The Movies Dataset Analysis — Proposal

Qingsong Wang

2019/3/3

1. Motivation

Movies, also known as films, are a type of visual communication which uses moving pictures and sound to tell stories or teach people something. Nowadays, movies are appealing a rapidly-increasing number of audience, thus I want to learn more about movies. The questions that I'm interested in are listed delow:

- Which factors are of importance in making the movies popular?
- How to recommend the most suitable movies for audience?
- How the network relationship among actors/actresses looks like?
- etc.

Fortunately, there several datasets regarding movies on the site of Kaggle(see *The Movies Dataset*, and *TMDb*), which contain comprehensive infomation about nearly 5000 movies and over 26 million ratings from The Movie Database(simplified as TMDb). What I'm going to do next is trying to find solutions to the questions mentioned above, with the help of these datasets.

2. Overview of the Datasets

The datasets consist of the following files:

file	description
tmdb_5000_movies.csv	Information of 5,000 movies including features like budget, revenue, release dates, languages, companies, etc.
tmdb_5000_credits.csv	Cast and crew information; stringified JSON object.
ratings.csv	Ratings from 270,000 users

First import the libraries needed in following analysis.

```
library(dplyr)
```

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

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

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

```
library(readr)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```r
library(jsonlite)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------------------- tid
```

```
## v ggplot2 3.1.0     v purrr   0.3.2
## v tibble  2.1.1     v stringr 1.4.0
## v tidyr   0.8.3     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------ tidyverse_
## x dplyr::filter()  masks stats::filter()
## x purrr::flatten() masks jsonlite::flatten()
## x dplyr::lag()     masks stats::lag()
```

```r
library(ggplot2)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

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

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```r
library(lubridate)
```

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

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

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(GGally)
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```r
library(network)
```

```
## network: Classes for Relational Data
## Version 1.15 created on 2019-04-01.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Martina Morris, University of Washington
##                     Skye Bender-deMoll, University of Washington
##  For citation information, type citation("network").
##  Type help("network-package") to get started.
```

Then load the datasets "*tmdb_5000_movies.csv*", and have a glimpse at it.

```r
movie <- read_csv("tmdb_5000_movies.csv", na = "NA")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   budget = col_double(),
##   id = col_double(),
##   popularity = col_double(),
##   release_date = col_date(format = ""),
##   revenue = col_double(),
##   runtime = col_double(),
##   vote_average = col_double(),
##   vote_count = col_double()
## )

## See spec(...) for full column specifications.
```

```r
glimpse(movie)
```

```
## Observations: 4,803
## Variables: 20
## $ budget               <dbl> 2.37e+08, 3.00e+08, 2.45e+08, 2.50e+08, 2...
## $ genres               <chr> "[{\"id\": 28, \"name\": \"Action\"}, {\"...
## $ homepage             <chr> "http://www.avatarmovie.com/", "http://di...
## $ id                   <dbl> 19995, 285, 206647, 49026, 49529, 559, 38...
## $ keywords             <chr> "[{\"id\": 1463, \"name\": \"culture clas...
## $ original_language    <chr> "en", "en", "en", "en", "en", "en", "en",...
## $ original_title       <chr> "Avatar", "Pirates of the Caribbean: At W...
## $ overview             <chr> "In the 22nd century, a paraplegic Marine...
## $ popularity           <dbl> 150.43758, 139.08262, 107.37679, 112.3129...
## $ production_companies <chr> "[{\"name\": \"Ingenious Film Partners\",...
## $ production_countries <chr> "[{\"iso_3166_1\": \"US\", \"name\": \"Un...
## $ release_date         <date> 2009-12-10, 2007-05-19, 2015-10-26, 2012...
## $ revenue              <dbl> 2787965087, 961000000, 880674609, 1084939...
## $ runtime              <dbl> 162, 169, 148, 165, 132, 139, 100, 141, 1...
## $ spoken_languages     <chr> "[{\"iso_639_1\": \"en\", \"name\": \"Eng...
## $ status               <chr> "Released", "Released", "Released", "Rele...
## $ tagline              <chr> "Enter the World of Pandora.", "At the en...
## $ title                <chr> "Avatar", "Pirates of the Caribbean: At W...
## $ vote_average         <dbl> 7.2, 6.9, 6.3, 7.6, 6.1, 5.9, 7.4, 7.3, 7...
## $ vote_count           <dbl> 11800, 4500, 4466, 9106, 2124, 3576, 3330...
```

The meanings of each variable are listed below.

- budget: The budget of the movie in dollars.

- genres: A stringified list of dictionaries that list out all the genres associated with the movie.
- homepage: The Official Homepage of the move.
- id: The ID of the move.
- original_language: The language in which the movie was originally shot in.
- original_title: The original title of the movie.
- overview: A brief blurb of the movie.
- popularity: The Popularity Score assigned by TMDB.
- poster_path: The URL of the poster image.
- production_companies: A stringified list of production companies involved with the making of the movie.
- production_countries: A stringified list of countries where the movie was shot/produced in.
- release_date: Theatrical Release Date of the movie.
- revenue: The total revenue of the movie in dollars.
- runtime: The runtime of the movie in minutes.
- spoken_languages: A stringified list of spoken languages in the film.
- status: The status of the movie (Released, To Be Released, Announced, etc.)
- tagline: The tagline of the movie.
- title: The Official Title of the movie.
- vote_average: The average rating of the movie.
- vote_count: The number of votes by users, as counted by TMDB.

Then we check the number of NAs and fill in the 3 missing values by searching on the Internet.

