233 wins & 312 nominations." Thus, I first extract `c('won', 'wins', 'nominations')`. Then I extract `c(3, 233, 312)`. Since the first two values correspond to 'won' and 'wins' respectively, the algorithm adds these 2 together for a total of `236` wins. Since the last value corresponds to nominations, the nominations pulled for this movie is `312` nominations.

I'm interpreting the second part of the question as I can choose whether to report how many rows had valid or non-zero wins or nominations, so I am choosing to report how many rows had non-zero wins and how many rows had non-zero nominations. The number of rows with non-zero wins is 3587 and the number of rows with non-zero nominations is 3235.

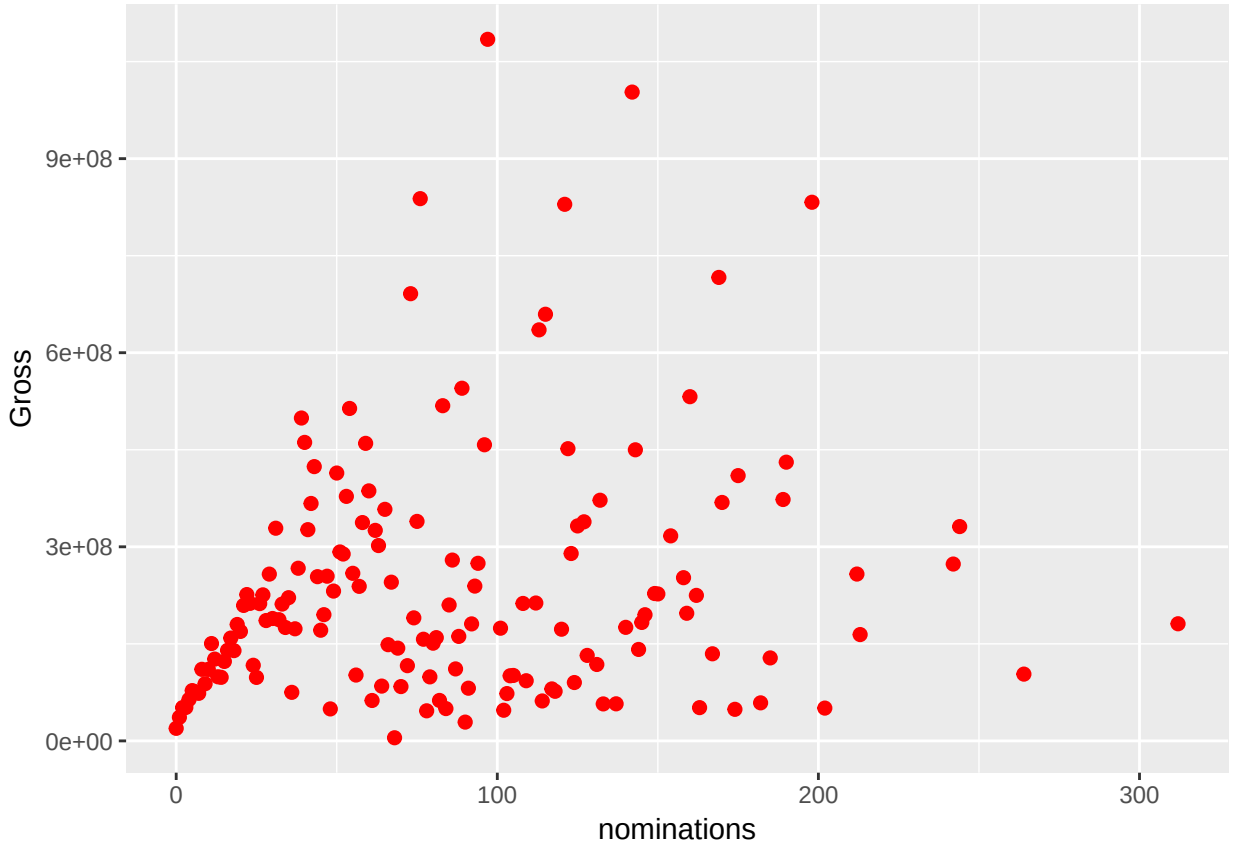
```
# TODO: Plot Gross revenue against wins and nominations
ggplot(df, aes(wins, Gross)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 30485 rows containing non-finite values (stat_summary).
```



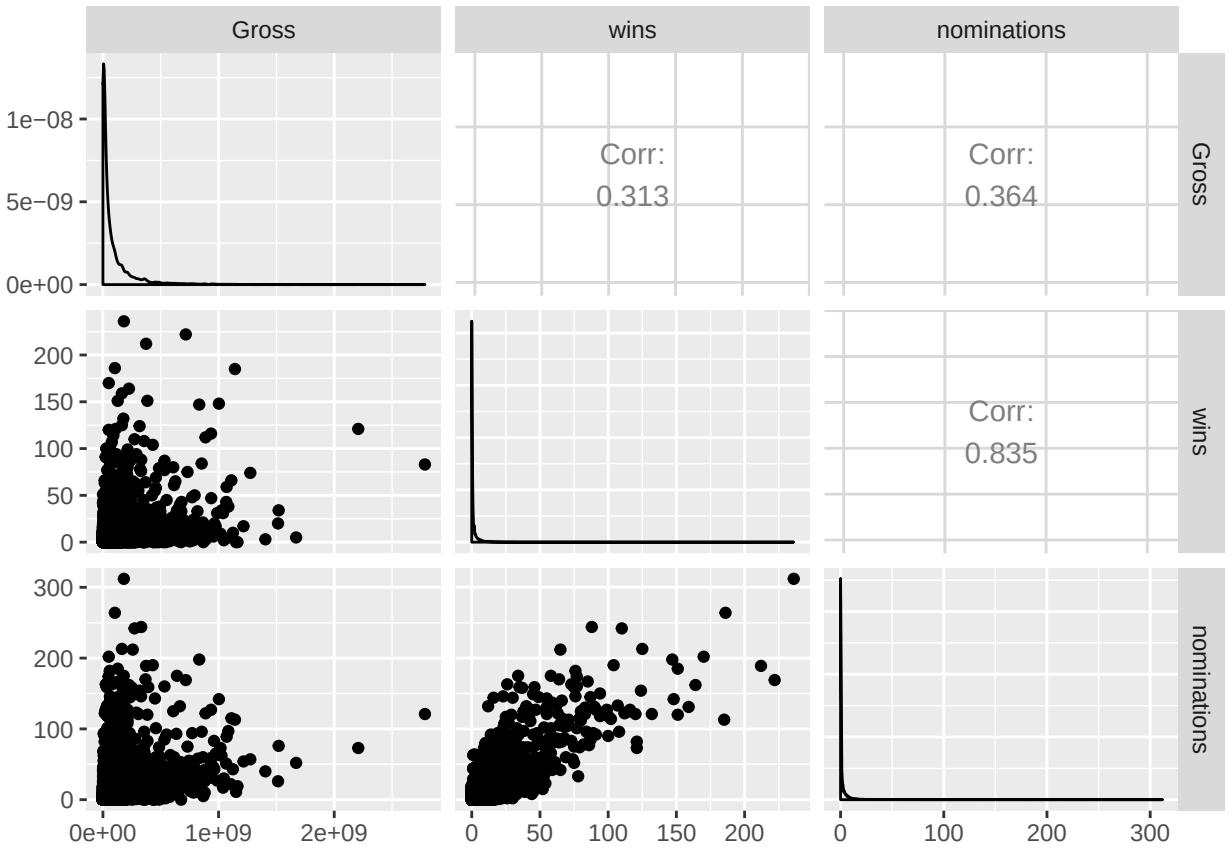
```
ggplot(df, aes(nominations, Gross)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 30485 rows containing non-finite values (stat_summary).
```




```
ggpairs(df, columns=c('Gross', 'wins', 'nominations'))
```

```
## Warning: Removed 30485 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30485 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 30485 rows containing missing values

## Warning: Removed 30485 rows containing missing values (geom_point).

## Warning: Removed 30485 rows containing missing values (geom_point).
```



