Capstone Edx - Movielens Project

Francielle Mina

17/05/2021

Contents

1. Introduction:

The current project is a final assessment to complete a Data Science certificate professional. There is a manny different language commonly used for data analysis and data science. According Zumel N and Mount Jhon (2019), data science is a definition as managing the process that can transform hypothesis and data into actionable predictions. With the advance on the internet and production of data, today is a need to filter, prioritize, and deliver efficient information to the customer or user. Nowadays, big company such as Netflix, Amazon, Spotify utilize the Recommendation system or Recommender system. This system is a class of algorithms that can suggest "relevant" items to users by searching through the large volume of dynamically generated information to provide users with personalized content and services. One of the parameters recommender system is RMSE, the square root of the variance of the residuals. The RMSE computes the mean value of all the differences squared between the true and the predicted ratings and then proceeds to calculate the square root out of the result. Consequently, significant errors may dramatically affect the RMSE rating, rendering the RMSE metric most valuable when significant errors are unwanted. RMSE is a measure to show how accurately the model predicts the response, and it is the essential criteria for fit if the primary purpose of the model is prediction. In this project, we will be using the data set Movielens provided by the Edx course. Also, the dataset Movielens has 25 million ratings and one million tag applications applied to 62,000 movies by 162,000 users.

2. Objective: Predict movie to users by rating from the dataset with low accurately RMSE.

2.1 Dataset: The dataset can be found:https://grouplens.org/datasets/movielens/10m/ http://files.grouplens.org/datasets/movielens/ml-10m.zip

3. Method and analysis:

For this project, we are using several packages from CRAN to assist our analysis. All the packages will be load along with the development of the project. First of all, we downloaded the dataset from the website, split the data in validation and train, and called edx. After that, the data edx also split into train and test. When the RMSE reaches the goal, the validation set will use for the final validation (unknow) model and predict results. The following steps for the project will be building, interpreting RMSE results and data exploration.

3.1 Explore dataset Edx.

First of all, explore and analyse the data set edx. It's essential to understand how the data are structured, characteristics for better knowledge.

```r
# MovieLens 10M dataset:
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
if(!require(RColorBrewer)) install.packages("RColorBrewer", repos = "http://cran.us.r-project.org")
```

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

```r
library(tidyverse)
library(caret)
library(data.table)
library(RColorBrewer)
#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(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) # if using R 3.5 or earlier, use `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)
```

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

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

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

```r
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000061 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983392 838983421 838983392 838983392 838984474 838983653 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (1994)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Action|Drama|Sci-
Fi|Thriller" ...
```

```
##  - attr(*, ".internal.selfref")=<externalptr>
```

The function strc show us information on the object's structure and information about the class, length and content for each class.

```r
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     231      5 838983392          Dumb & Dumber (1994)
## 4:      1     292      5 838983421               Outbreak (1995)
## 5:      1     316      5 838983392               Stargate (1994)
## 6:      1     329      5 838983392 Star Trek: Generations (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:                        Comedy
## 4:  Action|Drama|Sci-Fi|Thriller
## 5:       Action|Adventure|Sci-Fi
## 6: Action|Adventure|Drama|Sci-Fi
```

The head function shows the first 6 rows. We can observe the data has 6 columns, userId, movieId, rating, timestamp, title, genres.

```r
dim(edx)
```

```
## [1] 9000061       6
```

The dim function returns the vector with the number of rows in the first element and the numbers of columns in the second element.

```r
summary(edx)
```

