MovieLense - Movie recommendation Project

Khaled Mahmoud Elshamy

6/1/2020

Movie recommendation

Overview. This document is the report of the **Movielens** project in the course HarvardX: PH125.9x Data Science: Capstone project. This document describes my solution to develop a **recommendation** algorithm to predict ratings using Movielens dataset.

Section 1: describe project background, goal and overview on the dataset.
Section 2: describe the analysis on each variables, what's insights gained and concept of predicting model.
Section 3: evaluate and discuss on the result of the model.
Section 4: conclusion describe a brief summary of the report.

1. Introduction

1.1 Background

In general, recommendation systems use ratings that users have given items to make specific recommendations. Companies that sell many products to many customers and permit these customers to rate their products, like Amazon, are able to collect massive datasets that can be used to predict what rating a particular user will give a specific item. Items for which a high rating is predicted for a given user are then recommended to that user.

In particular, Netflix, the biggest video streaming service in the world today. It serves 190 countries with over 109 million users, with movies and series in 21 languages — and all without ads ruining your viewing experience. Netflix's main source of revenue is subscriptions, which cost between $7.99 and $13.99 per month. Therefore to serve the user (or customer) and keep them stay longer, Netflix uses a recommendation system to predict how many stars a user will give a specific movie and then recommend to that user. One star suggests it is not a good movie, whereas five stars suggests it is an excellent movie.

In this project, we will be creating a movie recommendation system using the MovieLens dataset, which is a small subset of the data used in the Netflix Prize competition, and also trying to reduce **RMSE** (root mean squared error) as much as possible.

1.2 Project Goal

The Goal for this project are as following:

> To create a movie recommendation system and reduce the RMSE less than 0.9849. To practice skills data wrangling, data visualization, reporting using tidyverse, ggplot2, rmarkdown.

1.3 Data set overview

1.3.1 Create train and test data set The data set is generated by following code.

```
################################
# Create edx set, validation set
################################
```

```
# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data

set.seed(1, sample.kind = "Rounding"
         )
```

```
Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler used
```

```
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)
```

```
Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
# Config
library(plotly)
library(flexdashboard)
if(!require(gridExtra)) install.packages('https://cran.rstudio.com/bin/windows/contrib/3.6/gridExtra_2.3
```

Compind edx and removed:

```
edx <- rbind(edx, removed)

# KH # rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

1.3.2 Data overview There are 2 datasets used in this project.

- The edx dataset is used to analyze the predictors, find insights and develop the recommendation system.
- The validation dataset is used to validate effectiveness of our developed recommendation system.

An overview on edx and validation datasets is provided as following.

```
knitr::kable(head(edx),caption = "Top rows of edx file")
```

Edx data set

Table 1: Top rows of edx file

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

The edx file have 9000055 rows and 6 columns. There are 6 columns where:

userId is an integer, identifies an user.

movieId is an integer, identifies an movie.

title is character, represents name of the movie.

rating is an integer, ranging from 1 to 5, made on 5-stars scale (whole and half-star ratings).

timestamp is represented in second since 1/1/1970 UTC.

genres is character, represents the genre.

There are total **69,878** users, **10,677** movies, **797** genres, the data is recorded in 14 years from the first rating date 1995-01-09 to the last rating date 2009-01-05.

```
#create summary table
edx_summary <- data.frame(number_of_rows = nrow(edx),
                          number_of_column = ncol(edx),
                          number_of_users = n_distinct(edx$userId),
                          number_of_movies = n_distinct(edx$movieId),
                          average_rating = round(mean(edx$rating),2),
                          number_of_genres = n_distinct(edx$genres),
                          the_first_rating_date =
                            as.Date(as.POSIXct(min(edx$timestamp),
                                               origin = "1970-01-01")),
                          the_last_rating_date =
                            as.Date(as.POSIXct(max(edx$timestamp),
```

```
                                          origin = "1970-01-01")))

knitr::kable(edx_summary[,1:6],caption = "Summary of edx set (part 1)")
```

Table 2: Summary of edx set (part 1)

number_of_rows	number_of_column	number_of_users	number_of_movies	average_rating	number_of_genres
9000055	6	69878	10677	3.51	797

```
knitr::kable(edx_summary[,7:8],caption = "Summary of edx set (part 2)")
```

Table 3: Summary of edx set (part 2)

the_first_rating_date	the_last_rating_date
1995-01-09	2009-01-05

Validation set The validation set has 999999 rows. There are 6 columns similar to the edx data set. We will use this validation set only at the final phase to evaluate our final recommendation model.

```
validation_summary <- data.frame(number_of_rows = nrow(validation),
                         number_of_column = ncol(validation),
                         number_of_users = n_distinct(validation$userId),
                         number_of_movies = n_distinct(validation$movieId),
                         average_rating = mean(validation$rating),
                         number_of_genres = n_distinct(validation$genres),
                         the_first_rating_date =
                           as.Date(as.POSIXct(min(validation$timestamp),
                                              origin = "1970-01-01")),
                         the_last_rating_date =
                           as.Date(as.POSIXct(max(validation$timestamp),
                                              origin = "1970-01-01")))

#create summary table


knitr::kable(validation_summary[,1:6],caption = "Summary of validation set (part 1)")
```

Table 4: Summary of validation set (part 1)

number_of_rows	number_of_column	number_of_users	number_of_movies	average_rating	number_of_genres
999999	6	68534	9809	3.512033	773

```
knitr::kable(validation_summary[,7:8],caption = "Summary of validation set (part 2)")
```

Table 5: Summary of validation set (part 2)

the_first_rating_date	the_last_rating_date
1995-01-09	2009-01-05

1.4 Project methodology

This project has 4 main steps:

- Step 1 variable analysis: explore and visualize the data to have an overview with-in and between the variables, what's insights gained after analysis. Main package for this step is tidyverse, to handle the cleaning, exploring and visualizing tasks.
- Step 2: develop concept of analysis and modeling. Inputs for this step are insights gained in previous step, the modeling concept from BellKor Pragmatic Chaos - Netflix Grand Prize Winning model, the techniques and model introduced by Dr. Rafael A. Irizarry in the book "Introduction to Data Science - Data Analysis and Prediction Algorithms with R".
- Step 3: analyze and build up the model. In parallel with step 2, we develop the model using edx data set and evaluate its effectiveness by using RMSE on its true rating. Because the edx data set has approximate 9 million rows and we want to get high productivity building up our recommendation model, therefore we use simple calculation method and function in R.
- Step 4: final evaluate the model on the validation set using RMSE. ## **2. Analysis**

2.1 Variable analysis

The objective of this subject is to have a good understanding about each variable of the data set and find any insights to develop the recommendation model.

2.1.1 Rating What is rating?

Rating is the ranking of someone or something based on a comparative assessment of their quality, standard, or performance. A rating is an evaluation or assessment of something, in terms of quality, quantity, or some combination of both.

In our data set, rating is given by users for the movie they watched. The rating is an ordinal scale of number from 0.5 to 5. If they like the movie, they can give 5 stars or if they don't like the movie, they can give lower value.

```
#create a summary table grouping by rating

rating_sum <- edx %>% group_by(rating) %>%
  summarize(count = n())
```

A descriptive summary of rating is as following:

- Average rating of all movies is 3.51
- Standard deviation 1.06

```
gg <- rating_sum %>% mutate(rating = factor(rating)) %>%
  ggplot(aes(rating, count)) +
  geom_col(fill = "steel blue", color = "white") +
  theme_classic() +
  labs(x = "Rating", y = "Count",
       title = "Number of rating",
       caption = "Figure 1 - Rating in edx dataset")
gg
```

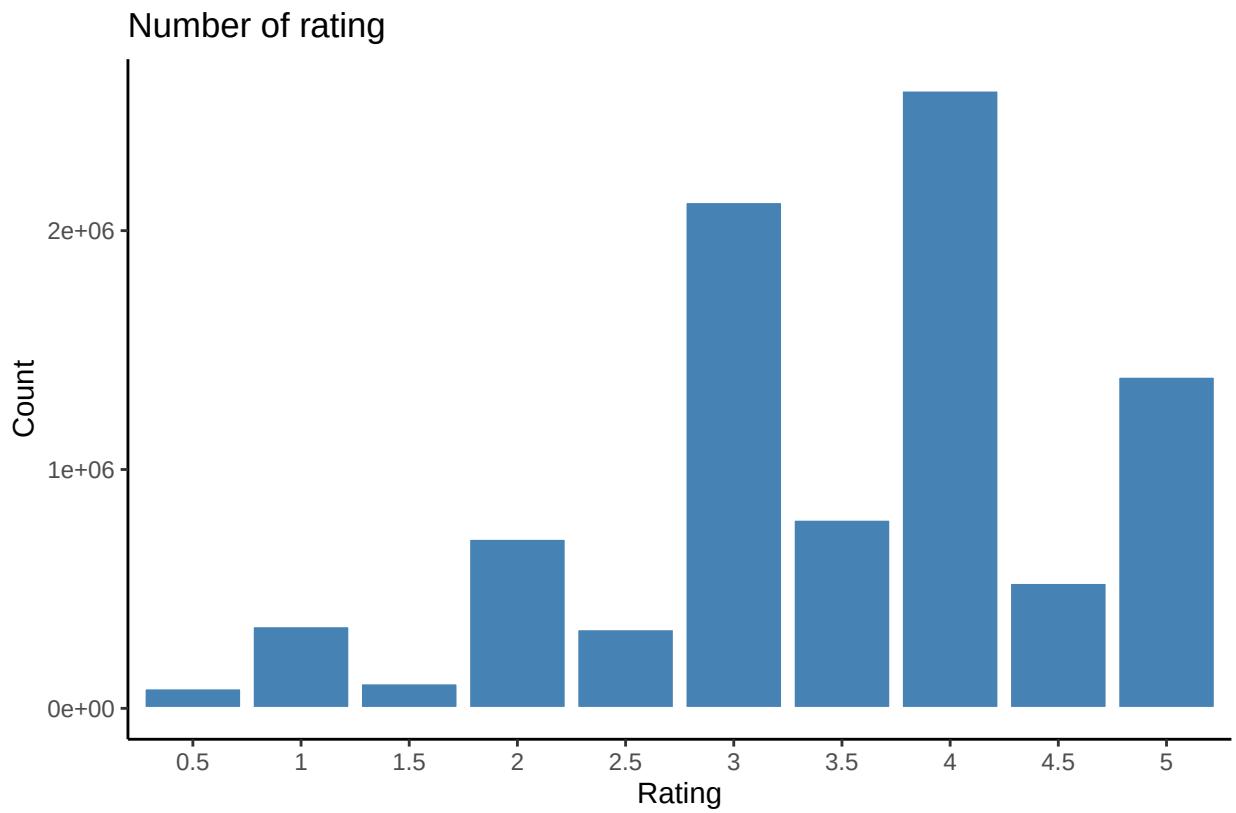


Figure 1 − Rating in edx dataset

We can see in figure 1 that in general, half star ratings are less common than whole star ratings (e.g., there are fewer ratings of 3.5 than there are ratings of 3 or 4, etc.).

The most rating value given by users is summary in figure 3.