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

A: Surprisingly, there does not seem to be that strong of a relationship between gross revenue and awards won or nominated. If we look at the `ggpairs` graph, we can see that the correlation between wins and nominations vs Gross is 0.313 and 0.364, respectively. While the positive correlation is expected, I would have thought it would be more strongly positive.

7. Movie ratings from IMDb and Rotten Tomatoes

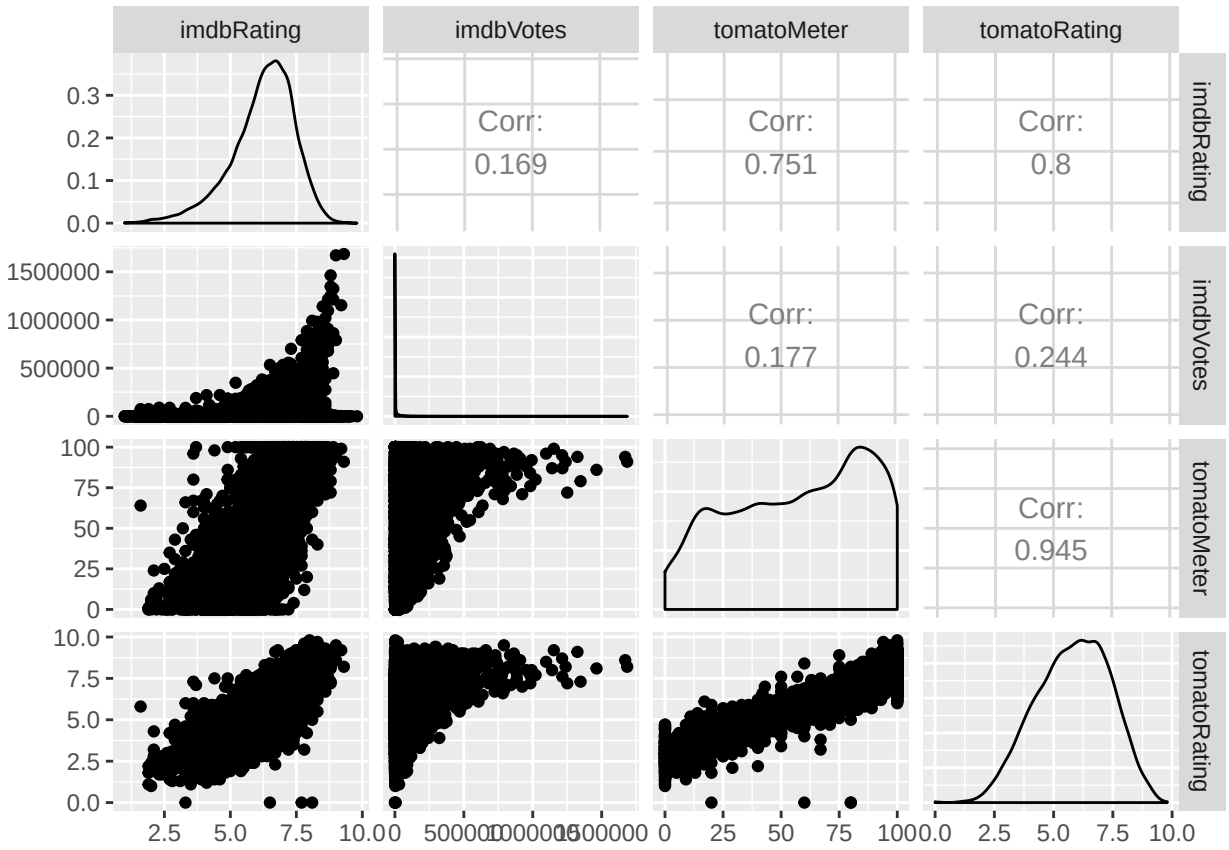
There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by several variables pre-fixed by `tomato`). Read up on such ratings on the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