```
apply(is.na(movie), 2, sum)
```

```
##               budget               genres             homepage
##                    0                    0                    0
##                   id             keywords    original_language
##                    0                    0                    0
##       original_title             overview           popularity
##                    0                    0                    0
## production_companies production_countries         release_date
##                    0                    0                    1
##              revenue              runtime     spoken_languages
##                    0                    2                    0
##               status              tagline                title
##                    0                    0                    0
##         vote_average           vote_count
##                    0                    0
```

```
movie$release_date[is.na(movie$release_date)] = "2014-06-01"
movie$runtime[is.na(movie$runtime)] = c(94, 240)
```

For "*tmdb_5000_credits.csv*", we do the same things and find no missing values. The variable **cast** and **crew** record actors/actresses and directors respectively.

```
credit <- read_csv("tmdb_5000_credits.csv", na = "NA")
```

```
## Parsed with column specification:
## cols(
##   movie_id = col_double(),
##   title = col_character(),
##   cast = col_character(),
##   crew = col_character()
## )
```

```r
glimpse(credit)
```

```
## Observations: 4,803
## Variables: 4
## $ movie_id <dbl> 19995, 285, 206647, 49026, 49529, 559, 38757, 99861, ...
## $ title    <chr> "Avatar", "Pirates of the Caribbean: At World's End",...
## $ cast     <chr> "[{\"cast_id\": 242, \"character\": \"Jake Sully\", \...
## $ crew     <chr> "[{\"credit_id\": \"52fe48009251416c750aca23\", \"dep...
```

```r
apply(is.na(credit), 2, sum)
```

```
## movie_id    title     cast     crew
##        0        0        0        0
```

Now we can combine the two datasets above and delete some covariates that have little with our analysis.

```r
credit <- credit %>% select(-title)
movie.full <- movie %>%
                inner_join(credit, by = c("id" = "movie_id")) %>%
                mutate(year = year(release_date)) %>%
                select(-homepage, -original_title, -spoken_languages,
                       -production_countries, -status, -release_date)
```

And lastly we load "*ratings.csv*", which contains ratings of nearly 45000 movies, meaning that we need to extract the movies that are recorded in all datasets. We check the NAs and find no one, too.

```r
rating <- read_csv("ratings.csv", na = "NA")
```

```
## Parsed with column specification:
## cols(
##   userId = col_double(),
##   movieId = col_double(),
##   rating = col_double(),
##   timestamp = col_double()
## )
```

```r
rating <- movie %>% select(id, title) %>%
              inner_join(rating, by = c("id" = "movieId")) %>%
              arrange(userId)
apply(is.na(rating), 2, sum)
```

```
##        id     title    userId    rating timestamp
##         0         0         0         0         0
```

There are some variables in the format of JSON object, so we transform them into character string.

