Pongsasit Thongpramoon

Index

1. Introduction

1.1Introduction

For this project, I created a movie recommendation system using the MovieLens dataset. The version of movielens included in the dslabs package is just a small subset of a much larger dataset with millions of ratings.

I used the 10M version of the MovieLens dataset, collected by GroupLens Research (https://grouplens.org/datasets/movielens/10m/).

The target of this project is to train a machine learning algorithm using the inputs of a provided subset to predict movie ratings in the provided validation set.

Using the MovieLens data set and penalized least squares, calculates the RMSE based on user ratings, movieId and the age of the movie.

Pongsasit Thongpramoon

1.2 Know the data

1.2.1 Download and Build the dataset.

```
#PONGSASIT THONGPRAMOON
##################################
# Create edx set and 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")

# 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 <- read.table(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")
```

The dataset splitted into 2 subsets, “edX” is train set and “validation” is test set.

```
# Validation set will be 10% of MovieLens data
set.seed(1)
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)
edx <- rbind(edx, removed)

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

I trained my model only on “edX”, “validation” set will be used to test the algorithm.

1.2.2 Know Dataset

At first I use head function to see overall data in this dataset. Then find the dimension of edX, then find the amount of 0 and 3 as given as rating in this dataset as below.

```
head(edx)
```

	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

```
#know Dataset
dim(edx)
```

```
9000055  6
```

```
#So we know that edx has 9000055 rows and 6 column
```

```
#Find zeros were given as ratings in the edx dataset
edx %>% filter(rating == 0) %>% tally()
#Find threes were given as rating in the edx dataset
edx %>% filter(rating == 3) %>% tally()
```

n
0

n
2121240

Then I find number of movie rating in each genres in edx dataset, rank the movies by movie ratings, find top five of the rating and show that half stars rating is less than whole star rating. As below.

```
#Show the number of movie rating in each genres in edx dataset
edx %>% separate_rows(genres, sep = "¥¥|") %>% group_by(genres) %>% summarize(count = n()) %>% arrange(desc(count))
```

genres	count
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183

```
#Rank the movies by movie ratings
edx %>% group_by(movieId, title) %>% summarize(count =n()) %>% arrange(desc(count))
```

movieId	title	count
296	Pulp Fiction (1994)	31362
356	Forrest Gump (1994)	31079
593	Silence of the Lambs, The (1991)	30382
480	Jurassic Park (1993)	29360
318	Shawshank Redemption, The (1994)	28015
110	Braveheart (1995)	26212
457	Fugitive, The (1993)	25998

```
#top five rating
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5) %>% arrange(desc(count))
```

