Project 1: Explore and Prepare Data

GID mmendiola3 CSE6242 - Data and Visual Analytics - Spring 2017 Due: Sunday, March 5, 2017 at 11:59 PM UTC-12:00 on T-Square

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

Data

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

> The file `movies_merged` 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.

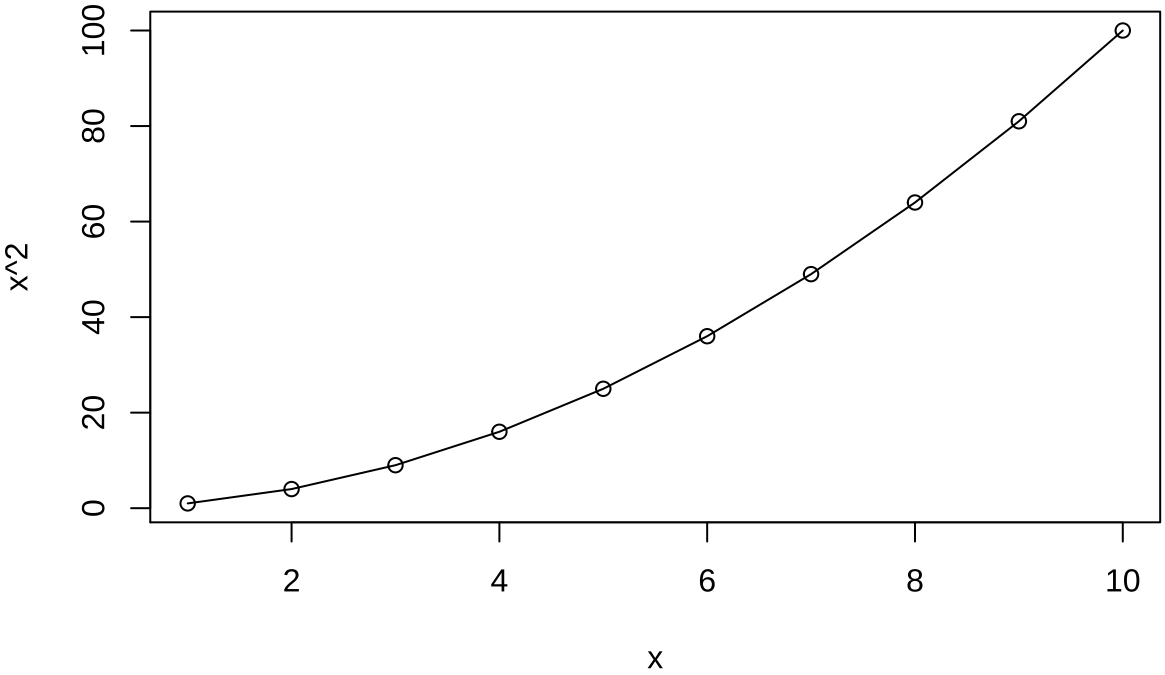
When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```r
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*.

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

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

Setup

Load data

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

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

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

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

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

```
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.

```r
options(warn=-1)
library(ggplot2)
library(lattice)
library(GGally)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:GGally':
##
##     nasa

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(qdap)
```

```
## Loading required package: qdapDictionaries

## Loading required package: qdapRegex

##
## Attaching package: 'qdapRegex'

## The following objects are masked from 'package:dplyr':
##
##     escape, explain

## The following object is masked from 'package:ggplot2':
##
##     %+%

## Loading required package: qdapTools

##
## Attaching package: 'qdapTools'

## The following object is masked from 'package:dplyr':
##
##     id

## Loading required package: RColorBrewer
```

```
##
## Attaching package: 'qdap'

## The following object is masked from 'package:dplyr':
##
##     %>%

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

```
library(reshape2)
library(stringr)
```

```
##
## Attaching package: 'stringr'

## The following object is masked from 'package:qdap':
##
##     %>%
```

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:

- ggplot2
- lattice
- GGally
- dplyr
- qdap
- reshape2
- stringr

Tasks

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

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

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

1. Remove non-movie rows

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

```
# TODO: Remove all rows from df that do not correspond to movies
df = subset(df, Type == 'movie')
nrow(df)
```