```
gg <- rating_sum %>%
  mutate(rating = factor(rating),
         top_rating = ifelse(rating %in% c(3,4,5),"high","low")) %>%
  ggplot(aes(x = reorder(rating, count), count, fill = top_rating)) +
  geom_col(color = "white") +
  coord_flip() +
  scale_fill_manual(values = c("steel blue","grey")) +
  theme_classic() +
  theme(legend.position = "none",
        axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12)) +
  labs(title = "Ranking most rating value",
       x = "Rating",
       caption = "Figure 2 - Data source edx")
gg
```

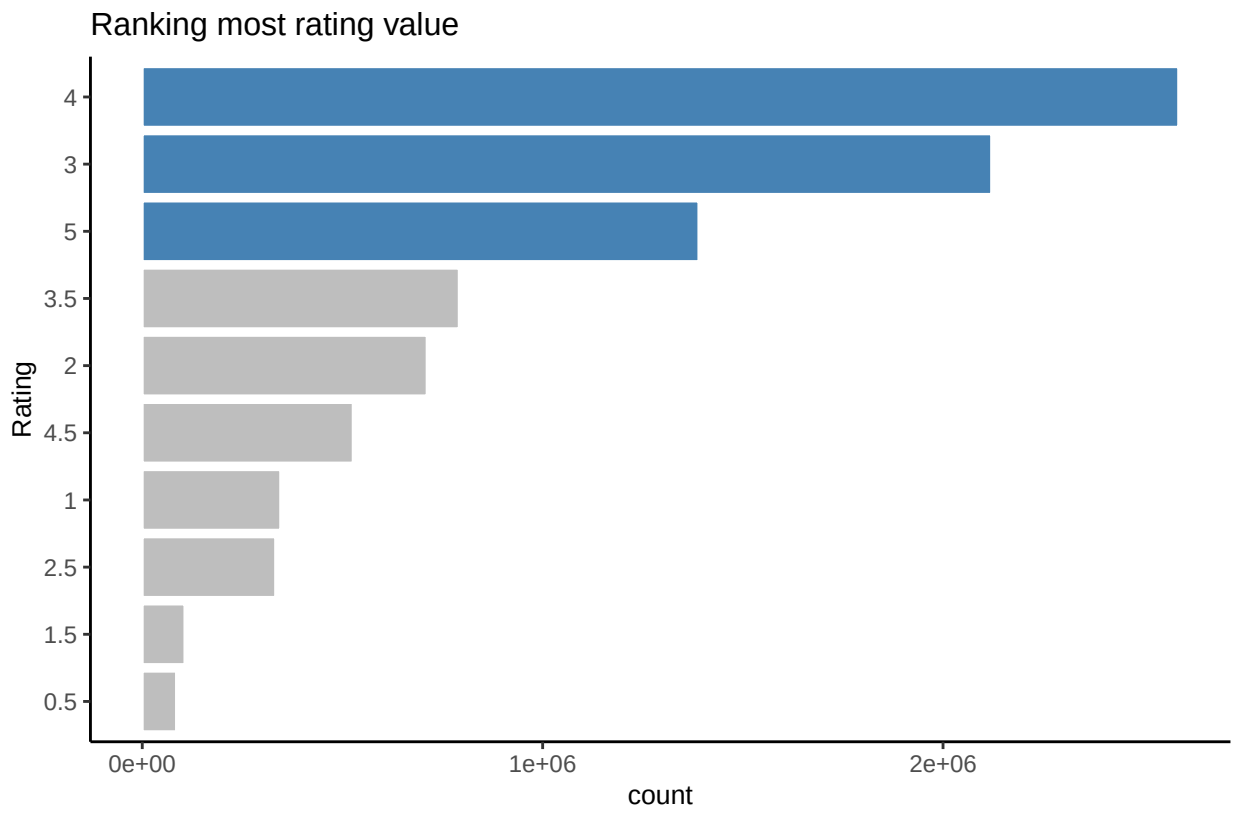


Figure 2 – Data source edx

Since there are not required the predicted ratings are ordinal values, we can develop model which allowed to return continuous values. The advantage of the returned continuous values is that it's easy to set it in order (descence or ascense) and recommend the top highest predicted ratings.

2.1.2 Movie What's data tell us?

The edx data set have total 10,677 movies, each is represented by a movieId.

```
#create summary table grouping by movieId

movie_sum <- edx %>% group_by(movieId) %>%
  summarize(n_rating_of_movie = n(),
            mu_movie = mean(rating),
            sd_movie = sd(rating))
```

```
#create figure of number of rating

gg <- movie_sum %>% ggplot(aes(n_rating_of_movie)) +
  geom_density(fill = "gold1") +
  labs(title = "Density plot - number of rating",
       x = "number of rating",
       y = "density",
       caption = "Figure 3 - The long tail number of rating") +
  geom_vline(aes(xintercept = mean(movie_sum$n_rating_of_movie)), color = "red")+
  annotate("text", x = 2000, y = 0.0022,
           label = print(round(mean(movie_sum$n_rating_of_movie),0)),
           color = "red", size = 3) +
```

```
  theme_classic() +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12),
        legend.position = "none")
```

```
[1] 843
```

```
gg
```

```
Warning: Use of `movie_sum$n_rating_of_movie` is discouraged. Use `n_rating_of_movie` instead.
```

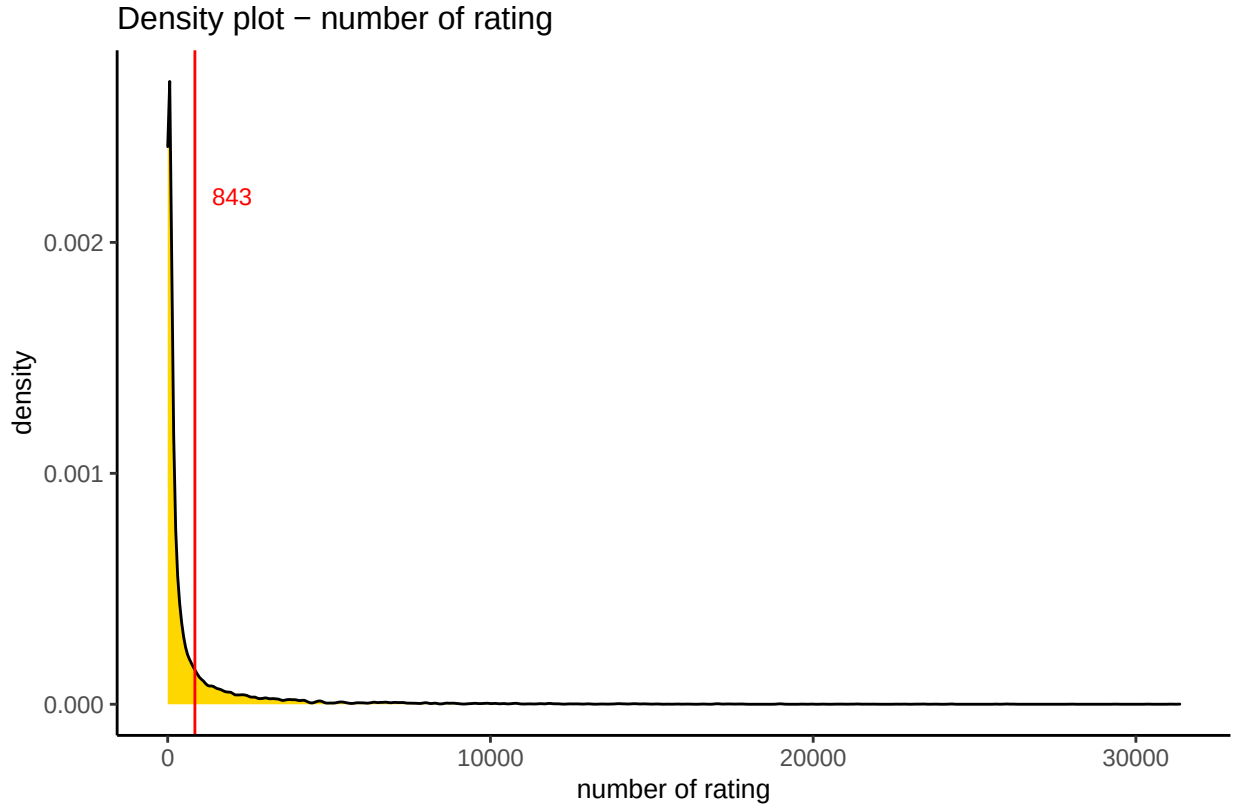


Figure 3 − The long tail number of rating

We can see in figure 3 the number of rating per movie is vary from 1 to 31362, while in average a movie get only 843 ratings by user.

There are 8553 movies, approximate 80 % number of movies, have number of rating less than 843 but only contribute 15 % number of rating.

The remain 2124 movies, which number of rating is greater than 843, contribute 85 number of rating in whole data set.

Prediction accuracy for movies with less number of rating is probably a challenge. The first ratings of those movies, probably come from its production, owners, respective people… In the edx set, there are 5% movies with less than 5 number of ratings, 9.9% movies with less than 10 number of ratings. This is not a high ratio.

Another perspective from figure 4, this first one is rating of the movies with less number of rating are widely varied from 0.5 star to 5 stars, white rating of the movies with high number of rating are more focused. This is probably a challenge to predict the true rating for movie with very less number of rating. The second one

is average rating is increased when the number of rating in increasing.