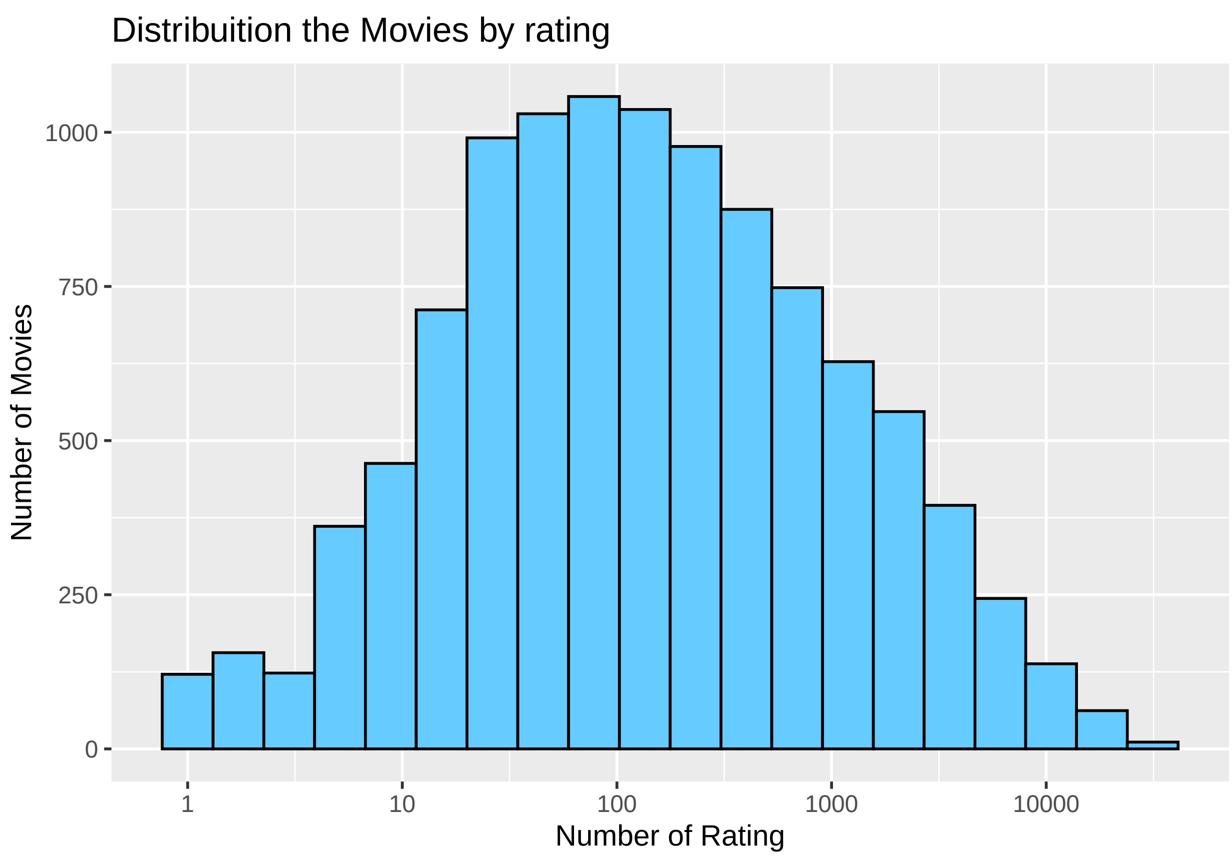
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000061     Length:9000061
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The summary function is essential. Exhibits the statistics for each column. It can observe the minimum and maximum value in each column and the mean, median, and 3rd quartile. The column title and genres show us the length class and mode because these two columns are categorical data.

3.1.2 Data analysis Edx data

3.1.2.1 Distribuition the Movies by ratings

```
library(RColorBrewer)
edx %>% group_by(movieId) %>% summarise(n = n()) %>% ggplot(aes(n)) + geom_histogram(color = "black", bi
  scale_x_log10() + xlab("Number of Rating") + ylab("Number of Movies")
```




```
n_distinct(edx$movieId)
```

```
## [1] 10677
```