Investigate the pairwise relationships between these different descriptors using graphs.

```
library(GGally)

# TODO: Illustrate how ratings from IMDb and Rotten Tomatoes are related
cols = c('imdbRating', 'imdbVotes', 'tomatoMeter', 'tomatoImage', 'tomatoRating', 'tomatoReviews',
         'tomatoFresh', 'tomatoRotten', 'tomatoConsensus', 'tomatoUserMeter', 'tomatoUserRating',
         'tomatoUserReviews')
df7 = df[cols]

ggpairs(df7, columns=c('imdbRating', 'imdbVotes', 'tomatoMeter', 'tomatoRating'), aes())
```

```
## Warning: Removed 1123 rows containing non-finite values (stat_density).
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 1157 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 26843 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 26850 rows containing missing values

## Warning: Removed 1157 rows containing missing values (geom_point).

## Warning: Removed 1157 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 26844 rows containing missing values

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 26851 rows containing missing values

## Warning: Removed 26843 rows containing missing values (geom_point).

## Warning: Removed 26844 rows containing missing values (geom_point).

## Warning: Removed 26841 rows containing non-finite values (stat_density).

## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 26848 rows containing missing values

## Warning: Removed 26850 rows containing missing values (geom_point).

## Warning: Removed 26851 rows containing missing values (geom_point).

## Warning: Removed 26848 rows containing missing values (geom_point).

## Warning: Removed 26848 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 off, it's interesting to note that we have significantly more data of IMDB ratings than Rotten tomatoes. In our data, we have about 1100 movies with missing IMDB ratings. However, we have close to 30,000 movies with missing Rotten Tomatoes ratings.

In the plot above we plot the `ggpairs` of `imdbRating`, `tomatoMeter`, and `tomatoRating`. Notice that the correlation between `tomatoMeter` and `tomatoRating` is very high, at 0.94. This makes sense as it's from the same source. The correlation between `tomatoMeter` and `imdbRating` is a bit lower, at around 0.746 vs 0.794 which is the correlation between `tomatoRating` and `imdbRating`.

Looking at the summary of the variables as well, it looks like `imdbRating` and `tomatoRating` are rated on similar scales. `imdbRating` looks like it's rated on a scale of 1 to 10, while `tomatoRating` looks like it's rated on a scale of 0 to 9.8. However, `tomatoMeter` looks like it's scaled from 0 to 100.

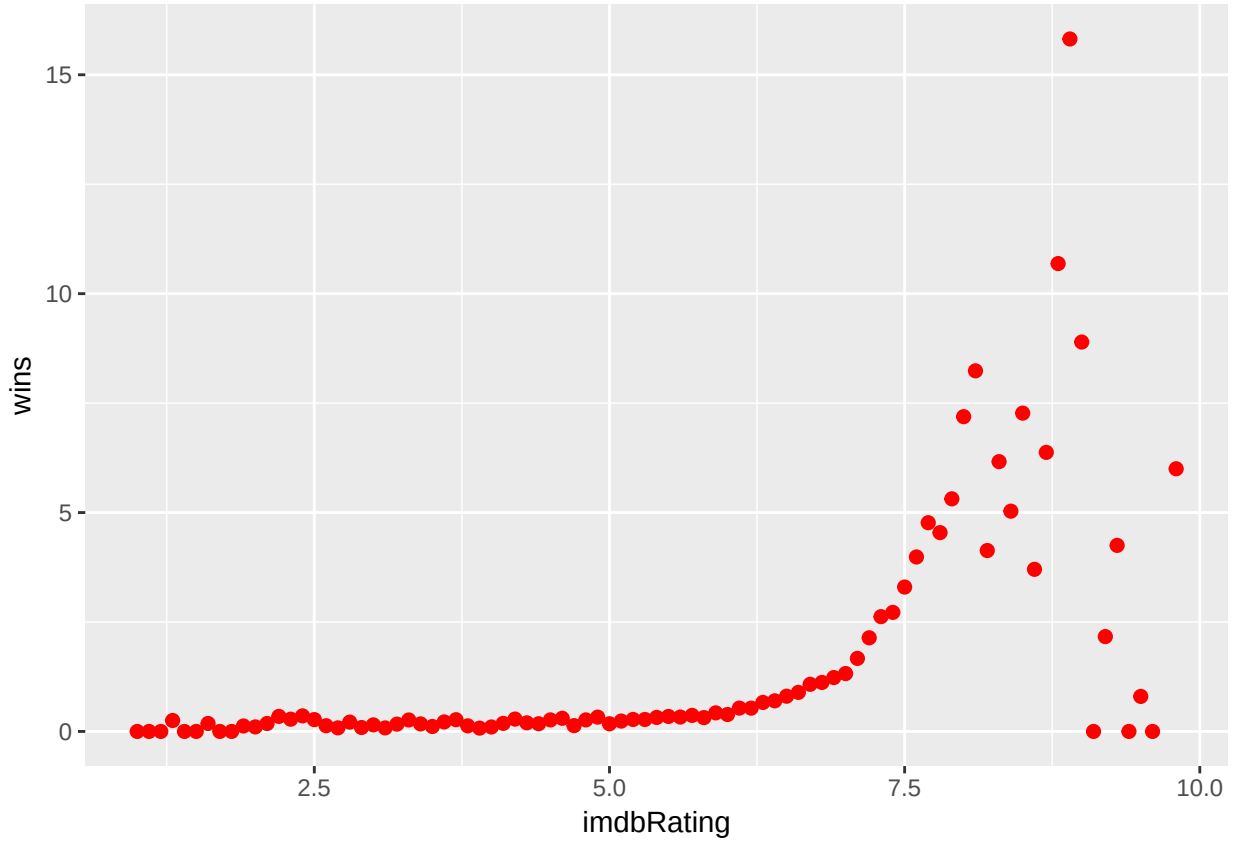
8. Ratings and awards

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).

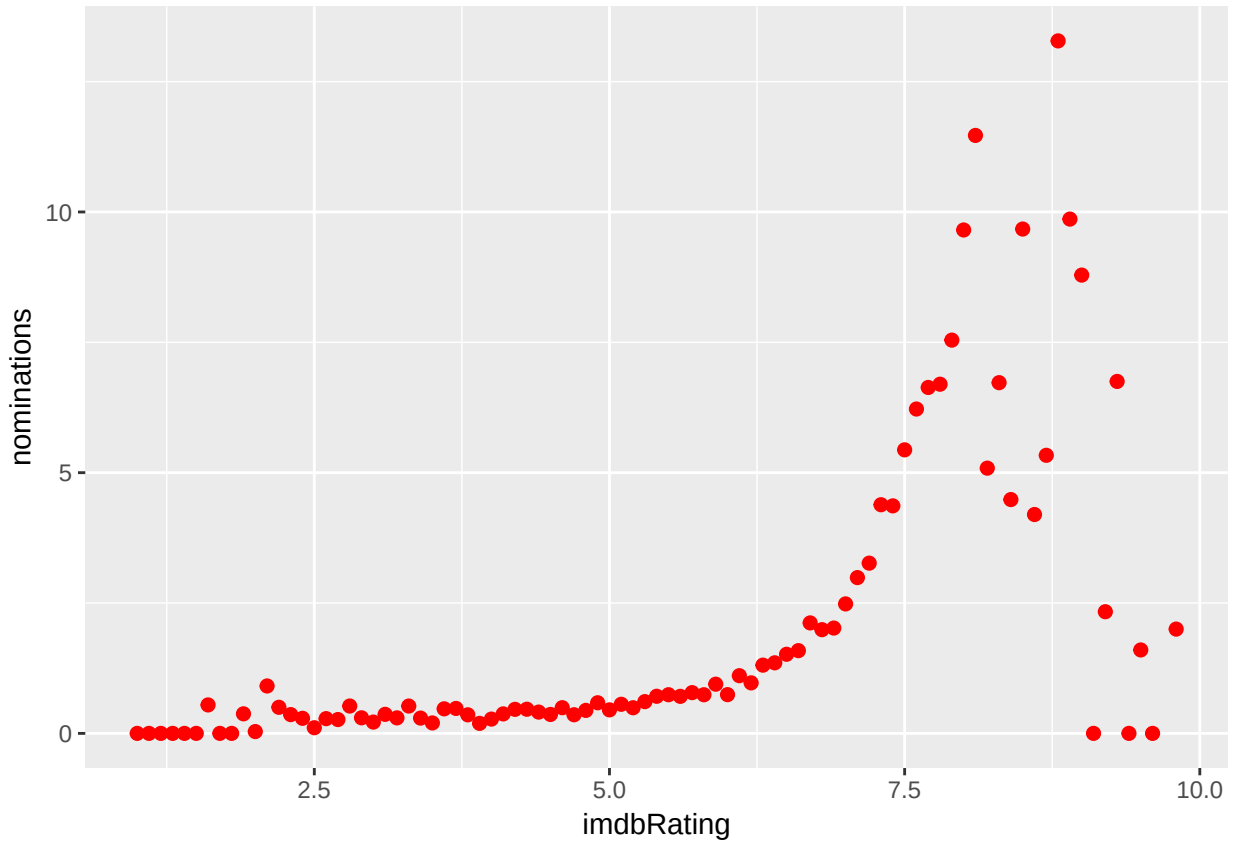
```
# TODO: Show how ratings and awards are related
ggplot(df, aes(imdbRating, wins)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 1123 rows containing non-finite values (stat_summary).
```



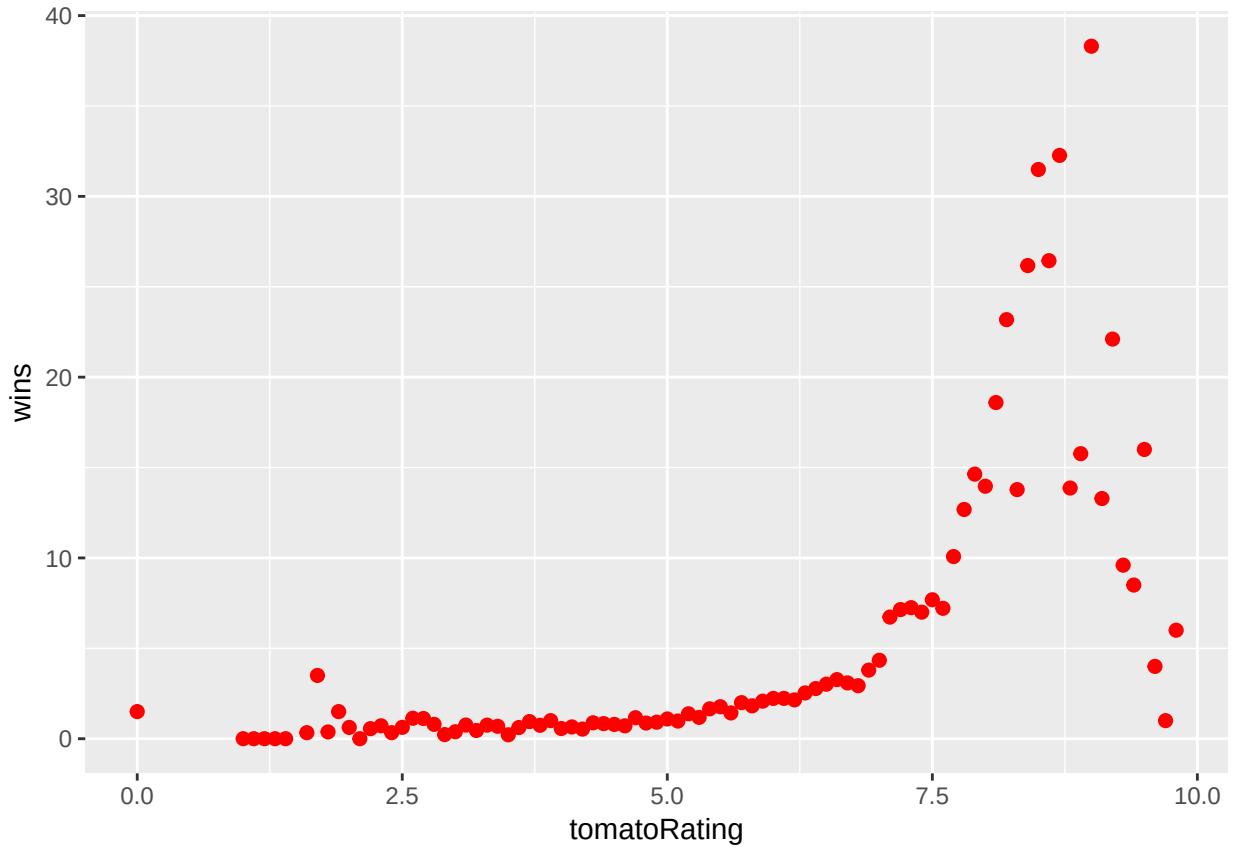