```
gg <- movie_sum %>%
  ggplot(aes(n_rating_of_movie, mu_movie)) +
  geom_point(color = "steel blue", alpha = 0.3) +
  geom_smooth()+
  geom_vline(aes(xintercept = mean(movie_sum$n_rating_of_movie)), color = "red")+
  annotate("text", x = 2000, y = 5,
           label = print(round(mean(movie_sum$n_rating_of_movie),0)),
           color = "red", size = 3) +
  theme_classic() +
  labs(title = "Scatter plot - Average rating vs number of rating",
       x = "Number of rating / movie",
       y = "Average rating",
       caption = "Figure 4") +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12))
```

```
[1] 843
```

```
gg
```

```
Warning: Use of `movie_sum$n_rating_of_movie` is discouraged. Use `n_rating_of_movie` instead.
```

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

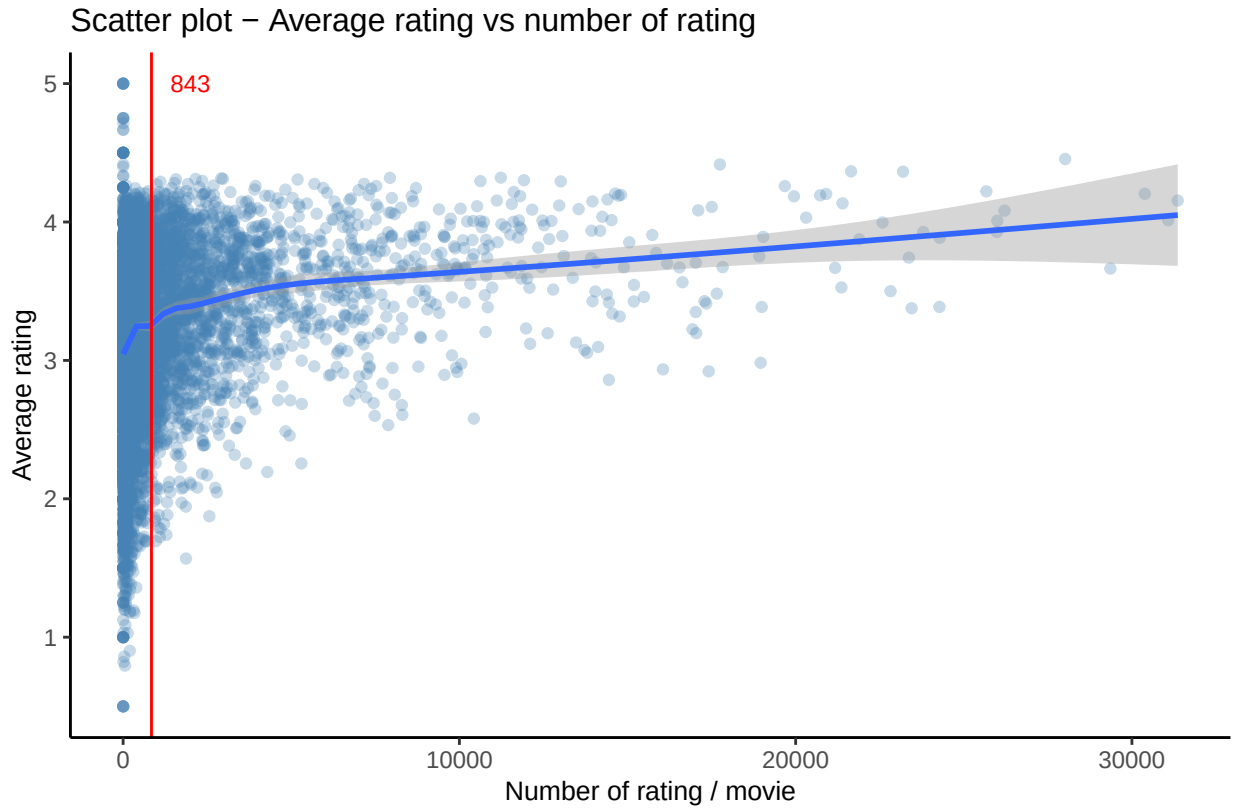


Figure 4

```
gg <- movie_sum %>%
  mutate(group = cut(n_rating_of_movie,
```

```
                    breaks = c(-Inf, mean(n_rating_of_movie),Inf),
                    labels = c("n < 843", "n > 843"))) %>%
  ggplot(aes(sd_movie, fill = group)) +
  geom_density(alpha = 0.5) +
  labs(title = "Standard deviation of rating",
       x = "Standard deviation",
       y = "count",
       caption = "Figure 7 -
       N < 843 number of rating less than average,
       N > 843 number of rating greater than average") +
  theme_classic() +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12))

gg
```

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

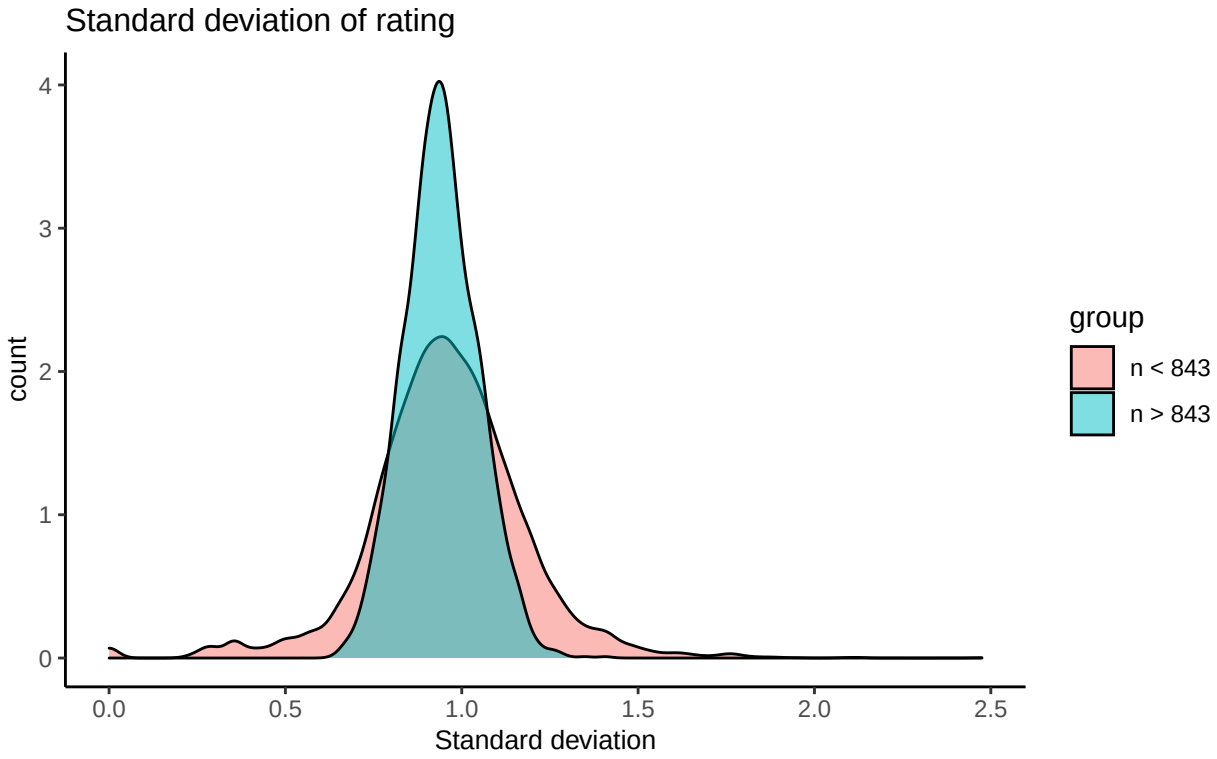


Figure 7 -
N < 843 number of rating less than average,
N > 843 number of rating greater than average

What's insights

Data is unbalancing since 19.9% movies, which has number of rating greater than 843, contribute 85% number of rating in whole data set.

Average rating of a movie tend to increase when the number of rating in increasing.

Rating values of a movie is less varied when the number of rating is increased.

2.1.3 User What's data tell us?

The second variable impact to the rating of a movie is effect of each specific user. In summary, the edx set have 69,878 users represented by an userId.

```
#create summary table grouping by userId

user_sum <- edx %>% group_by(userId) %>%
  summarize(n_user_rated = n(),
            mu_user = mean(rating),
            sd_user = sd(rating))
```

```
#create figure of number of rating

gg <- user_sum %>% ggplot(aes(n_user_rated)) +
  geom_density(fill = "steel blue", alpha = 0.8) +
  labs(title = "Density plot - number of user rated",
       x = "number of rating",
       y = "density",
       caption = "Figure 8") +
  geom_vline(aes(xintercept = mean(user_sum$n_user_rated)), color = "red")+
  annotate("text", x = 400, y = 0.009,
           label = print(round(mean(user_sum$n_user_rated),0)),
           color = "red", size = 3) +
  theme_classic() +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12),
        legend.position = "none")
```

```
[1] 129
```

```
gg
```

```
Warning: Use of `user_sum$n_user_rated` is discouraged. Use `n_user_rated` instead.
```

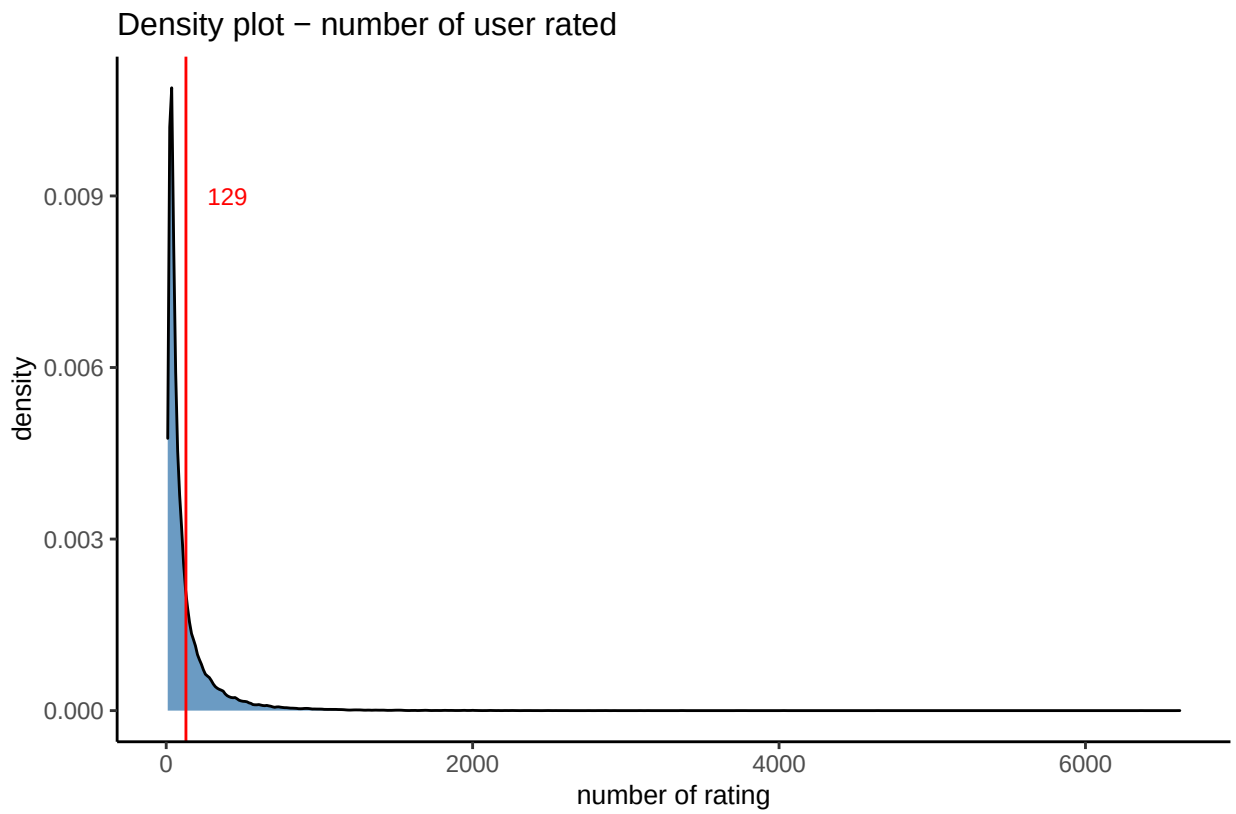


Figure 8

A same density plot number of rating given by each users in figure 8, 30 % users have the number of rating given higher than the average of all users, contribute over 70 %number of rating. This mean that some users are very active than other users.

The top 10 users have highest number of rating as as below:

```
top_10_users <- user_sum %>% arrange(desc(n_user_rated)) %>% head(10)


knitr::kable(top_10_users,caption = "Top 10 users rating")
```

Table 6: Top 10 users rating

userId	n_user_rated	mu_user	sd_user
59269	6616	3.264586	0.6386508
67385	6360	3.197720	0.9568548
14463	4648	2.403615	0.6881860
68259	4036	3.576933	1.0513298
27468	4023	3.826871	0.7343875
19635	3771	3.498807	0.7783109
3817	3733	3.112510	0.5787978
63134	3371	3.268170	0.9568703
58357	3361	3.000744	0.7984822
27584	3142	3.001432	0.7187717

Compare to the top 10 users with lowest number of rating given.