```r
genres <- movie.full %>%
  filter(nchar(genres) > 2) %>%
  mutate(js = lapply(genres, fromJSON)) %>%
  unnest(js) %>%
  select(id, title, genres = name) %>%
  mutate_if(is.character, factor)
keywords <- movie.full %>%
  filter(nchar(keywords) > 2) %>%
  mutate(js = lapply(keywords, fromJSON)) %>%
  unnest(js) %>%
  select(id, title, keywords = name) %>%
  mutate_if(is.character, factor)
```

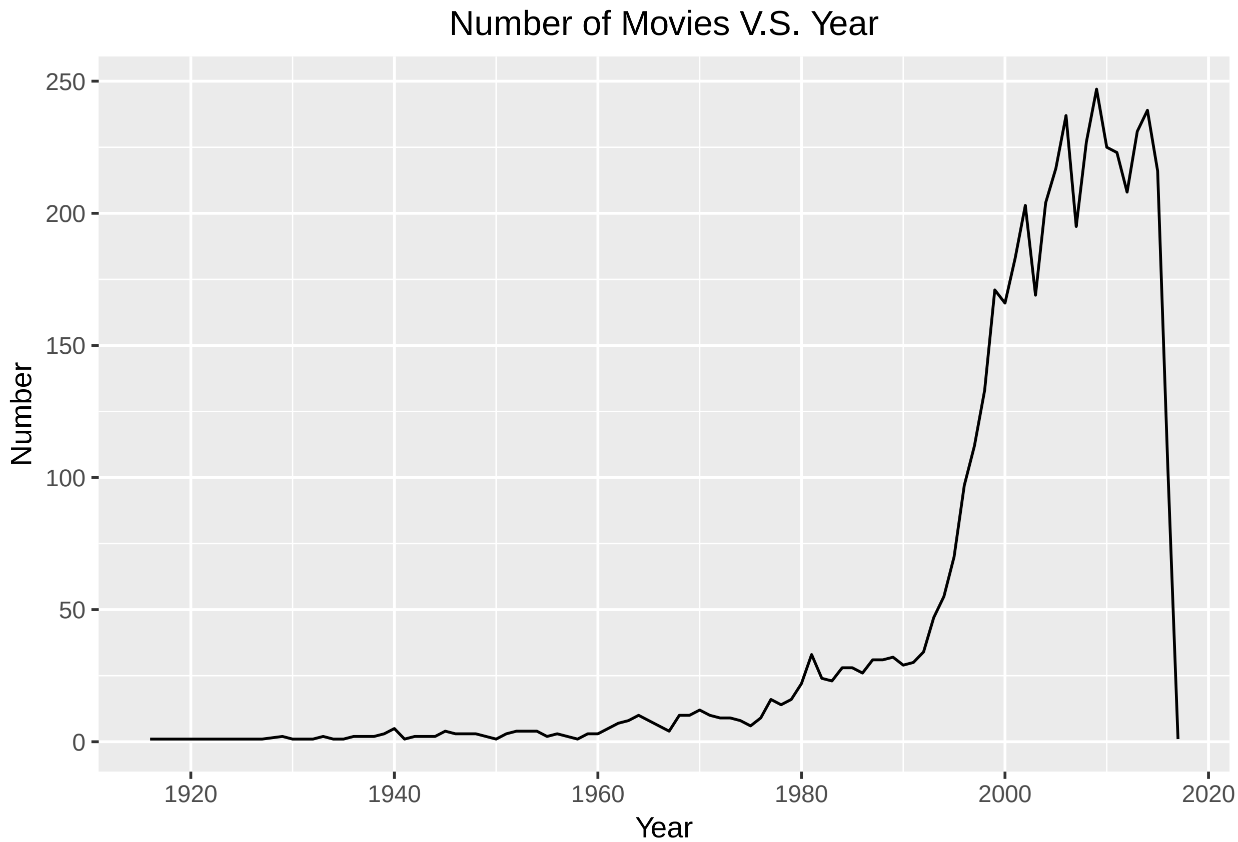
```r
company <- movie.full %>%
  filter(nchar(production_companies) > 2) %>%
  mutate(js = lapply(production_companies, fromJSON)) %>%
  unnest(js) %>%
  select(id, title, company = name) %>%
  mutate_if(is.character, factor)
cast <- movie.full %>%
  filter(nchar(cast) > 2) %>%
  mutate(js = lapply(cast, fromJSON)) %>%
  unnest(js) %>%
  select(id, title, cast = name) %>%
  mutate_if(is.character, factor)
crew <- movie.full %>%
  filter(nchar(crew) > 2) %>%
  mutate(js = lapply(crew, fromJSON)) %>%
  unnest(js) %>%
  select(id, title, crew = name) %>%
  mutate_if(is.character, factor)
```

3. Exploratory Data Analysis

We do data visualization in this section, trying to find some patterns of the data.

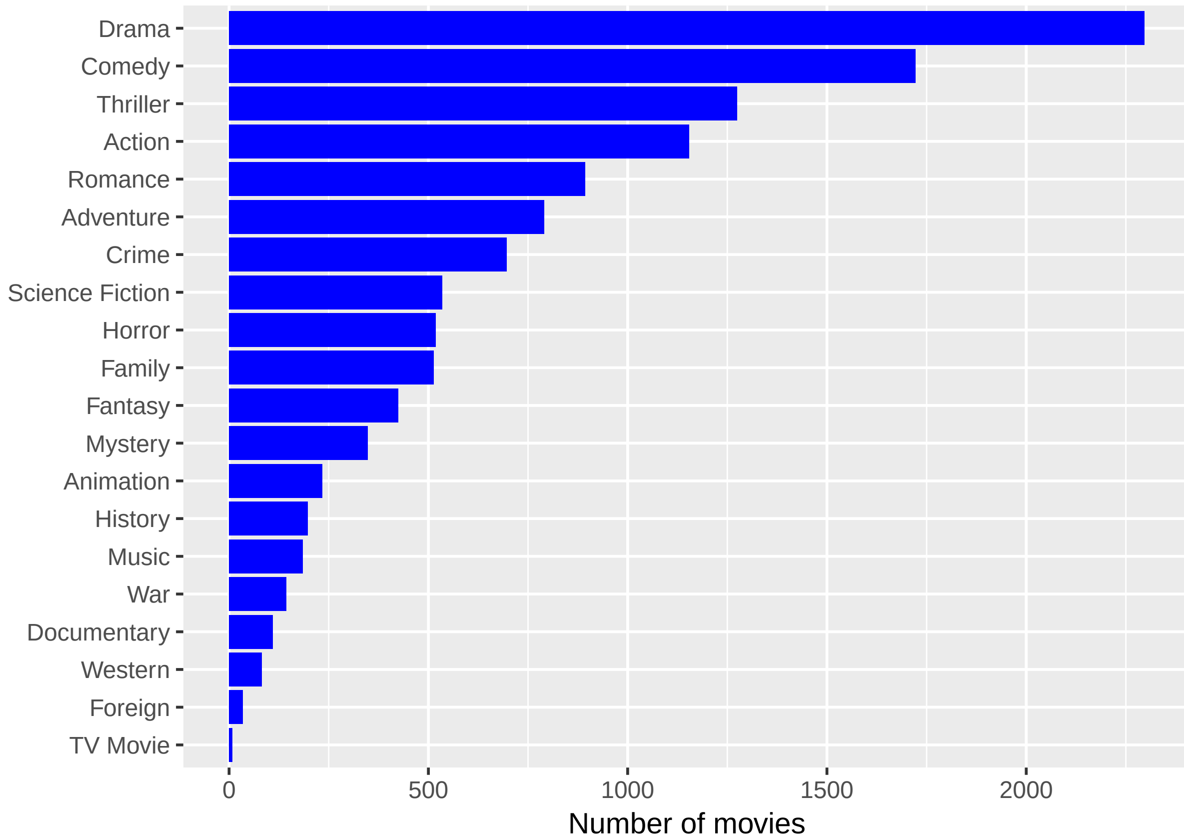
First, we draw a figure of the trend of movie numbers against time, where we find an apparent increasing trend.