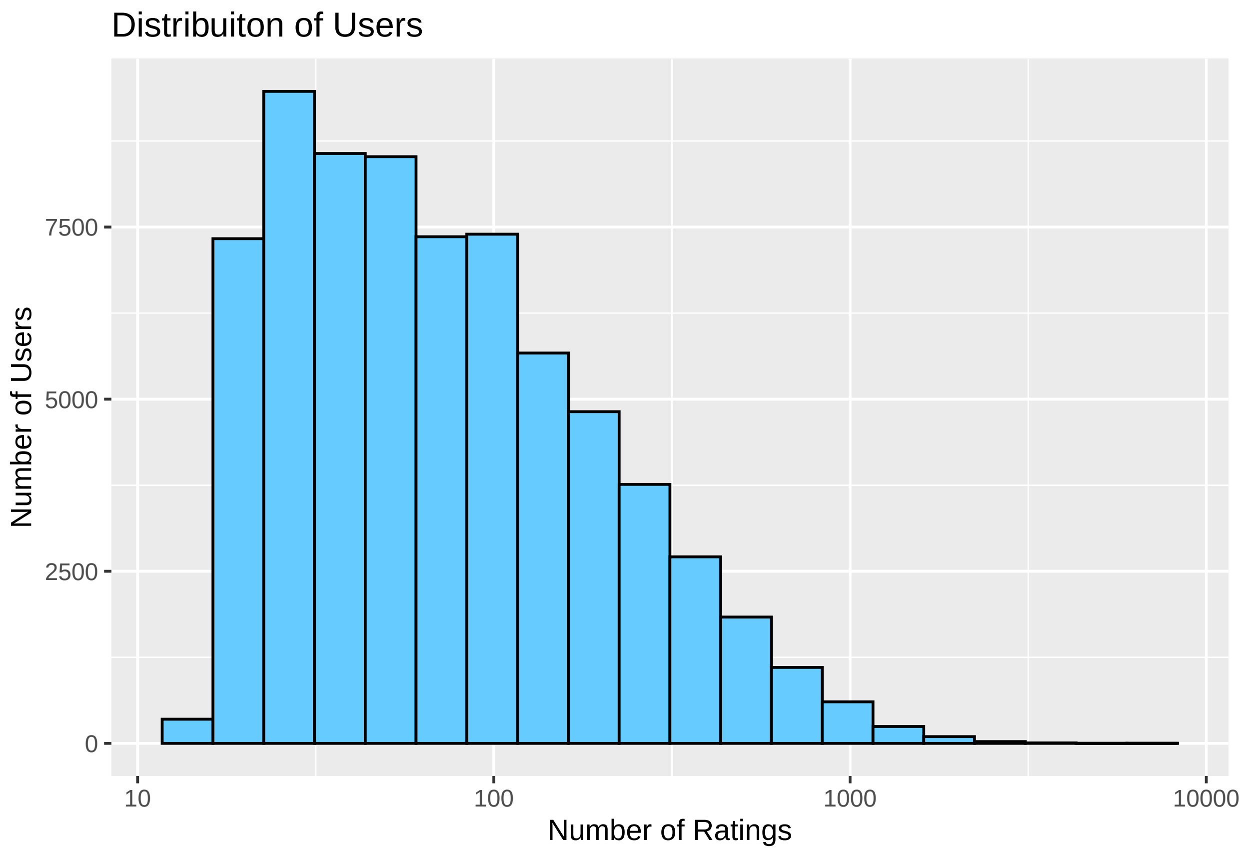
```
edx %>% group_by(movieId) %>% summarise(n = n()) %>% head()
```

```
## # A tibble: 6 x 2
##   movieId     n
##     <dbl> <int>
## 1       1 23826
## 2       2 10717
## 3       3  7053
## 4       4  1579
## 5       5  6415
## 6       6 12385
```

The histogram shows how are distributed movies in the edx data set. Using the function n_distinct, it can observe there are 10677 movies in the edx data set. As well, it followed at distribution on the histogram.