```
#top 10 users have lowest number of rating

lowest_10_users <- user_sum %>% arrange(n_user_rated) %>% head(10)
 knitr::kable(lowest_10_users,caption = "Lowest 10 users rating")
```

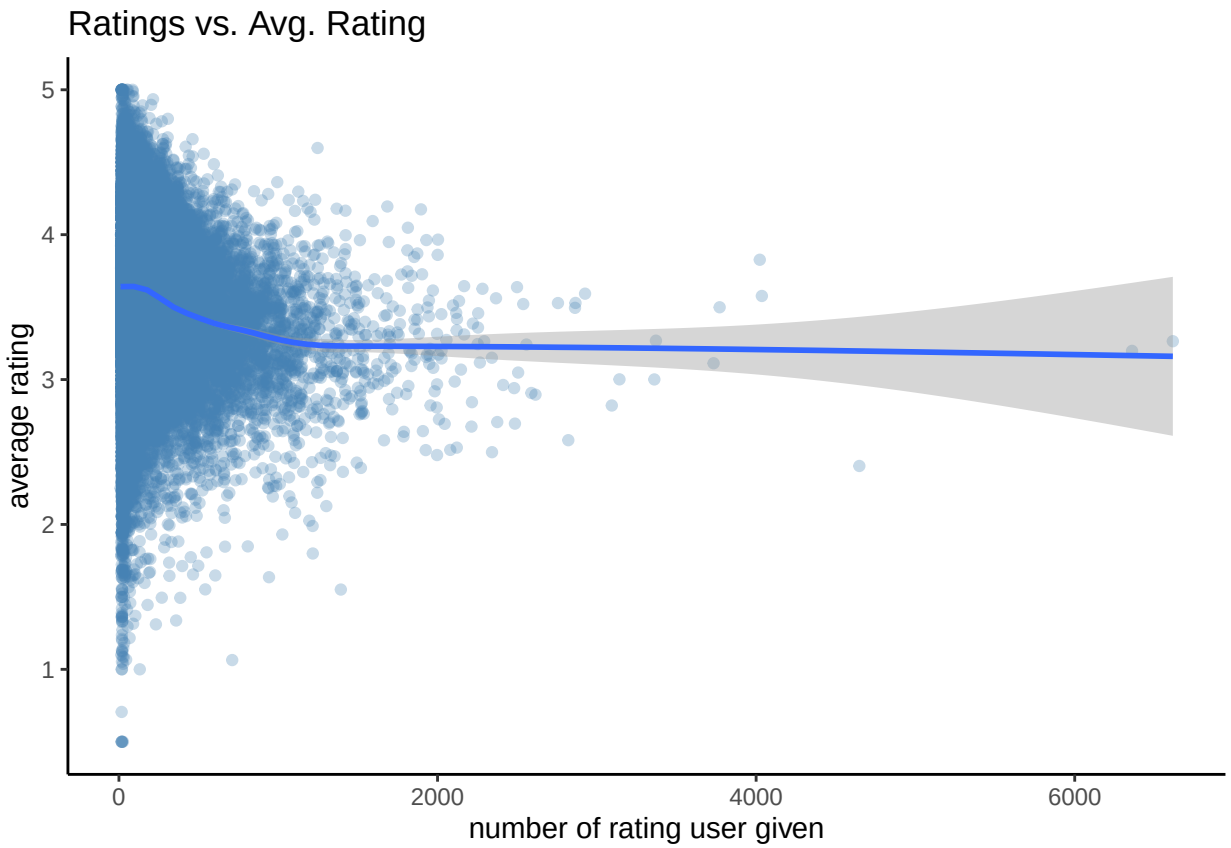
Table 7: Lowest 10 users rating

userId	n_user_rated	mu_user	sd_user
62516	10	2.250000	1.1365151
22170	12	4.000000	0.7385489
15719	13	3.769231	1.2351684
50608	13	3.923077	1.4411534
901	14	4.714286	0.4688072
1833	14	3.000000	1.2403473
2476	14	2.928571	1.3280573
5214	14	1.785714	1.2202504
9689	14	3.571429	1.2224997
10364	14	4.321429	1.2650197

In average, an user given 129 ratings, while the highest number of rating given by a user was 6616 and the lowest number of rating given by a user was 10. This is very unbalance.

```
user_sum %>%
  ggplot(aes(n_user_rated, mu_user)) +
  geom_point(color = "steel blue", alpha = 0.3) +
  geom_smooth() +
  theme_classic() +
  ylab("average rating") +
  xlab("number of rating user given") + ggtitle("Ratings vs. Avg. Rating")
```

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



We see in the figure 9 that when the number of rating user given is low, the average rating is more varied compared to when the number of rating user given is high. Some users tend to give rating lower than average, and some users tend to give rating higher than average. Compare to average rating of whole edx data set is 3.51 , there are 62 % users have average rating greater than the averate rating of whole edx data set. In the meanwhile, there are 38 % users have average rating less than the averate rating of whole edx data set.

Furthermore, there is no different of average rating's standard deviation between group users with number of rating given less than and greater than the average rating (129 rating per user). We can see it in figure 10 as following.

```
gg <- user_sum %>%
  mutate(group = cut(n_user_rated,
                     breaks = c(-Inf, mean(n_user_rated), Inf),
                     label = c("< 129", ">129"))) %>%
  ggplot(aes(sd_user, fill = group)) +
  geom_density(alpha = 0.5) +
  labs(title = "Standard deviation of rating by user",
       x = "Standard deviation", y = "count",
       caption = "Figure 10") +
  theme_classic() +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12),
        legend.position = "none")

gg
```

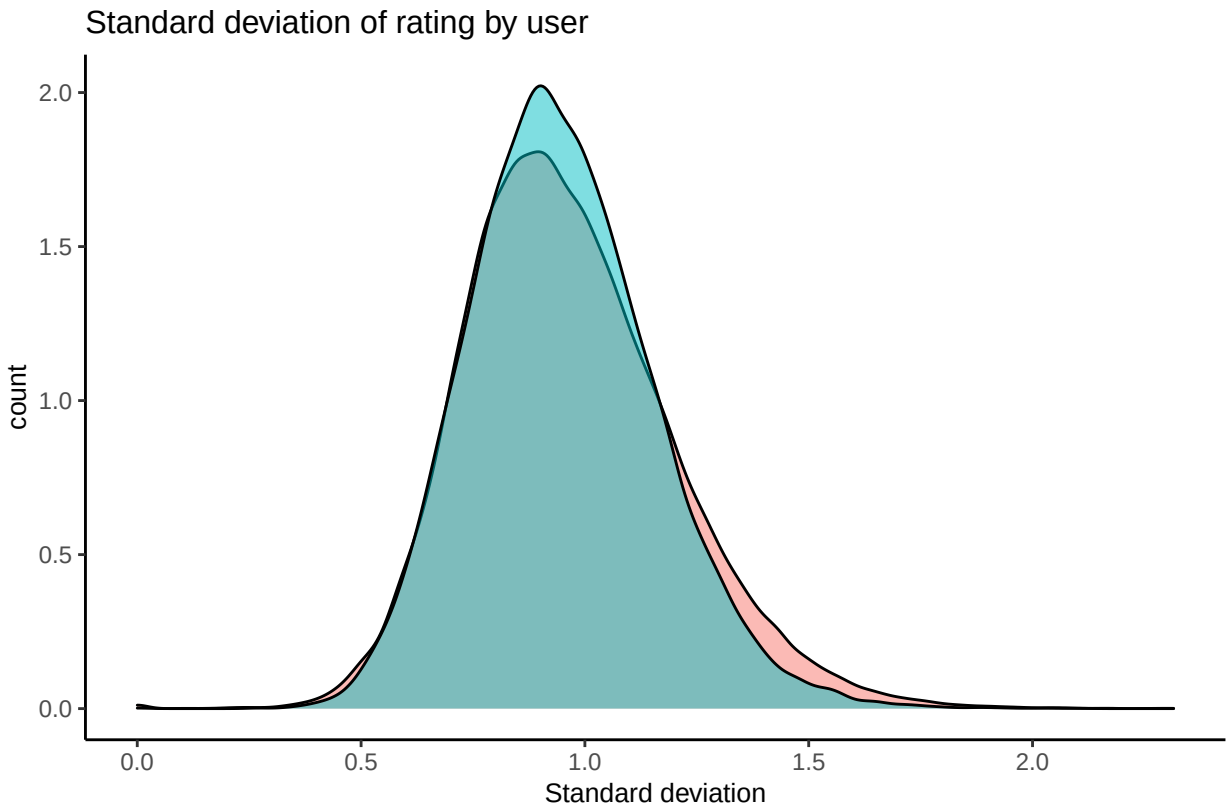


Figure 10

What's insights

The number of rating given by each user is unbalance. 30 % users contribute 70 % number of rating in whole edx dataset.

Variation of user is more varied when the number of rating is small, and is closer to the overall average of all users when the number of rating is increasing.

A specific user tend to give rating score less than or greater than the overall average of all users.

2.1.4 Time effect

What's data tell us?

In our dataset, time is recorded as the Unix timestamp, which is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC. Therefore, the Unix timestamp is merely the number of seconds between a particular date and the Unix Epoch.

To make this variable be more friendly, we convert this to date-time format and assigned it to new variable rating year.

```
edx_up1 <- edx %>%
  mutate(rating_time = as.Date(as.POSIXct(timestamp, origin = "1970-01-01"))) %>%
  mutate(rating_year = year(rating_time))
```

Adding the release year of each movie.

```
edx_up2 <- edx_up1 %>%
  mutate(release_year = as.integer(substr(title, str_length(title) - 4,
                                  str_length(title) - 1)))
```

Overview the rating trend by the release year and rating year.