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

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

A: 40000

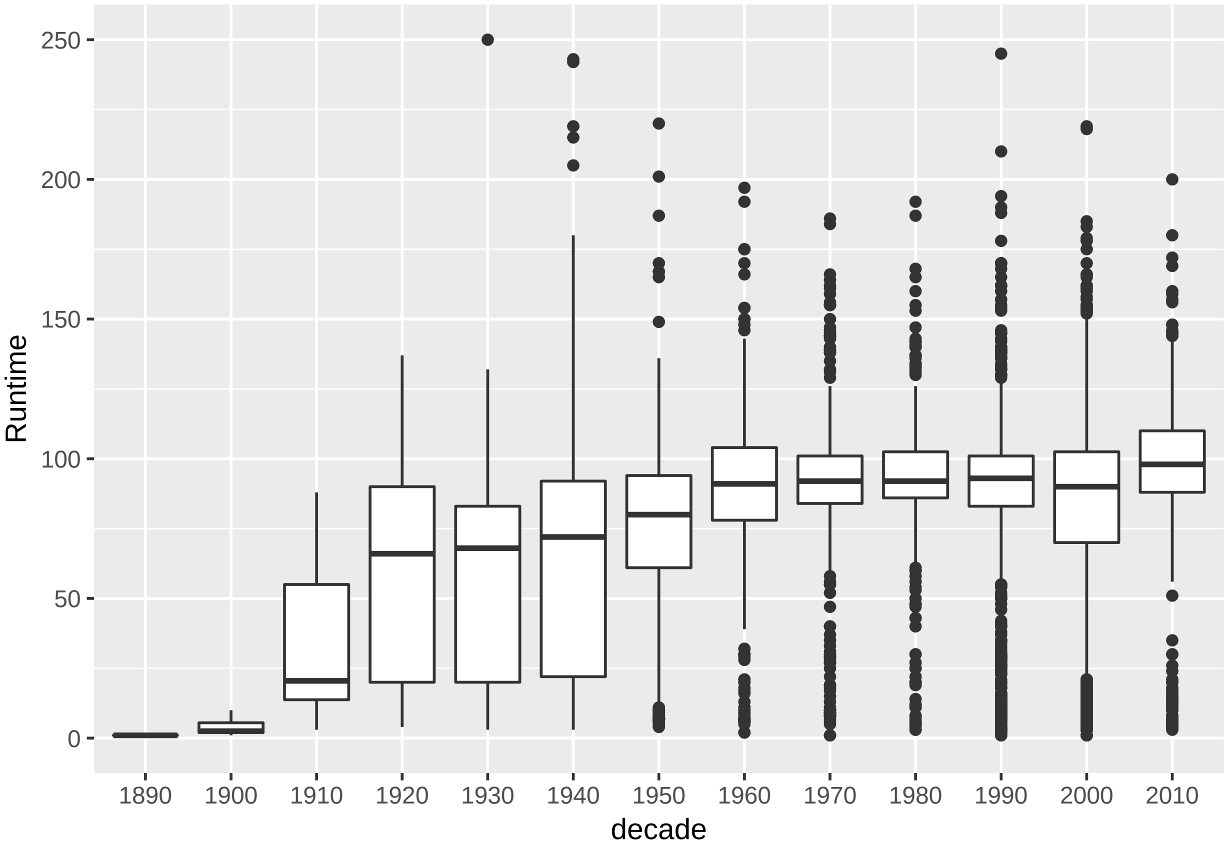
2. Process Runtime column

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

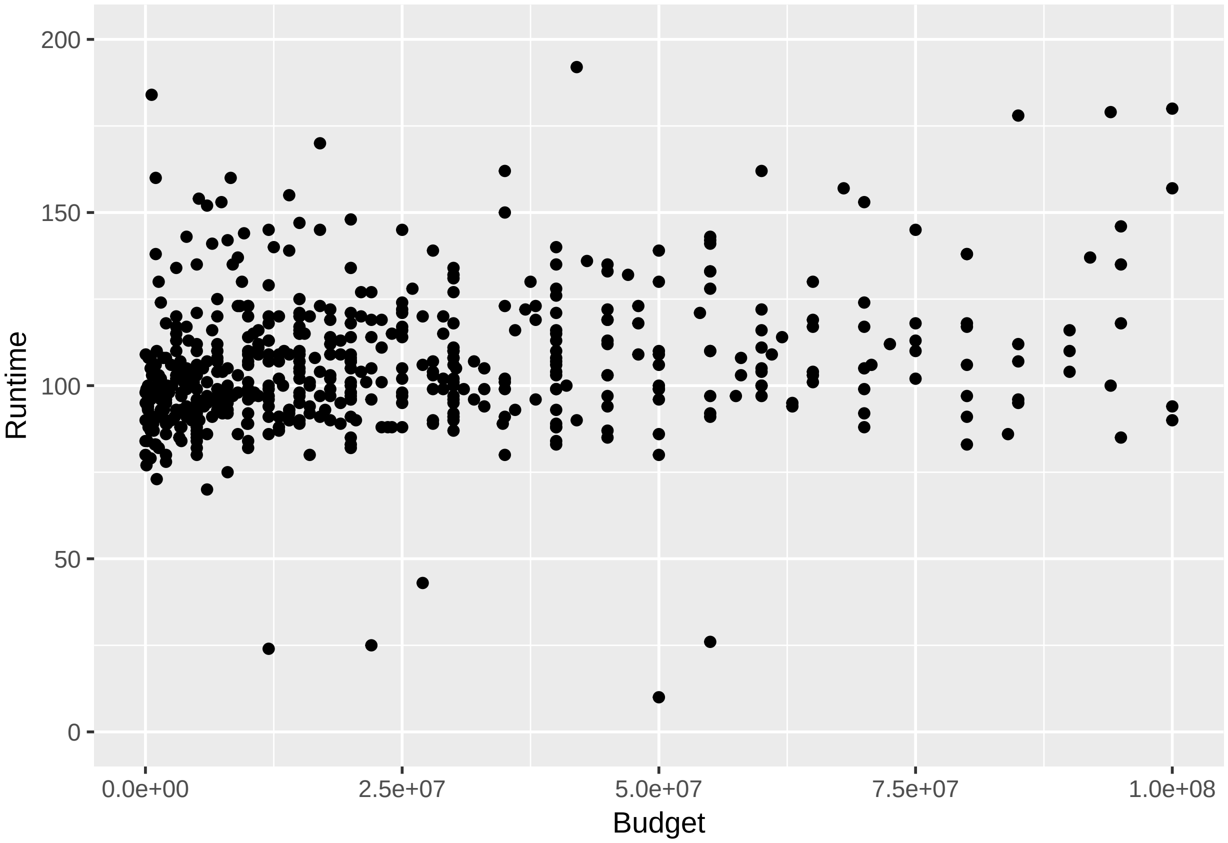
```
# TODO: Replace df$Runtime with a numeric column containing the runtime in minutes
parseRuntime <- function(string) {
  parts = unlist(strsplit(string, ' '))
  if (length(parts) == 2) {
    return(as.numeric(parts[1]))
  } else if (length(parts) == 4) {
    return(as.numeric(parts[1]) * 60 + as.numeric(parts[3]))
  } else {
    return(NA)
  }
}

df$Runtime = sapply(df$Runtime, parseRuntime)
```

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
set.seed(2)
df$decade = factor(df$Year - (df$Year %% 10))
# ggpairs(df[c('Runtime', 'decade', 'Budget')])
df_sample = df[sample(nrow(df), 5000),]
runtime_decade = ggplot(df_sample, aes(decade, Runtime, na.rm=TRUE)) +
  geom_boxplot() +
  ylim(0, 250)
print(runtime_decade)
```



```r
runtime_budget = ggplot(df_sample, aes(Budget, Runtime, na.rm=TRUE)) +
  geom_point() +
  xlim(0, 100000000) +
  ylim(0, 200)
print(runtime_budget)
```



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:

- Median runtimes increase over each decade, with a slight drop in the 2000s.
- There are many outliers, with some Runtimes over 10 hours!
- IQR decreases over the decades, while outliers increase. Indicates many more movies being made at or near the median in later dacades. Runtimes had greater variance between 1910 and 1950.
- There does not seem to be a corelation between Runtime and Budget.

3. Encode `Genre` column

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

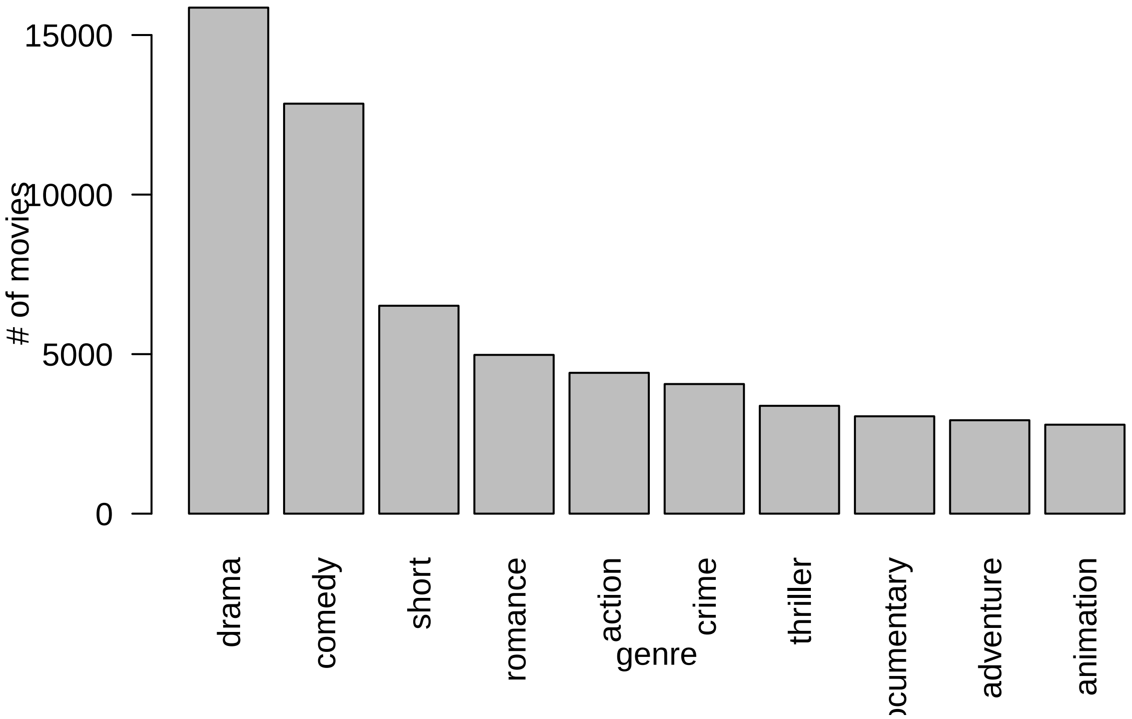
For example, if there are a total of 3 genres: Drama, Comedy, and Action, a movie that is both Action and Comedy should be represented by a binary vector <0, 1, 1>. Note that you need to first compile a dictionary of all possible genres and then figure out which movie has which genres (you can use the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

```r
# TODO: Replace Genre with a collection of binary columns
freqs = t(wfm(df$Genre, 1:nrow(df)))
```

```
df = data.frame(df, freqs, check.names = FALSE)
df[c('Genre', 'na')] = NULL
```

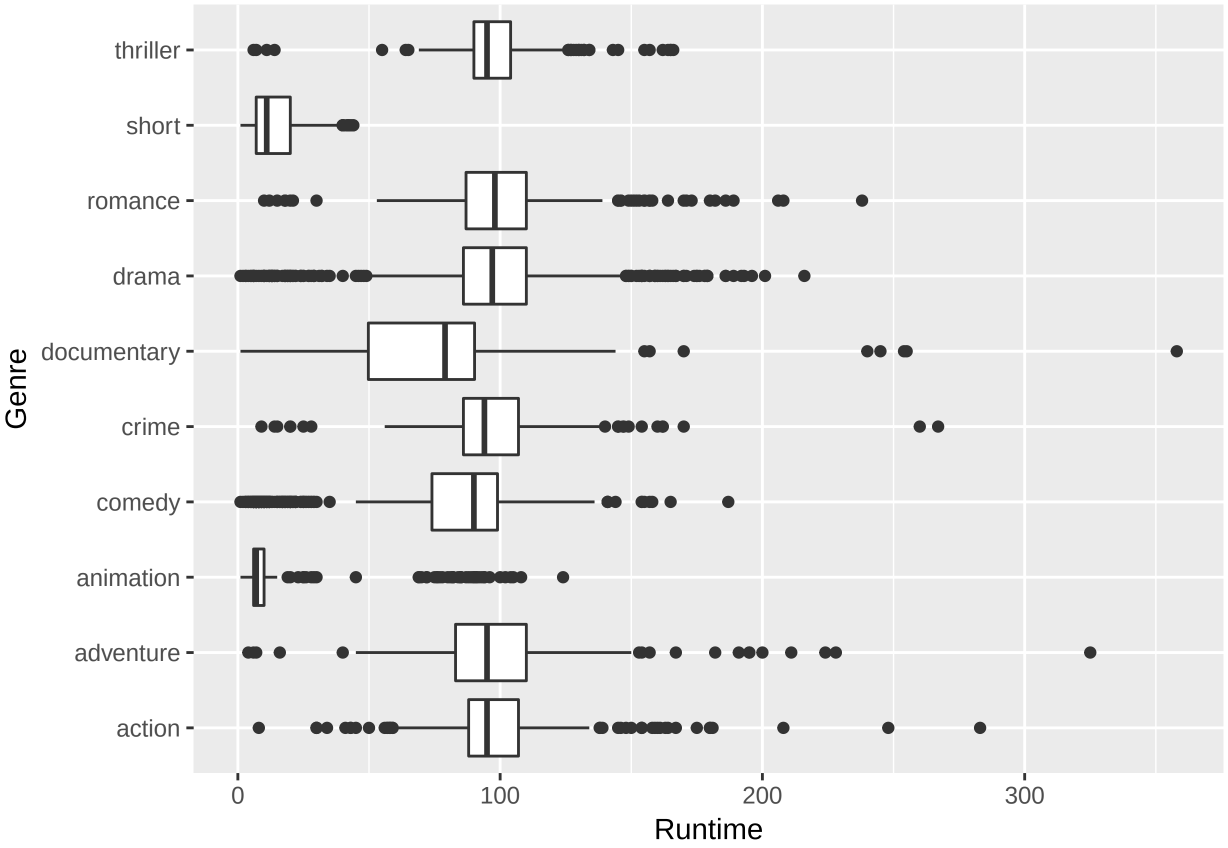
Plot the relative proportions of movies having the top 10 most common genres.

