TMDB Box Office revenue prediction

Nguyen Bao Long

8/19/2019

Contents

Abstract This document belongs to the second project - Choose Your Own in the course HarvardX: PH125.9x Data Science: Capstone. Inspired by the challenging in Kaggle's competitions, as well as my expectation to practice as much as possible data science techniques I have learned, in this project I used the dataset from Kaggle TMDb Box Office Competition and publicly additional data to develop a machine learning model to predict movie's revenue.

This documents consist 4 sections:

- Section 1: describe project background, goal, key steps and overview on the dataset.
- Section 2: describe the data pre-processing, data exploratory analysis, what's insights gained and modeling approach.
- Section 3: evaluate and discuss on the result of final models.
- Section 4: conclusion describe a brief summary of the report.

1 Introduction

1.1 Background

In general, the film-industry is having a long history of development and in a constant growth trend now. Starting in 1929, during the Great Depression and the Golden Age of Hollywood, an insight began to evolve related to the consumption of movie tickets. It appeared that even in that bad economic period, the film industry kept growing. The phenomenon repeated in the 2008 recession. The global box office was worth 41.7 billion in 2018, that would rep a 2.7% upwards shift from 2017 $40.6B and mark only the second time ever that it's cracked $40B. Revenue from North America and Canada came in at a best-ever $11.9B, a mammoth 7% spike over 2017, while attendance was up 5%.

From business point of view, one of the main interests of the film studios and its related stakeholders is a prediction of revenue that a new movie can generate based on a few given input attributes before its released date.

1.1.1 TMDB Box Office Competition

"Can you predict a movie's worldwide box office revenue?"

On February 9^{th}, 2019 Kaggle launch a competition, using the TMDB Box Office dataset to predict a movie's worldwide box office revenue. This competition was end by May 31^{th}, 2019 which total 1398 teams, 1618 competitors and 19,034 entries. Here come the introduction from Kaggle:

"In a world... where movies made an estimated $41.7 billion in 2018, the film industry is more popular than ever. But what movies make the most money at the box office? How much does a director matter? Or the budget? For some movies, it's"You had me at 'Hello.'" For others, the trailer falls short of expectations and you think "What we have here is a failure to communicate."

In this competition, you're presented with metadata on over 7,000 past films from The Movie Database to try and predict their overall worldwide box office revenue. Data points provided include cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries. You can collect other publicly available data to use in your model predictions, but in the spirit of this competition, use only data that would have been available before a movie's release." [1]

1.1.2 Competition evaluation

To evaluate results in Kaggle's competition, competitors must develop machine learning model to predict the international box office revenue for each movie. For each id in the test set, competitors must predict the value of the revenue variable.

Submissions are evaluated on Root-Mean-Squared-Logarithmic-Error (RMSLE) between the predicted value and the actual revenue. Logs are taken to not overweight blockbuster revenue movies. [1]

$$RMSLE = \sqrt{\frac{\sum_{i=1}^{n}(lny_i - ln\hat{y}_i)^2}{n}}$$

With:

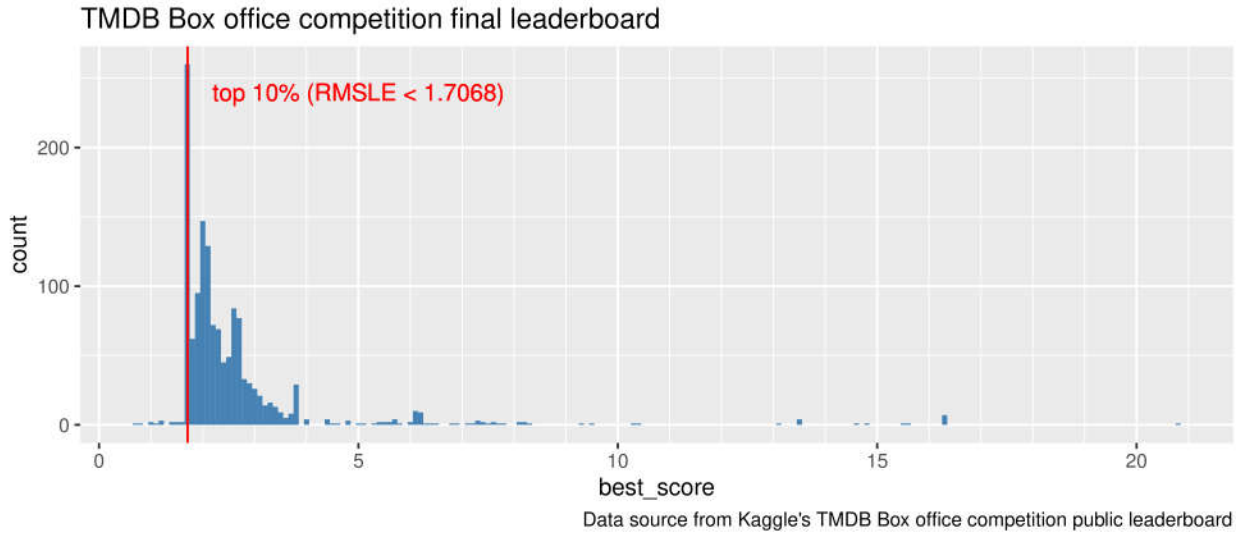
- y_i the true revenue of movie i
- $\hat{y}_i$ the predicted revenue of movie i

```r
# write function RMSLE to evaluate modeling performance
RMSLE <- function(predicted_revenue, true_revenue){
  sqrt(mean((logb(true_revenue + 1)- logb(predicted_revenue + 1))^2))
}
```

```r
# write function RMSL to evaluate modeling performance
RMSE <- function(predicted_revenue, true_revenue){
  sqrt(mean((true_revenue- predicted_revenue)^2))
}
```

Since this competition completed on May 31^{st}, following is the final public leaderboard.

```r
final_board %>%
  ggplot(aes(best_score)) +
  geom_histogram(fill = "steel blue", binwidth = 0.1) +
  scale_x_continuous(minor_breaks = seq(0, 30, 1)) +
  geom_vline(xintercept = 1.7068, color = "red") +
  geom_text(aes(x = 5, y = 240, label = "top 10% (RMSLE < 1.7068)"), color = "red") +
  labs(title = "TMDB Box office competition final leaderboard",
       caption = "Data source from Kaggle's TMDB Box office competition public leaderboard")
```



Noted: even Kaggle's competition rules mentioned *"use only data that would have been available before a movie's release"*, many competitors trained there model with the data available after the movie's released date. [2]

1.2 Project objective

The primary goal is to build a machine-learning model to predict the revenue of a new movie given such features as budget, release dates, genres, production companies, production countries... The modeling performance is evaluating based on the RMSLE, which is same with the Kaggle TMDb Box Office Competition. There is no targeted RMSLE required to achieve in this project.

The secondary goal is to practice skills data wrangling, data visualization, reporting using stringr, dplyr, ggplot2, rmarkdown, kableExtra.

1.3 Project methodology

This project has 4 high-level steps:

- Step 1: **data overview and pre-processing** overviewing and cleaning data using additional data from The Movie Database and Wikipedia.
- Step 2: **data exploratory analysis and features engineering** explore and visualize the data to have an overview with-in and between the variables, what's insights gained and what's new features added in. Main package for this step is tidyverse, to handle the cleaning, exploring and visualizing tasks. Output of this step is a set of variables for modeling experiment and training.
- Step 3: **modeling experiments** design and conduct a set of experiments to evaluate performance and select machine learning method, compare and select features selection approach, evaluate modeling performance before and after apply log-transformation.
- Step 4: **final evaluate the model** on the validation set using RMSLE.

1.4 Data overview

1.4.1 Kaggle competition's dataset

The dataset used in this competition has been collected from the Movies Database [7]. The movie details, credits and keywords have been collected from the TMDB Open API. This competition uses the TMDB API but is not endorsed or certified by TMDB. Their API also provides access to data on many additional movies, actors and actresses, crew members, and TV shows.

This data contain 7398 movies and a variety of metadata. Movies are labeled with id. Data points include cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries. The dataset was subset to 2 dataset, the `train.csv` has 3000 movies and `test.csv` with 4398 movies.

All datas which were used in this project, are able to download from my github repo.

Import data

```
# download raw data from my github repo
url_train <-
  url("https://raw.githubusercontent.com/billynguyenlss/TMDB-Box/master/data/train.csv")
train_set <- read.csv(url_train, na.strings=c("", '#N/A', '[]', '0'))

url_test <-
  url("https://raw.githubusercontent.com/billynguyenlss/TMDB-Box/master/data/test.csv")
test_set <- read.csv(url_test, na.strings=c("", '#N/A', '[]', '0'))

url_sample_submission <-
  url("https://raw.githubusercontent.com/billynguyenlss/TMDB-Box/master/data/sample_submission.csv")
sample_submission <- read.csv(url_sample_submission, na.strings=c("", '#N/A', '[]', '0'))
```

Data overview

The **train_set** have 3000 observations of 23 variables.

```
dim(train_set)
```

```
## [1] 3000   23
```

The **test_set** has 4398 observations of 22 variables.

```
dim(test_set)
```

```
## [1] 4398   22
```

Combine train and test dataset to one data to save time for any data pre-processing and transforming later on.

```
# combine train_set and test_set to reduce time to pre-processing data
train_set <- train_set %>% mutate_if(is.factor, as.character)
test_set <- test_set %>% mutate_if(is.factor, as.character)
df <- bind_rows(train_set, test_set)
```

Lets take a glimpse at Kaggle's dataset to get a feel of how it looks like.

```
glimpse(df)
```

```
## Observations: 7,398
## Variables: 23
## $ ï..id                 <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1...
## $ belongs_to_collection <chr> "[{'id': 313576, 'name': 'Hot Tub Time M...
## $ budget                <int> 14000000, 40000000, 3300000, 1200000, NA...
## $ genres                <chr> "[{'id': 35, 'name': 'Comedy'}]", "[{'id...
## $ homepage              <chr> NA, NA, "http://sonyclassics.com/whiplas...
## $ imdb_id               <chr> "tt2637294", "tt0368933", "tt2582802", "...
## $ original_language     <chr> "en", "en", "en", "hi", "ko", "en", "en"...
## $ original_title        <chr> "Hot Tub Time Machine 2", "The Princess ...
## $ overview              <chr> "When Lou, who has become the \"father o...
## $ popularity            <dbl> 6.575393, 8.248895, 64.299990, 3.174936,...
## $ poster_path           <chr> "/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg", "/w9...
## $ production_companies  <chr> "[{'name': 'Paramount Pictures', 'id': 4...
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'United S...
## $ release_date          <chr> "2/20/15", "8/6/04", "10/10/14", "3/9/12...
## $ runtime               <int> 93, 113, 105, 122, 118, 83, 92, 84, 100,...
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'English'}...
## $ status                <chr> "Released", "Released", "Released", "Rel...
## $ tagline               <chr> "The Laws of Space and Time are About to...
## $ title                 <chr> "Hot Tub Time Machine 2", "The Princess ...
## $ Keywords              <chr> "[{'id': 4379, 'name': 'time travel'}, {...
## $ cast                  <chr> "[{'cast_id': 4, 'character': 'Lou', 'cr...
## $ crew                  <chr> "[{'credit_id': '59ac067c92514107af02c8c...
## $ revenue               <int> 12314651, 95149435, 13092000, 16000000, ...
```

Those variables are classifying to two main type of data, character/text and numeric/integer.

- The character/text variables are belong to collection, genres, crew, cast... It is worth noting that the text structure of those character variables look like complicated. For example in the first value of genres "[{'id': 35, 'name': 'Comedy'}]", , only "Comedy" is informative. This first problem leads to a required pre-processing step to remove all unnecessary character and/or extract only the required data for analysis.
- The numeric variables are budget, revenue, popularity, runtime.

Following is the summary table on variables:

Variables	Type	Description
budget	integer	Budget of a movie in American Dollar (USD) and not be adjusted for inflation.
popularity	numeric	Popularity was based on user interactions on the TMDb website
runtime	integer	Duration in minutes of a movie
revenue	integer	Revenue of a movie in American Dollar (USD) and not be adjusted for inflation. Recommended resources for box office revenue information: Box Office Mojo and The Numbers.
release_date	date / time	Release date of a movie
original_language	character	The original language of a movie
original_title	character	The original title is usually the title of the original version of the film when it is first officially released locally.
overview	character	Overviews should describe the plot of the movie. They should be to the point, spoiler-free and brief. A few lines at most.
poster_path	character	link to movie's póter
production_companies	character	list of production companies
production_countries	character	list of production countries
spoken_languages	character	Only the languages spoken in the original version. No translated/dubbed languages.
tagline	character	A movie tagline is usually a short promotional text used on the poster.
Keywords	character	to describe the plot of movie. Around 5-10 keywords for TV zshows and 15-20 keywords maximum for movies is reasonable.
cast	character	Only the cast of the original (not dubbed or extended) version.
crew	character	Only the crew credited in the original version.