```
release_year_sum <- edx_up2 %>% group_by(release_year) %>%
  summarize(n = n(), average_rating = mean(rating))
```

```
p1 <- ggplot(release_year_sum, aes(release_year, n)) +
      geom_point(color = "steel blue", alpha = 0.6) +
      geom_line(color = "steel blue") +
      theme_classic() +
      labs(title = "number of movies by release year",
         caption = "Figure 11") +
      theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12),
        legend.position = "none")

p2 <- release_year_sum %>% ggplot(aes(release_year, average_rating)) +
      geom_point(color = "steel blue", alpha = 0.6) +
      theme_classic() +
      geom_smooth() +
      labs(title = "average rating by release year",
         caption = "Figure 12") +
      theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12),
        legend.position = "none")

 grid.arrange(p1, p2, nrow = 1)
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

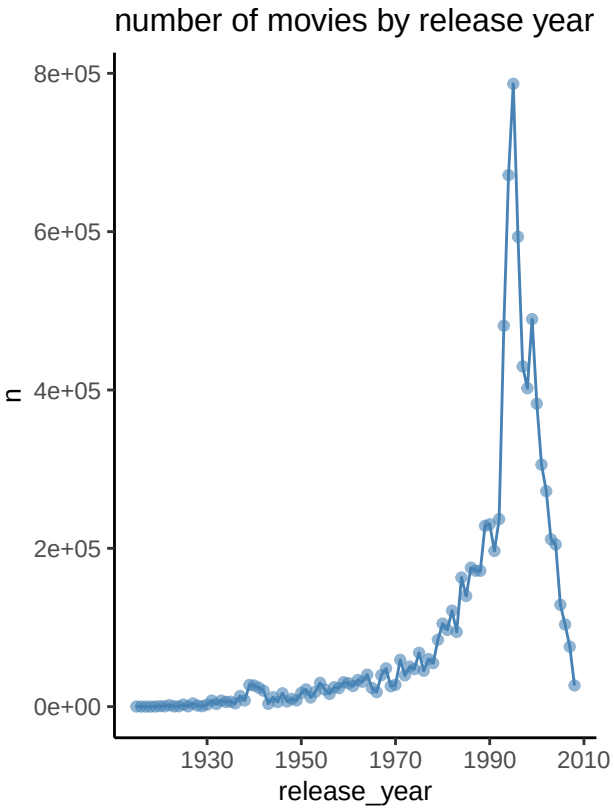


Figure 11

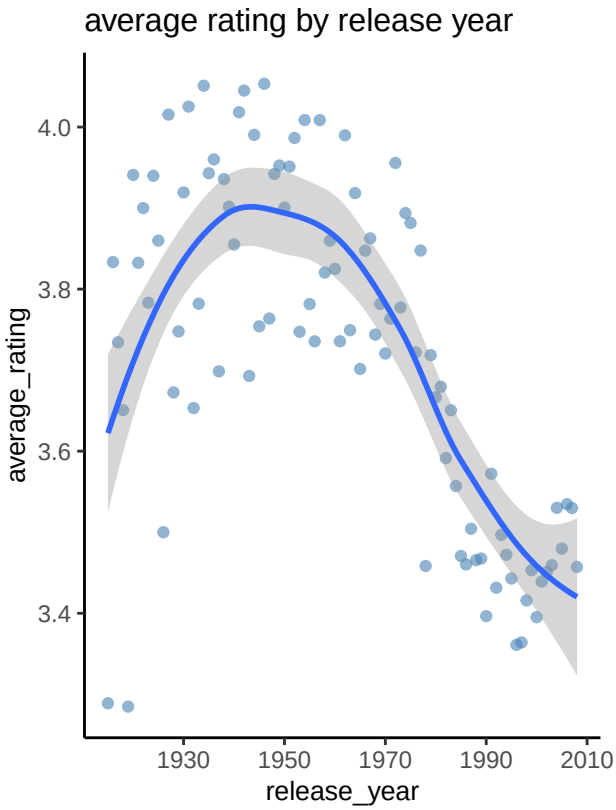


Figure 12

We see in figure 11 that almost movies were released after decade 1980s. The figure 12 show that the average rating is increased from 1915 to approximate 1945, and decreased after that. It look like that the rating is correlated to the released year. The correlation between the rating and release can be explained by following code.

```
fit_lm <- lm(average_rating ~ I(release_year^3) + I(release_year^2) +
             I(release_year),
           data = release_year_sum)
summary(fit_lm)
```

```
Call:
lm(formula = average_rating ~ I(release_year^3) + I(release_year^2) +
    I(release_year), data = release_year_sum)

Residuals:
     Min       1Q   Median       3Q      Max
-0.38963 -0.07844  0.01702  0.08197  0.24390

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)       -2.730e+04  6.290e+03  -4.341 3.70e-05 ***
I(release_year^3)  3.542e-06  8.337e-07   4.249 5.22e-05 ***
I(release_year^2) -2.100e-02  4.906e-03  -4.281 4.65e-05 ***
I(release_year)    4.148e+01  9.622e+00   4.311 4.14e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.1268 on 90 degrees of freedom
Multiple R-squared:  0.647, Adjusted R-squared:  0.6353
F-statistic: 54.99 on 3 and 90 DF,  p-value: < 2.2e-16
```

Another point of view from time effect is the effect of aging time, which is defined by the period of time from the first rating given by anyone to a specific rating time after that. We will calculated this new variable aging time with unit of measure by month.

```
#calculate the first rating time of each movie
movie_sum <- edx_up2 %>% group_by(movieId) %>%
  summarize(n_rating_of_movie = n(),
            mu_movie = mean(rating),
            first_rating_time = min(timestamp))

#calculate the aging time
edx_up3 <- edx_up2 %>% left_join(movie_sum, by = "movieId")
edx_up4 <- edx_up3 %>%
  mutate(aging_time = round((timestamp - first_rating_time)/60/60/24/30,0))

#create a summary table grouping by aging time
aging_time_sum <- edx_up4 %>% group_by(aging_time) %>%
  summarize(n_aging_time = n(),
            average_rating = mean(rating))
```

```
#visualize by ggplot
p1 <- ggplot(aging_time_sum, aes(aging_time, n_aging_time)) +
               geom_point(color = "steel blue") +
               geom_line(color = "steel blue") +
               theme_classic() +
               labs(title = "number of rating per aging time",
                    x = "aging time (month)",
                    y = "count",
                    caption = "Figure 13") +
               theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12),
        legend.position = "none")

p2 <- ggplot(aging_time_sum, aes(aging_time, average_rating)) +
               geom_point(color = "steel blue") +
               geom_line(color = "steel blue") +
               theme_classic() +
               labs(title = "average rating per aging time",
                    x = "aging time (month)",
                    y = "average rating",
                    caption = "Figure 14") +
          theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12),
        legend.position = "none")
grid.arrange(p1, p2, nrow = 1)
```

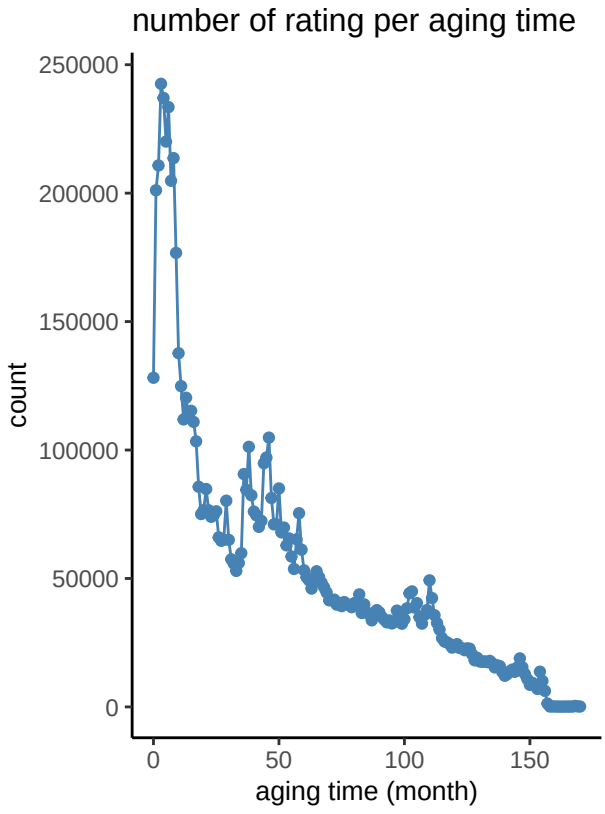


Figure 13

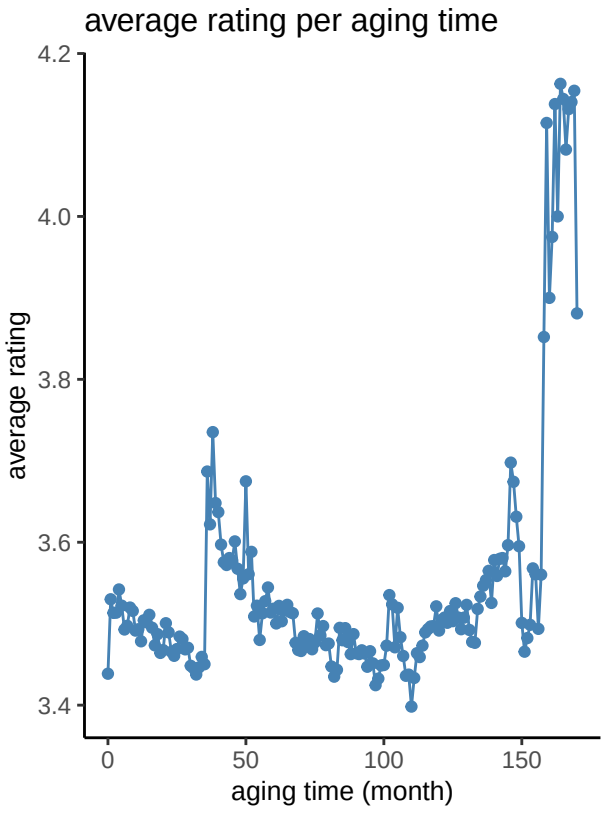


Figure 14

The figure 13 show that the number of rating of a movie is increased after its first rating got by any user, from month 0 to month 3 th. After month 3 th, the number of rating is decreased by the time.

The figure 14 show three trends on the data.

The first one is the average rating is reduced when the aging time increased from 0 to around 40 month.

The second one is the average rating have a increasing step and higher average rating at the aging time 40 month, but the trend is also decreased from month 40th to month 100th.

From month 100th to later, the average rating is increased. But number of rating user given after month 100th is not much, about 14 % of the total rating user given in the edx dataset. And the number of rating user given after month 150th is very low, about 0.6 % of the total rating user given in the edx dataset. Therefore we can ignore those uncertainty.

What's insights?

Rating of a movie is correlated to its release year (R2 = 0.647).

The number of rating of a movie is increased after aging time month 0 and get highest value at aging time month 3, after that it reduce by the time.

Rating of a movie is higher when aging time is close to month 0, and decreased by the time when aging time is increased.

2.1.5 Genres effect Genre is the term for any category of literature or other forms of art or entertainment, e.g. music, whether written or spoken, audio or visual, based on some set of stylistic criteria. Genres are formed by conventions that change over time as new genres are invented and the use of old ones are discontinued. Often, works fit into multiple genres by way of borrowing and recombining these conventions.

What's data tell us?