3.1.2.2 Analysis of Distribuition by Users

```
edx %>% group_by(userId) %>% summarise(n = n()) %>% ggplot(aes(n)) + geom_histogram(color = "black", bi
```




```
n_distinct(edx$userId)
```

```
## [1] 69878
```

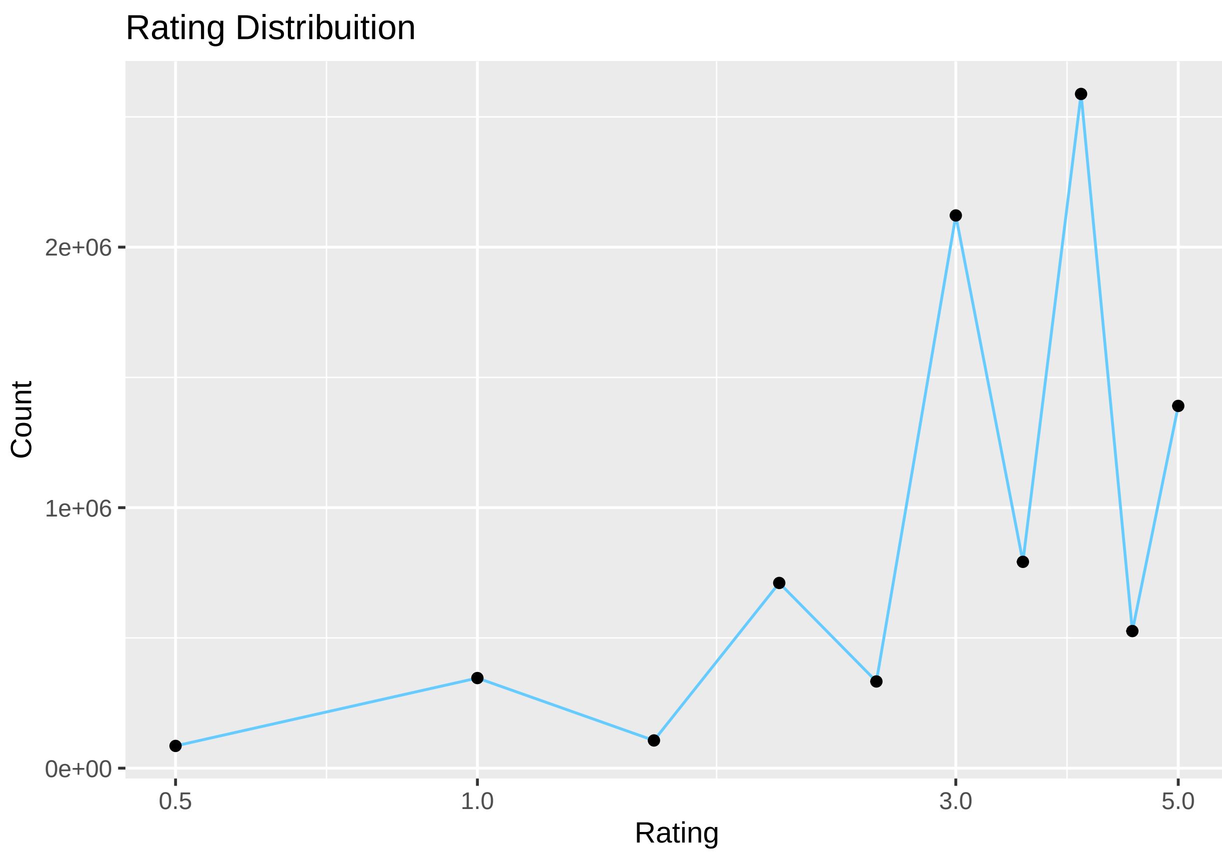
```
edx %>% group_by(userId) %>% summarise(n = n()) %>% head()
```

```
## # A tibble: 6 x 2
##   userId     n
##    <int> <int>
## 1      1    21
## 2      2    18
## 3      3    30
## 4      4    35
## 5      5    80
## 6      6    39
```

The histogram graph show us the distribuition of user on edx dataset, 69878 users

3.1.2.3 Analysis of Ratings

```r
edx %>% group_by(rating) %>% summarise(n = n()) %>% ggplot(aes(rating, n)) + geom_line(color = "#66CCFF
  xlab("Rating") + ylab("Count")
```




```r
n_distinct(edx$rating)
```

```
## [1] 10
```

```r
edx %>% group_by(rating) %>% summarise(n = n()) %>% head()
```

```
## # A tibble: 6 x 2
##   rating       n
##    <dbl>   <int>
## 1    0.5   85420
## 2    1    345935
## 3    1.5  106379
## 4    2    710998
## 5    2.5  332783
## 6    3   2121638
```

The graph and the n_distinct shows the 10 possibilities that the user can rantings from 0.5 to 5.0.