```
# TODO: Select movies from top 10 most common genres and plot their relative proportions
genres = colSums(subset(df, select=action:western))
top_genres = sort(genres, decreasing = TRUE)[1:10]
barplot(top_genres, las=2, ylab="# of movies", xlab="genre")
```



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
genre_select <- function(genre_query) {
  genre_sample = df[df[genre_query] == 1, 'Runtime']
  return(genre_sample)
}
r = sapply(names(top_genres), genre_select)
runtimes = melt(r)
colnames(runtimes) <- c('Runtime', 'Genre')
runtimes_sample = runtimes[sample(nrow(runtimes), 5000),]
ggplot(runtimes_sample, aes(reorder(Genre, -Runtime, median), Runtime)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("Genre")
```



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

A:

- Unsurprisingly, movies in the short and animation genres have a significantly shorter median runtime.
- Documentaries seem to have a large variance in runtimes, with a few large outliers. The median runtime for documentaries are shorter than most other genres.
- All other genres seem to follow a similar distribution, with median runtimes just under 100 minutes, small variance, and a number of outliers.

4. Eliminate mismatched rows

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

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

```
# TODO: Remove rows with Released-Year mismatch
year_delta <- function(year_, date) {
  if (is.na(year_) || is.na(date)) {
    return(0)
  } else if (year_ == as.numeric(format(date, "%Y"))) {
    return(0)
  }
  date = as.Date(date)
  year_start = as.Date(ISOdate(year_, 1, 1))
  year_end = as.Date(ISOdate(year_, 12, 31))
  diff = min(abs(c(date - year_start, date - year_end)))
  return(diff)
}
before_count = nrow(df)
df$DaysDiff = mapply(year_delta, df$Year, df$Released)
```

```
# Threshold of days between Release and Year
threshold = 67
df_removed = subset(df, DaysDiff > threshold)
df_remaining = subset(df, DaysDiff <= threshold)
print(nrow(df_removed))
```

```
## [1] 4318
```

```
total_gross_count = nrow(subset(df, !is.na(Gross)))
removed_gross_count = nrow(subset(df_removed, !is.na(Gross)))
sprintf("percent of rows with Gross value removed: %f", 100 * removed_gross_count/total_gross_count)
```

```
## [1] "percent of rows with Gross value removed: 10.004388"
```

```
df = df_remaining
```

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

A:

1. If Year or Released are NA, keep row
2. If Year matches the year portion of Released, keep row
3. If Released date is within 67 days of the first or last date of the Year, keep row
4. If Released date is greater than 67 days of the first or last date of the Year, remove row

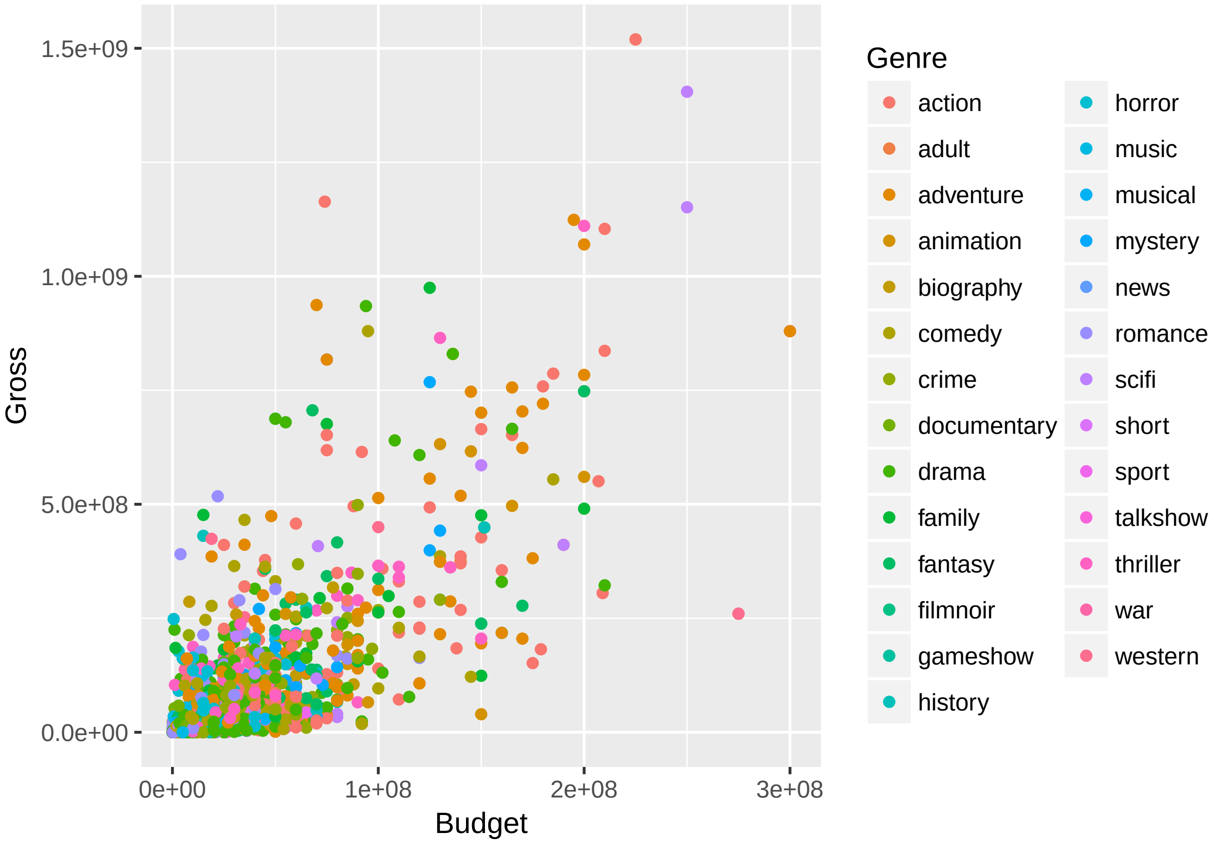
- Removes 4,318 rows.
- Removes 10% of rows with non NA values for Gross.