```
ggplot(df, aes(imdbRating, nominations)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 1123 rows containing non-finite values (stat_summary).
```



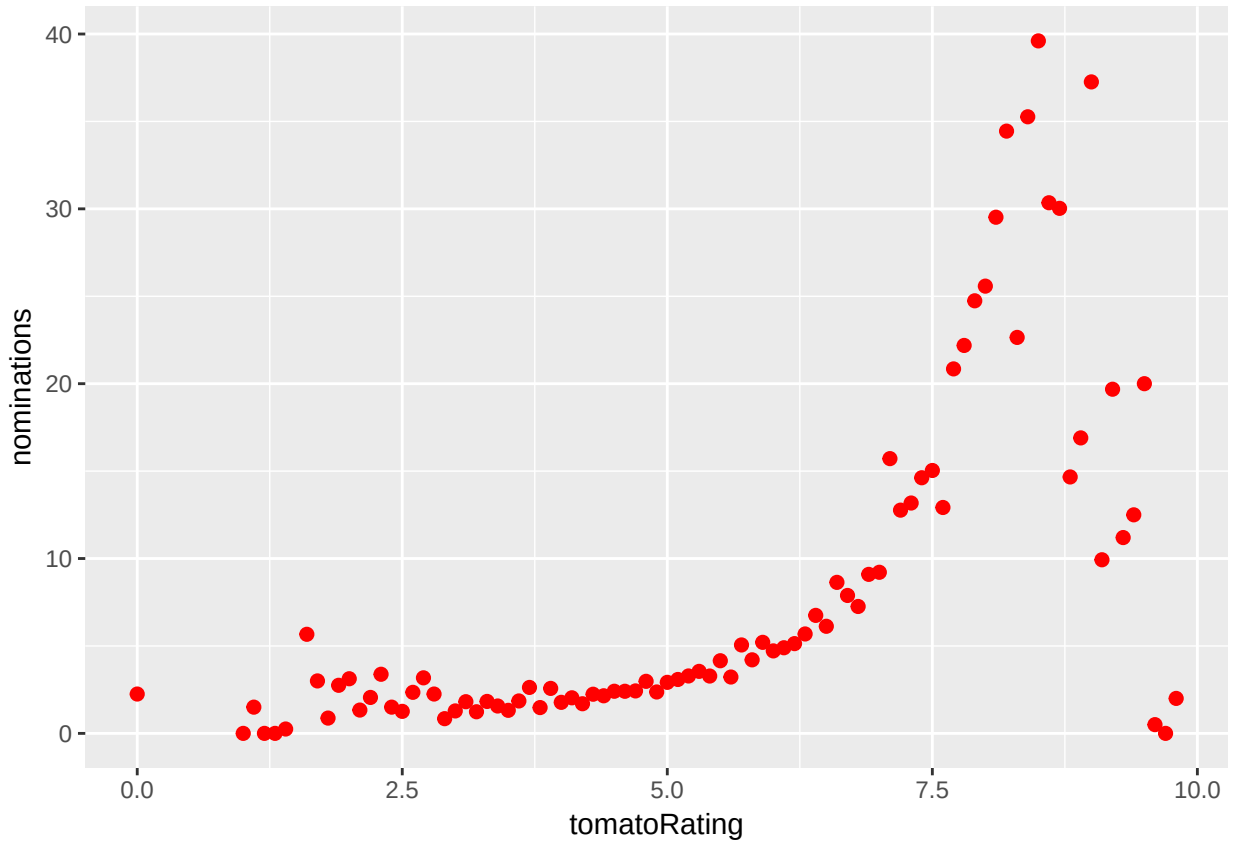