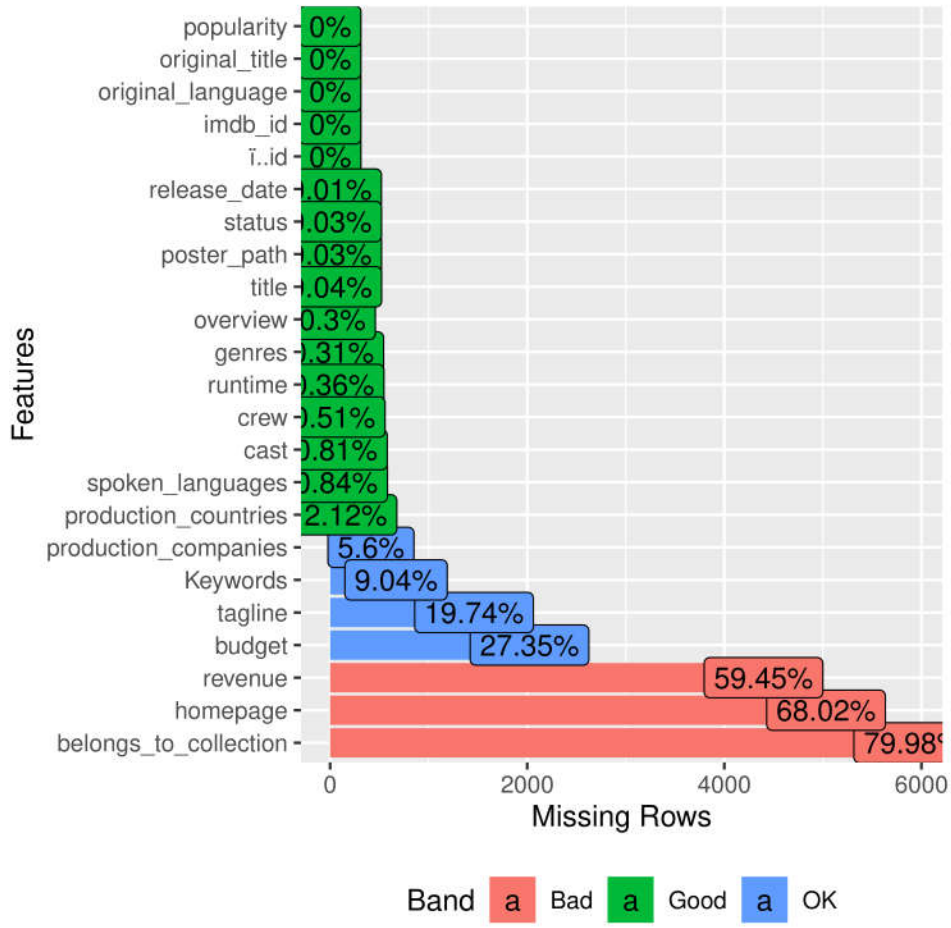
More description on variables could be reference from following link

https://www.themoviedb.org/bible/movie/

NA value status

Missing values must be dropped or replaced in order to draw correct conclusion from the data. The second problem of this dataset is missing value in many variables.

```
plot_missing(df)
```



Two highest NA ratio variables are`belongs_to_collection` and `homepage`, which are more than 60%. `budget`, which suppose the most important variables, also have 27.07% NA values. Other `tagline`, `Keywords`, `production_companies` have lower NA value ratio, less than 20%.

Since there are many available publicly dataset in internet on budget number, movies description... as well as the competition rules allow to use any publicly dataset which are available before the movie's release data. I decide to use the additional data from The Movie Database itself and Wikipedia website to fulfill all missing values.

1.4.2 Additional dataset

The third problem is that there were many concern on the quality of dataset provided in this Kaggle competition, especially the accuracy of budget, revenue, popularity... [2]

In addition, the moment I have been working in this project later on the Competition timeline. Therefore, this might lead to unforseen circumstance due to inaccuracy data. Hence, to assure the accuracy and consistency of the dataset, in this project I set priority to use the additional TMDb data at first, then the second priority is Kaggle's dataset, the Wikipedia is to use ultimately. The publicly additional data are from following sources:

- the Movies Database (https://www.themoviedb.org/)
- Wikipedia (https://www.wikipedia.org/)

1.4.2.1 The Movies Database API

The additional data from TMDB website is openned and able to download by following their instruction.

The instruction to retrieve data from TMDb website is reference from following link (http://www.planetanalytics.in/2017/05/how-to-extract-movie-data-from-movie.html).

The TMDb packages was also available in R libraries [10] which you can retrieve other features for this competition.

Import data

Due to very long processing time to download data from TMDb website, I did not include the code in this report, but you can download data direct from my github repo.

```
# download additional data from my github repo
url_additional <- url("https://raw.githubusercontent.com/billynguyenlss/TMDB-Box/master/data/personal%2
additional_data <- read.csv(url_additional, na.strings=c("", '#N/A', '[]', '0'))
```

Data overview

This data set have 7398 observations of 14 variables.

```
glimpse(additional_data)
```

```
## Observations: 7,398
## Variables: 14
## $ imdb_id          <fct> tt1380152, tt0391024, tt0117110, tt0093857, t...
## $ new_budget       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ vote_count       <int> 10, 30, 170, 117, 114, 626, 15, 331, 55, 450,...
## $ vote_average     <dbl> 5.6, 7.4, 6.4, 5.4, 6.0, 5.9, 5.0, 5.6, 5.5, ...
## $ new_popularity   <dbl> 4.077, 2.293, 8.695, 6.974, 6.378, 10.905, 1....
## $ new_revenue      <dbl> 3923970, 2586511, 34327391, 22642033, 1234254...
## $ new_runtime      <int> 118, 84, 100, 98, 111, 116, 92, 87, 95, 117, ...
## $ new_release_date <fct> 2/5/2009, 1/15/2004, 2/16/1996, 7/10/1987, 12...
## $ genres1          <fct> Adventure, Adventure, Drama, Action, Horror, ...
## $ genres2          <fct> Family, Comedy, Comedy, Crime, Thriller, Anim...
## $ genres3          <fct> Horror, Music, Drama, Drama, Mystery, Science...
## $ company1         <fct> Ghost House Pictures, Amercent Films, DreamWo...
## $ company2         <fct> North Box Productions, 20th Century Fox, Doub...
## $ company3         <fct> Walt Disney Pictures, Jinks/Cohen Company, Je...
```

Summary table of additional data:

Variables	Type	Description
imdb_id	character	identified number in imdb system, extract from Kaggle's dataset
new_budget	integer	the budget number downloaded from TMDb
vote_count	integer	the number of vote given by users after movie was released
vote_average	double	the average vote given by users after movie was released
new_popularity	double	the popularity number downloaded from TMDb
new_revenue	integer	the revenue downloaded from TMDb
new_runtime	integer	runtime number downloaded from TMDb
new_release_date	date/time	release date downloaded from TMDb
genres1	character	first genre of a movie
genres2	character	second genre of a movie

Variables	Type	Description
genres3	character	third genre of a movie
company1	character	first company of a movie
company2	character	second company of a movie
company3	character	third company of a movie

1.4.2.2 Wikipedia

The second additional data was scrapped from wikipedia, using `rvest` package. Because of long processing time to scrapping and pre-processing, I didn't copied those scrapping code in my report. If you're interested with this code, you can refer it and/or download the completed data from my github repo. The code to download and import the additional data was as following:

Import data

```
# download data from my github repo
url_wikipedia_budget <-
  url("https://raw.githubusercontent.com/billynguyenlss/TMDB-Box/master/wikipedia_us_budget.csv")

wikipedia_budget <- read.csv(url_wikipedia_budget, na.strings=c("", '#N/A', '[]', '0'))

# to assure the final budget as numeric type
wikipedia_budget$final_budget <- as.numeric(wikipedia_budget$final_budget)
```

Overview data

```
glimpse(wikipedia_budget)
```

```
## Observations: 1,842
## Variables: 5
## $ imdb_id       <fct> tt3950078, tt0099581, tt1440292, tt0456020, tt12...
## $ wiki_budget   <fct> "US$35 Million", "US$35 Million", "US$35 Million...
## $ new_budget    <dbl> 35, 35, 35, 35, 35, 35, 25000, 25000, 25000, 250...
## $ currency_unit <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ final_budget  <dbl> 3.5e+07, 3.5e+07, 3.5e+07, 3.5e+07, 3.5e+07, 3.5...
```

Summary table of wikipedia dataset:

Variables	Type	Description
imdb_id	character	the identified number in imdb system
wiki_budget	character	the original scrapping from wikipedia
new_budget	integer	extracted number from wiki_budget (unit probably single, thousand, million usd/jpy/cad/...)
currency_unit	integer	currency exchange number from other currency to usd
final_budget	double	final budget after transforming to usd

1.4.3 Data overview summary

In summary, top 3 problems were recognized in this section and the respective solution to deal with those problems were as following:

No	Problem	Solution
1	missing/NA values in many variables, especial in budget	use new budget from additional data (TMDb & wikipedia) to update/replace for missing/wrong value
2	complicated text structure in character variables (belongs_to_collection, genres, crew, cast, production_companies...)	use text mining function in package `stringr` to replace unnecessary characters and/or extract the required data for analysis
3	inaccuracy data	priority to use data from additional data (TMDb & wikipedia) instead of original data from Kaggle competition

2 Analysis

2.1 Data exploratory analysis

2.1.1 Release date

The first noting that release data was as type character, then we have to convert it to date-and-time format.

```
df$release_date <- mdy(df$release_date)
summary(df$release_date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "1969-01-01" "1995-09-15" "2006-02-19" "2005-01-15" "2012-08-31"
##         Max.         NA's
## "2068-12-30"          "1"
```

The second noting that the time frame was really wide, the soonest released date was 1969-01-01 and the latest was "2068-12-30". In addition, the latest release_date "2068-12-30" was impossible. Moreover, there were total 357 days later than the time I was doing this report on August 27^{th}, 2019 and 1 NA value. Hence, it's certainly not enough to assure the original release date was reliable or not.

For all these reasons, I used the release date in TMDb additional data set to replace the original release data from Kaggle data.

```
# left joining the addition data into df
df <- df %>% left_join(additional_data, by = "imdb_id")
df$new_release_date <- as.character(df$new_release_date)

df$release_date <- df$new_release_date
df$release_date <- mdy(df$release_date)
```

2.1.2 Revenue

```
summary(train_set$revenue)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 1.000e+00 2.380e+06 1.681e+07 6.673e+07 6.892e+07 1.520e+09
```

First of all, we could see the minimum revenue was 1 and the maximum revenue was 1519557910. In a little more detail,, there were total 57 movies had revenue value less than 1000. Those abnormal revenue number's caused by the currency unit was recorded in million USD while almost movies currency unit was recorded in single USD.

Hence, to assure all revenue number was up-to-date and its reliability in my project, in like manner to budget number, I used the revenue from TMDb data to replace the original revenue number in Kaggle data.

```
df <- df %>%
  mutate(new_revenue = ifelse(is.na(new_revenue), revenue, new_revenue))

df$revenue[1:3000] <- df$new_revenue[1:3000]
```

The revenue number was used to evaluate the final modeling performance was also extracted as following:

```
# extract the true_rating value from additional data
test_y <- test_set %>% left_join(additional_data, by = "imdb_id") %>%

  # replace any NA value by median value
  mutate(new_revenue = ifelse(is.na(new_revenue), median(new_revenue, na.rm = T), new_revenue)) %>%
  pull(new_revenue)
```

2.1.3 Budget

Certainly, budget would be the most important variable to predict revenue of a movie.

```
summary(df$budget)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
##         1   5053316  17000000  31108016  40000000 380000000      2023
```

Firstly, we could see 2023 NA values, approximate 27.3 % of total observation. Then to deal with those NA values, I replaced them by the budget number from additional data, which were downloaded from The Movie Database and Wikipedia.

Secondly, the minimum budget is just 1, this is the same issue and caused by the same reason similar to abnormal revenue number in previous section. In addition, some movies have budget recorded in million USD (value less than $1,000), but its revenue recorded in single USD (value up to $1,000,000). In contrast, some movies have budget record in single USD (value up to $1,000,000) but its revenue recorded in million USD (value less than $1,000).

On the whole, I summarized all issues related to budget number as following table.

```
train_set %>% dplyr::select(budget, revenue) %>%
  mutate(budget_status = ifelse(is.na(budget), "missing budget",
                                ifelse(budget <=1000,"low budget","normal budget")),
         revenue_status = ifelse(revenue <= 1000,"low revenue","normal revenue")) %>%
  group_by(budget_status, revenue_status) %>%
  summarize(count = n(),
            percent = 100*round(n()/nrow(train_set),3)) %>%
  mutate(remarks = ifelse(count == 5,"wrong budget",
                   ifelse(count == 34, "NA impact",
                   ifelse(count == 778, "NA impact",
```

```
                    ifelse(count == 10, "wrong revenue","-"))))) %>%
  kable(caption = "Summarized table of abnormal budget and revenue") %>%
  kable_styling(latex_options = c("HOLD_position"),
                bootstrap_options = c("striped", "hover"),
                full_width = F, position = "center")
```

Table 5: Summarized table of abnormal budget and revenue

budget_status	revenue_status	count	percent	remarks
low budget	low revenue	13	0.4	-
low budget	normal revenue	5	0.2	wrong budget
missing budget	low revenue	34	1.1	NA impact
missing budget	normal revenue	778	25.9	NA impact
normal budget	low revenue	10	0.3	wrong revenue
normal budget	normal revenue	2160	72.0	-

Hence, to prevent those mistakes impact to the performance of final predicting model, I replaced them by the value from additional data (TMDb & Wikipedia).

New budget

Firstly, I replaced abnormal budget numbers (value less than $1,000) if its respective revenue was normal (value greater than $10,000). Then, I replace the NA value. This step dropped NA value from 27.07% to 12%.

```
# joining addition wikipedia budget
df <- df %>% mutate(imdb_id = as.factor(imdb_id)) %>%
  left_join(wikipedia_budget[,c(1,2,5)],by = "imdb_id")
```

```
## Warning: Column `imdb_id` joining factors with different levels, coercing
## to character vector
```

```
#replace low budget value
df <- df %>%
  mutate(new_budget = ifelse((new_budget <= 1000 & revenue > 10000), final_budget, new_budget))

# replace NA budget value by wikipedia budget
df <- df %>% mutate(new_budget = ifelse(is.na(new_budget),
                                        final_budget, new_budget))

mean(is.na(df$new_budget))
```

```
## [1] 0.1197621
```

The NA value was reduced from 27.07% to 12% . Because the Wikipedia budget number is not available for every movies, therefore I replaced remain NA value by median budget value. The median budget values were calculated by each released year to minimize impact of time.

```
# create summary budget table by release year
df$release_year <- year(df$release_date)
```

```
budget_by_year <- df %>%
  group_by(release_year) %>%
  summarize(avg_budget = mean(new_budget, na.rm = T),
            median_budget = median(new_budget, na.rm = T))

# replace NA budget by median_budget value
df <- df %>% left_join(budget_by_year, by = "release_year") %>%
  mutate(new_budget = ifelse(is.na(new_budget), median_budget,
                             new_budget))
df$new_budget[is.na(df$new_budget)] <- df$median_budget
```

```
## Warning in df$new_budget[is.na(df$new_budget)] <- df$median_budget: number
## of items to replace is not a multiple of replacement length
```

```
# re-check NA value in budget
mean(is.na(df$new_budget))
```

```
## [1] 0
```

2.1.4 Runtime

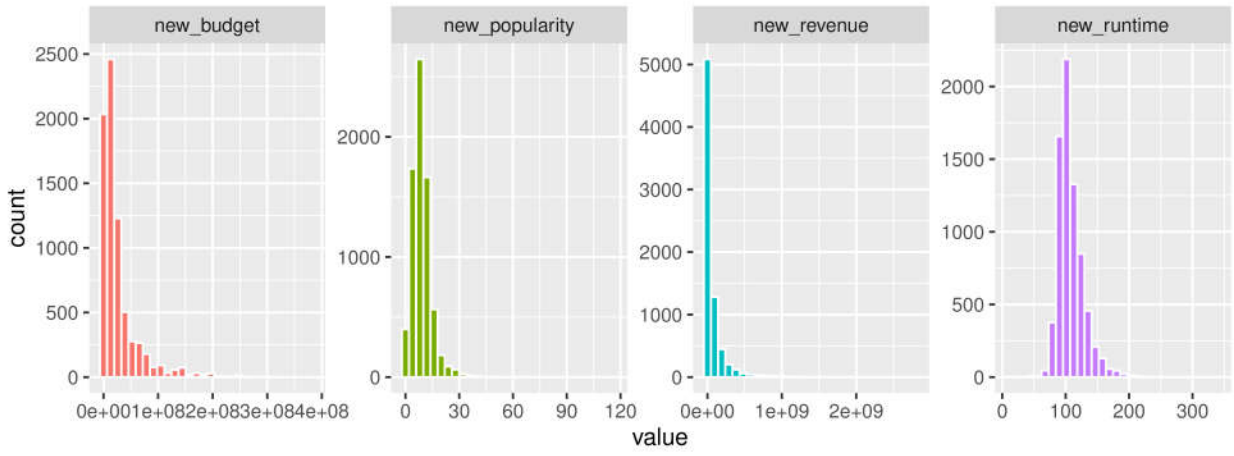
The missing values in `runtime` was to be replaced by median run time value.

```
df$runtime[is.na(df$runtime)] <- median(df$runtime, na.rm = T)
```

Now we can take the first look on the corrected value of budget, revenue, popularity and runtime.

```
df %>% dplyr::select(new_revenue, new_budget, new_popularity, new_runtime) %>%
  gather(1:4, key = "variables", value = "value") %>%
  ggplot(aes(value, fill = variables)) +
  geom_histogram(color = "white") + facet_wrap(.~variables, scales = "free", ncol = 4) +
  theme(legend.position = "none")
```

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



We could see:

- Not many movies have high revenue. 75.1 % movies in the training dataset have revenue lower than the average revenue. Only 24.9 % movies have revenue greater than the average revenue.
- Budget, revenue, popularity seem to have positive skew distribution.