```r
temp <- movie.full %>% group_by(year) %>% summarise(n = n())
p <- ggplot(temp, aes(x = year, y = n)) + geom_line() +
     labs(title = "Number of Movies V.S. Year", x = "Year", y = "Number") +
     theme(plot.title = element_text(hjust = 0.5))
p
```

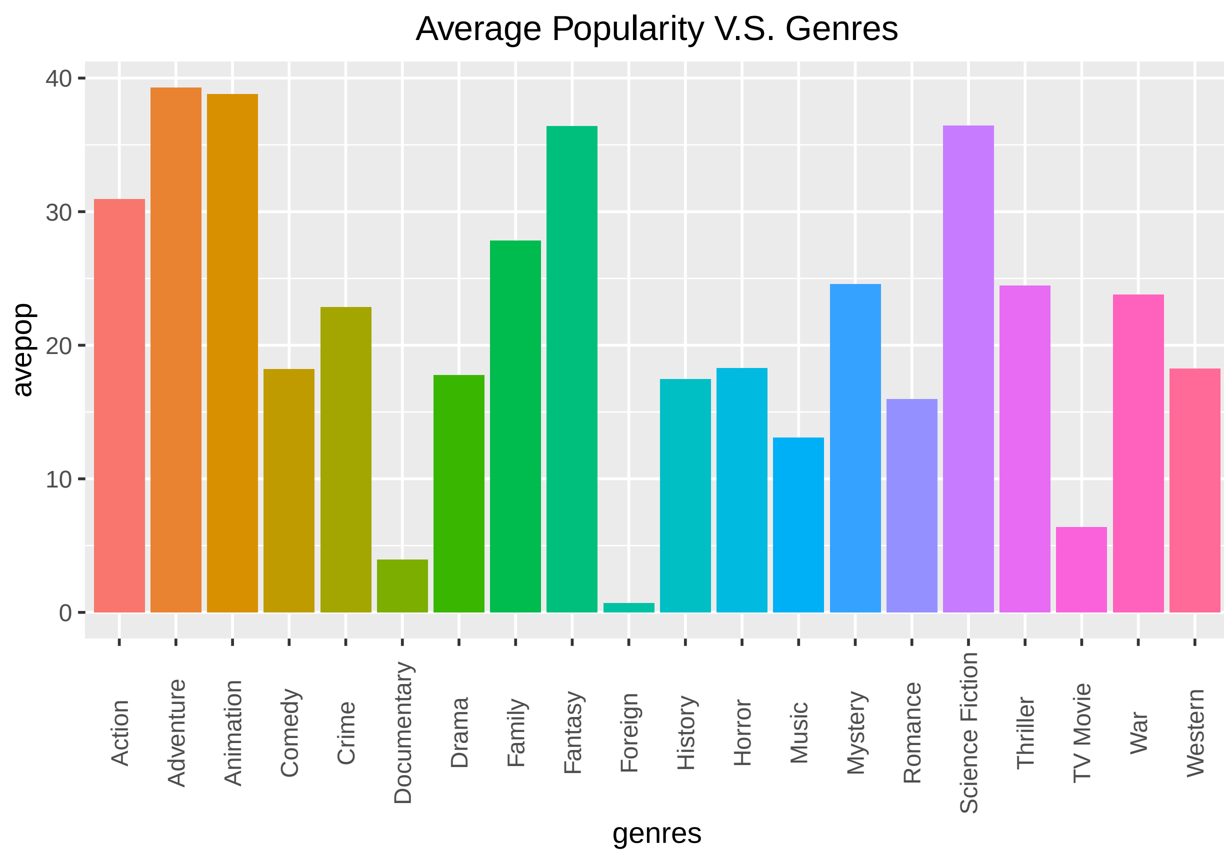


We plot the number of movies, the average revenue and the average votes grouped by genres.

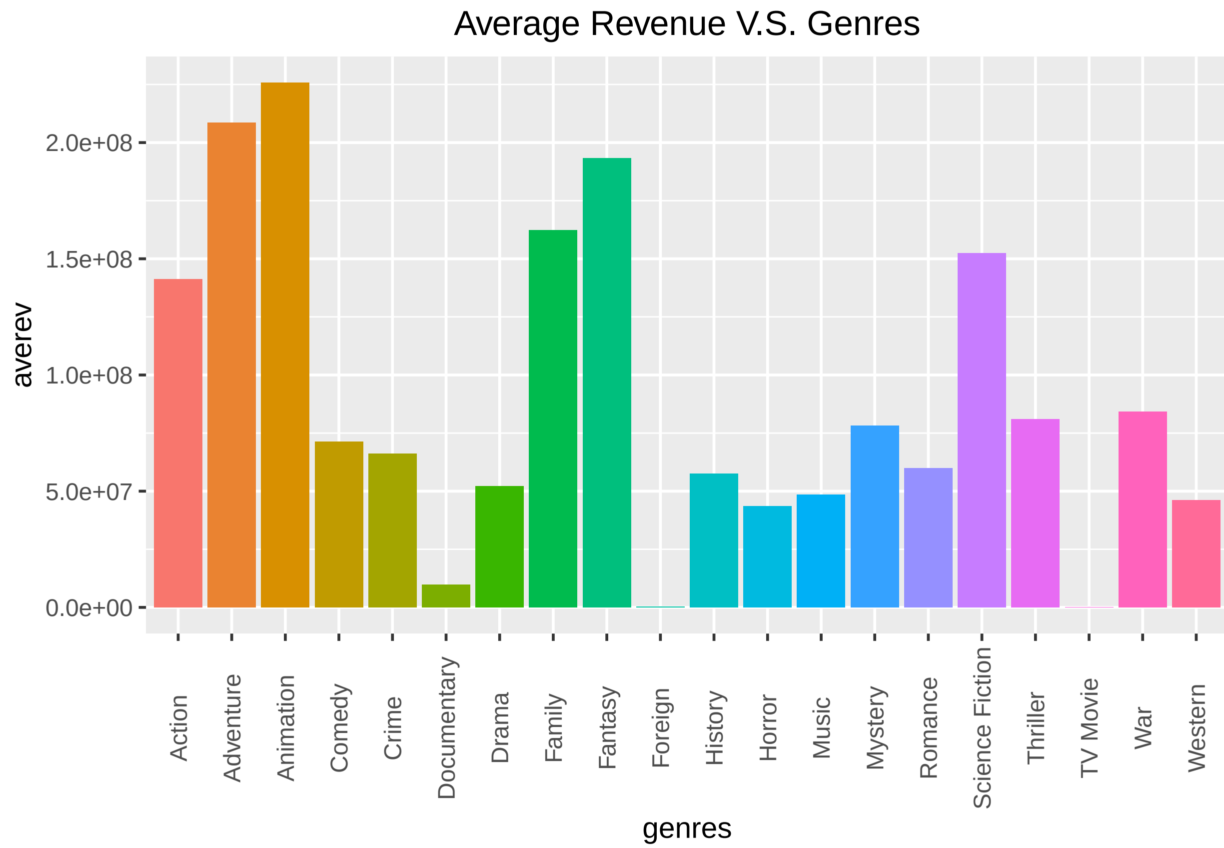
```
genres %>% group_by(genres) %>% count() %>%
        ggplot(aes(x=reorder(genres, n), y=n)) +
        geom_col(fill="blue") + coord_flip() +
        labs(x="", y="Number of movies")
```