3.2.1 Explore datset on Validation

Until now, we can analyse the data set from edx before the split into train and test. Also, we can run data analysis on the validation dataset before the final validation and predict.

```r
str(validation)
```

```
## Classes 'data.table' and 'data.frame':   999993 obs. of  6 variables:
##  $ userId   : int  1 2 2 3 3 3 4 4 4 5 ...
##  $ movieId  : num  588 1210 1544 151 1288 ...
##  $ rating   : num  5 4 3 4.5 3 3 3 3 5 3 ...
##  $ timestamp: int  838983339 868245644 868245920 1133571026 1133571035 1164885617 844416656 84441707
##  $ title    : chr  "Aladdin (1992)" "Star Wars: Episode VI - Return of the Jedi (1983)" "Lost World:
##  $ genres   : chr  "Adventure|Animation|Children|Comedy|Musical" "Action|Adventure|Sci-
Fi" "Action|Adventure|Horror|Sci-Fi|Thriller" "Action|Drama|Romance|War" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The dataset on validation has 999999 obs and 6 variables. This part of data has 90% from the original data.

```r
head(validation)
```

```
##    userId movieId rating  timestamp
## 1:      1     588    5.0  838983339
## 2:      2    1210    4.0  868245644
## 3:      2    1544    3.0  868245920
## 4:      3     151    4.5 1133571026
## 5:      3    1288    3.0 1133571035
## 6:      3    5299    3.0 1164885617
##                                                title
## 1:                                    Aladdin (1992)
## 2:       Star Wars: Episode VI - Return of the Jedi (1983)
## 3: Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
## 4:                                    Rob Roy (1995)
## 5:                          This Is Spinal Tap (1984)
## 6:                    My Big Fat Greek Wedding (2002)
##                                          genres
## 1: Adventure|Animation|Children|Comedy|Musical
## 2:                     Action|Adventure|Sci-Fi
## 3:     Action|Adventure|Horror|Sci-Fi|Thriller
## 4:                    Action|Drama|Romance|War
## 5:                              Comedy|Musical
## 6:                              Comedy|Romance
```

The head function show the first 6 rows. We can observed the data has 6 columns, userId, movieId, rating, timestamp, title, genres.

```r
dim(validation)
```

```
## [1] 999993      6
```

The dim function returns the vector with the number of rows in the first element, and the numbers of columns the second element. The first argument has 999999 entries, and the second argument has 6 entries.