```
ggplot(df, aes(tomatoRating, wins)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 26848 rows containing non-finite values (stat_summary).
```




```
ggplot(df, aes(tomatoRating, nominations)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 26848 rows containing non-finite values (stat_summary).
```




```
mod1 = lm(wins ~ imdbRating, data = df)
summary(mod1)
```

```
##
## Call:
## lm(formula = wins ~ imdbRating, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -4.752  -1.841  -1.038   0.166 232.754
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.88412    0.18149  -26.91   <2e-16 ***
## imdbRating   1.00377    0.02858   35.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.307 on 33918 degrees of freedom
##   (1123 observations deleted due to missingness)
## Multiple R-squared:  0.03509,    Adjusted R-squared:  0.03506
## F-statistic:  1233 on 1 and 33918 DF,  p-value: < 2.2e-16
```

```
mod1 = lm(nominations ~ imdbRating, data = df)
summary(mod1)
```

```
##
## Call:
```

```
## lm(formula = nominations ~ imdbRating, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -6.904  -2.850  -1.732  -0.012 307.193
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.51463    0.28342  -22.99   <2e-16 ***
## imdbRating   1.39776    0.04464   31.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.849 on 33918 degrees of freedom
##   (1123 observations deleted due to missingness)
## Multiple R-squared:  0.0281, Adjusted R-squared:  0.02807
## F-statistic: 980.6 on 1 and 33918 DF,  p-value: < 2.2e-16
```

```
mod1 = lm(wins ~ tomatoRating, data = df)
summary(mod1)
```

```
##
## Call:
## lm(formula = wins ~ tomatoRating, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.004  -4.583  -1.703   1.526 222.935
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -11.87485    0.49878  -23.81   <2e-16 ***
## tomatoRating   2.77106    0.08289   33.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.55 on 8193 degrees of freedom
##   (26848 observations deleted due to missingness)
## Multiple R-squared:   0.12,  Adjusted R-squared:  0.1199
## F-statistic:  1118 on 1 and 8193 DF,  p-value: < 2.2e-16
```

```
mod1 = lm(nominations ~ tomatoRating, data = df)
summary(mod1)
```

```
##
## Call:
## lm(formula = nominations ~ tomatoRating, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.842  -7.808  -2.930   2.238 291.036
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -16.0386     0.7733  -20.74   <2e-16 ***
```