```
temp <- movie.full %>% select(-genres, -title) %>%
          right_join(genres, by = "id") %>%
          group_by(genres) %>% summarise(avepop = mean(popularity),
                                          averev = mean(revenue))
plot1 <- ggplot(temp, aes(x = genres, y = avepop, fill = genres)) + geom_bar(stat = "identity") + theme
plot1
```

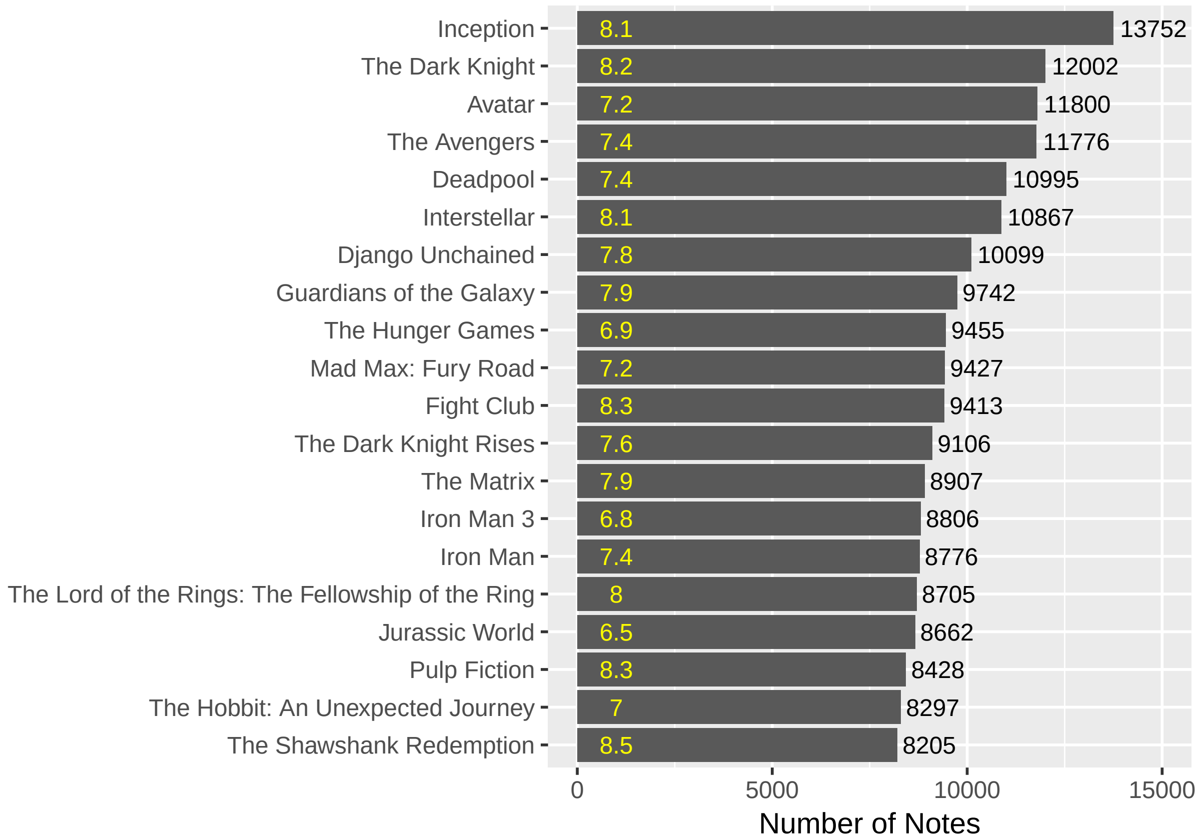