```r
summary(validation)
```

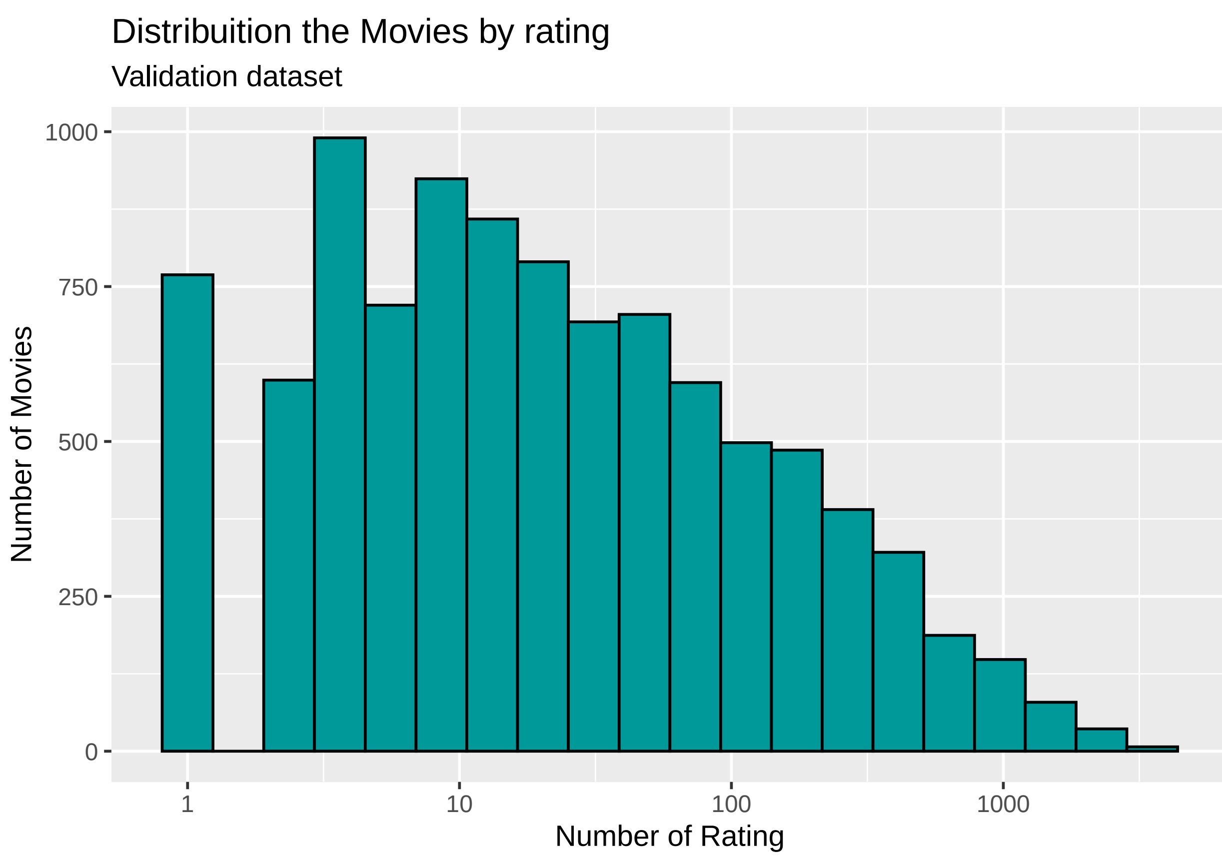
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18127   1st Qu.:  653   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35719   Median : 1835   Median :4.000   Median :1.036e+09
##  Mean   :35878   Mean   : 4121   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53649   3rd Qu.: 3633   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:999993      Length:999993
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The summary function show the statistics on dataset. Showing the mean, median and quartiles.

3.2.2 Data analysis on Validation dataset

3.2.3 Analysis of Distribuition by Movies