```
Selecting by count
```

rating	count
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422

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

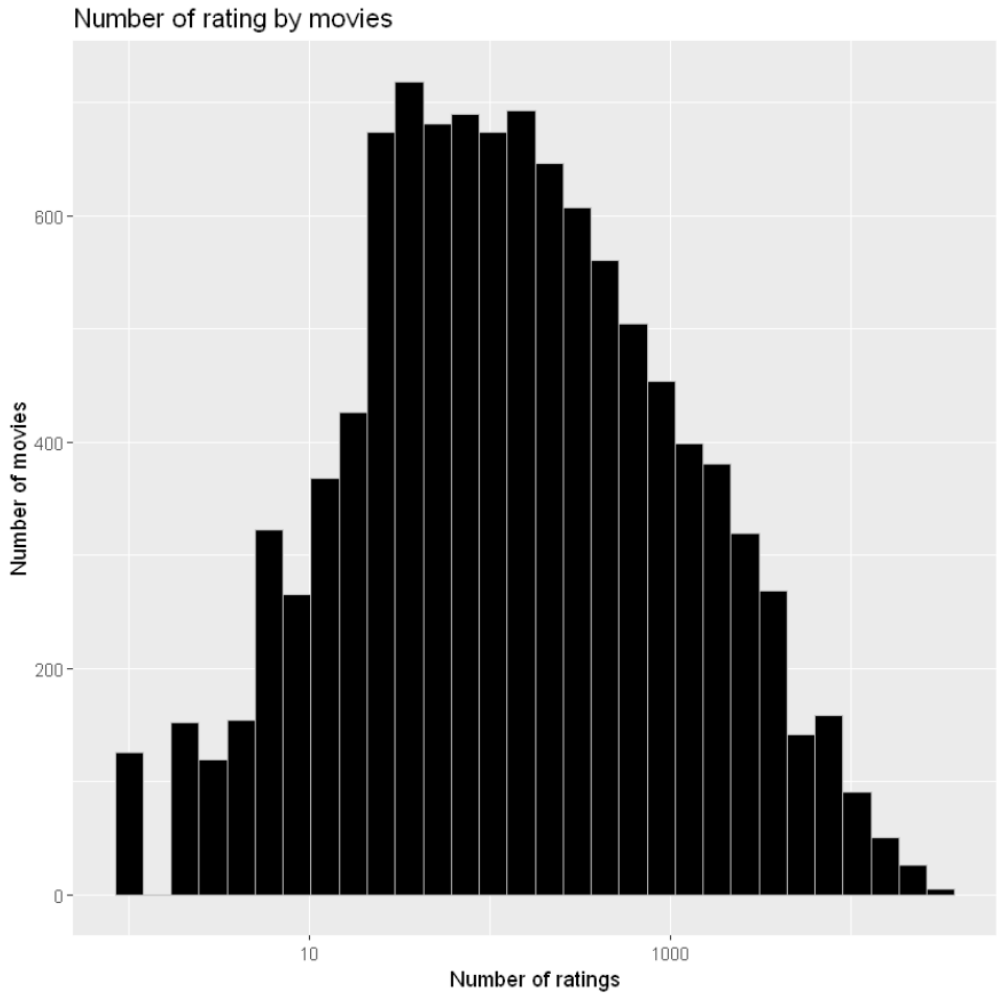
rating	count
0.5	85374
1.0	345679
1.5	106426
2.0	711422
2.5	333010
3.0	2121240
3.5	791624
4.0	2588430
4.5	526736

So now we know about the basic info of this edx dataset. Then find more informative data in the next section.

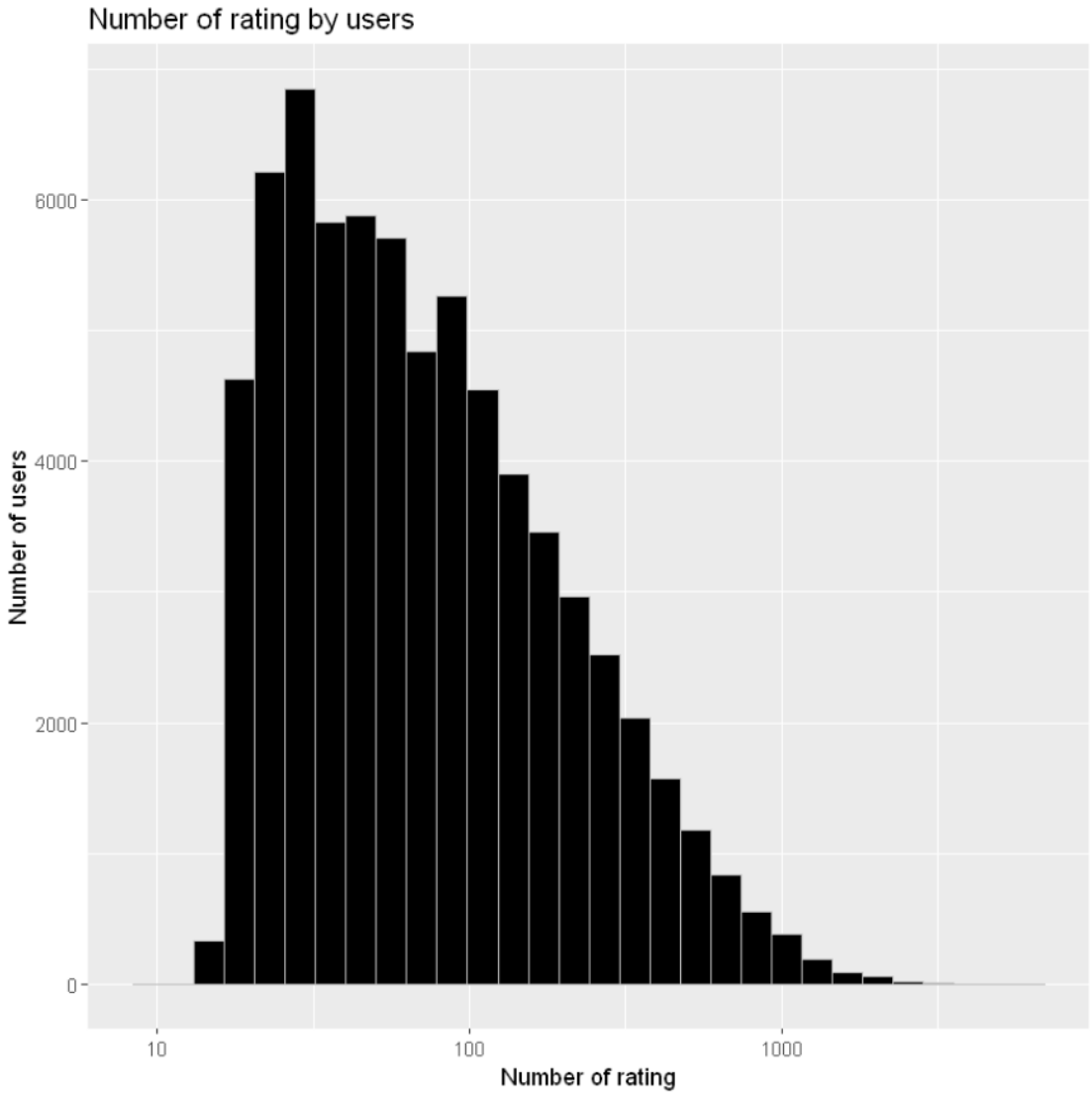
2. Analysis

2.1 Data analysis