A movie could be classified to one or more genres, grouping by variable genres in the edx data set give us 797 levels of genres. If we separated it out, there are only 20 levels of genre.

```
#create a vector of genres
genres <- str_replace(edx_up4$genres,"\\|.*","")
genres <- genres[!duplicated(genres)]
genres
```

```
 [1] "Comedy"             "Action"             "Children"           "Adventure"          "Animation"
 [6] "Drama"              "Crime"              "Sci-Fi"             "Horror"             "Thriller"
[11] "Film-Noir"          "Mystery"            "Western"            "Documentary"        "Romance"
[16] "Fantasy"            "Musical"            "War"                "IMAX"               "(no genres lis
```

Calculate the number of movies per each genre and average rating per each genres

```
#calculate the number of movies per each genres
n_genres <- sapply(genres, function(ge){
  index <- str_which(edx_up4$genres, ge)
    length(edx_up4$rating[index])

})

#calculate the average rating of each genres
genres_rating <- sapply(genres, function(ge){
  index <- str_which(edx_up4$genres, ge)
  mean(edx_up4$rating[index], na.rm = T)
})

#create a summary data by genres
genres_sum <- data.frame(genres = genres,
                         n_genres = n_genres,
                         average_rating = genres_rating)

#print out the summary table by genres
sum_by_geners <- genres_sum %>% arrange(desc(n_genres)) %>% head

knitr::kable(sum_by_geners,caption = "Summary By Genners")
```

Table 8: Summary By Genners

genres	n_genres	average_rating
Drama	3910127	3.673131
Comedy	3540930	3.436908
Action	2560545	3.421405
Thriller	2325899	3.507676
Adventure	1908892	3.493544
Romance	1712100	3.553813

```
#ranking genres by number of each appear in the edx data set
g1 <-  genres_sum %>%
    mutate(top5 = ifelse(genres %in% c("Comedy","Drama","Action","Thriller","Adventure"),
                         "top5","non")) %>%
    ggplot(aes(x = reorder(genres, n_genres), n_genres, fill = top5)) +
               geom_col(color = "white") +
```

```
                theme_classic() +
                coord_flip() +
                labs(title = "number of movie by genres",
                     y = "number of rating",
                     x = "genres",
                     caption = "Figure 15") +
    scale_fill_manual(values = c("grey","steel blue")) +
    theme(legend.position = "none")
# comparing average rating of each genres in edx data set
g2 <-  ggplot(genres_sum,
         aes(x = reorder(genres, average_rating), average_rating)) +
                geom_col(fill = "steel blue", color = "white") +
                theme_classic() +
                coord_flip() +
                labs(title = "average rating by genres",
                     y = "average rating", x = "genres",
                     caption = "Figure 16")

grid.arrange(g1, g2, nrow = 1)
```

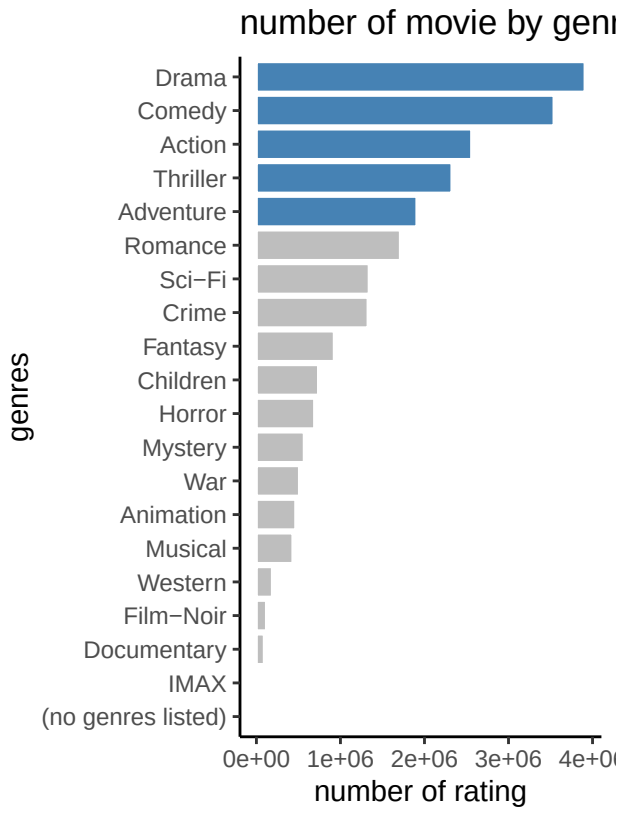


Figure 15

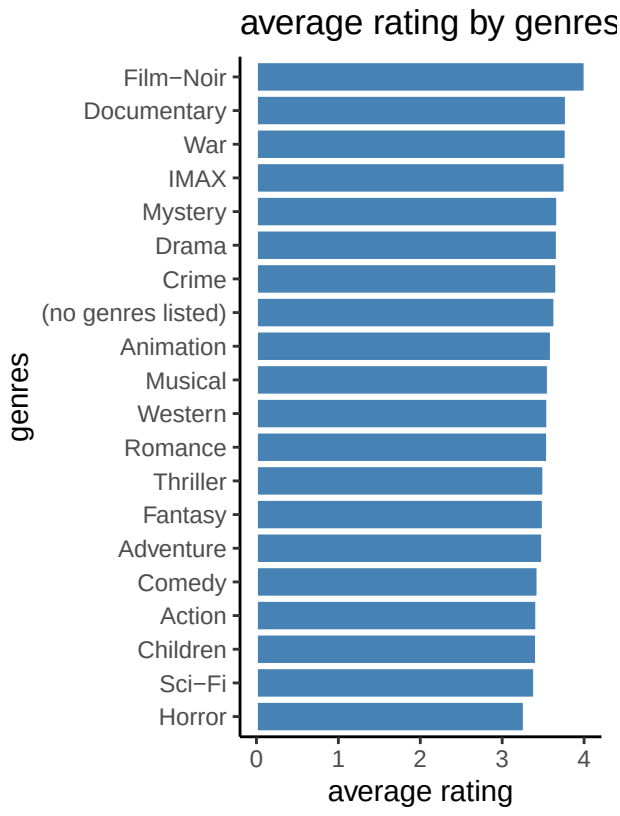


Figure 16

Top 5 popular genres Drama, Comedy, Action, Thriller, Adventure appear 61 % in the edx data set. Film-Noir have highest average rating and Horror have lowest average rating compared to other movies.

Because almost movies have more than 1 genre, we will analyze their rating change by original multiple genres instead of single genre.

```r
gg <- edx_up4 %>% group_by(genres) %>%
  summarize(count = n(), rating = mean(rating)) %>%
  ggplot(aes(x = reorder(genres, rating), rating)) +
           geom_col(fill = "steel blue") +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12)) +
  labs(x = "multiple genres",
       y = "average rating",
       title = "average rating by genres in edx data",
       caption = "Figure 17")
gg
```

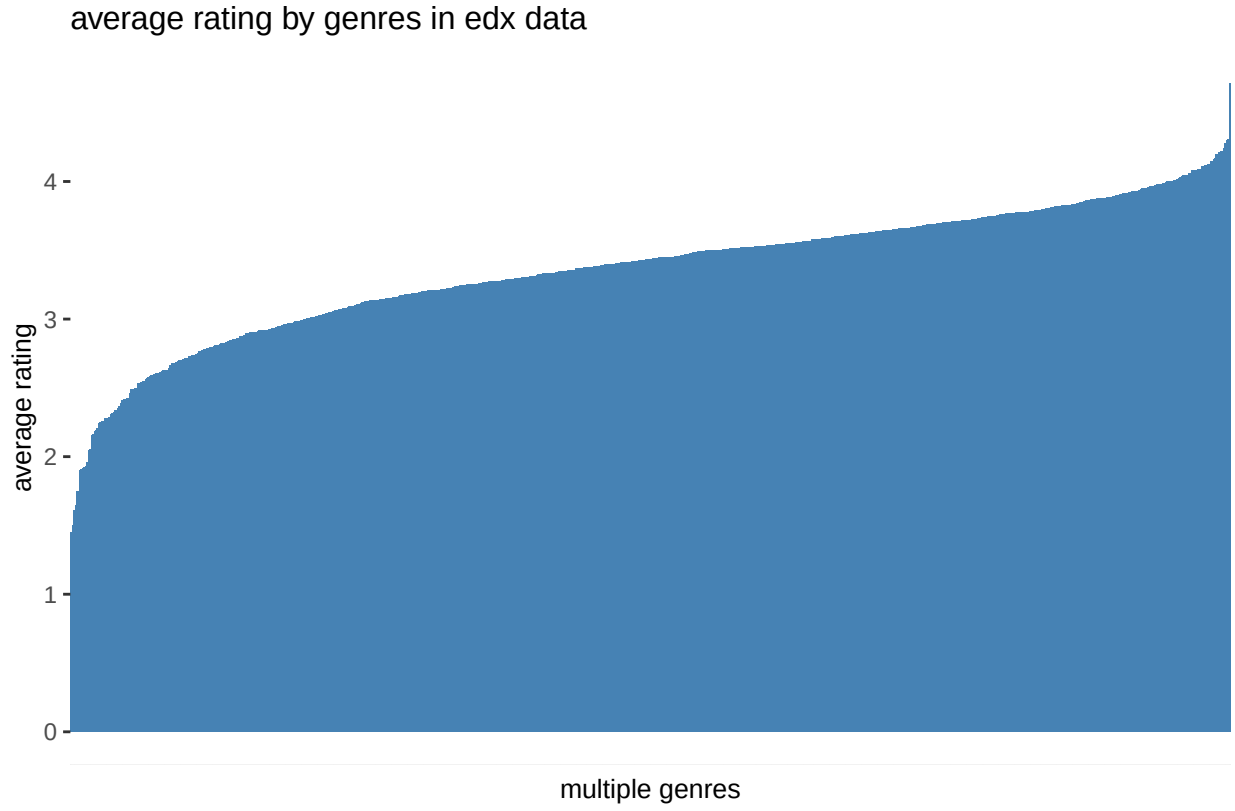


Figure 17

Similarly, average rating is varied across movie genres. Suppose this variation is due to user effect, we will select a group of users to validate this hypothesis. The users is sampling based on the number of rating they given, to choose group of average users, the number of rating given is approximate the average value 129 +/- 1.

```r
#create list average users
avg_user_list <- user_sum %>%
  filter(n_user_rated >= round(mean(n_user_rated),2)-1,
         n_user_rated <= round(mean(n_user_rated),2)+1) %>%
  select(userId, mu_user)
```

```
#select randomly 4 users
set.seed(1, sample.kind = "Rounding")
```

```
Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler used
```

```
avg_user_list <- sample(avg_user_list$userId, 10)
avg_user_list
```

```
 [1] 18984 26573 40923 63214 13487 62715 65315 46709 45207  5035
```

```
#create figure of rating change by genres of average users
gg <- edx_up4 %>% filter(userId %in% avg_user_list) %>%
  group_by(genres) %>%
  summarize(rating = mean(rating), count = n()) %>%
  ggplot(aes(reorder(genres, rating), rating))+
  geom_col(fill = "steel blue") +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank()) +
  labs(x = "genres", title = "Rating change by genres",
       caption = "Figure 18") +
  theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12),
        legend.position = "none")

gg
```