5. Explore Gross revenue

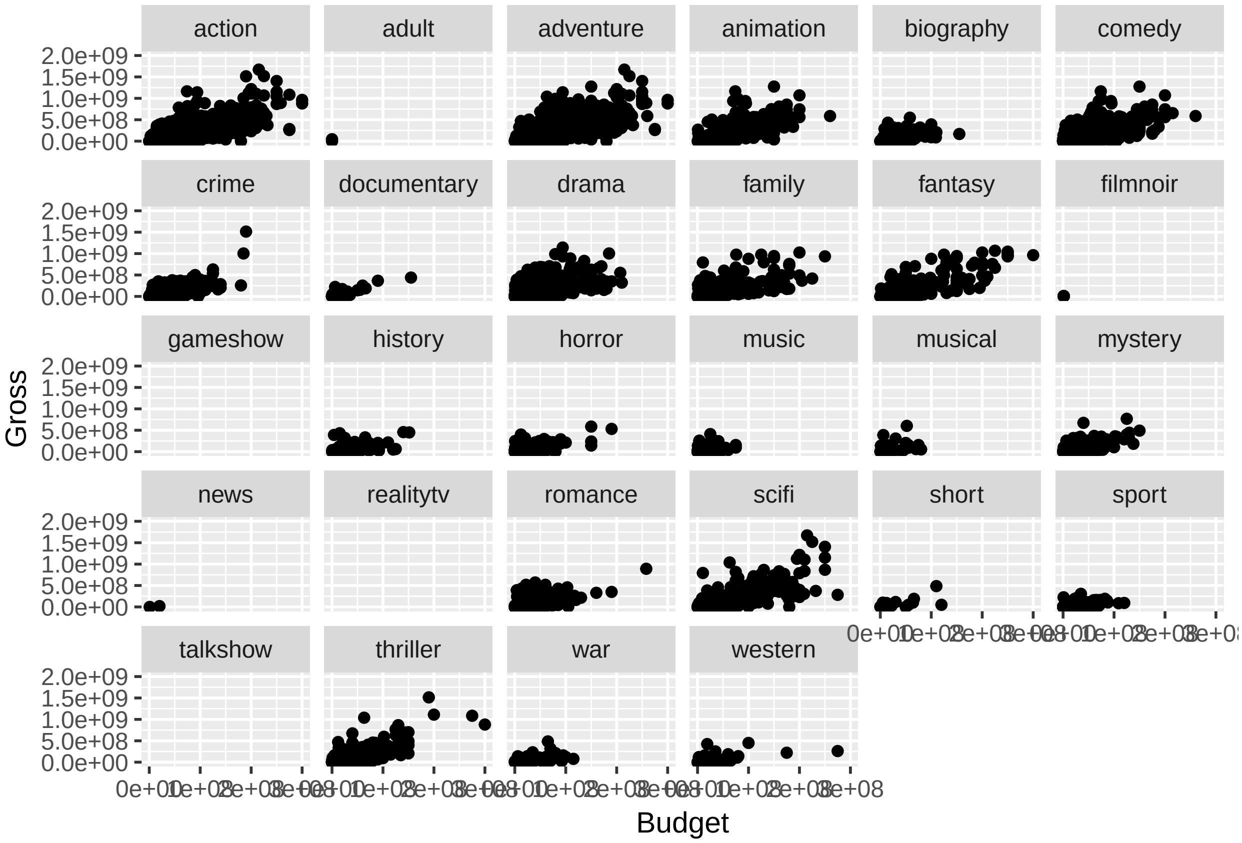
For the commercial success of a movie, production houses want to maximize Gross revenue. Investigate if Gross revenue is related to Budget, Runtime or Genre in any way.

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

```
# TODO: Investigate if Gross Revenue is related to Budget, Runtime or Genre
df_long = melt(df, variable.name="Genre", measure.vars=names(genres))
df_long = subset(df_long, value == 1)
df_long$value = NULL
df_long$Profit = df_long$Gross - df_long$Budget
df_long$Short = df_long$Runtime < 50
df_sample = df_long[sample(nrow(df_long), 10000),]

# No visually observable strong relationship between Gross and Budget across all Genres
ggplot(df_sample, aes(Budget, Gross, color=Genre)) +
  geom_point()
```



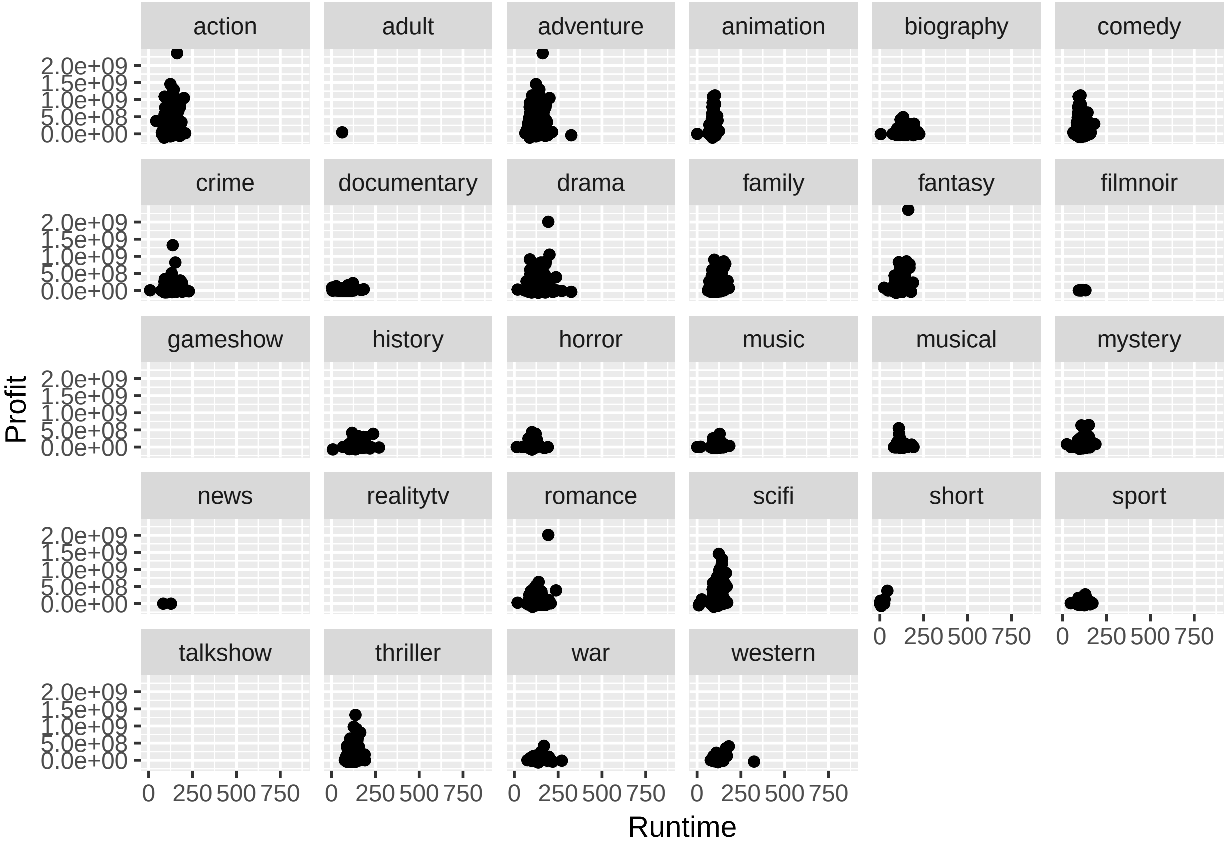
```r
# No visually observable strong relationship between Gross and Budget within most Genres
ggplot(df_long, aes(Budget, Gross)) +
  geom_point(stat="identity") +
  xlim(0, 3e+8) +
  ylim(0, 2e+9) +
  facet_wrap(~ Genre)
```