```
## tomatoRating   4.1114     0.1285   31.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.91 on 8193 degrees of freedom
##   (26848 observations deleted due to missingness)
## Multiple R-squared:  0.1111, Adjusted R-squared:  0.1109
## F-statistic:  1024 on 1 and 8193 DF,  p-value: < 2.2e-16
```

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: One extremely interesting trend from the graph above is that for both `imdbRatings` and `tomatoRatings` generally there is a positive correlation between ratings and both wins and nominations. But very interestingly enough, the relationship is actually not monotonic. In the extremely upper regions of ratings, the number of wins and nominations actually decrease!

I fit a linear regression between wins and nominations against ratings field. Notice that in each regression the coefficient is significant, which means that there is a significant positive relationship between rating and nominations and wins. However, the R^2 is relatively low, so the correlation is relatively low.

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.

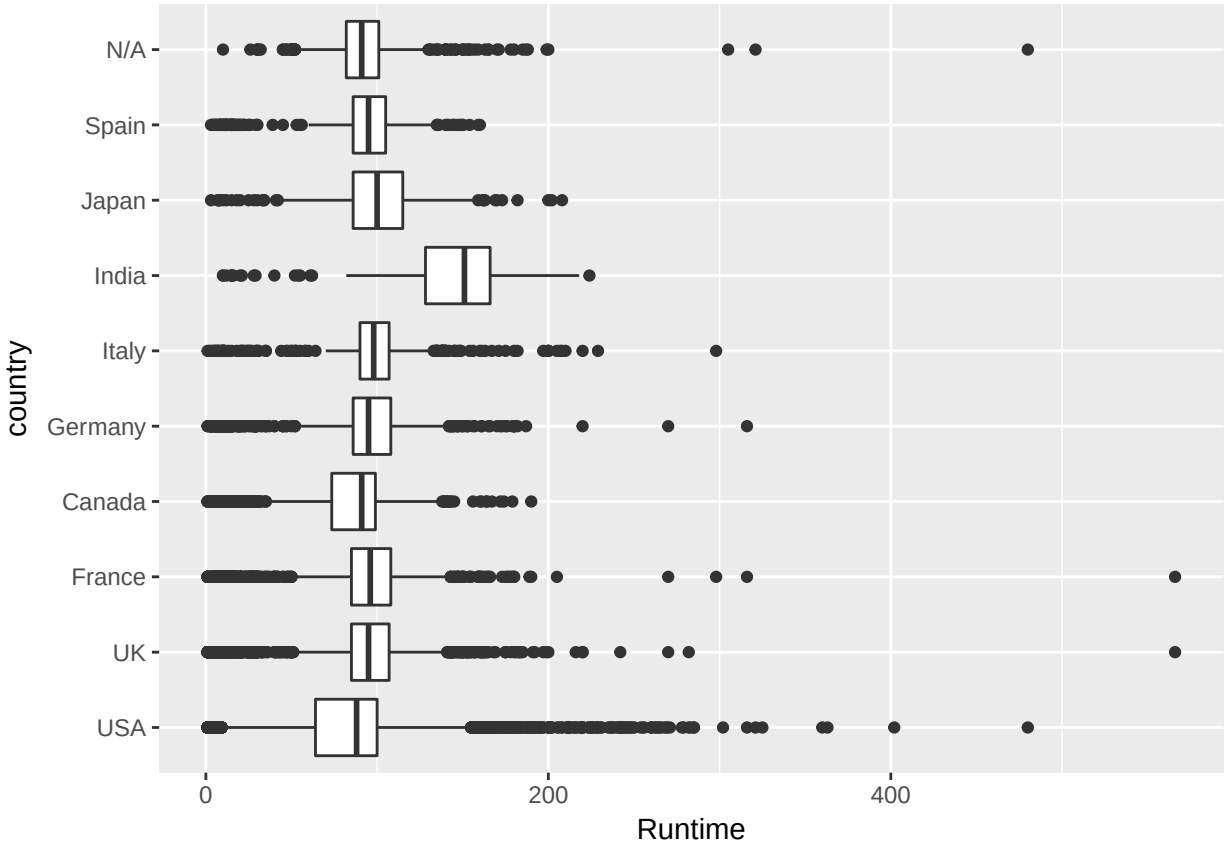
```
# TODO: Find and illustrate two expected insights
df = cbind(df, mtabulate(strsplit(df$Country, ", ")))

# cols = c('Genre', 'Action', 'Adult', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime', 'Docum

# Remove column from dataframe
df$Country = NULL

# cols = setdiff(cols, 'Genre')
colCount = colSums(df[, 80:ncol(df)])
topTenIds = order(colCount,decreasing=TRUE)[1:10]
topTenCols = names(df[, 80:ncol(df)][topTenIds])

# The top 10 most common countries are
print(topTenCols)
```

```
##  [1] "USA"     "UK"      "France"  "Canada"  "Germany" "Italy"   "India"
##  [8] "Japan"   "Spain"   "N/A"
```

```
cols = c('Title', 'Runtime', 'Gross', 'Domestic_Gross')
cols = c(cols, topTenCols)

df9 = df[cols]
df9.long = melt(df9, id.vars=c('Title', 'Runtime', 'Gross', 'Domestic_Gross'), variable.name='country')
df9.long = df9.long[apply(df9.long['value'],1,function(z) !any(z==0)),]

ggplot(df9.long, aes(as.factor(country), Runtime)) +
  geom_boxplot() +
```

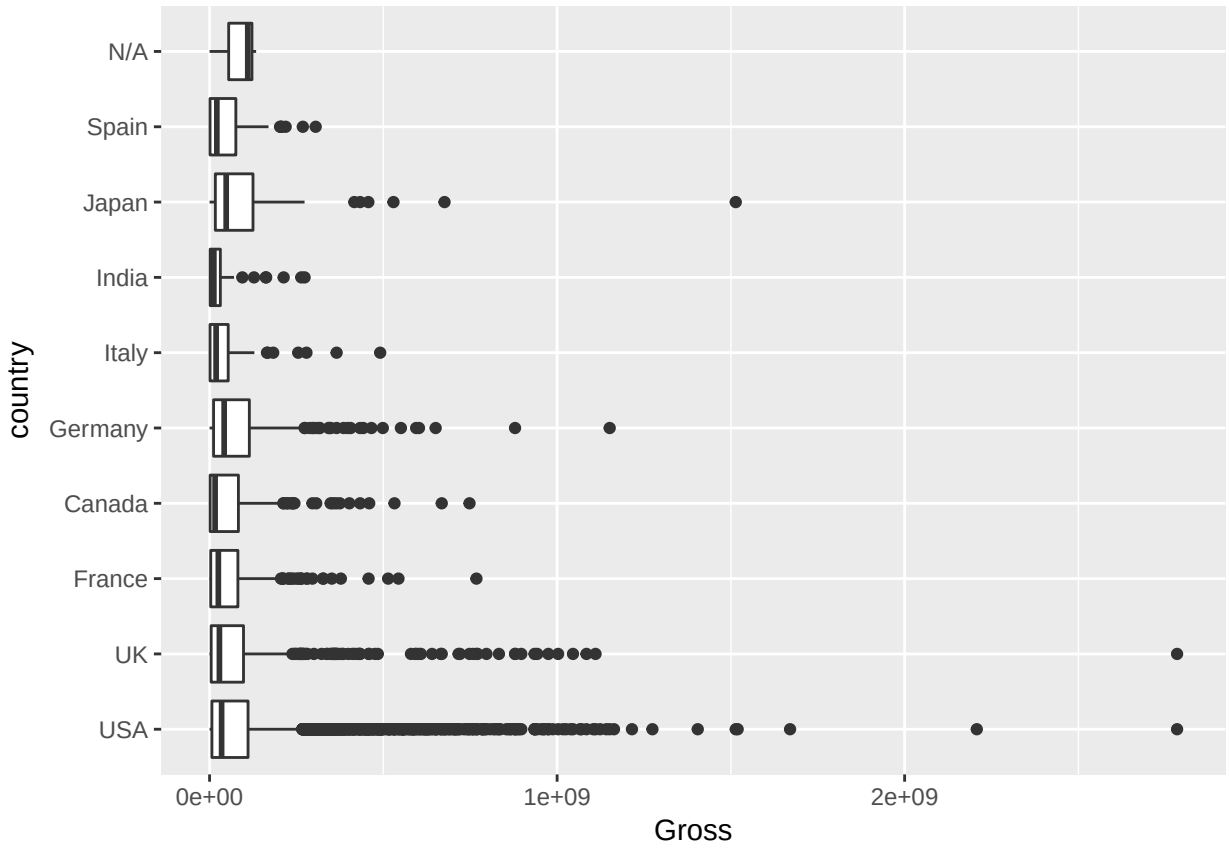
```
  coord_flip() +
  scale_x_discrete("country")
```