2.1.5 Profit ratio & Return on Investment (ROI)

Return on investment (ROI) is a ratio between net profit and cost of investment. A high ROI means the investment's gains compare favorably to its cost. As a performance measure, ROI is used to evaluate the efficiency of an investment or to compare the efficiencies of several different investments.

```
# calculate average_profit_ratio
avg_profit_ratio <- sum(df$revenue[1:3000], na.rm = T)/sum(df$new_budget[1:3000])
avg_profit_ratio
```

```
## [1] 2.599791
```

In average, the profit ratio of a movie was 260%.

To evaluate the ROI of different movies, following was the summarized table of top 10 highest ROI movies:

```
df %>% mutate(ROI = (revenue - budget)/budget) %>%
  dplyr::select(title, budget, revenue, ROI, popularity) %>%
  arrange(desc(ROI)) %>% head(10) %>%
  kable(caption = "Top movies with greatest ROI", digits = 2) %>%
  kable_styling(latex_options = c("HOLD_position"),
                bootstrap_options = c("striped", "hover"),
                full_width = F, position = "center")
```

Table 6: Top movies with greatest ROI

title	budget	revenue	ROI	popularity
Paranormal Activity	15000	193355800	12889.39	12.71
The Blair Witch Project	60000	248000000	4132.33	14.84
The Tiger: An Old Hunter's Tale	5000	11083449	2215.69	3.45
Madea Goes to Jail	80000	90508336	1130.35	2.84
Fuck You Goethe 2	121000	83027924	685.18	9.22
Pink Flamingos	12000	6000000	499.00	5.74
Gunfight at the O.K. Corral	25000	11750000	469.00	10.46
We Are from the Future 2	30000	8910819	296.03	1.22
Beaches	200000	57041866	284.21	16.18
The Legend of Boggy Creek	100000	22000000	219.00	1.52

The most profitable movie was "Paranormal Activity", which was a 2007 American supernatural horror film co-produced, written, directed, photographed and edited by Oren Peli. It centerred on a young couple (Katie Featherston and Micah Sloat) who were haunted by a supernatural presence in their home. The film had earned nearly $108 million at the U.S. box office and a further $85 million internationally for a worldwide total of $193 million [8].

```r
which(df$title == "Paranormal Activity")
```

```
## [1] 1231
```

The second most profitable movie is "The Blair Witch Project". Reference from wikipedia, The Blair Witch Project grossed nearly $250 million worldwide on a modest budget of $60,000, making it one of the most successful independent films of all time [9].

```r
which(df$title == "The Blair Witch Project")
```

```
## [1] 1680
```

Not too many movies could be successful like both movies. To drop out its impact to final predicting model, I noted its row indexes in a vector named outlier_rows.

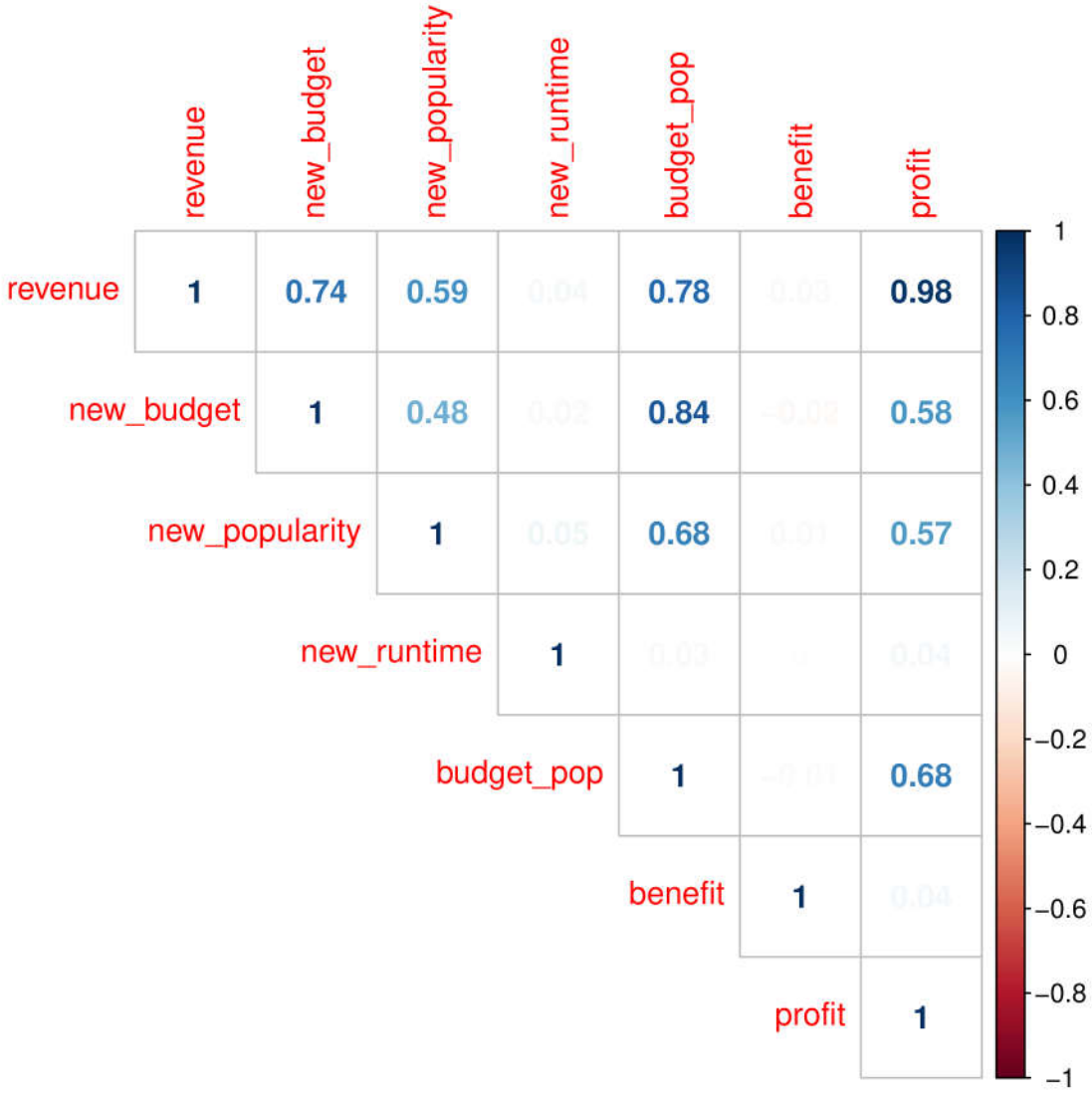
```r
outlier_rows <- c(1231,1680)
```

2.1.6 Correlation between revenue and budget, popularity, runtime

To evaluate the correlation and interaction between revenue and budget, popularity, runtime, I added new features includes:

- `budget_pop` represent for interaction between budget and popularity.
- `benefit` represent for the ratio between revenue and budget.
- `profit` represent for the margin between revenue and budget.

```r
# create correlation plot between revenue, budget, popularity

corrplot(cor(df[1:3000,] %>%
  # select separated variables
  dplyr::select(revenue, new_budget, new_popularity,
         new_runtime) %>%

   # to evaluate the correlation between variables
  mutate(budget_pop = new_budget*new_popularity,
         benefit = revenue/new_budget,
         profit = (revenue - new_budget))
),
type = "upper", method = "number")
```



We could see:

- revenue was highest correlated to budget_pop (0.78), and new_budget (0.74).
- it's surprise that revenue/budget ratio seem not to correlate to any variables.
- runtime seem not to correlate to revenue and other variables.

The following figures used to visualize the correlation between revenue vs budget, popularity:

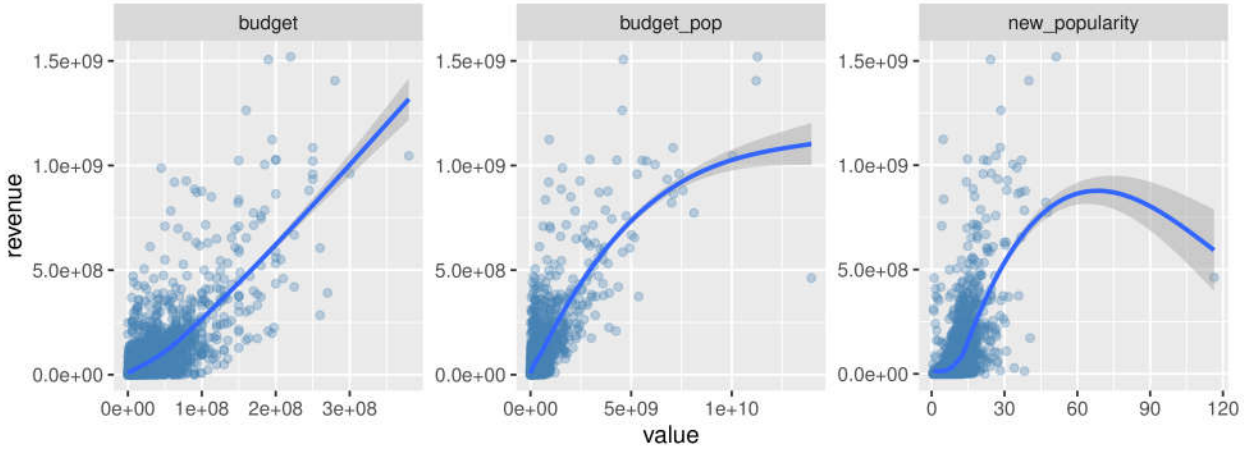
```
df %>% mutate(budget_pop = new_budget*new_popularity) %>%
  dplyr::select(revenue, new_popularity, budget, budget_pop) %>%
  gather(2:4, key = "variables", value = "value") %>%
```

```
  ggplot(aes(value, revenue)) +
  geom_point(color = "steel blue", alpha = 0.3) +
  geom_smooth() + facet_wrap(.~variables, ncol = 3, scales = "free")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 13194 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 13194 rows containing missing values (geom_point).
```



We could see:

- The scatter between budget & revenue was more diffuse than the scatter between budget_pop & revenue.
- One popularity outlier (value > 100) break the smooth fitting line. The row index of this outlier could be determined by following code:

```
# determine popularity outlier rows
which(df$new_popularity > 100)
```

```
## [1] 2859
```

Adding this outlier to vector outlier_rows.

```
# add popularity outier rows to outlier_rows vector
outlier_rows <- c(outlier_rows, which(df$new_popularity > 100))
```

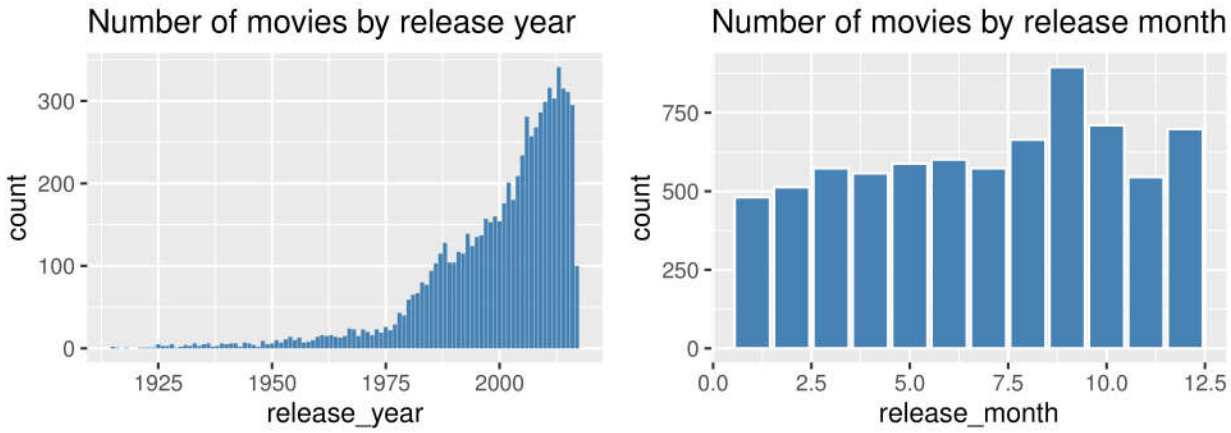
2.1.7 Change across time

```
summary(df$release_date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "1915-02-08" "1992-10-26" "2004-07-15" "2000-02-25" "2011-06-17"
##         Max.
## "2017-08-11"
```

The soonest release date was "1915-02-08" and the latest release date was "2017-08-11", the period of time were 97 years. Consumer behaviours have changed over the years: the MeToo movement, as well as other social developments, have surfaced in our society, and that would impact on the movies box office industry as well.