```
edx %>% count(movieId) %>%
  ggplot(aes(n)) + geom_histogram(fill = "black", color = "grey", bins = 30) +
  scale_x_log10() + xlab("Number of ratings") + ylab("Number of movies") +
  ggtitle("Number of rating by movies")
```

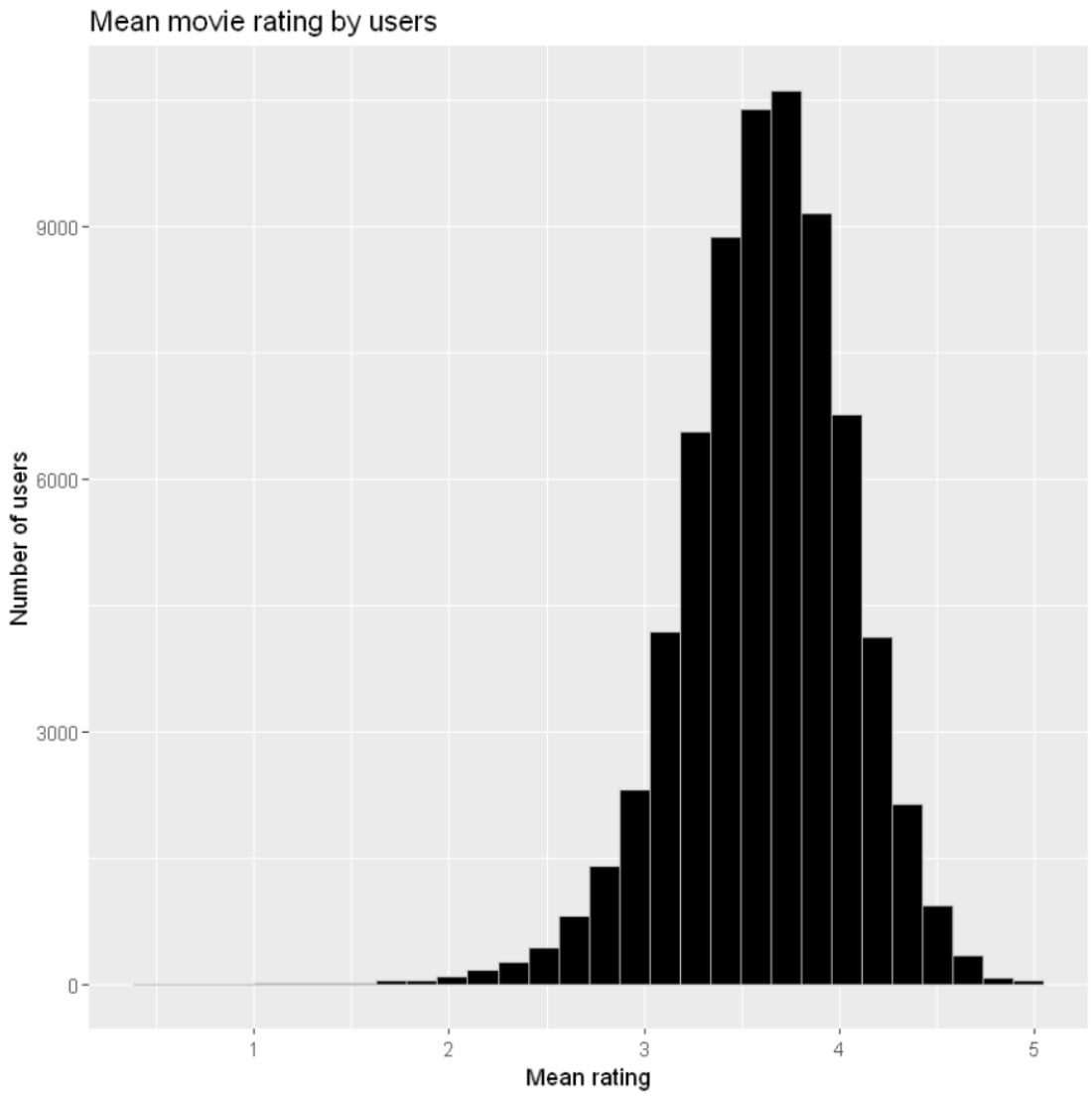