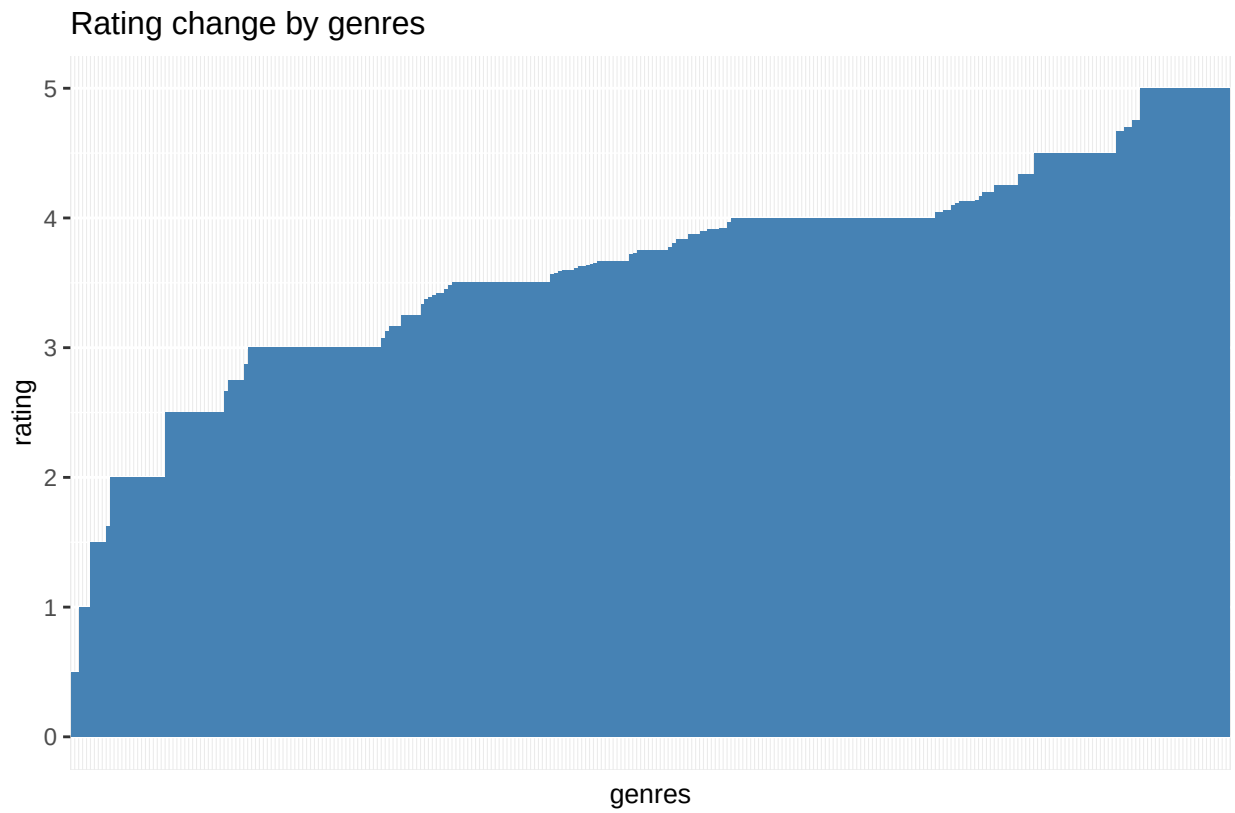


Figure 18

Similarly, the average rating is varied across genres, and we can see the variation is bigger, some genres are

rated at 5 stars, and some genres are rated at 1 star.

What's insights?

Rating is varied across genres. Some genres's rating are higher or lower compared to other genres.

A user given high rating for some genres (suppose those are his/her favor) and low rating for some genres (not his/her favor) compared to other genres.

2.2 Concept of model

2.2.1 BellKor's model - Netflix Prize Winner On September 18, 2009, Netflix announced team "BellKor's Pragmatic Chaos" as the prize winner (a Test RMSE of 0.8567), and the prize was awarded to the team in a ceremony on September 21, 2009. [1]

In a summary of the BellKor Solution to the Netflix Gran Prize [2], Yehuda Koren highlighted that the collaborative filtering (CF) models try to capture the interactions between users and items that produce the different rating values. However, many of the observed rating values are due to effects associated with either users or items, independently of their interaction. A prime example is that typical CF data exhibit large user and item biases – i.e., systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others.

To encapsulate those effects, which do not involve user-item interaction, one technique introduced by BellKor's Pragmatic Chaos was baseline prediction model.

The predicted value from baseline prediction model can be decomposed to following formula:

Where:

$y_{i,j}$: is the true rating value of $movie_i$ and $user_i$.
$\hat{y}_{i,j}$: is the predicted value of $movie_i$ and $user_j$.
b_i : is a movie-specific effect and b_j is a user-specific effect.
$\varepsilon_{i,j}$: is residual.

2.2.2 Model: baseline is average rating of each movie From our analysis in previous section, we know:

Ratings of a movie are less varied when the number of rating is increased.

Data is unbalancing: In our edx data set, we have total 10677 movies, and 7989 have the number of rating more than 30. This ratio is approximate 75 %. And we have 9538 movies have the number of rating more than 10, approximate 89 % of total number of movies.

The Central Limit Theorem (CLT) tells us that when the sample size is large enough, the probability distribution of the sum of the independent sample is approximately normal. Large is a relative term, however in many circumstances as few as sample size is more than 30 is enough to make the CLT useful. In some specific instances, as few as 10 is enough.[3]

By the law of the large number, we know that the standard error of the average becomes smaller and smaller as the sample size grows larger. When the sample size is very large, then the standard error is practically 0 and the average of the sample converges to the average of the population. Therefore when a movie have the number of rating increased, the overall rating of the movie would be closed to the true rating.

We assume that the true quality of moviei represented by the average rating value μ_i. We develope our model which the baseline is the average rating of each movie:

with:

μ_i : the average rating of $movie_i$

$\varepsilon_{i,j}$: the residual.

Now we calculated the RMSE, effected only by the average rating of each movie ($\hat{y}_{i,j} = \mu_i$) :

```
model_1_movie <- RMSE(edx_up4$mu_movie, edx_up4$rating)

rmse_results <- data_frame(method="Only baseline is movie average",
                                     RMSE = model_1_movie)
```

```
knitr::kable(rmse_results,caption = "RSME by Method Result")
```

Table 9: RSME by Method Result

method	RMSE
Only baseline is movie average	0.9423475

RMSE is 0.9423, still far compared to our target (reducing RMSE less than 0.8649).

2.2.3 Adding specific effect by user From what we known in previous section:

30 % users contribute 70 % number of rating in whole data set.

The rating of each user is very varied.

A specific user tend to give rating score less than or greater than the overall average of all users.

We add the specific-effect by user b_j to improve our model accuracy. The specific-user effect can be calculated as following:

The b_j is calculated by following code:

```
b_j_sum <- edx_up4 %>% mutate(yhat = rating - mu_movie) %>%
  group_by(userId) %>%
  summarize(n_user_rated = n(),
            b_j = mean(yhat))
```

```
  p1<- b_j_sum %>% ggplot(aes(b_j)) +
  geom_histogram(fill = "steel blue", color = "white") +
  theme_classic() +
  labs(title = "Distribution of user bias",
       caption = "Figure 19") +
    theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12))

  p2 <- b_j_sum %>% ggplot(aes(n_user_rated, b_j)) +
  geom_point(color = "steel blue", alpha = 0.5) +
  theme_classic() +
  labs(title = "Scatter plot of user bias and number of user rated",
       caption = "Figure 20") +
    theme(axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10),
        plot.title = element_text(size = 12))

grid.arrange(p1, p2, nrow = 1)
```

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

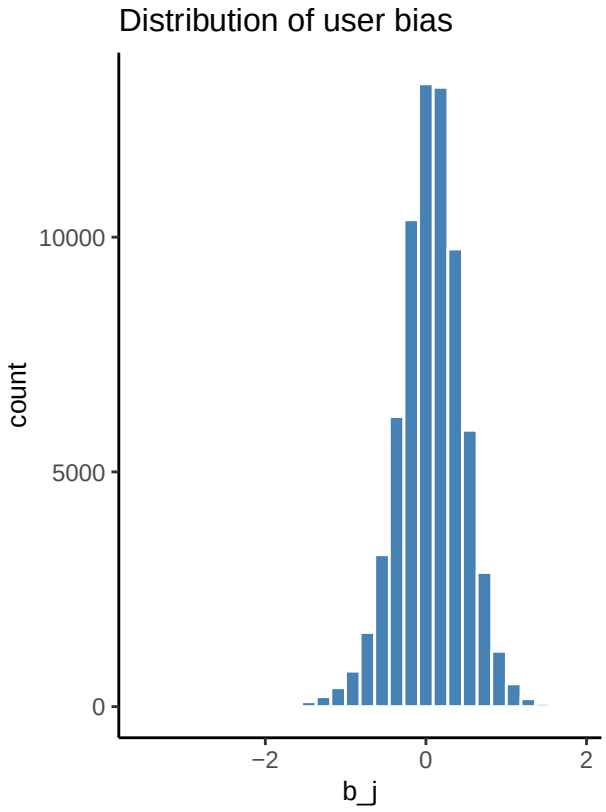


Figure 19

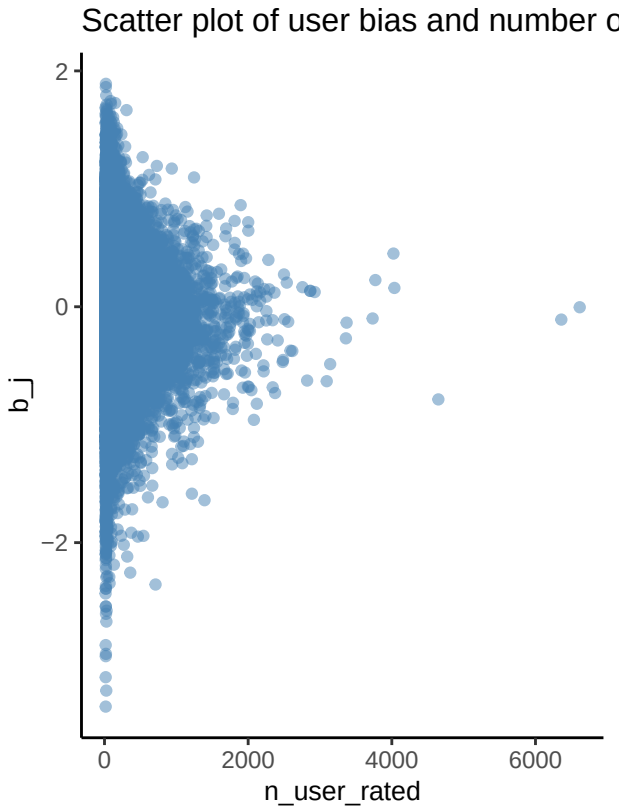


Figure 20

The scatter plot of b_j and number of ratings user given is similar to the scatter plot of average rating and number of ratings user given which we explore in previous section. Adding this specific-user effect to our predicting model, the predicted ratings will be calculated as following:

```
edx_up5 <- edx_up4 %>%
  left_join(b_j_sum, by = "userId") %>%
  mutate(mu_movie_user = mu_movie + b_j)
```

The RMSE is as following:

```
model_2_movie_user <- RMSE(edx_up5$mu_movie_user, edx_up5$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Add Specific-effect of user",
                                     RMSE = model_2_movie_user))



knitr::kable(rmse_results,caption = "RSME by Method effect of user")
```

Table 10: RSME by Method effect of user

method	RMSE
Only baseline is movie average	0.9423475
Add Specific-effect of user	0.8567039

2.2.4 Add specific-effect of time Refer to the BellKor winning model [2], Yehuda highlighted that the fact that an item's popularity may change over time. For example, movies can go in and out of popularity as triggered by external events such as the appearance of an actor in a new movie.