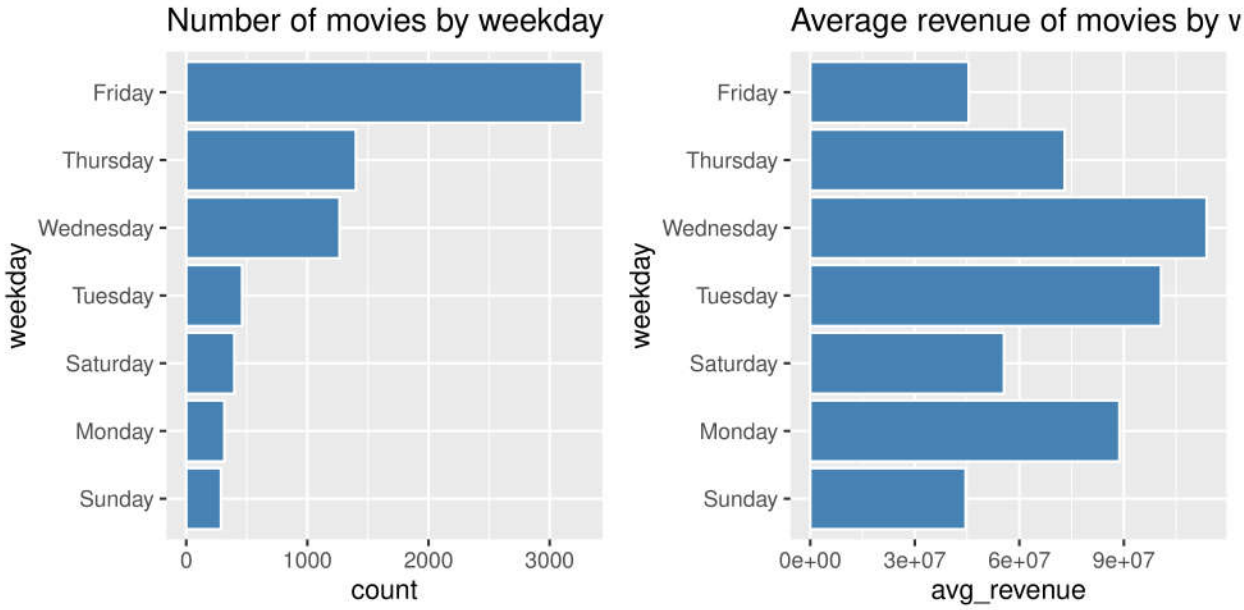
```
# calculate the release month
df$release_month <- month(df$release_date)
df$weekday <- weekdays(df$release_date)

# visualize the different number of movies by release year and release month
grid.arrange(
  df %>%
    ggplot(aes(release_year)) +
    geom_bar(fill = "steel blue") +
    labs(title = "Number of movies by release year"),

  df %>% ggplot(aes(release_month)) +
    geom_bar(fill = "steel blue", color = "white") +
    labs(title = "Number of movies by release month"),

  ncol = 2)
```



We could see:

- The movies box office industry growth slowly before decade 1980s , after this time it's continuous growing every year. 25% movies released in total 72 years from 1921 to 1993, but the next 25% movies released in only 8 years from 1993 to 2001.
- 50% remain movies released in 17 after 2001.

```r
grid.arrange(
    df %>% group_by(weekday) %>% summarize(count = n()) %>%
    mutate(weekday = reorder(weekday, count)) %>%
    ggplot(aes(weekday, count)) +
    geom_col(fill = "steel blue", color = "white") +
    labs(title = "Number of movies by weekday") + coord_flip(),

  df %>% group_by(weekday) %>%
    summarize(count = n(),
              avg_revenue = mean(revenue, na.rm = T)) %>%
    mutate(weekday = reorder(weekday, count)) %>%
    ggplot(aes(weekday, avg_revenue)) +
    geom_col(fill = "steel blue", color = "white") +
    labs(title = "Average revenue of movies by weekday") + coord_flip(),

  ncol = 2
)
```



We could see, in average:

- 44.2 movies released on Friday, 18.9 movies released on Thursday, 17.1 movies released on Wednesday, remain movies released on other weekdays.
- Movies released on Wednesday have highest revenue compared to other weekdays.

2.1.8 belongs_to_collection

This variable have greatest number of NA values. There are 79.98 % NA values and 20.02 % not-NA values. The NA values represented for a movie not belongs to any collection, otherwise a movie collection represent by following text:

[{'id': 313576, 'name': 'Hot Tub Time Machine Collection', 'poster_path': '/iEhb00TGPucF0b4joM1ieyY026U.jpg', 'backdrop_path': '/noeTVcgpBiD48fDjFVic1Vz7ope.jpg'}]

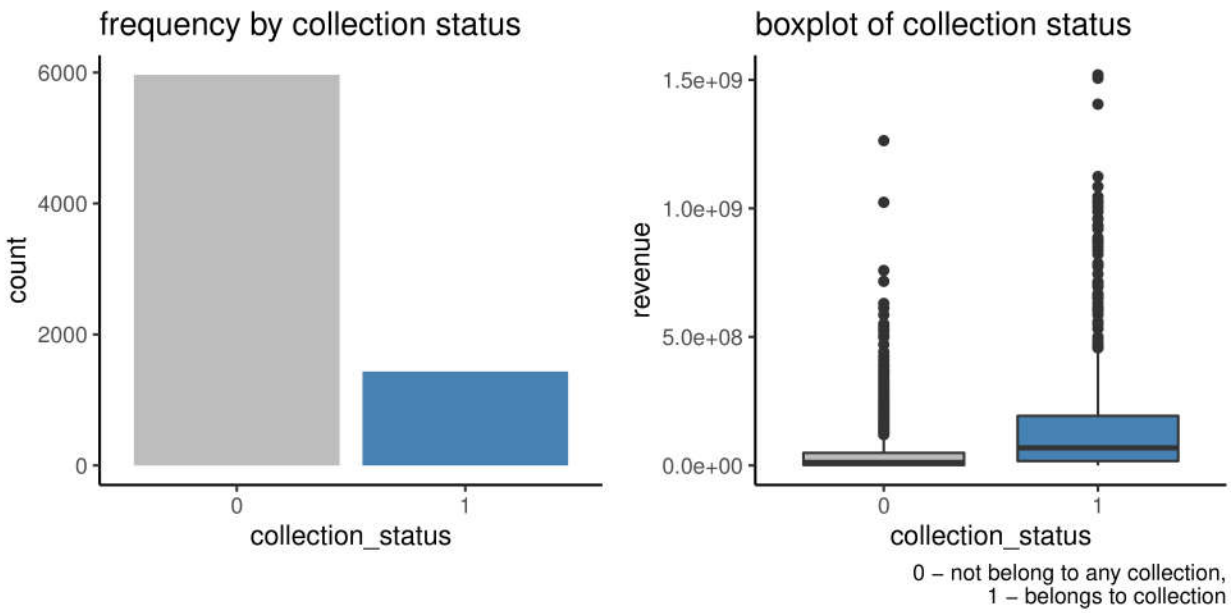
For each collection, I remove all unnecessary characters and kept only the collection's name (red text). The process performed by following code:

```
# extract collection
df$collection <- str_extract(df$belongs_to_collection,
            pattern = "(?<=name\\'\\:\\s{1}\\').+(?=\\'\\,\\s{1}\\'poster)")
df$collection[is.na(df$collection)] <- "no collection"
```

There were total 725 collection from both train_set and test_set. To visualize the difference between movies belong to a collection and movies not belong to any collection, I added one new feature `collection_status` to classify a movie belong to a collection or not. The following code and figure represented for this difference.

```
df <- df %>% mutate(collection_status = ifelse(collection == "no collection", 0, 1))
grid.arrange(
df %>% mutate(collection_status = factor(collection_status)) %>%
  ggplot(aes(x = collection_status, fill = collection_status)) +
  geom_bar() +
  theme_classic() +
  theme(legend.position = "none")+
  scale_fill_manual(values = c("grey","steel blue")) +
  labs(title = "frequency by collection status",
       caption = "
       "),

df %>% mutate(collection_status = factor(collection_status)) %>%
  ggplot(aes(x = collection_status, revenue, fill = collection_status)) +
  geom_boxplot() +
  theme_classic() +
  theme(legend.position = "none") +
  scale_fill_manual(values = c("grey","steel blue")) +
  labs(title = "boxplot of collection status",
       caption = "0 - not belong to any collection,
        1 - belongs to collection"),
ncol = 2)
```

```
## Warning: Removed 4398 rows containing non-finite values (stat_boxplot).
```



We could see:

In average, a movie belongs to a collection has higher median revenue than other movies not belong to any collection.

2.1.9 genres

Let's take a look on the genres.

```
head(df$genres,2)
```

```
## [1] "[{'id': 35, 'name': 'Comedy'}]"
## [2] "[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}, {'id': 10751, 'name': 'Family'}, {'
```

Some movies have one genre, other movies could had more than one genre. Each genre represent by. Each genre represents by following text "{'id': 35, 'name': 'Comedy'}" and separate by comma. For each genres value, I remove all unnecessary character and kept only the genre's name (red text).

Number of genres

To explore the correlation between number of genres vs movie revenue, I created the new variable to visualize the total genres of each movie.

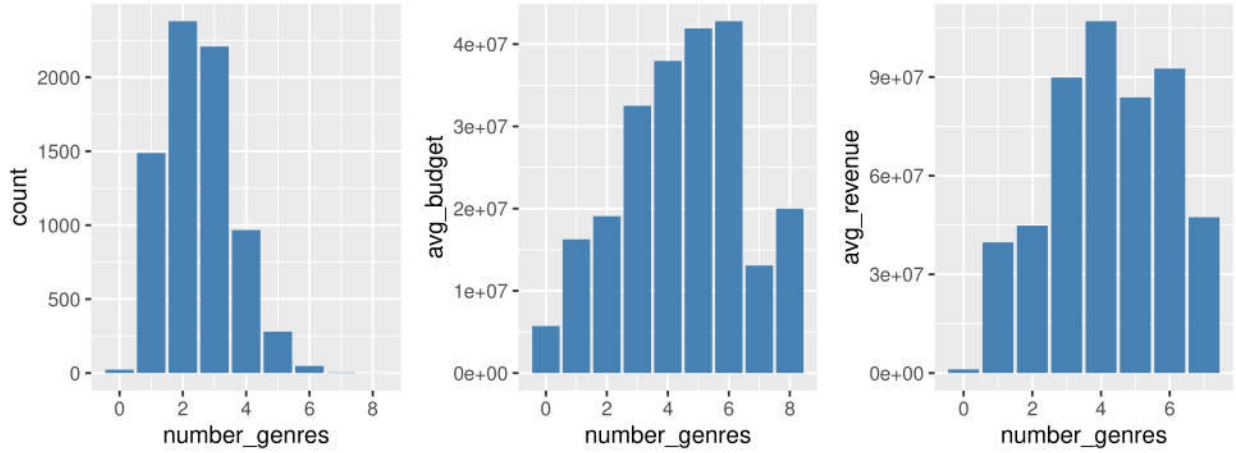
```
df$number_genres <- str_count(df$genres, pattern = "id")
df$number_genres[is.na(df$number_genres)] <- 0
```

Now we evaluate the correlation between budget, revenue on the number of genres.

```
grid.arrange(
  df %>%
    ggplot(aes(number_genres)) +
    geom_bar(fill = "steel blue"),
```

```
  df %>% group_by(number_genres) %>%
    summarize(avg_budget = mean(budget, na.rm = TRUE)) %>%
    ggplot(aes(number_genres, avg_budget)) +
    geom_col(fill = "steel blue"),

    df %>% group_by(number_genres) %>%
    summarize(avg_revenue = mean(revenue, na.rm = TRUE)) %>%
    ggplot(aes(number_genres, avg_revenue)) +
    geom_col(fill = "steel blue"),

  ncol = 3
)
```

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



We could see:

- Almost movies have two or three genres.
- In average, movies which number of genres were from 3 to 6 seem to have higher average revenue compared to other movies which less than 3 genres or greater than 6 genres.

NA value

There was also 23 NA value. I replaced those NA values by "no genre".

```
# replace NA value by "no genre"
df$genres[is.na(df$genres)] <- "no genre"
```

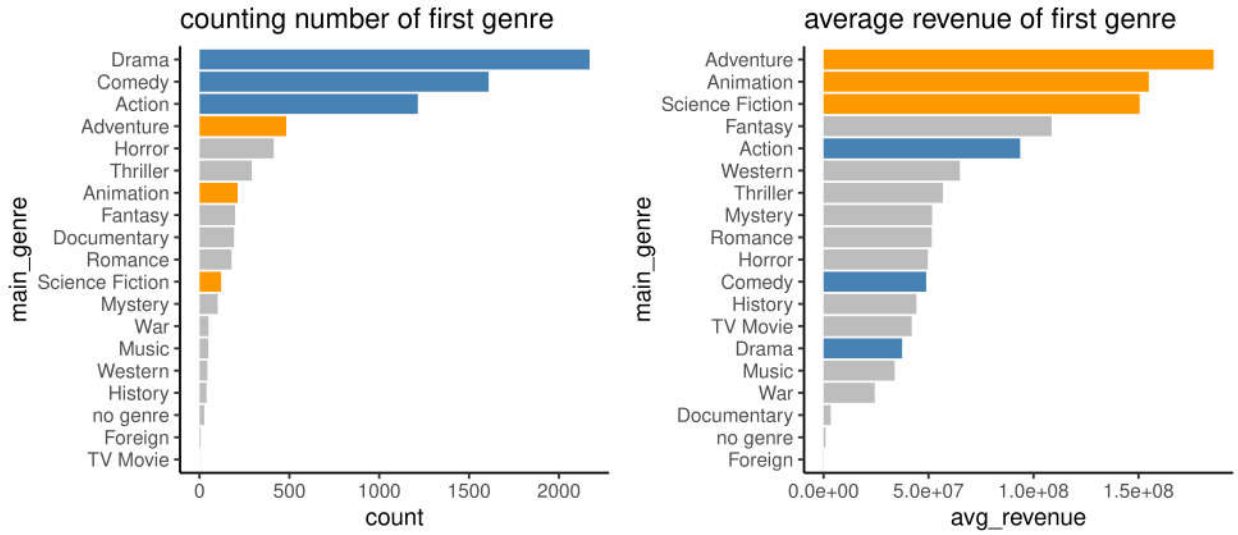
First genre

It is my impression that the first genre would be the main genre of each movie. To evaluate the difference between each genre, I extracted the first genre of each movie and add them to a new variable named `main_genre`.

```r
# create a vector with all genre levels
genres_matching_point <- "Comedy|Horror|Action|Drama|Documentary|Science Fiction|
              Crime|Fantasy|Thriller|Animation|Adventure|Mystery|War|Romance|Music|
              Family|Western|History|TV Movie|Foreign"

# extract the main genre from genres

df$main_genre <- str_extract(df$genres, genres_matching_point)
df$main_genre[is.na(df$main_genre)] <- "no genre"
```

Now figure out the difference of each main genre, by the number of movies and revenue.

```r
grid.arrange(

# barplot number of movies per genre
df %>%
  group_by(main_genre) %>%
  summarize(count = n()) %>%
  mutate(main_genre = reorder(main_genre, count),
         top_genre = ifelse(main_genre %in% c("Drama","Comedy","Action"),
                            "top count",
                            ifelse(main_genre %in% c("Adventure","Animation",
                            "Science Fiction"),"top revenue","common"))) %>%
  ggplot(aes(main_genre, count, fill = top_genre)) +
  geom_col() +
  scale_fill_manual(values = c("grey","steel blue","#FF9900"))+
  theme_classic() +
  coord_flip()+
  labs(title = "counting number of first genre") +
  theme(legend.position = "none"),

# barplot average revenue per genre
df %>%
  group_by(main_genre) %>%
  summarize(avg_revenue = mean(revenue, na.rm = TRUE)) %>%
  mutate(main_genre = reorder(main_genre, avg_revenue),
         top_genre = ifelse(main_genre %in% c("Drama","Comedy","Action"),
                            "top count",
                            ifelse(main_genre %in% c("Adventure","Animation",
                            "Science Fiction"),"top revenue","common"))) %>%
  ggplot(aes(main_genre, avg_revenue, fill = top_genre)) +
  geom_col() +
  scale_fill_manual(values = c("grey","steel blue","#FF9900"))+
  theme_classic() +
  coord_flip()+
  labs(title = "average revenue of first genre") +
  theme(legend.position = "none"),
ncol = 2)
```



We could see:

- Top three common genres with highest number of movies were *Drama, Comedy, Action.*
- Top three genres with greatest average revenue were *Adventure, Animation, Science Fiction*, and the average revenue of each genre are quite different compared to others.