```
edx %>% count(userId) %>%
  ggplot(aes(n)) + geom_histogram(fill = "black", color = "grey", bins = 30) +
  scale_x_log10() + xlab("Number of rating") + ylab("Number of users") +
  ggtitle("Number of rating by users")
```



This graph shown us the distribution of number of rating by userId. And the mean of the rating can visualize as the graph below.

```
edx %>% group_by(userId) %>% summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) + geom_histogram(fill = "black", color = "grey", bins = 30) +
  xlab("Mean rating") + ylab("Number of users") +
  ggtitle("Mean movie rating by users")
```



2.2 Modelling

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{u,i}\left(\hat{y}_{u,i} - y_{u,i}\right)^2}$$

Root mean Square Error, or RMSE is used to measure the differences between predicted values($\hat{y}$) as predicted rating of movie(i) by user(u) and observed values(y). If this number is larger than 1, it means our typical error is larger than 1 rating star. Which is not good. Which can write in code as below.

```
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
  }
```

2.2.1 **Average movie rating model**: the simplest model that assumes the same rating for all movies and users with all the differences explained by random variation would look like this:

$$Y_{u,i} = \mu + \varepsilon_{u,i}$$

With $\varepsilon_{i,u}$ independent errors sampled from the same distribution centered at 0 and μ the "true" rating for all movies. We know that the estimate that minimizes the RMSE is the least squares estimate of μ and, in this case, is the average of all ratings. Below is the code of this model.