```
## Warning: Removed 634 rows containing non-finite values (stat_boxplot).
```



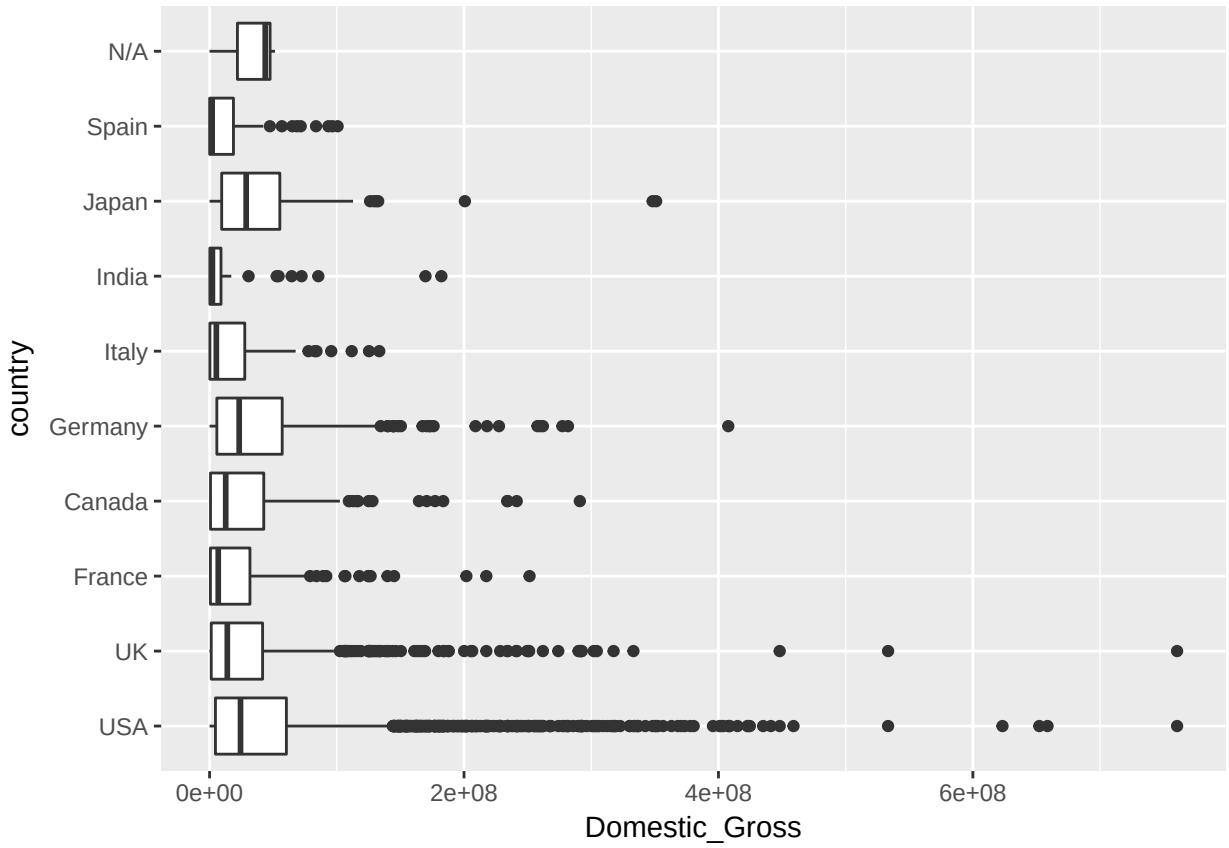

```
ggplot(df9.long, aes(as.factor(country), Gross)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("country")
```

```
## Warning: Removed 27833 rows containing non-finite values (stat_boxplot).
```




```
ggplot(df9.long, aes(as.factor(country), Domestic_Gross)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("country")
```

```
## Warning: Removed 27833 rows containing non-finite values (stat_boxplot).
```

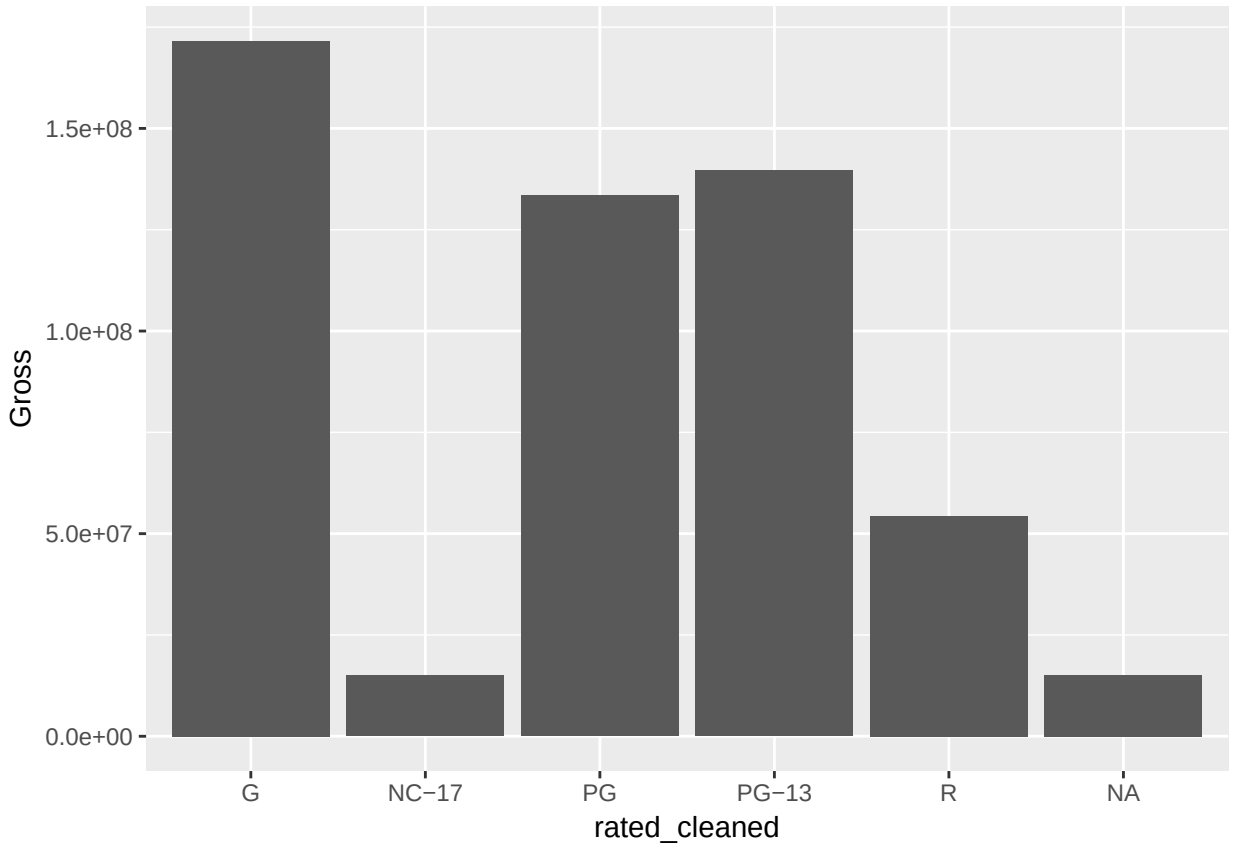