2.1.10 production_companies

From previous section, we known there were total
414 NA values, approximate 5.6 % of total observation. Let's take a quick look on the top rows of this variable.

```
head(df$production_companies, 3)
```

```
## [1] "[{'name': 'Paramount Pictures', 'id': 4}, {'name': 'United Artists', 'id': 60}, {'name': 'Metro
## [2] "[{'name': 'Walt Disney Pictures', 'id': 2}]"
## [3] "[{'name': 'Bold Films', 'id': 2266}, {'name': 'Blumhouse Productions', 'id': 3172}, {'name': 'R
```

Some movies have one production company, other movies could had more than one production company. Each company represent by a name (string), and its id (number), for example {'name': 'Paramount Pictures', 'id': 4}. For each production company, I remove all unnecessary character and kept only the production company's name (red text). For 414 NA values, I replaced by "no production companies info".

Number of production companies

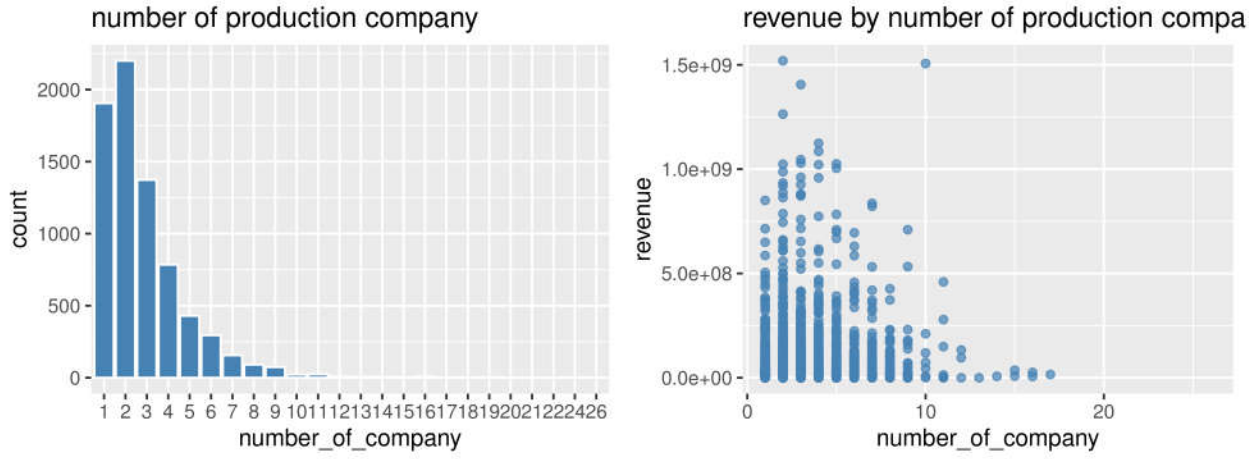
To explore the effect of number of production companies on the revenue, I created a new feature named number_of_company represent for the number of production companies of each movie. For NA value, I replaced by the median value. The process performed by following code:

```
# calculate number of company of a movie
df$number_of_company <- str_count(df$production_companies, pattern = "\\'name\\'")

# replace NA number by median
df$number_of_company[is.na(df$number_of_company)] <- median(df$number_of_company, na.rm = TRUE)
```

Figure out the number of production company per movie and how it correlated to the revenue.

```
# figure out the number of company per movies and revenue change
grid.arrange(
  df %>% mutate(number_of_company = factor(number_of_company)) %>%
    ggplot(aes(number_of_company)) +
    geom_bar(fill = "steel blue", color = "white") +
    labs(title = "number of production company"),

df %>%
  ggplot(aes(number_of_company, revenue)) +
  geom_point(color = "steel blue", alpha = 0.7) +
  labs(title = "revenue by number of production company"),

ncol = 2)
```

```
## Warning: Removed 4398 rows containing missing values (geom_point).
```



We could see:

90.5 % movies have less than or equal 5 production companies. The revenue seem to decrease when the number of production company greater than 5.

Top production companies effect

In the same manner, to evaluate the effect of top production companies on the revenue, I suppose that the first company would be the main company, who might have biggest impact on the production as well as quality of a movie. Then I created a summary data group by each production company to rank and evaluate their performance. The metric to evaluate performance of a production company were number of released movies, average budget, average revenue, ROI per company...

This process performed as following. The first step was to replaced all unnecessary characters and kept only the production company's name in a new variables named `companies`.

```
# remove all unnecessary character and keep only production company's name
df$companies <- gsub("(\\[?\\{\\'name\\'\\:\\s\\')|(\\'\\,\\s{1}\\'id\\'\\:\\s{1}\\d+\\}\\]?)",
                     "",df$production_companies)

# replace NA value in feature companies by "no production companies info"
```

```
df$companies[is.na(df$companies)] <- "no production companies info"

# create a list of all production companies
production_companies <- strsplit(df$companies, ", ")
production_companies <- unlist(production_companies, use.names=FALSE)
```

Extract the first company and store into variables named `first_company`.

```
# extract first company
df$first_company <- gsub("\\,\\s{1}.*","",df$companies)
```

The summary data of production companies was created by following code:

```
# calculate the total revenue from train data
total_revenue <- sum(train_set$revenue)

# create a summary table by first company
first_company_summary <- df[] %>% group_by(first_company) %>%
  summarize(movies_per_company = n(),
            avg_budget_per_company = mean(budget, na.rm = TRUE),
            avg_revenue_per_company = mean(revenue, na.rm = TRUE),
            ROI_per_company = round((mean(revenue, na.rm = TRUE) -
                          mean(budget, na.rm = TRUE))/
                    mean(budget, na.rm = TRUE),3))
```

Following was the summary table of top 10 production companies and their performance metrics.

```
first_company_summary %>% arrange(desc(movies_per_company)) %>%
  head(10) %>%
  kable(caption = "Top 10 production companies") %>%
  kable_styling(latex_options = c("HOLD_position","scale_down"),
                position = "center",
                bootstrap_options = c("striped", "hover"))
```

Table 7: Top 10 production companies

first_company	movies_per_company	avg_budget_per_company	avg_revenue_per_company	ROI_per_company
no production companies info	414	8897759	3715800	-0.582
Universal Pictures	401	34826952	106734116	2.065
Paramount Pictures	389	40354231	117855829	1.921
Twentieth Century Fox Film Corporation	291	33140639	99184949	1.993
Columbia Pictures	235	51558467	117958388	1.288
New Line Cinema	187	30698316	102169356	2.328
Warner Bros.	162	24934157	74005740	1.968
Walt Disney Pictures	146	83791171	308776496	2.685
Metro-Goldwyn-Mayer (MGM)	109	12042941	17718473	0.471
Columbia Pictures Corporation	105	27680048	66261848	1.394

We could see:

Although there were total 3000 movies in train data set and 1064 production companies in variables first company, but:

- Top 10 companies appeared as the first company in 826 movies, approximate 27.5 %.

- only “Universal Pictures”, “Paramount Pictures” and “Walt Disney Pictures” contribute approximate 27% of total revenue.
- 2.1% movies which were missing Production company information, had negative ROI (-0.547).

2.1.11 production_countries

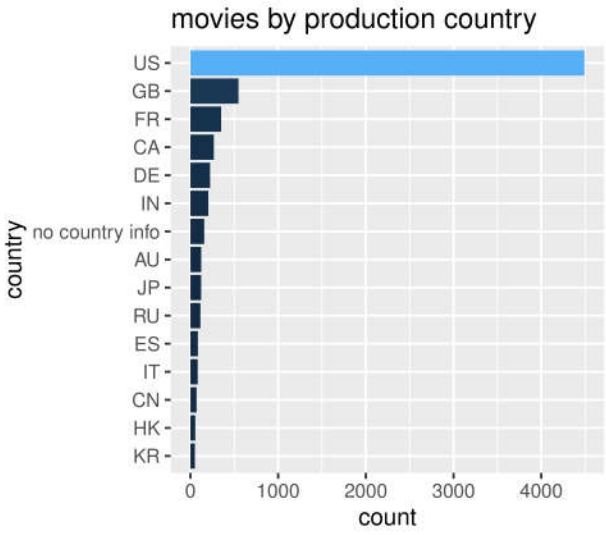
```
head(df$production_countries, 1)
```

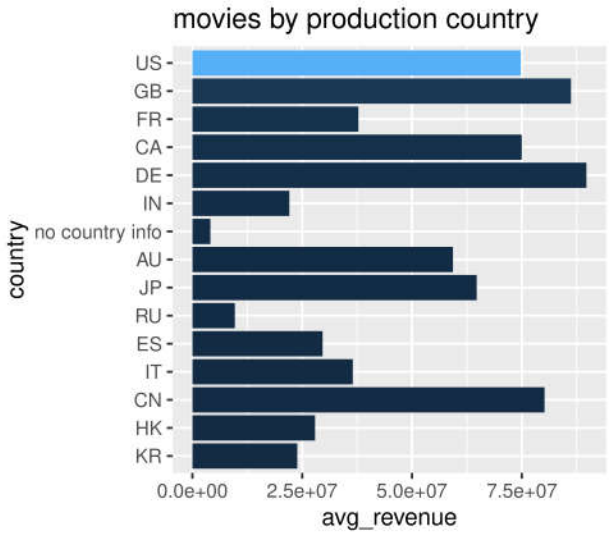
```
## [1] "[{'iso_3166_1': 'US', 'name': 'United States of America'}]"
```

Similarly, some movies had one production country, while other movies had more than one production country. Each company represent by following text “{‘iso_3166_1’: ‘US’, ‘name’: ‘United States of America’}” and separate by comma. For each production company, I remove all unnecessary character and kept only the production company’s name (red text). For NA values, I replaced by “no country info”.

The process performed by following code:

```
# extract the first production country
df$country <- gsub("(\\[?\\{\\'iso\\_3166\\_1\\'\\:\\s{1}\\')|(\\'\\,\\s{1}\\'name.*\\}\\]?)",
                   "",df$production_countries)
df$country[is.na(df$country)] <- "no country info"

# create summary table by first production country
country_summary <- df %>% mutate(country = factor(country)) %>%
  group_by(country) %>%
  summarize(count = n(),avg_revenue = mean(revenue, na.rm = T)) %>%
  arrange(desc(count))

# visualize country performance
grid.arrange(
  country_summary %>% head(15) %>%
  ggplot(aes(x = reorder(country, count), count, fill = count)) +
  labs(x = "country", title = "movies by production country") +
  geom_col() + coord_flip() + theme(legend.position = "none"),

  country_summary %>% head(15) %>%
  ggplot(aes(x = reorder(country, count), avg_revenue, fill = count)) +
  labs(x = "country", title = "movies by production country") +
  geom_col() + coord_flip() + theme(legend.position = "none"),

  ncol = 2)
```





We could see, in average:

- 60.7% movies produced in United State of America.
- Movies from countries US, GB, CA, DE, AU, JP, CN have better revenue than movies from other countries.
- Movies with no country information have lower revenue than other movies.

To add and evaluate effect of production country later on predicting model, a vector named `top_countries` to be created as following:

```
top_countries <- c("US","GB","CA","DE","AU","JP","CN","no country info")
```

2.2 Features engineering

Following is the summary table of what insighs gained from previous section and its respective possible engineering process.

No	Factors	Insights	Possible engineering process
1	Time-effect	Movies box office industry growth after that. Almost movies released on Friday, movies released on Wednesday had higher average revenue.	Add new feature to minimize time effect. Add time features to predicting model.
2	Budget, revenue, popularity	Budget, revenue, popularity have positive skew distribution. Revenue is highest correlated to budget x popularity & budget	Logarithm transformation Add new feature represent for for interaction between budget and popularity
3	Average profit ratio	In average, a movie have profit ratio 2.5997906 to calculat	Multiply budget by profit ratio e predicted revenue.
4	Belong_to_collection	Movies belongs to a collection seem to have higher revenue.	Add collection status to predicting model.

No	Factors	Insights	Possible engineering process
5	Genres	Different genres have significant different revenue. A movie could have more than one genre.	Add new feature to represent for all genres of a movie.
		Movies which number of genres were from 3 to 6 seem to have higher average revenue.	Add features number of genres to predicting model.
6	Production companies	Top 10 production companies contribute 27% total release movies Top production companies dominate company, the revenue percentage. A movie missing production company info have negative ROI.	Add features to recognize a movie belong to a top production or missing production company info.
7	Production countries	Almost movie produced from US. Movies from top countries have better revenue than others. Movies missing production countries info have lower revenue	Add features to recognize a movie belong to a top production country, or missing production country info.