From our analysis in post sections (2.1.4 and 2.1.5) we know that the average rating was decreasing when the aging time increased.

Therefore we can study the relationship between the residual $\varepsilon_{u,i}$,i and remain variables which we did not add to our above model (time - effect and genres - effect). We can decompose the true rating value as following:

Where the time - effect b_{time} can be calculated by:

The new predicted rating will be calculated by following formular:

The b_{time} is calculated by following code:

```
b_time_sum <- edx_up5 %>%
  mutate(error = rating - mu_movie_user) %>%
  group_by(aging_time) %>%
  summarize(b_time = mean(error))
```

We can compare out current prediction with the time effect variable b_{time} by following graph:

```
p1 <-  edx_up5 %>% ggplot(aes(mu_movie_user))+
    geom_histogram(fill = "steel blue", color = "white") +
    theme_classic() +
    labs(title = "predicted rating, movie - user effect",
         caption = "Figure 20")

p2 <- b_time_sum %>% ggplot(aes(b_time)) +
    geom_histogram(fill = "steel blue", color = "white") +
    theme_classic() +
    labs(title = "time effect parameter",
         caption = "Figure 21")


grid.arrange(p1, p2, nrow = 1)
```

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

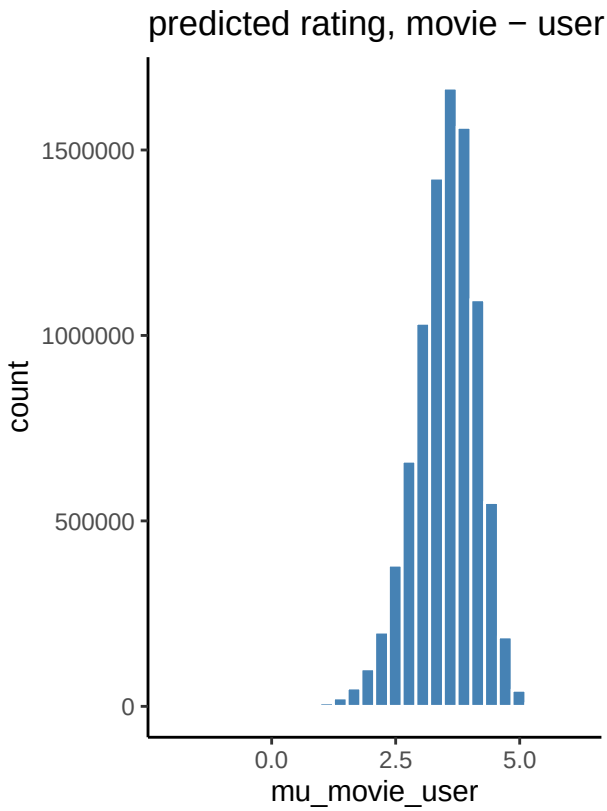


Figure 20

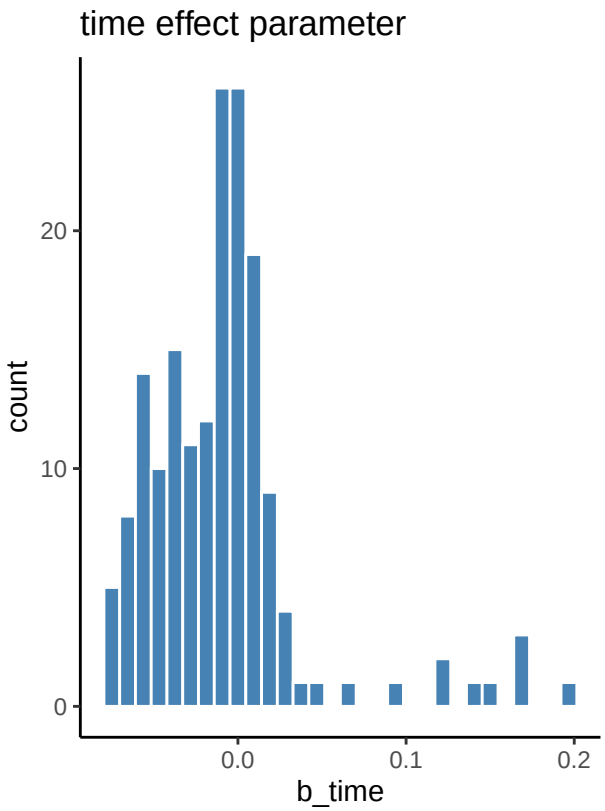


Figure 21

The predicted rating is calculated as following:

```
#calculate predicted rating
edx_up6 <- edx_up5 %>% left_join(b_time_sum, by = "aging_time")
edx_up6$b_time[is.na(edx_up5$b_time)] <- 0

edx_up7 <- edx_up6 %>%
  mutate(mu_movie_user_time = mu_movie_user + b_time)
```

```
model_3_movie_user_time <- RMSE(edx_up7$mu_movie_user_time, edx_up7$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie - User - Time Effect Model",
                                     RMSE = model_3_movie_user_time))


knitr::kable(rmse_results,caption = "RSME by Method movie-user-Time effict")
```

Table 11: RSME by Method movie-user-Time effict

method	RMSE
Only baseline is movie average	0.9423475
Add Specific-effect of user	0.8567039
Movie - User - Time Effect Model	0.8562755

Now the RMSE is 0.8563, with a little bit improvement. In next section we will add the specific-effect by

genres to our model and see how much we can improve.

2.2.5 Add specific-effect by genres From previous analysis, we know a user given higher rating for some genres (suppose those are his/her favor) and lower rating for some genres (not his/her favor) compared to other genres.

By the same approach, we add the genres-effect and the true rating can be decomposed to:

Where the genres - effect b_{genres} can be calculated by:

$$b_{genres} = y_{i,j,time,genres} - \hat{y}_{i,j,time}$$

$$= y_{u,i} - (\mu + b_i + b_u + b_{time})(9)$$

The b_{genres} is calculated by following method:

```
#calculate the genres effect bias
b_genres_sum <- edx_up7 %>% mutate(error = rating - mu_movie_user_time) %>%
  group_by(genres) %>%
  summarize(b_genres = mean(error))
```

The predicted rating is calculated as following:

```
edx_up8 <- edx_up7 %>% left_join(b_genres_sum, by = "genres")

edx_up9 <- edx_up8 %>%
  mutate(mu_movie_user_time_genres = mu_movie_user_time + b_genres)
```

The RMSE is calculated as following:

```
model_4_movie_user_time_genres <- RMSE(edx_up9$mu_movie_user_time_genres, edx_up9$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie - User - Time - Genres Effect Model",
                                     RMSE = model_4_movie_user_time_genres))


knitr::kable(rmse_results,caption = "RSME by Method movie-user-time-genner effict Model")
```

Table 12: RSME by Method movie-user-time-genner effict Model

method	RMSE
Only baseline is movie average	0.9423475
Add Specific-effect of user	0.8567039
Movie - User - Time Effect Model	0.8562755
Movie - User - Time - Genres Effect Model	0.8559753

Now, our model reduce the RMSE to 0.856.

3. Results

3.1 Model summary

Our final predicting model used 4 predictors, the baseline as average rating of each movie, the specific effect by each user, the time effect and genres effect.

The predicted rating calculation formular is as following:

With:

$\hat{y}_{i,j,time,genres}$: the predicted rating value.

μ_i : the baseline as average rating of movie i.

b_j : the specific effect of user j.

b_{time} : the effect of aging time.

b_{genres} : the effect of genres.

3.2 Modeling performance evaluation

Adding the new features in our validation set.

```
#calculate the rating time
validation <- validation %>%
  mutate(rating_time = as.Date(as.POSIXct(timestamp, origin = "1970-01-01"))) %>%
  mutate(rating_year = year(rating_time))

#calculate the aging time
validation <- validation %>% left_join(movie_sum, by = "movieId")
validation <- validation %>%
  mutate(aging_time = round((timestamp - first_rating_time)/60/60/24/30,0))
```

Adding user-effect b_j, time-effect b_{time}, genres-effect b_{genres} to validation dataset.

```
validation <- validation %>% left_join(b_j_sum, by = "userId") %>%
  left_join(b_time_sum, by = "aging_time") %>%
  left_join(b_genres_sum, by = "genres")
```

Check if any NA value and replace by the mean value.

```
knitr::kable(head(validation[,14:16]),caption = "validation")
```

Table 13: validation

b_j	b_time	b_genres
1.6792347	0.0226741	-0.0070401
1.6792347	0.0226741	-0.0228665
1.6792347	0.0288979	-0.0231494
-0.2364086	0.0045294	0.0014985
-0.2364086	0.0083779	0.0264940
-0.2364086	0.0288979	-0.0144323

NA value check in validation set

We will replace the NA value in b_{time} by the average.

```
validation$b_time[is.na(validation$b_time)] <- mean(validation$b_time, na.rm = T)
```

The predicted rating is as following:

```
validation <- validation %>%
  mutate(predicted_rating = mu_movie + b_j + b_time + b_genres)
```

The RMSE is as following:

```
RMSE(validation$rating, validation$predicted_rating)
```

```
[1] 0.8645411
```

The final RMSE is 0.8645 less than our target 0.8650 as well as the the required RMSE to get the maximum point for the EDX Capstone Movielens projects (required RMSE $\leq$ 0.8649).

4. Conclusion

In this report we described a way to build up the recommendation algorithm to predict movie ratings using Movielens data set step by step. Four predictors were used in our algorithm included: (i) the baseline as average rating of each movie, (ii) the specific-effect by user, (iii) the specific-effect by aging time, (iv) the specific-effect by genres. Two main predictors have highest impact to the results are the average rating of each movie and the specific-effect by user. The final RMSE from our algorithm is 0.8645. This result achieved our project goals and the initial criteria of the course HarvardX - PH125.9X Data Science: Capstone project - All learners (RMSE $\leq$ 0.8649).

Because our algorithm is based on the average rating of each movie, therefore if a movie have very less number of rating, this algorithm is limited to provide an accuracy results. A better result is received only when the number of rating of a movie and/or the number of rating given by a user is increased high enough.