```
df$rated_cleaned = ifelse(df$Rated %in% c('G', 'PG-13', 'PG', 'R', 'NC-17'), df$Rated, NA)
ggplot(df, aes(rated_cleaned, Gross)) +
  geom_bar(stat='summary', fill.y='mean')
```

```
## Warning: Ignoring unknown parameters: fill.y
```

```
## Warning: Removed 30485 rows containing non-finite values (stat_summary).
```

```
## No summary function supplied, defaulting to `mean_se()
```



Q: Expected insight #1.

A: First, we pick the top 10 countires that show up in our data set. Notice that we mostly see "major" countires (e.g. USA, UK, France, Canada). This is expected because I would imagine that the countries producing the most movies are from more major countries with higher GDP. Second, we plot runtime by country. While there is different distribution and sample size by country, notice that the median runtime by country is actually really similar. I would expect this because I am not aware of differences in runtime trends by country. Third, notice that most of the big box office hits (positive) outliers, seem to come from the USA. This is also expected as one would expect almost all the blockbuster movies to come from USA.

Q: Expected insight #2.

A:

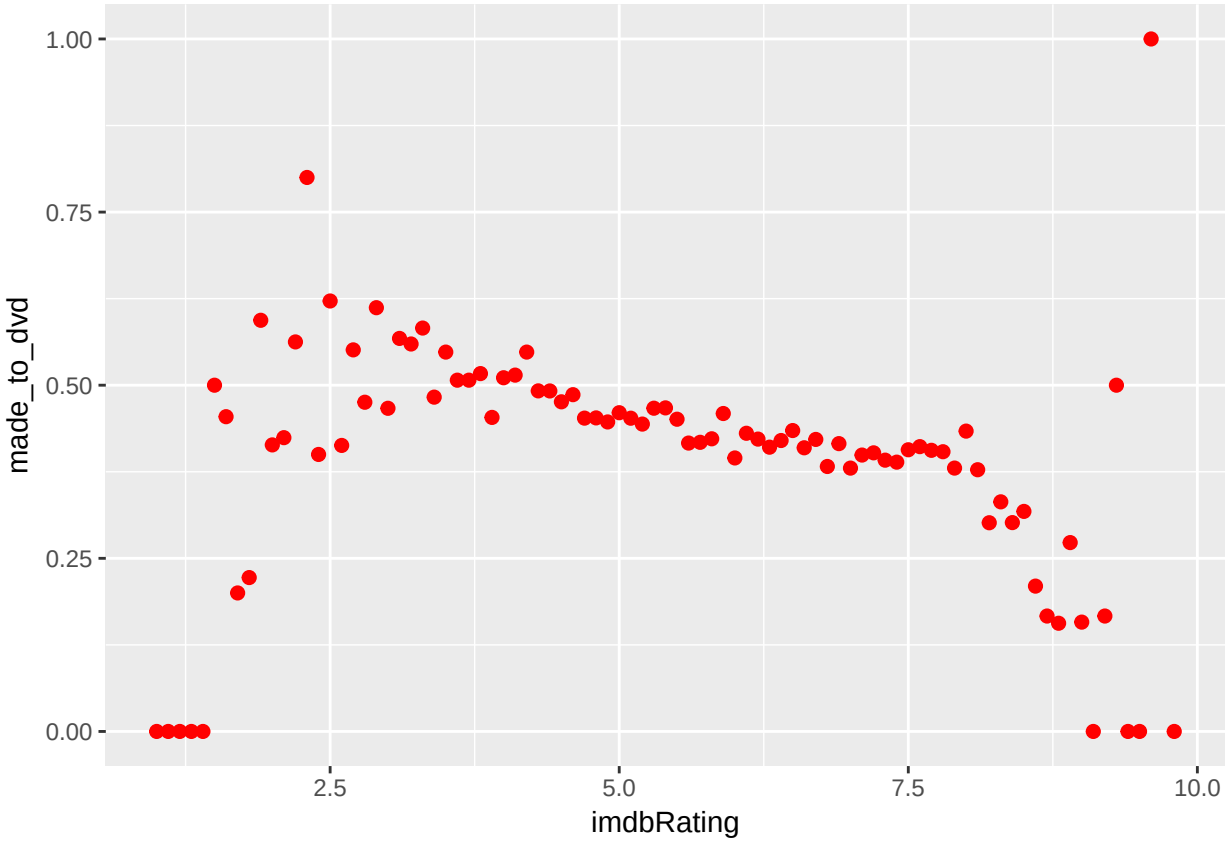
Plotting the Gross Revenue by Rated Type, notice that G rated movies on average gross the highest. This is expected as G rated movies attract entire families so more people tend to watch G rated movies. 2nd and 3rd highest are PG13 and PG, respectively.

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$made_to_dvd = as.numeric(!is.na(df$DVD))

ggplot(df, aes(imdbRating, made_to_dvd)) +
  stat_summary(fun.y = "mean", colour = "red", size = 2, geom = "point")
```

```
## Warning: Removed 1123 rows containing non-finite values (stat_summary).
```



Q: Unexpected insight.

A: One thing that was unexpected to me was I would have expected that there would be a positive correlation between ratings and whether the move was made to a DVD (better movies would more likely be made into a DVD). The relationship I found was somewhat of an "inversed-U" shape. At very low ratings, the proportion of DVD's made is very low. Suprisingly, highly rated ratings also have lower made to DVD ratings.