2.2.1 Time-effect: Normallized popularity

From previous section, we known the period of time from first movie released date to the latest movie released date was 97 years. Furthermore, the popularity was calculated based on user interactions on the TMDb website, while the internet users and their time in internet increased by the time. To reduce the change across time, I added one more feature `normalized_popularity` by dividing `popularity` by the average popularity per release year. The calculation was as following:

```
# create the summarized table for popularity
df$popularity <- df$new_popularity

popularity_sum <- df %>%
  group_by(release_year) %>%
  summarize(avg_popularity = mean(new_popularity, na.rm = T))
# create new normallized popularity
df <- df %>%
  left_join(popularity_sum, by = "release_year") %>%
  mutate(normallized_popularity = new_popularity/avg_popularity)
```

Adding second time normallized popularity feature.

```
# create 2nd summarize table for normallized_popularity
norm_popularity_sum <- df %>%
  group_by(release_year) %>%
  summarize(max_norm_pop = max(normallized_popularity, na.rm = T))
# add 2nd normallized popularity
df <- df %>%
  left_join(norm_popularity_sum, by = "release_year") %>%
  mutate(second_norm_popularity = normallized_popularity/max_norm_pop)
```

2.2.2 Budget and popularity interaction

New features named `budget_norm_pop` represent for the interaction between budget and normalized popularity to be added by following code:

```
df <- df %>%
   # add interaction features between variables
  mutate(budget_pop = budget*new_popularity,
         budget_norm_pop = budget*normallized_popularity)
```

2.2.3 Expected revenue based on average profit ratio

The expected revenue based on average profit ratio was calculated by following code:

```
# calculate expected revenue based on average profit
df <- df %>%
  mutate(expected_revenue = new_budget*avg_profit_ratio)
```

2.2.4 Genres

To add the effect by genres, I created 19 new features named by each genre. Each feature had two levels 0 and 1, which:

- 0: if this genre didn't present in the genres list of movie
- 1: if this genre present in the genres list of movie

```
# create a vector contain all genre name
genres <- levels(factor(df$main_genre))
```

```
# create features for each genre
for (i in 1:19){
  df[,genres[i]] <- ifelse(str_detect(df$genres,genres[i]),
                           1,0)
}

# calculate the column index for genres
from_genre <- grep("Action", colnames(df))
to_genre <- grep("Mystery", colnames(df))
```

2.2.5 Top production companies

To add the effect by top 10 production companies, I created 10 new features named by each production company. Each feature had two levels 0 and 1, which:

- 0: if the production company didn't present in the production companies list of movie
- 1: if the production company present in the production companies list of movie

For other production companies did not belong to top 10 production companies, I assigned to a variables named "other_production_company", with similarly two levels 0 and 1.

```
# create top production companies vector
top_production_companies <- first_company_summary %>%
  arrange(desc(movies_per_company)) %>% head(10) %>%
  pull(first_company)

# create Dummy features for each production company
for (i in 1:length(top_production_companies)){
  df[,top_production_companies[i]] <-
    ifelse(str_detect(df$production_companies,top_production_companies[i]),
                          1,0)
}

df$`no production companies info` <-
  ifelse(str_detect(df$companies, "no production companies info"),1,0)

from_company <- grep(head(top_production_companies,1), colnames(df))

to_company <- grep(tail(top_production_companies,1), colnames(df))

df$other_production_company <-
  ifelse(rowMeans((df[,from_company:to_company])) >0,0,1)

for (i in from_company:to_company){
  df[is.na(df[,i]),i] <- 0
}
```

2.2.6 Top production countries

To add the effect by top production country, I created new features named by each production country. Each feature had two levels 0 and 1, which:

- 0: if the production country didn't present in the production countries list of movie
- 1: if the production country present in the production countries list of movie

For other production countries did not belong to top production countries, I assigned to a variables named "other_production_countries", with similarly two levels 0 and 1.

```
# create Dummy features for each production country
for (i in 1:length(top_countries)){
  df[,top_countries[i]] <-
    ifelse(str_detect(df$production_countries,top_countries[i]),
                          1,0)
}

df$`no country info` <-
  ifelse(str_detect(df$first_company, "no country info"),1,0)

# determine the column index of production country features
from_country <- grep(head(top_countries,1), colnames(df))

to_country <- grep(tail(top_countries,1), colnames(df))
```

```
# create feature other_production_countries
df$other_production_countries <-
  ifelse(rowMeans((df[,from_country:to_country])) >0,0,1)

# replace any NA value by zero
for (i in from_country:to_country){
  df[is.na(df[,i]),i] <- 0
}
```

2.2.7 Features selection

The features for modeling were selected as following:

```
# selecting features
dat <- df %>%
  mutate(revenue_pop = expected_revenue*new_popularity,
         revenue_norm_pop = expected_revenue*normallized_popularity) %>%
  dplyr::select(revenue,
         expected_revenue, revenue_pop, revenue_norm_pop,
         new_budget,
         budget_pop,
         budget_norm_pop,
         new_popularity, normallized_popularity,
         collection_status,
         number_of_company,
         number_genres,
         from_genre:to_genre,(to_genre + 2):(to_genre + 7),
        from_company:to_company, to_company + 1,
        from_country:to_country, to_country + 1,
        release_year, release_month, weekday)

# replace NA
for (i in 2:(ncol(dat)-1)){
  dat[is.na(dat[,i]),i] <- median(dat[,i], na.rm = T)
}
```

Separate the full data df to train and test set and also replace the outliers.

```
# create train data
dat_train <- dat[1:3000,]

# replace outlier in train data
dat_train <- dat_train[-outlier_rows,]
dat_train <- filter(dat_train, new_budget > 1000 & revenue > 1000)

# create test data
dat_test <- dat[3001:7398,]
```

Create a small data partition from `dat_train` for cross validation in training modeling.

```
# create data for cross validation in modeling training
index <- createDataPartition(dat_train$revenue, times = 1, p = 0.8, list = FALSE)
```

```
train_set <- dat_train[index,]
test_set <- dat_train[-index,]
```

2.3 Modeling

2.3.1 Design of experiment

The experiments were designed to evaluate (i) the effect of different variables, (ii) different machine learning methods and the best performance model, (iii) effect of the logarit transformation on modeling performance. The RMSLE was used to evaluate modeling performance. Following was summarized table of experimented models in this section.

Exp#	Model name	Description	Objective
1	Naive model	use average revenue as predicted revenue.	the RMSLE of this model to be used as the baseline to evaluate performance of other experiments.
2	Best ML methods	compare different machine learning method performance. The methods to be experimented were random forest (rf), bayesian generalized linear model (bayesglm), generalized linear model boosting (glmboost), linear model (lm).	to select the machine learning method with best performance.
3	Features selection	use multiple features to predict revenue.	to select features for final predicting model.
4	Logarithm transform	use logarit to transform the data.	to evaluate the effect of logarithm on machine learning performance.

2.3.2 Experiment 1: Naive model

The first experiment using average revenue as predicted revenue. The RMSLE are as following:

```
predicted_revenue <- rep(mean(train_set$revenue, na.rm = T), times = nrow(test_set))

results_exp1 <- data.frame(Exp_No = 1,
                      Experiment = "Naive model",
                      normal_calculation = RMSLE(predicted_revenue, test_set$revenue))

results_exp1 %>%  kable(digits = 3) %>%
  kable_styling(latex_options = c("HOLD_position"),
                bootstrap_options = c("striped", "hover"),
                full_width = F, position = "center")
```

Exp_No	Experiment	normal_calculation
1	Naive model	3.212

2.3.3 Experiment 2: Best ML methods

In this experiment, I compared performance of random forest (rf), bayesian generalized linear model (bayesglm), generalized linear model boosting (glmboosting), linear model (lm). At the end of this

experiment, we would determine what method have best performance.

```
# modeling with different methods
# NOTED: this process will take some minute
fit_rf_exp2 <- train(revenue ~ .,
             data = train_set,
             method = "rf",
             importance = TRUE,
             verbose = TRUE,
             trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))

fit_bayesglm_exp2 <- train(revenue~.,
                 data = train_set,
                 method = "bayesglm",
                 trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))
fit_glmboost_exp2 <- train(revenue~.,
                 data = train_set,
                 method = "glmboost",
                 trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))
fit_lm_exp2 <- train(revenue~.,
                 data = train_set,
                 method = "lm",
                 trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))
```

The predicted revenues were calculated as following:

```
# calculate predicted revenue
yhat_rf <- predict(fit_rf_exp2, newdata = test_set)
yhat_bayesglm <- predict(fit_bayesglm_exp2, newdata = test_set)
yhat_glmboost <- predict(fit_glmboost_exp2, newdata = test_set)
yhat_lm <- predict(fit_lm_exp2, newdata = test_set)
```

Create a data frame to contain all predicted revenue from different machine learning method.

```
# ensembles
ensembles <- data.frame(rf = yhat_rf,
                        bayesglm = yhat_bayesglm,
                        glmboost = yhat_glmboost,
                        lm = yhat_lm)
```

```
# to replace negative predicted revenue by median
ensembles$bayesglm[ensembles$bayesglm < 0] <- median(ensembles$bayesglm)
ensembles$glmboost[ensembles$glmboost < 0] <- median(ensembles$glmboost)
ensembles$lm[ensembles$lm < 0] <- median(ensembles$lm)
```

Random forest performed better results than other methods and previous models. The RMSLE of random forest was 3.5203488. Therefore I used random forest for remain experiments.

Following was the RMSLE table of experimented methods:

```
results_exp2<- data.frame(Exp_No = 2,
             Experiment = "Evaluate machine learning methods",
             rf = RMSLE(ensembles$rf, test_set$revenue),
           bayesglm = RMSLE(ensembles$bayesglm, test_set$revenue),
           glmboost = RMSLE(ensembles$glmboost, test_set$revenue),
           lm = RMSLE(ensembles$lm, test_set$revenue))

results_exp2 %>%  kable(digits = 3) %>%
  kable_styling(latex_options = c("HOLD_position"),
                bootstrap_options = c("striped", "hover"),
                full_width = F, position = "center")
```

Exp_No	Experiment	rf	bayesglm	glmboost	lm
2	Evaluate machine learning methods	2.191	2.628	2.645	2.672

2.3.4 Experiment 3: Features selection

Objective of this experiment was to evaluate the importance and select the features for predicting model.

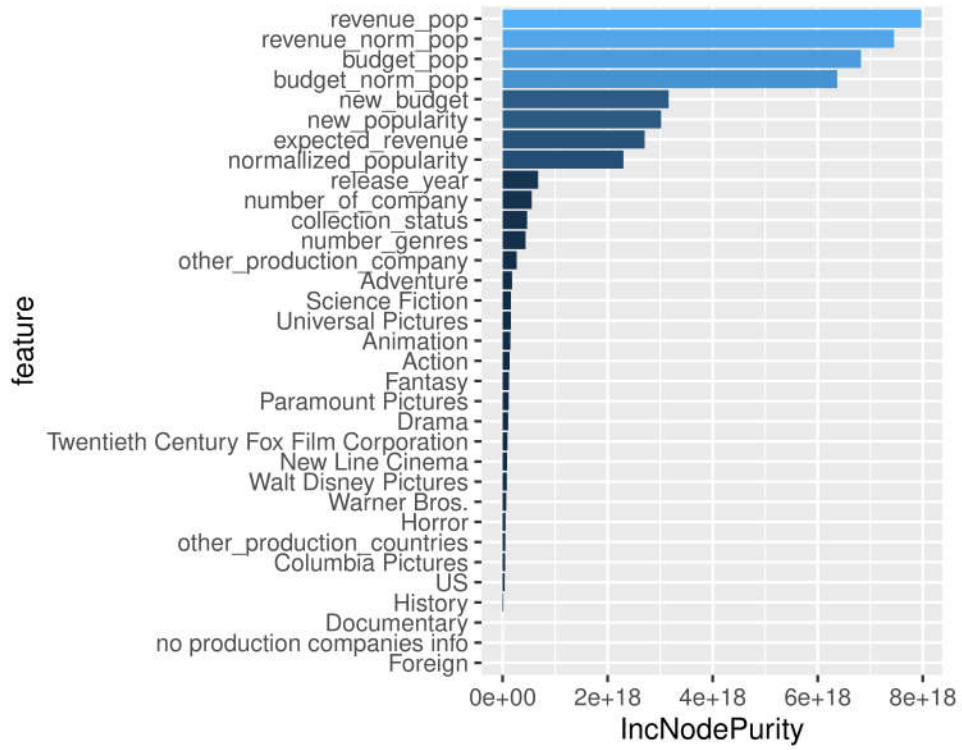
Many models that can be accessed using caret's train function produce prediction equations that do not necessarily use all the predictors. In many cases, using these models with built-in feature selection will be more efficient than algorithms where the search routine for the right predictors is external to the model. Built-in feature selection typically couples the predictor search algorithm with the parameter estimation and are usually optimized with a single objective function (e.g. error rates or likelihood). [3]

Feature Selection using Univariate Filters

An approach to feature selection is to pre-screen the predictors using simple univariate statistical methods then only use those that pass some criterion in the subsequent model steps. Similar to recursive selection, cross-validation of the subsequent models will be biased as the remaining predictors have already been evaluate on the data set. Proper performance estimates via resampling should include the feature selection step. [4]

```
# use sbf function from Caret to select features
# NOTED: this process will take few minutes
filterCtrl <- sbfControl(functions = rfSBF, method = "repeatedcv", repeats = 5)
set.seed(10)
rfWithFilter <- sbf(train_set[,-1], train_set$revenue, sbfControl = filterCtrl)
rfWithFilter
```

```
# visualize feature importance
data.frame(rfWithFilter$fit$importance, feature = row.names(rfWithFilter$fit$importance)) %>%
  mutate(feature = reorder(feature, IncNodePurity)) %>%
  ggplot(aes(feature,IncNodePurity, fill = IncNodePurity)) + geom_col() +
  coord_flip() + theme(legend.position = "none")
```