```
validation %>% group_by(movieId) %>% summarise(n = n()) %>% ggplot(aes(n)) + geom_histogram(color = "bla
  scale_x_log10() + xlab("Number of Rating") + ylab("Number of Movies")
```



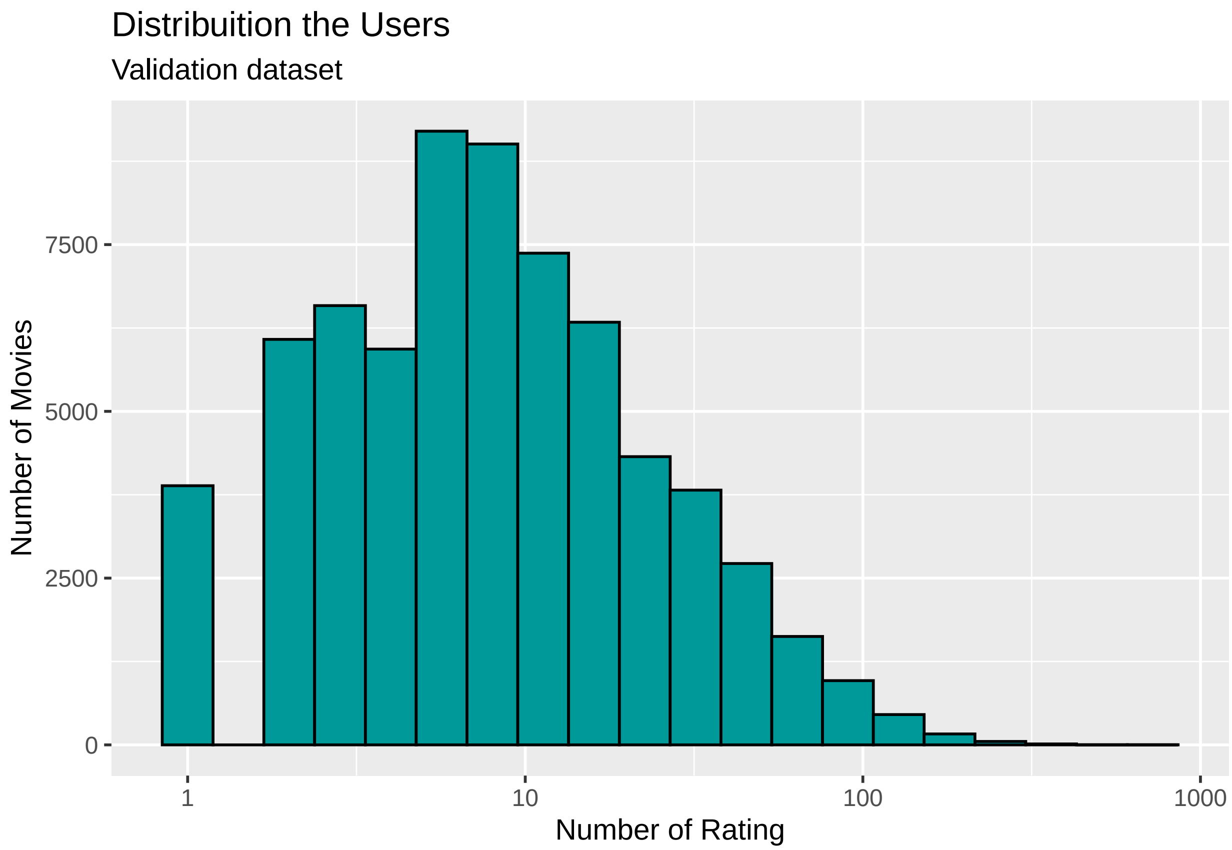

```
n_distinct(validation$movieId)
```

```
## [1] 9796
```

It can observe we have 9809 movies on dataset validation.

3.2.4 Analysis of Distribuition by Users