```
#mean of all rating
mu <- mean(edx$rating)
mu
#If we predict all unknown ratings with mu we obtain the following RMSE:
naive_rmse <- RMSE(validation$rating,mu)
naive_rmse
#make a result table for every method:
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse)
rmse_results
```

3.51246520160155

1.06120181029262

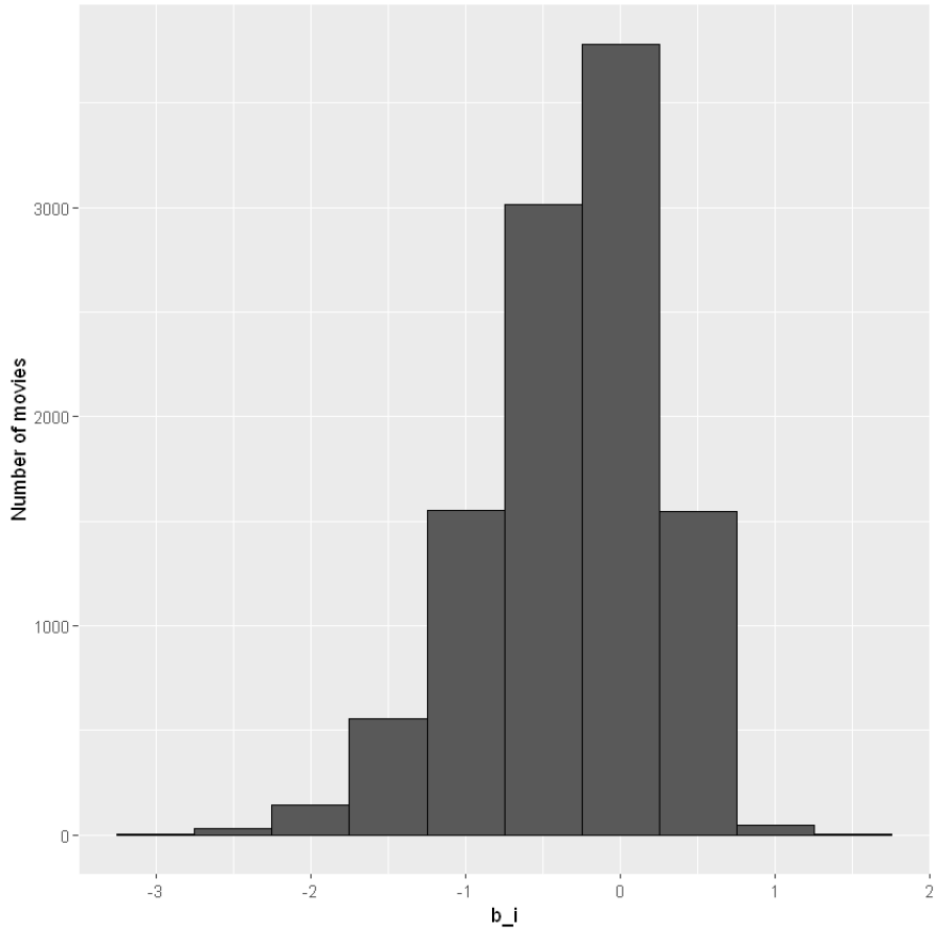
method	RMSE
Just the average	1.061202

As above the result of this model is the average of rating equal 3.512 … and RSME is about 1.061 that is very high. Which is not good.

2.2.2 **Movie effects model:** augment the previous model by adding them b_i (bias)to represent average ranking for movie (i)

$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$$

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"), ylab = "Number of movies")
```



By the graph above, can see that these estimates vary substantially. So check the prediction result as the code below.