33 variables were selected includes:

```
rfWithFilter$optVariables
```

```
##  [1] "expected_revenue"
##  [2] "revenue_pop"
##  [3] "revenue_norm_pop"
##  [4] "new_budget"
##  [5] "budget_pop"
##  [6] "budget_norm_pop"
##  [7] "new_popularity"
##  [8] "normallized_popularity"
##  [9] "collection_status"
## [10] "number_of_company"
## [11] "number_genres"
## [12] "Action"
## [13] "Adventure"
## [14] "Animation"
## [15] "Documentary"
## [16] "Drama"
## [17] "Fantasy"
## [18] "Foreign"
## [19] "History"
## [20] "Horror"
## [21] "Science Fiction"
## [22] "no production companies info"
## [23] "Universal Pictures"
## [24] "Paramount Pictures"
```

```
## [25] "Twentieth Century Fox Film Corporation"
## [26] "Columbia Pictures"
## [27] "New Line Cinema"
## [28] "Warner Bros."
## [29] "Walt Disney Pictures"
## [30] "other_production_company"
## [31] "US"
## [32] "other_production_countries"
## [33] "release_year"
```

The predicted revenue was calculated as following.

```
yhat_sbf <- predict(rfWithFilter, test_set)
results_exp3 <-data.frame(Exp_No = 3,
                          Experiment = "Feature selection",
                          rf = RMSLE(yhat_sbf, test_set$revenue))
```

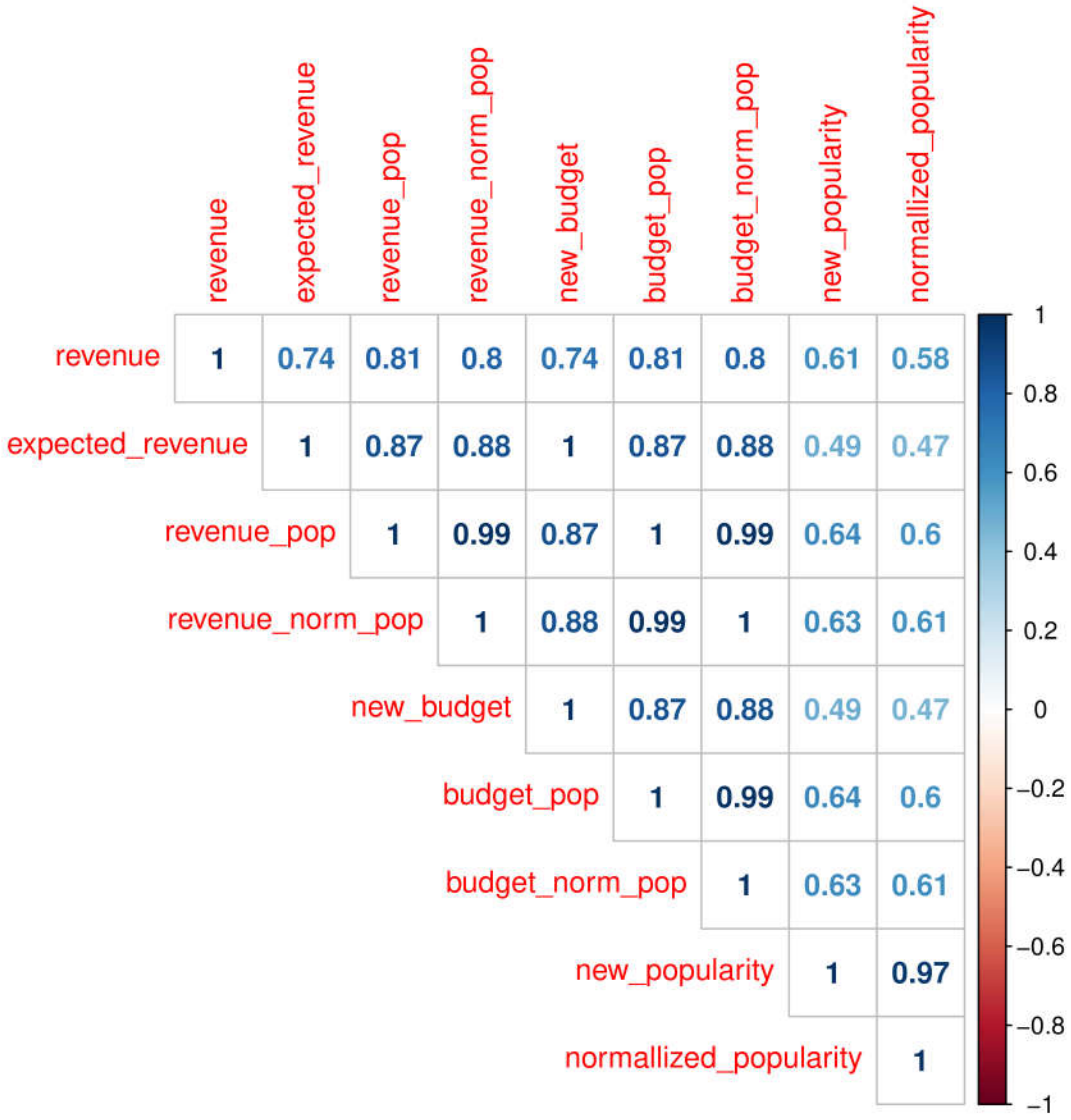
The RMSLE was 2.1652098 , which was significant improvement on the performance.

Confounding between variables

In statistics, a confounder (also confounding variable, confounding factor, or lurking variable) is a variable that influences both the dependent variable and independent variable, causing a spurious association. Confounding is a causal concept, and as such, cannot be described in terms of correlations or associations. [5]

In previous section, I added the interaction features between revenue, budget, popularity... The following correlation matrix showed the correlation between features related to budget, revenue, popularity...

```
corrplot(cor(dat_train[,1:9]), type = "upper", method = "number")
```



We could see high correlation number between features represented for interaction related to revenue (revenue-group features includes expected_revenue, revenue_pop, revenue_norm_pop) & features represented for interaction related to budget (budget-group features includes new_budget, budget_pop, budget_norm_pop).

To explore the effect of include/exclude above features (revenue-group features and budget-group features), I evaluated performance of three following models:

Model_No	Machine Learning method	Feature selection	Preprocessing
1	Random Forest (rf)	using Univariate Filters	
2	Random Forest (rf)	using revenue-group features	exclude new_budget, budget_pop, budget_norm_pop
3	Random Forest (rf)	using budget-group features	exclude expected_revenue, revenue_pop, revenue_norm_pop

The experiment process performed by following code:

```
# training 3 models: all features, budget_group features, revenue_group features
fit_all_features <- train(revenue ~ .,
             data = train_set,
             method = "rf",
             importance = TRUE,
             verbose = TRUE,
             trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))

fit_revenue_exp2 <- train(revenue ~ .,
             data = train_set[,-c(5,6,7)],
             method = "rf",
             importance = TRUE,
             verbose = TRUE,
             trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))

fit_budget_exp2 <- train(revenue ~ .,
             data = train_set[,-c(2,3,4)],
             method = "rf",
             importance = TRUE,
             verbose = TRUE,
             trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))

# calculate the yhat of 3 models
yhat_all_features <- predict(fit_all_features, test_set)
yhat_revenue <- predict(fit_revenue_exp2, test_set)
yhat_budget <- predict(fit_budget_exp2, test_set)
```

The results were summarized as following table:

```
confounding_features_results <-
  data.frame(Univariate_Filters = RMSLE(yhat_sbf, test_set$revenue),
            all_features = RMSLE(yhat_all_features, test_set$revenue),
            Revenue_group_Features = RMSLE(yhat_revenue, test_set$revenue),
            Budget_group_Features = RMSLE(yhat_budget, test_set$revenue))

confounding_features_results %>% kable(digits = 3) %>%
  kable_styling(latex_options = c("HOLD_position"),
                bootstrap_options = c("striped", "hover"),
                full_width = F, position = "center")
```

Univariate_Filters	all_features	Revenue_group_Features	Budget_group_Features
2.165	2.19	2.202	2.192

We could see the lowest RMSLE in model - Feature Selection using Univariate Filters, but the RMSLE gap between all models were not quite different.

2.3.5 Experiment 4: Logarithm transformation

One insight from previous section that the budget and revenue had a positive skewed distribution shape. However, many statistic methods have been developed to test the normality assumption of observed data. When the distribution of the continuous data is non-normal, transformations of data are applied to make the data as "normal" as possible and, thus, increase the validity of the associated statistical analyses. The log transformation is, arguably, the most popular among the different types of transformations used to transform skewed data to approximately conform to normality. [3]

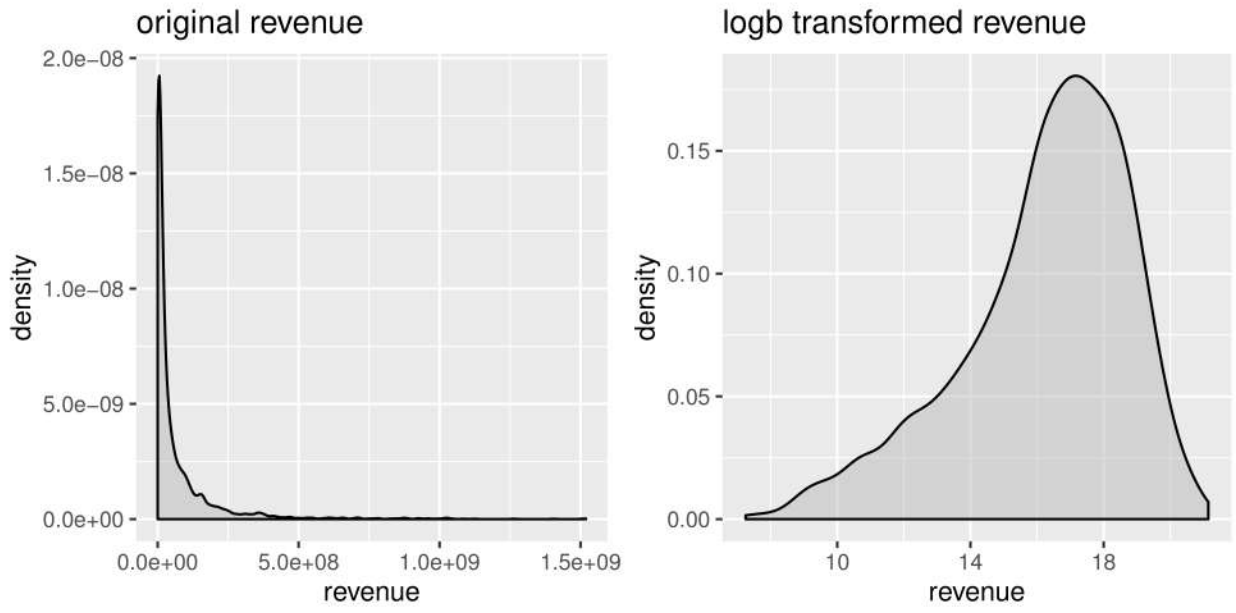
Therefore, the objective of this experiment is to evaluate the effect of logarit transformation on the results.

Before process modeling, I transformed the revenue, budget and its respective features using logarit function `logb`.

```
# logarit transforming
for (i in 1:8){
  train_set[,i] <- logb(train_set[,i]+1)
  test_set[,i] <- logb(test_set[,i]+1)
}
```

We could see the distribution shape changed before and after log transformation.

```
grid.arrange(
  dat_train %>% ggplot(aes(revenue)) + geom_density(fill = "grey", alpha = 0.5) +
    labs(title = "original revenue"),

  train_set %>% ggplot(aes(revenue)) + geom_density(fill = "grey", alpha = 0.5) +
    labs(title = "logb transformed revenue"),

  ncol = 2)
```



The machine learning method using in this experiment was Random Forest, using Univariate Filter.

```
# experiment 4: training model
filterCtrl <- sbfControl(functions = rfSBF, method = "repeatedcv", repeats = 5)
set.seed(10)
rf_with_filter_logb <- sbf(train_set[,-1], train_set$revenue, sbfControl = filterCtrl)
```

The predicted revenue was calculated as following:

```
# experiment 4: predict revenue
yhat_exp4 <- predict(rf_with_filter_logb, test_set)
results_exp4 <- data.frame(Exp_No = 4,
                           Experiment = "Logarit transformation",
                           rf = RMSE(yhat_exp4, test_set$revenue))

results_exp4 %>%  kable(digits = 3) %>%
  kable_styling(latex_options = c("HOLD_position"),
                bootstrap_options = c("striped", "hover"),
                full_width = F, position = "center")
```

Exp_No	Experiment	rf
4	Logarit transformation	1.764

We could see the RMSLE was 1.7635992 , which was a significant improvement compare to other models in previous experiments.

2.3.6 Experiments summary

The experiment results was summarized as following:

- Random forest was performed better than bayesian generalized linear model (bayesglm), generalized linear model boosting (glmboost), linear model (lm). Therefore, I selected random forest as the machine learning method for final models.
- Model using Univariate Filters to select features performed better model using revenue-group features and budget-group features, but the gap was not significant different.
- Models using logarit transformation had better results (lower RMSLE) than other non-logarit transformation models in previous experiments.

```
final_results <- bind_rows(results_exp1, results_exp2, results_exp3, results_exp4)
final_results %>%
  kable(caption = "Experiments Summary", digits = 3) %>%
  kable_styling(latex_options = c("HOLD_position","scale_down"),
                position = "center", full_width = F,
                bootstrap_options = c("striped", "hover"))
```

Table 11: Experiments Summary

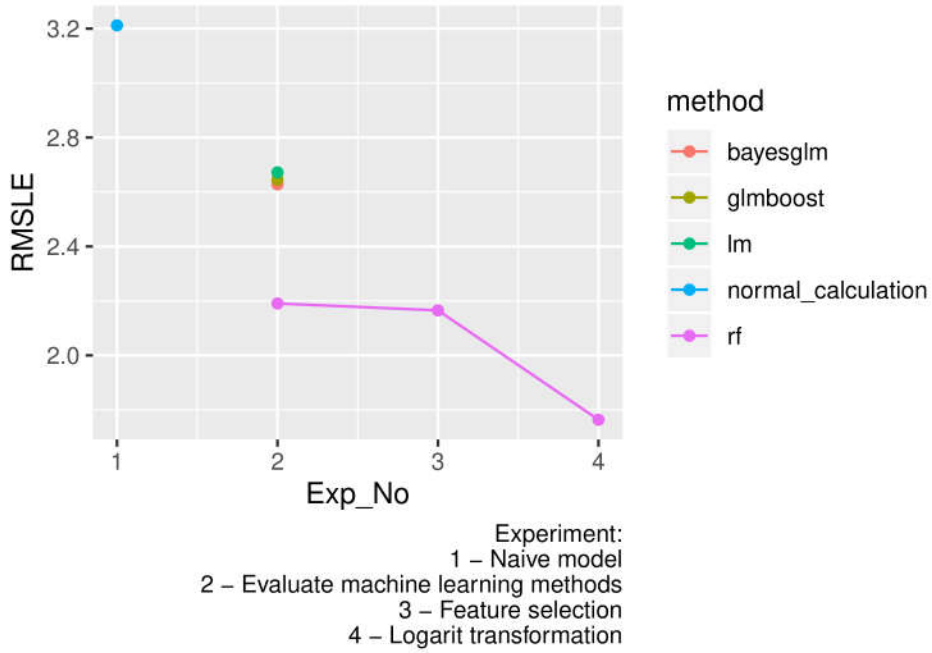
Exp_No	Experiment	normal_calculation	rf	bayesglm	glmboost	lm
1	Naive model	3.212	-	-	-	-
2	Evaluate machine learning methods	-	2.191	2.628	2.645	2.672
3	Feature selection	-	2.165	-	-	-
4	Logarit transformation	-	1.764	-	-	-