```
plot2 <- ggplot(temp, aes(x = genres, y = averev, fill = genres)) + geom_bar(stat = "identity") + theme
plot2
```



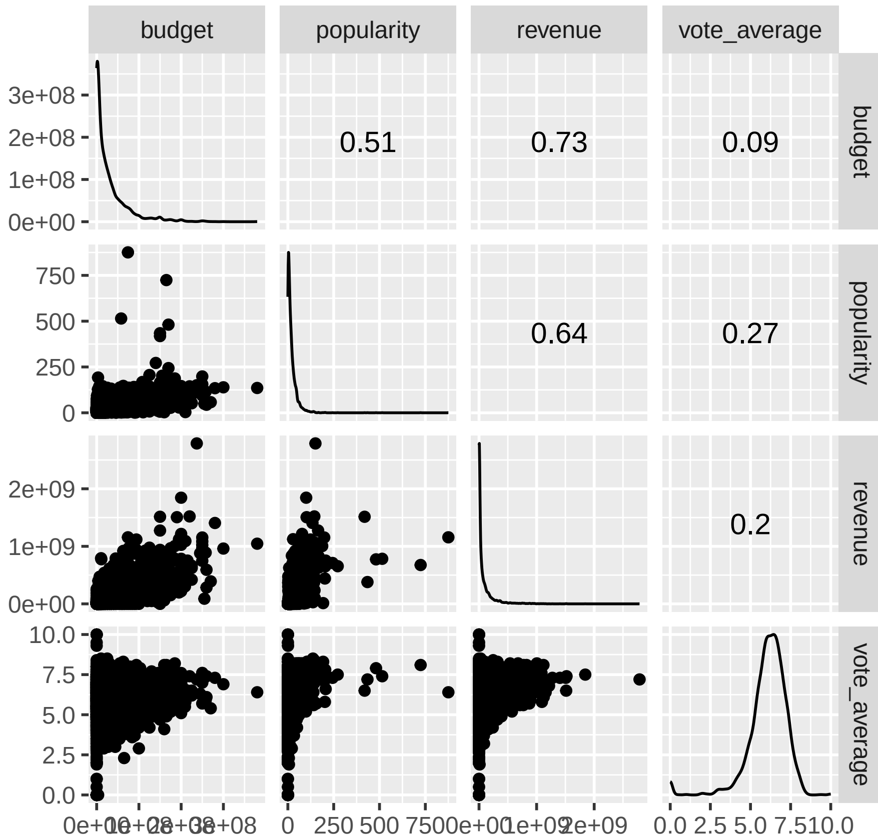
The following is the top 20 most-voted movies.