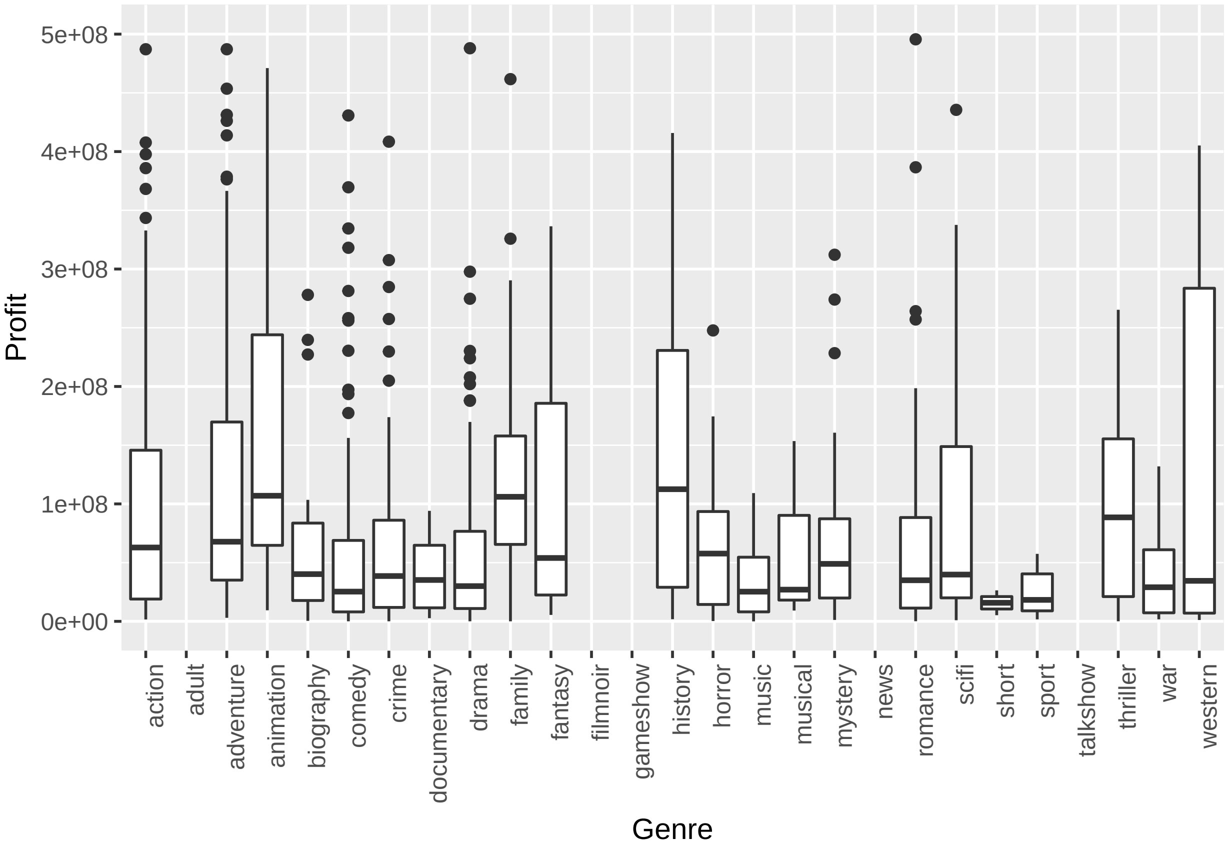
```r
# Documentaries have a strong positive correlation between Gross and Budget
# Action, thriller, adventure, scifi, and fantasy each have a moderate positive relationship between Gr
for (g in names(sort(genres, decreasing = TRUE)[1:25])) {
  m = lm(Gross ~ Budget, subset(df_long, Genre == g))
  print(sprintf("%s Adj-R2: %f", g, summary(m)$adj.r.squared))
}
```

```
## [1] "drama Adj-R2: 0.369895"
## [1] "comedy Adj-R2: 0.450935"
## [1] "short Adj-R2: 0.385869"
## [1] "romance Adj-R2: 0.364293"
## [1] "action Adj-R2: 0.571262"
## [1] "crime Adj-R2: 0.442565"
## [1] "thriller Adj-R2: 0.555336"
## [1] "documentary Adj-R2: 0.787261"
## [1] "adventure Adj-R2: 0.551370"
## [1] "animation Adj-R2: 0.378125"
## [1] "horror Adj-R2: 0.349578"
## [1] "family Adj-R2: 0.440425"
## [1] "mystery Adj-R2: 0.441089"
## [1] "scifi Adj-R2: 0.550420"
## [1] "fantasy Adj-R2: 0.665078"
## [1] "musical Adj-R2: 0.169010"
## [1] "western Adj-R2: 0.283697"
## [1] "music Adj-R2: 0.129483"
## [1] "biography Adj-R2: 0.265162"
## [1] "war Adj-R2: 0.292011"
## [1] "history Adj-R2: 0.275242"
## [1] "sport Adj-R2: 0.176138"
## [1] "adult Adj-R2: NaN"
## [1] "filmnoir Adj-R2: -0.187958"
## [1] "news Adj-R2: NaN"
```

```
# No relationship between Profit and runtime
ggplot(df_long, aes(Runtime, Profit)) +
  geom_point(stat="identity") +
  facet_wrap(~ Genre)
```



```
#
ggplot(df_sample, aes(Genre, Profit)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylim(0, 5e+8)
```

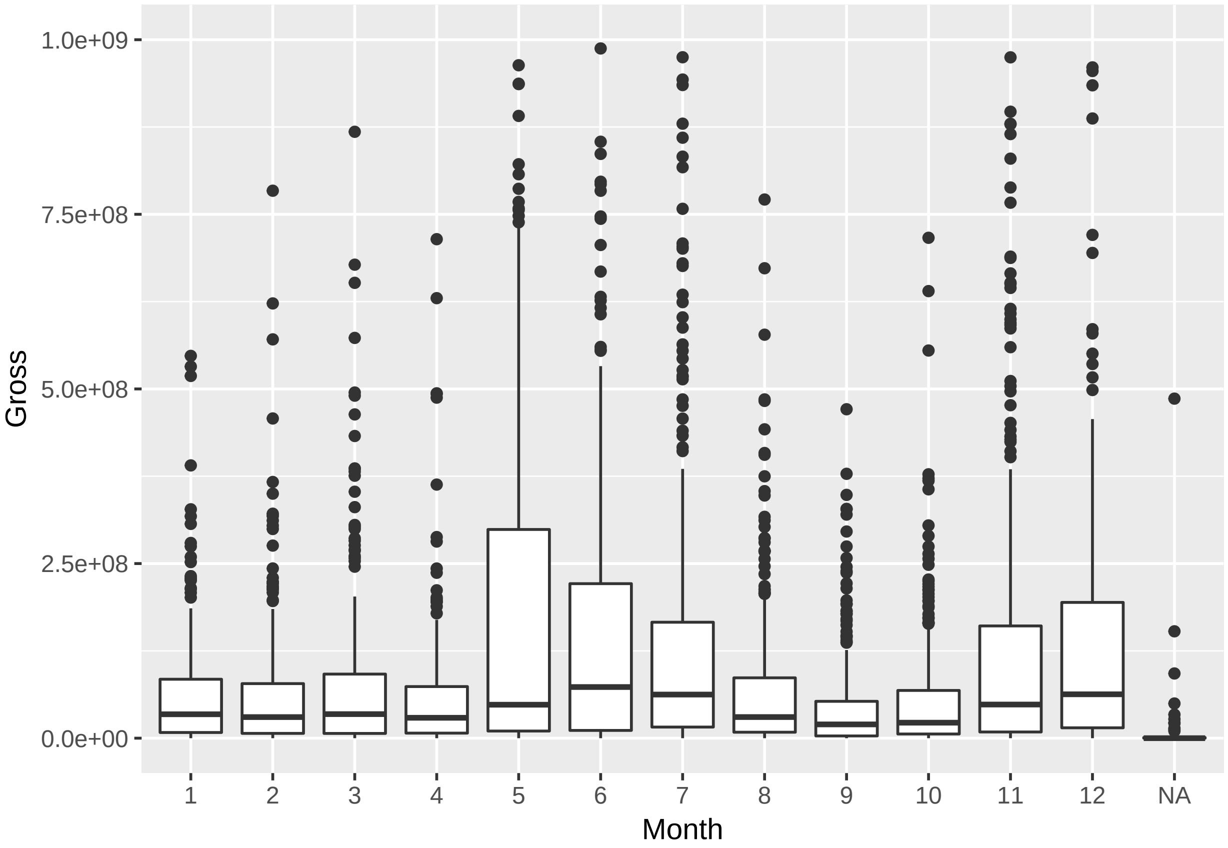


Q: Did you find any observable relationships or combinations of Budget/Runtime/Genre that result in high Gross revenue? If you divided the movies into different subsets, you may get different answers for them - point out interesting ones.

A:

- Documentaries have a strong positive correlation between Gross and Budget (~0.8 Adj R squared)
- Action, thriller, adventure, scifi, and fantasy movies each have a moderate positive relationship between Gross and Budget (~0.5 Adj R squared)
- There is no correlation between Profit (Gross - Budget) and Runtime
- Profit distribution across Genres varies significantly
- Animations have the largest Profit variance and highest median Profit
- Music movies have the lowest Profit variance and the second lowest median Profit
- Profit distributions for most genres are skewed upward (low median with large possitive outliers)

```
# TODO: Investigate if Gross Revenue is related to Release Month
df$Month = factor(as.numeric(format(df$Released, "%m")))
df$Profit = df$Gross - df$Budget
ggplot(df, aes(Month, Gross)) +
  geom_boxplot() +
  ylim(0, 1e+9)
```



Observations:

- A pattern in the distribution of Gross values can be observed over each month
- There is a pattern of low Gross distributions from January to April
- A spike happens in May/June
- A decease occurs from June to September, and an increase from September to Deceber