Visualize the final results

```
final_results %>% gather(3:7, key = "method", value = "RMSLE") %>%
  ggplot(aes(Exp_No, RMSLE, color = method)) +
  geom_line() +
  geom_point() + labs(caption = "Experiment:
                      1 - Naive model
                      2 - Evaluate machine learning methods
                      3 - Feature selection
                      4 - Logarit transformation")
```

```
## Warning: Removed 13 rows containing missing values (geom_path).
```

```
## Warning: Removed 13 rows containing missing values (geom_point).
```



3 Results & discussion

3.1 Final models for validation

As a consequence, I trained 3 models for final validation in test data. The machine learning method, feature selection, pre-processing were summarized as following:

Model_No	Machine Learning method	Feature selection	Preprocessing
1	Random Forest (rf)	using Univariate Filters	logb transforming
2	Random Forest (rf)	using revenue-group features	logb transforming
3	Random Forest (rf)	using budget-group features	logb transforming

Pre-processing data using logarith transformation method.

```
# logarit transforming
for (i in 1:8){
  dat_train[,i] <- logb(dat_train[,i]+1)
  dat_test[,i] <- logb(dat_test[,i]+1)
}
```

Model 1: Feature Selection using Univariate Filters

Train model:

```
# train model 1 with Univariate Filters to select features
filterCtrl <- sbfControl(functions = rfSBF, method = "repeatedcv", repeats = 5)
set.seed(10)
final_model_1 <- sbf(dat_train[,-c(1,3,4,5,6,7)], dat_train$revenue, sbfControl = filterCtrl)
```

Final predicted revenue:

```
yhat_final_1 <- predict(final_model_1, dat_test)
final_model1_RMSLE <- RMSE(yhat_final_1, logb(test_y+1))
```

Model 2: Revenue-group features

Train model:

```
# train model 2 with revenue-group features
final_model_2 <- train(revenue ~ .,
             data = train_set[,-c(5,6,7)],
             method = "rf",
             importance = TRUE,
             verbose = TRUE,
             trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))
```

Final predicted revenue:

```
yhat_final_2 <- predict(final_model_2, dat_test)
final_model2_RMSLE <- RMSE(yhat_final_2, logb(test_y+1))
```

Model 3: Budget-group features

Train model:

```
# train model 3 with budget-group features
final_model_3 <- train(revenue ~ .,
             data = train_set[,-c(2:4)],
             method = "rf",
             importance = TRUE,
             verbose = TRUE,
             trControl = trainControl(method = "cv",
                                      number = 5,
                                      p = 0.8))
```

Final predicted revenue:

```
yhat_final_3 <- predict(final_model_3, dat_test)
final_model3_RMSLE <- RMSE(yhat_final_3, logb(test_y+1))
```

3.2 Results

To conclude with, the RMSLE from 3 models were summarized as following:

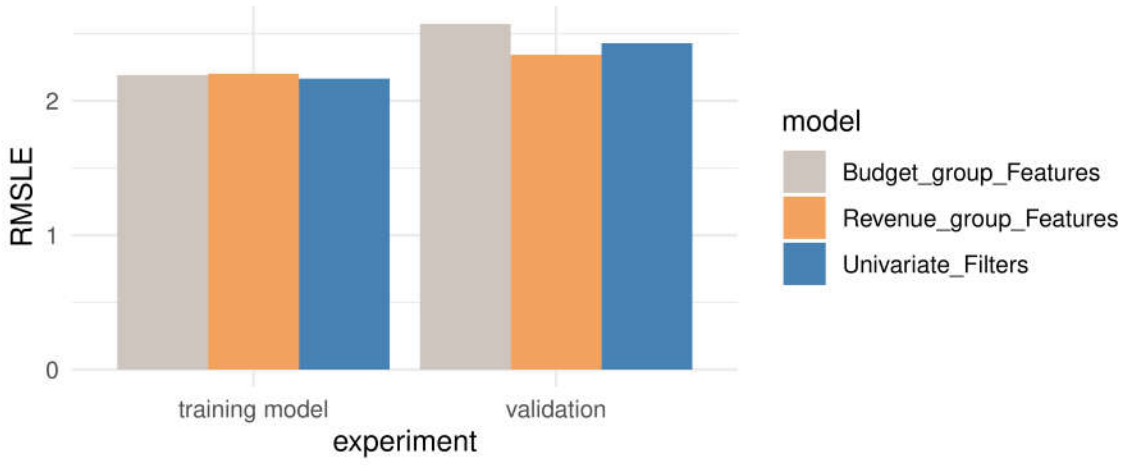
```
validation_results <- data.frame(Univariate_Filters = final_model1_RMSLE,
           Revenue_group_Features = final_model2_RMSLE,
           Budget_group_Features = final_model3_RMSLE)

validation_results %>%  kable(digits = 3) %>%
  kable_styling(latex_options = c("HOLD_position"),
                bootstrap_options = c("striped", "hover"),
                full_width = F, position = "center")
```

Univariate_Filters	Revenue_group_Features	Budget_group_Features
2.429	2.343	2.572

3.3 Discussion on modeling performance

In previous experiments, the RMSLE from three models were not quite different. Nevertheless, in the final validation, the best performance results was received from the model using revenue-group features, with RMSLE = 2.3426967 . Following was the model using Univariate Filters for feature selection, with RMSLE = 2.4292049. The worst performance was received by model using Budget-group features, with RMSLE = 2.5724968.

```
bind_rows(confounding_features_results[,-2], validation_results) %>%
  mutate(experiment = c("training model","validation")) %>%
  gather(1:3,key = "model", value = "RMSLE") %>%
  ggplot(aes(experiment, RMSLE, fill = model)) +
  geom_col(position = "dodge") +
  scale_fill_manual(values = c("seashell3","sandybrown","steel blue")) +
  theme_minimal()
```



On the other hand, the best RMSLE on validated on test data was 2.343 , which was higher than the RMSLE receive from experiment 3, RMSLE = 1.764. It might cause of the unbalancing data between train and test data. Certainly, the train data have 3000 rows (approx ~40% total observations) and test data have 4398 observations (approx ~60% total observations). To be more precise, approx 65% movie's collection appeared in both train and test data. This issue might lead to an inaccuracy predicted results for a movie with unseen predictors as well.

```
mean(na.omit(df$collection[1:3000]) == na.omit(df$collection[3001:7398]))
```

```
## Warning in na.omit(df$collection[1:3000]) ==
## na.omit(df$collection[3001:7398]): longer object length is not a multiple
## of shorter object length
```

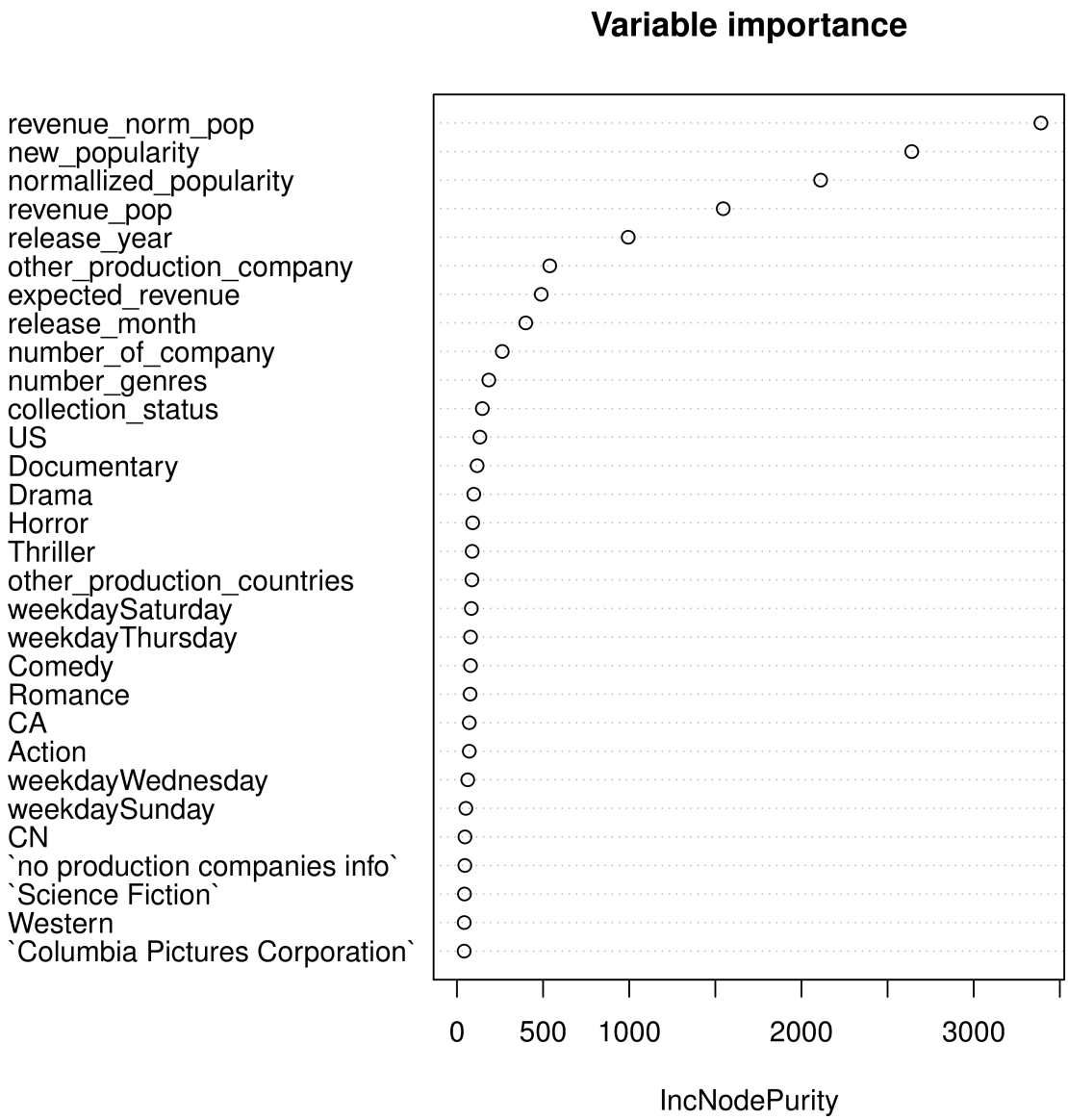
```
## [1] 0.6507503
```

To deal with imbalance data issue, one approach is to collect more data, which are available in The Movie Database. Furthermore, in this model I didn't analyze and include remain features such as cast, crew, Keywords... which might effect on the movie revenue.

3.4 What's most important features?

The important features plot are as following:

```
varImpPlot(final_model_2$finalModel,type = 2,main = "Variable importance")
```



The top 3 important features were:

- **revenue_norm_pop**, represents for the interaction between revenue and normalized popularity,
- **normallized_popularity**, represents for the normalized popularity by release year,
- **new_popularity**, represents for the popularity of a movies.

4 Conclusion

In this report I described a way to build up the machine learning model to predict a movie revenue before its release date, using data from Kaggle TMDb Box Office Competition and other publicly data from The Movie Database website and Wikipedia. The original features, which were analyzed and included in final predicting model, were budget, popularity, genres, belong to collection, production companies, production countries. Four machine learning methods were experimented in a cross-validation dataset, Random Forest (rf) give better results (RMSLE 2.191) than Generalized Linear Model Boosting (glmboost) (RMSLE 2.645), Bayesian Generalized Linear Model (bayesglm) (RMSLE 2.628), Linear Model (lm) (RMSLE 2.672). Three approach for feature selection were experimented with no significant difference on modeling performance (Univariate Filter RMSLE 2.165 , revenue-group features RMSLE 2.202 , budget-group features RMSLE 2.192). Since revenue and budget had a skewed-distribution, a data pre-processing approach using log transformation (base e) on the model using Random Forest and Univariate Filters were experimented and give best performance (RMSLE 1.764) on cross validation dataset.

Finally, three models were developed and validated on the test data, with best performance (RMSLE 2.343) belong to model using revenue-group features. The most importance features on best performance model are revenue_norm_pop represents for the interaction between revenue and normalized popularity, normallized_popularity represents for the normalized popularity by release year, new_popularity represents for the popularity of a movies.

Because those models were developed on a 3000 observation train dataset and validated on a 4398 observation test dataset, it's limited to provide an accuracy results for a movie with unseen predictors. However, this is also an opportunity to improve modeling performance in the future by adding more observation in the training set. Furthermore, other features were not analyzed and included in predicting model, which are also analyzed and added-in for further improvement.

Another point that although random forest give better performance on RMSLE than other machine learning methods, its processing time is quite longer and might be limited if the training dataset is more bigger. For this reason, alternative machine learning methods are able to experiment to improve modeling speed in the future.

5 Reference

[1] TMDB Box Office Prediction - https://www.kaggle.com/c/tmdb-box-office-prediction/overview

[2] TMDB Box Office Prediction - Discussion - https://www.kaggle.com/c/tmdb-box-office-prediction/discussion

[3] *The caret Package* - Max Kuhn - 2019-03-27 - Feature Selection Overview - https://topepo.github.io/caret/feature-selection-overview.html

[4] *The caret Package* - Max Kuhn - 2019-03-27 - Feature Selection using Univariate Filters - https://topepo.github.io/caret/feature-selection-using-univariate-filters.html

[5] Confounding - https://en.wikipedia.org/wiki/Confounding

[6] *Log-transformation and its implications for data analysis* - Changyong FENG,1,, Hongyue WANG,1 Naiji LU,1 Tian CHEN,1 Hua HE,1 Ying LU,2 and Xin M. TU1 - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4120293/

[7] *The Movie Database* - https://www.themoviedb.org/

[8] https://en.wikipedia.org/wiki/Paranormal_Activity

[9] https://en.wikipedia.org/wiki/The_Blair_Witch_Project

[10] TMDb package - https://cran.r-project.org/web/packages/TMDb/index.html