```
movie.full  %>% top_n(20, wt = vote_count) %>%
        ggplot(aes(x = reorder(title, vote_count), y = vote_count)) +
        geom_bar(stat='identity') + coord_flip(y=c(0, 15000)) +
        labs(x = "", y = "Number of Notes") +
        geom_text(aes(label = vote_count), hjust = -0.1, size = 3) +
        geom_text(aes(label = vote_average), y = 1000, size = 3, col = "yellow")
```



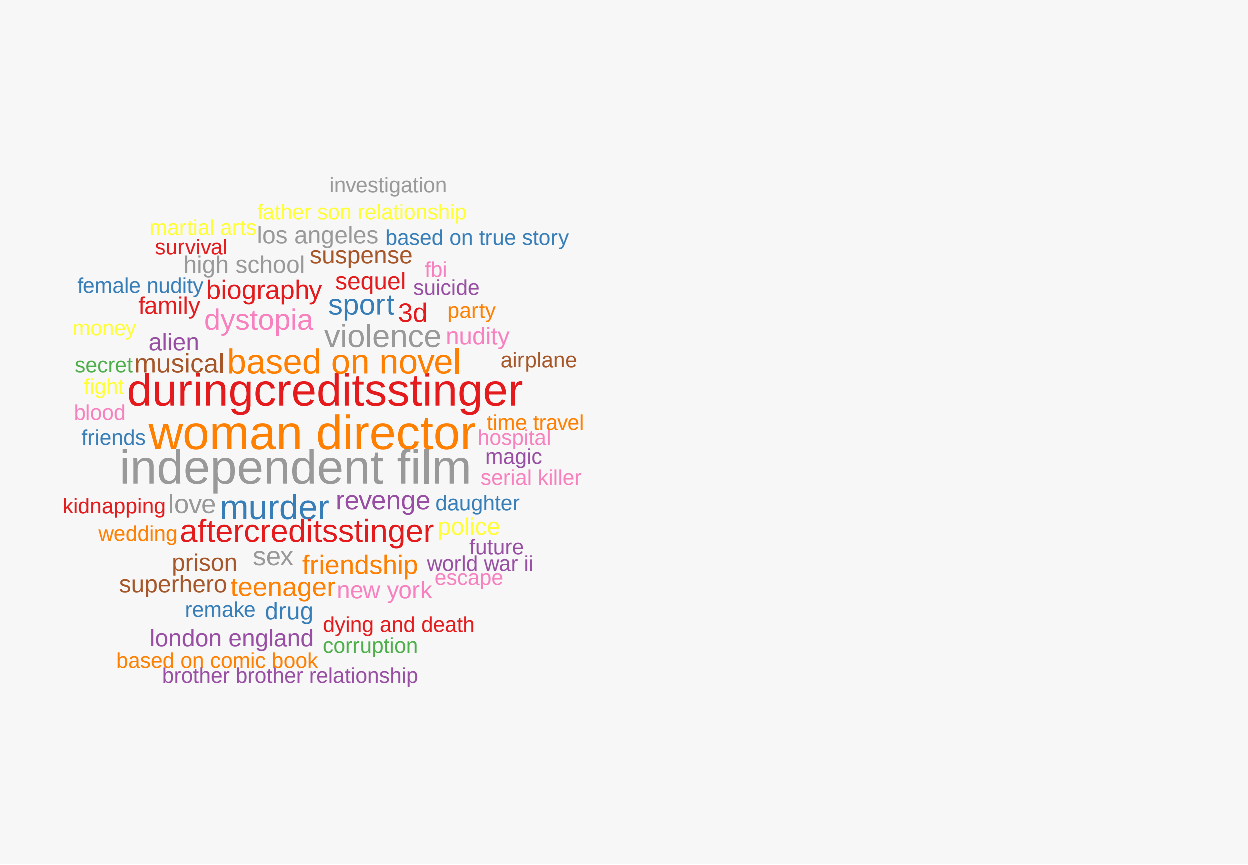
Then we draw several scatterplots to capture the correlation between variables.