6. Process `Awards` column

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

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

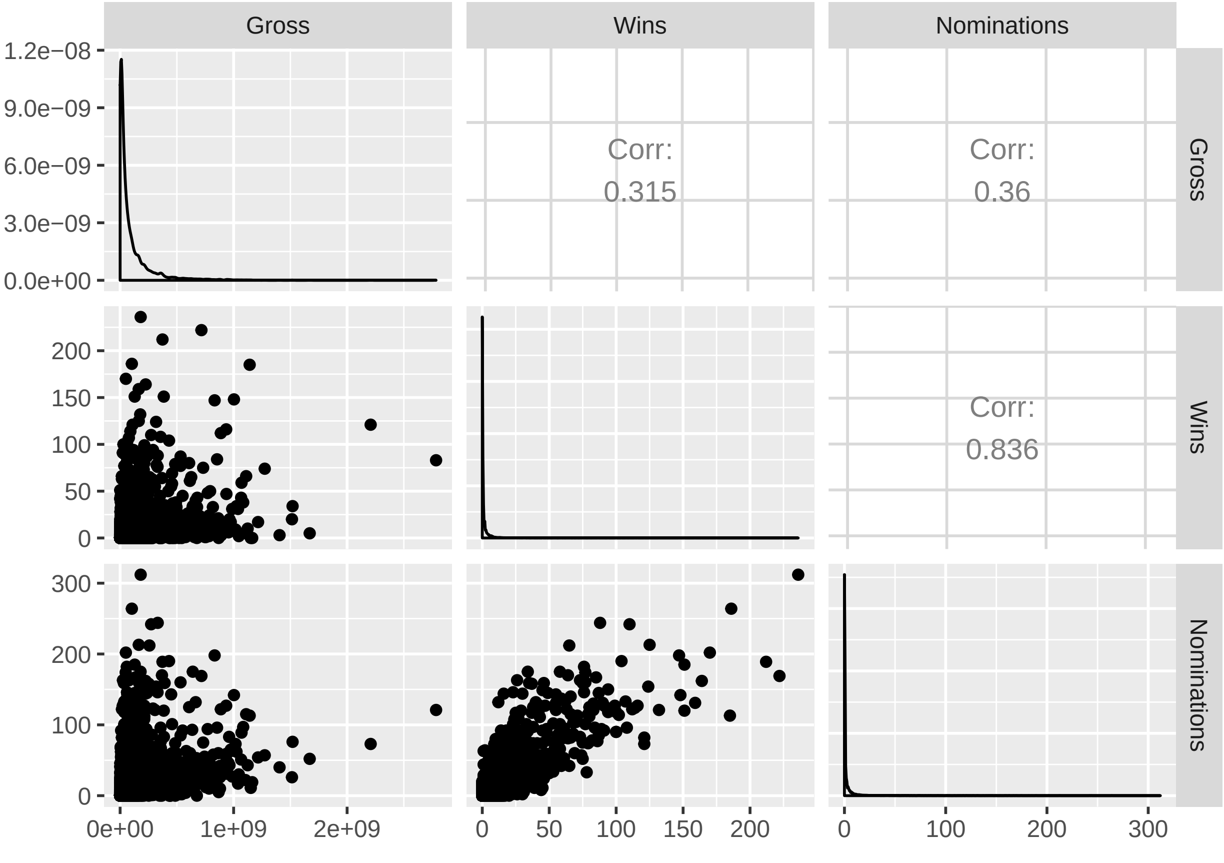
```
# TODO: Convert Awards to 2 numeric columns: wins and nominations

parse_awards <- function(msg, patterns) {
  if (is.na(msg)) {
    return(NA)
  }
  msg = tolower(msg)
  value = 0
  for (pattern in patterns) {
    m = str_match(msg, pattern)[1, 2]
    if (!is.na(m)) {
      value = value + as.integer(m)
    }
  }
  return(value)
}

awards_patterns = c('won (\\d+)', "(\\d+) win")
nomination_patterns = c('nominated for (\\d+)', "(\\d+) nomination")
df$Wins = sapply(df$Awards, parse_awards, pattern=awards_patterns)
df$Nominations = sapply(df$Awards, parse_awards, pattern=nomination_patterns)
df['Awards'] = NULL
nrow(subset(df, Wins > 0 | Nominations > 0))
```

```
## [1] 12525
```

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

A: Four regexes were crafted to extract wins and nominations from the observed variations. Each of the two regexes for each type are run against the Awards column. Extracted values are summed and entered into new columns. 11,653 rows had valid/non-zero wins or nominations.

```
# TODO: Plot Gross revenue against wins and nominations
ggpairs(df[c('Gross', 'Wins', 'Nominations')])
```



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

A:

- Gross varies widely across both number of wins and nominations

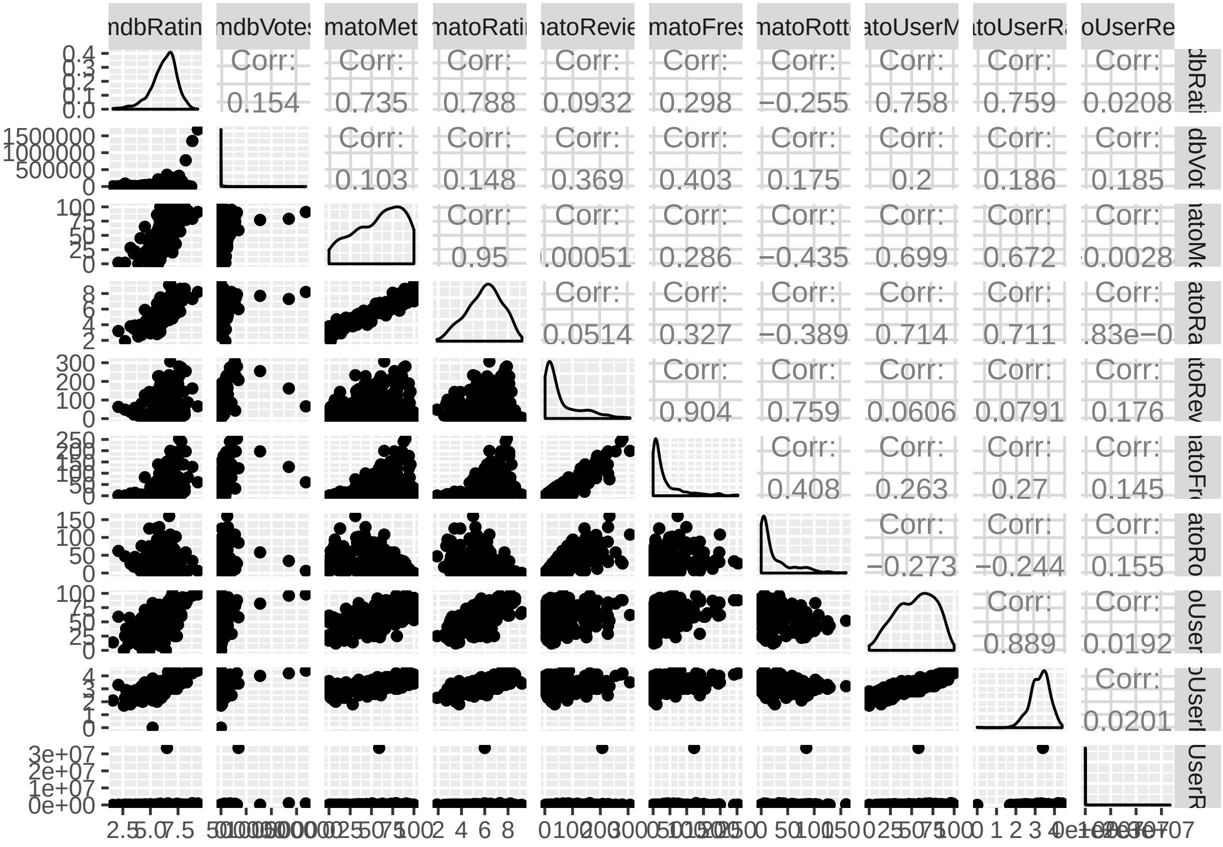
- Movies with > 50 Wins cluster toward lower Gross values
- Movies with > 75 Nominations cluster toward lower Gross values

7. Movie ratings from IMDb and Rotten Tomatoes

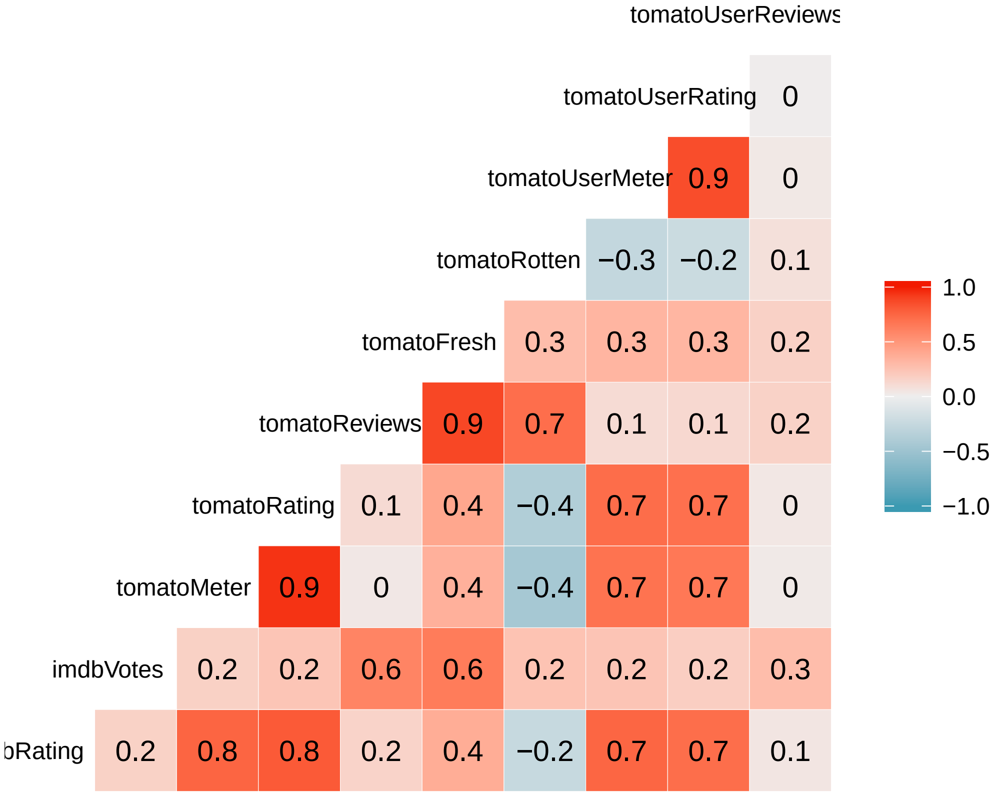
There are several variables that describe ratings, including IMDb ratings (`imdbRating` represents average user ratings and `imdbVotes` represents the number of user ratings), and multiple Rotten Tomatoes ratings (represented by several variables pre-fixed by `tomato`). Read up on such ratings on the 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
ratingColumns = c('imdbRating', 'imdbVotes', 'tomatoMeter', 'tomatoRating', 'tomatoReviews', 'tomatoFres
df_sample = df[sample(nrow(df), 1000),]
ggpairs(df_sample, columns=ratingColumns)
```