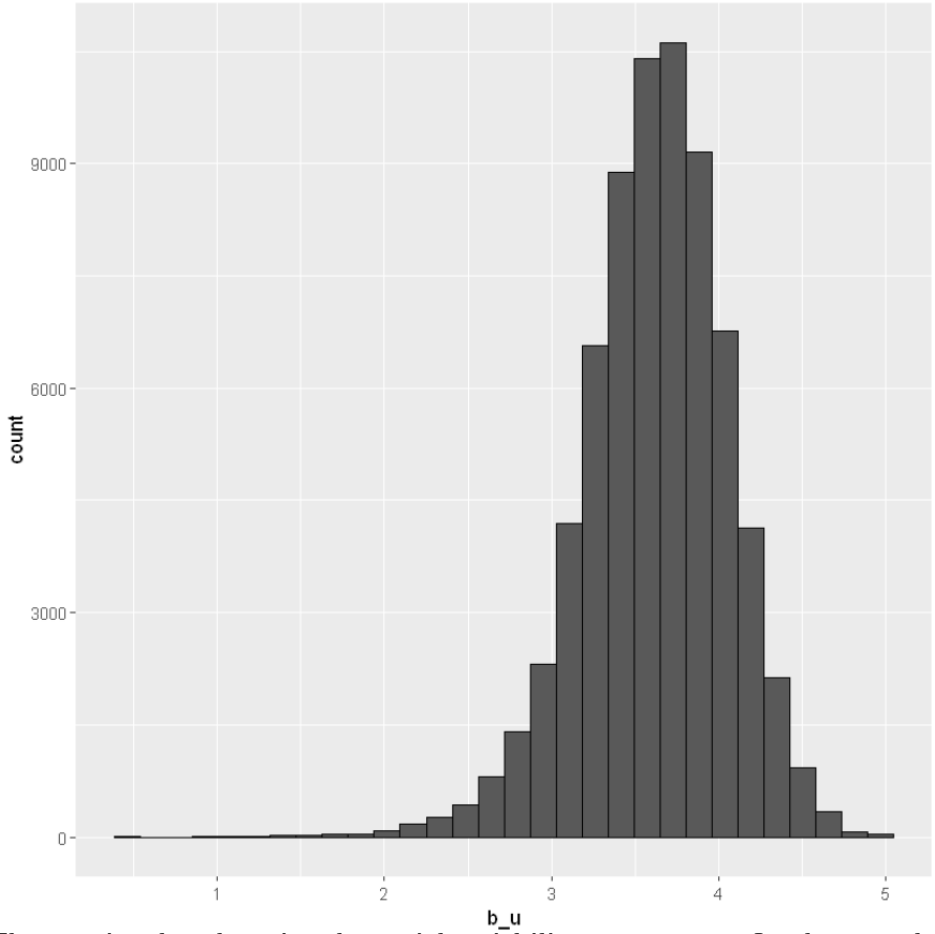
```
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse))
rmse_results
```

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

2.2.3 **Movie and user effects model:** compute the average rating for users.

```
#user effects model
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



Then notice that there is substantial variability across users. Imply to model could be:

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$$

When b_u is a user-specific effect and b_i is movie-specific effect. We compute an approximation by compute $\hat{\mu}$ and $\hat{b}_i$ and estimating $\hat{b}_u$ as the average of $y_{u,i} - \hat{\mu} - \hat{b}_i$. To create this model, I used the code below.

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

#construct predictors and see how much the RMSE

predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)


model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and User Effects Model",
                                     RMSE = model_2_rmse))
rmse_results
```

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488

The RSME is less than the previous model.

3. Result

The RMSE values of the previous model shown below:

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488

The lowest RMSE values is 0.8653488, when used **Movie and user effects model.**

4. Conclusion

Finally, I can made the machine learning model that can achieve to goal (RMSE <= 0.87750). This is very good assignment that make me use many skills that I learned before. For the next project I will try my best like this (or better than this) project. And this project inspires me the “I can do” attitude.

5. Environment & Reference

5.1 Environment

```
print("Operating System:")
version
```

```
[1] "Operating System:"

               _
platform       x86_64-w64-mingw32
arch           x86_64
os             mingw32
system         x86_64, mingw32
status
major          3
minor          5.1
year           2018
month          07
day            02
svn rev        74947
language       R
version.string R version 3.5.1 (2018-07-02)
nickname       Feather Spray
```

5.2 Reference

https://rafalab.github.io/dsbook/large-datasets.html#netflix-loss-function