```
p <- ggscatmat(movie.full %>% select(budget, popularity, revenue, vote_average, vote_average))
p
```

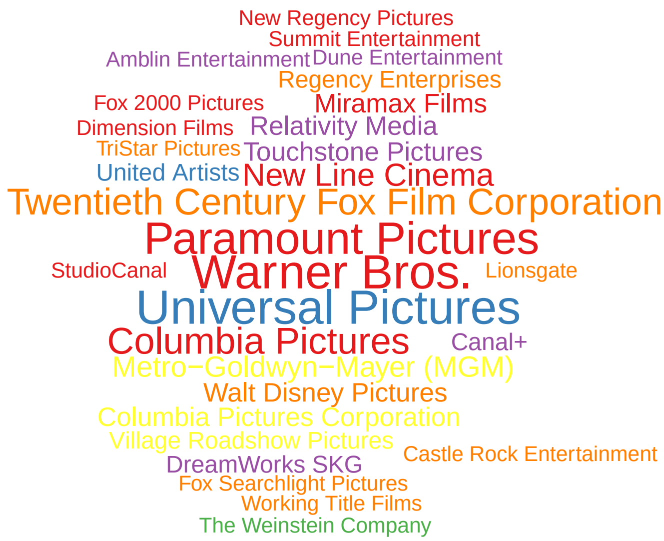


We also analyze the keywords and obtain the following wordcloud.

```
set.seed(2019)
keywords_counts <- keywords %>% count(keywords)
par(mfrow=c(1, 2),bg="grey97")
wordcloud(keywords_counts$keywords, keywords_counts$n, max.words = 60, scale=c(1.5, .5), random.color =
```




```
company_counts <- company %>% count(company)
wordcloud(company_counts$company, company_counts$n, max.words = 30, scale=c(1.5,.5), random.color = TRU
```



To construct a recommendation engine, we need to quantify the votes of each movie reasonably, thus we use the IMDB formula

$$W = \frac{v}{v+m}R + \frac{m}{v+m}C,$$

where

- W = weighted rating
- R = average rating for the movie as a number from 1 to 10 (vote_average)
- v = number of votes for the movie (vote_count)
- m = minimum votes required to be listed in the Top 250
- C = the mean vote across the whole report

```
C <- mean(movie.full$vote_average)
m <- quantile(movie.full$vote_count, 0.75)
movie.full <- movie.full %>% mutate(weighted_score = (vote_average * vote_count + C * m) / (vote_count
```

We

4. Plan for Future Analysis

For the first question, we will try various machine learning algorithms to train predictors/classifiers in order to predict how good a movie can be. One mail goal is to find the most significant features.

For the second question, we will apply different recommendation engine design, including **content-based engine**, **popularity-based engine** and **collaborative filtering engine**. The first two engines do not need ratings data, while the last one does.

For the last question, we will introduce network/graphical models to capture the connection among actors/actresses. Since the number of actors/actresses is huge, we might partition them according to release date of movies they starred.