```
ggcorr(df[ratingColumns], label=TRUE, size=3, hjust=0.75)
```



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

A:

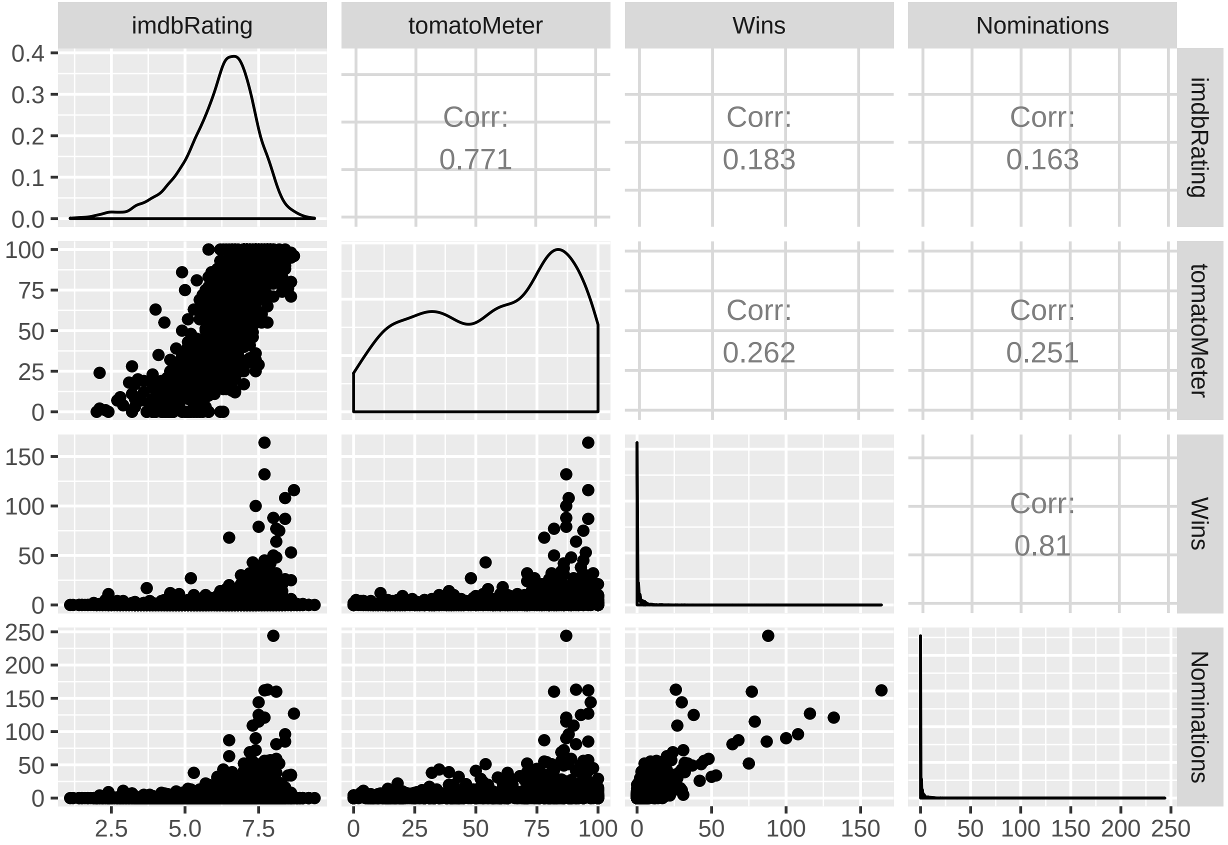
- The strongest relationship is betwen imdbRating and tomatoRating/tomatoMeter (corr 0.8)
- imdbRating also correlates strongly with tomatoUserRating/tomatoUserMeter (corr 0.7)
- Imdb and tomato ratings do not correlate to their respective number of critic or user reviews.

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).

```r
# TODO: Show how ratings and awards are related
columns = c('imdbRating', 'tomatoMeter', 'Wins', 'Nominations')
df_sample = df[sample(nrow(df), 5000),]
ggpairs(df_sample, columns=columns)
```



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?

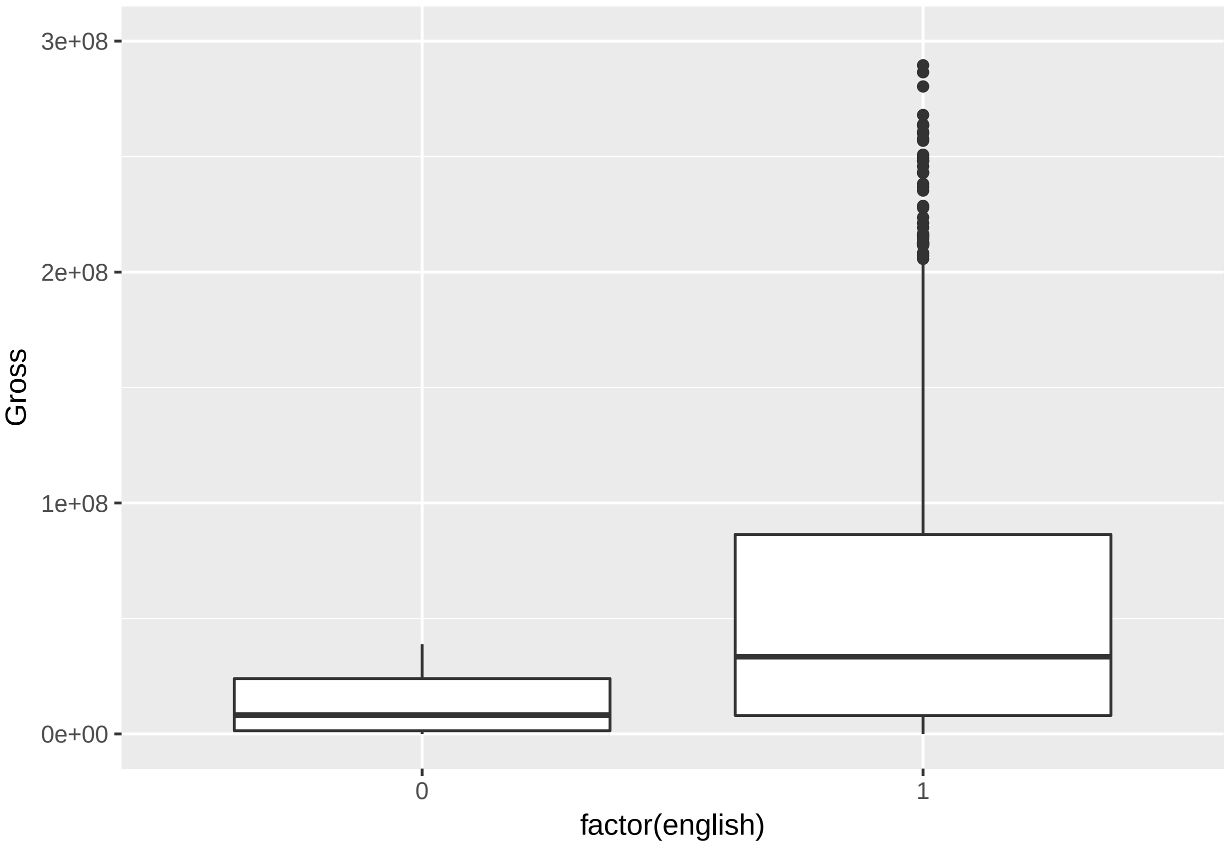
A:

- imdbRating and tomatoMeter each have low correlation with either Wins or Nominations.
- A linear relationship does not exist between ratings an awards
- Movies with higher ratings have a wider range of Wins and Nominations
- A predictive model could partially determine award success (low ratings predicts low awards)

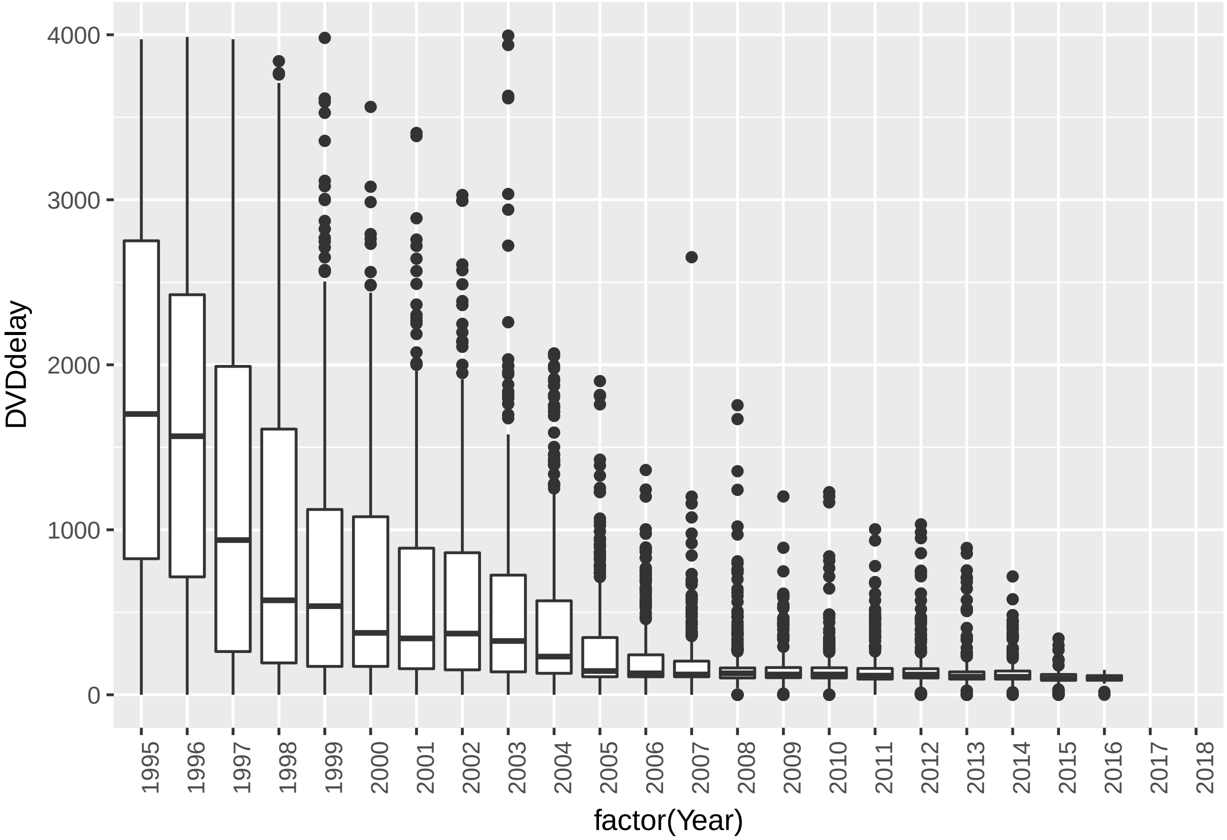
9. Expected insights

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
freqs = t(wfm(df$Language, 1:nrow(df)))
freqs[freqs > 1] = 1
top_n_lang = sort(colSums(freqs), decreasing = TRUE)[1:10]
df = data.frame(df, freqs[, names(top_n_lang)])
# df[names(top_n_lang) > 1] <- 1

df_sample = df[sample(nrow(df), 5000),]

ggplot(df_sample, aes(factor(english), Gross)) +
  geom_boxplot() +
  ylim(0, 3e+8)
```



```r
# DVD delay
df$DVDdelay = df$DVD - df$Released

df_sample = subset(df, Year >= 1995)

ggplot(df_sample, aes(factor(Year), DVDdelay)) +
  geom_boxplot() +
  ylim(0, 4000) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



```
fivenum(df_sample[df_sample$Year == 2016, 'DVDdelay'])
```

```
## Time differences in days
## [1]   0  88 102 116 151
```

Q: Expected insight #1.

A: The distribution of Gross values is lower for movies without an English language version.

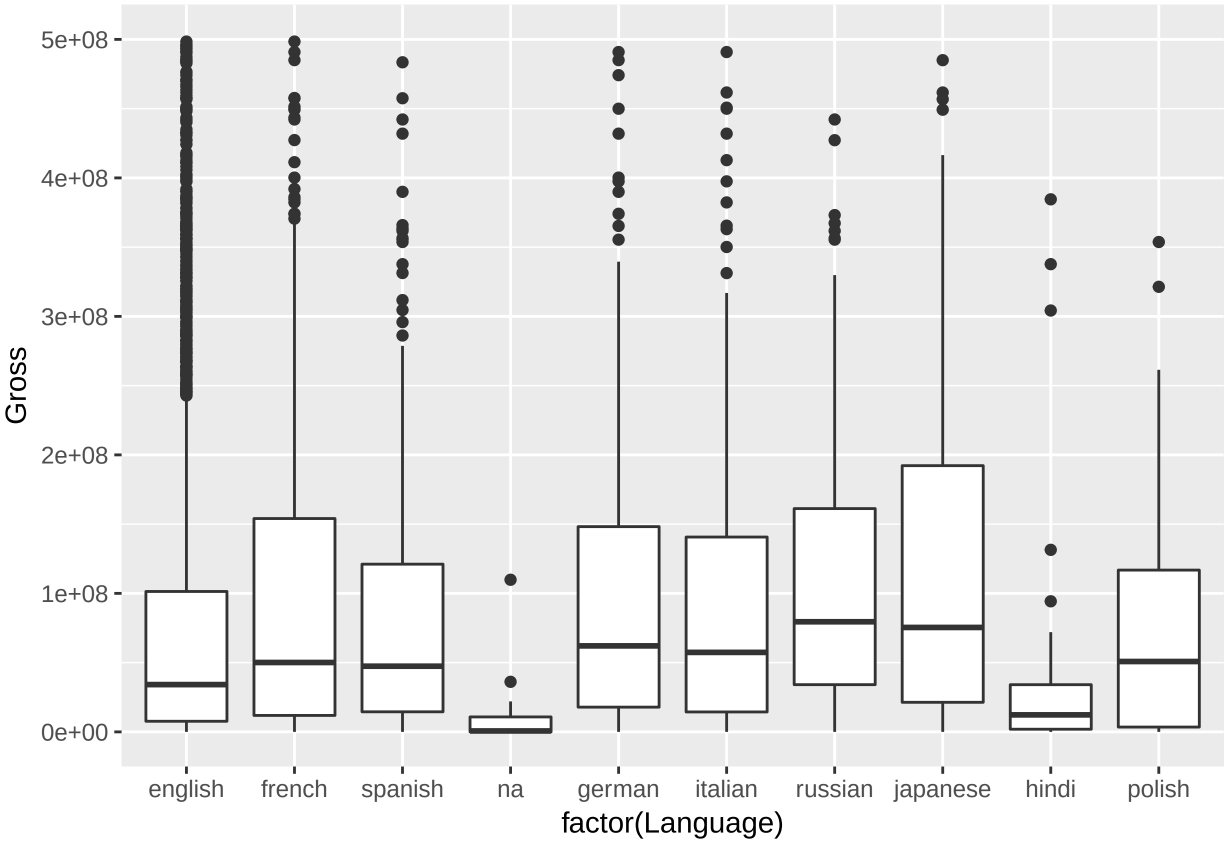
Q: Expected insight #2.

A: The distribution in delay between theater release and DVD release have decrease steadily since DVDs were first created in 1995.

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_temp = cbind(df)
df_temp[c('Language')] = NULL
df_long = melt(df_temp, variable.name="Language", measure.vars=names(top_n_lang))
df_long = subset(df_long, value == 1)
df_long$value = NULL

ggplot(df_long, aes(factor(Language), Gross)) +
  geom_boxplot() +
  ylim(0, 5e+8)
```



Q: Unexpected insight.

A: English language movies do not have the highest median Gross. Movies with a Russian, Japanese, French, Spanish, Italian, or Polish language version have higher median Gross values. This is explained by the fact that successful domestic movies are probably translated for export in other languages, while poor performing movies remain only in English versions. This pulls the distribution of Gross values up for movies in these other languages.