```
validation %>% group_by(userId) %>% summarise(n = n()) %>% ggplot(aes(n)) + geom_histogram(color = "bla
  scale_x_log10() + xlab("Number of Rating") + ylab("Number of Movies")
```



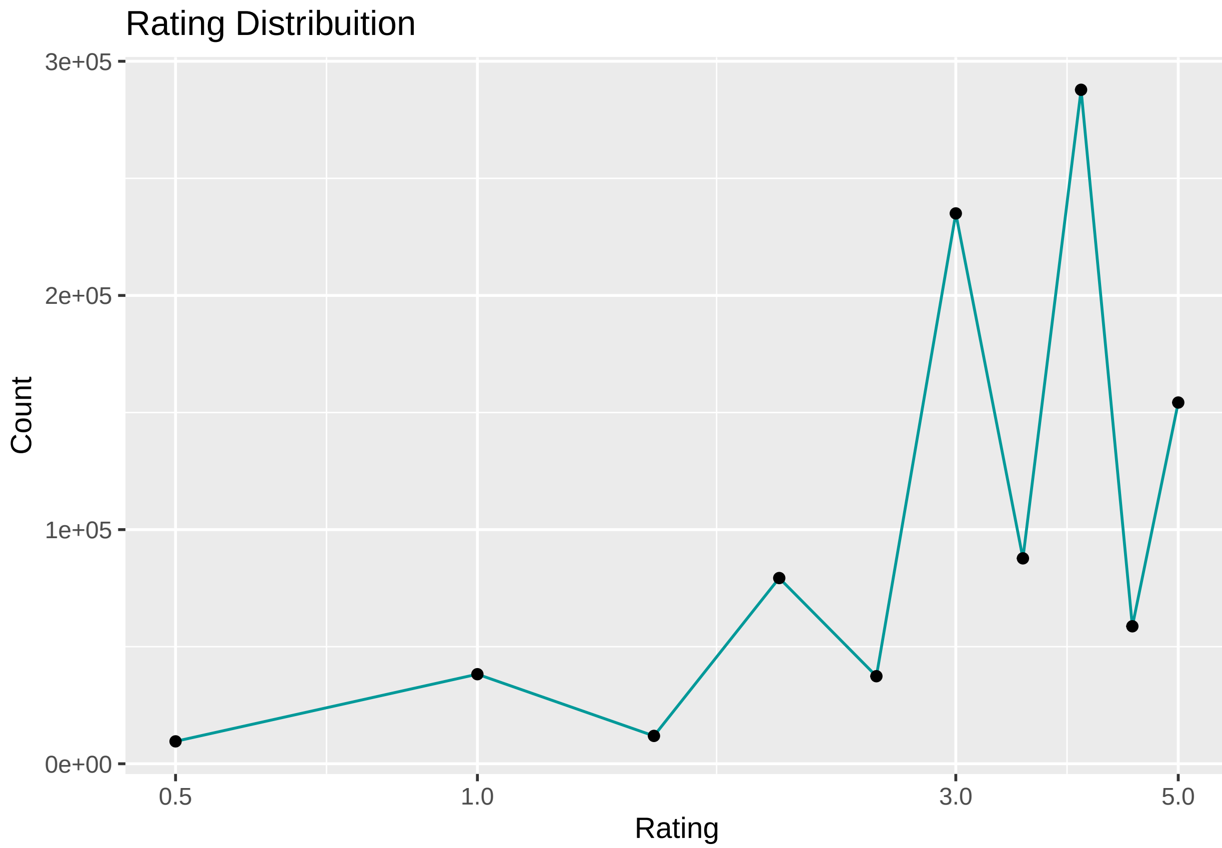

```
n_distinct(validation$userId)
```

```
## [1] 68531
```

It can observe the distriution the user on validation set is 68534.

3.2.5 Analysis of Ratings

```
validation %>% group_by(rating) %>% summarise(n = n()) %>% ggplot(aes(rating, n)) + geom_line(color = "
  xlab("Rating") + ylab("Count")
```




```r
n_distinct(validation$rating)
```

```
## [1] 10
```

```r
validation %>% group_by(rating) %>% summarise(n = n()) %>% head()
```

```
## # A tibble: 6 x 2
##   rating      n
##    <dbl>  <int>
## 1    0.5   9568
## 2    1    38245
## 3    1.5  11899
## 4    2    79308
## 5    2.5  37395
## 6    3   235038
```

It observed in this graph the most significant rating it was to from 3 to 5.

4.0 Results

For this step, we are split the edx data set into train and test. The train data set has 10%, and the test has 90% of the original data. Also, it is an important method that evaluates the accuracy of the dataset. As explained before, train the part of data to allow the algorithm to predict the outcome.

4.1 Recommendations system

Exploring all the available digital data has created a challenge for big companies such as Google and Netflix to personalised and prioritise information to the user. For to solve this problem, the Recommendation system solves this. The Recommender system can predict whether a particular user would prefer an item or not based on the user's profile. Netflix uses a recommendation system to predict how many stars a user will give a specific movie. In 2006 Netflix launched a challenge to the data science community, offering one million dollars to improve 10 % of the recommendation algorithm. This is a more complicated code. To see this, we are predicting the rating for movie i by user u, in principle, all other ratings related to movie i and by user u.

Following this, the Netflix challenge is based on RMSE, residual mean squared error. Where is defined yu,i as the rating for movie i by user u and denote our prediction with ŷu,i. The RMSE is then defined as:

RMSE=√1N u,i(ŷu,i−yu,i)2

Also, we have discussed that RMSE can interpret similar to standard deviation, which means that when RMSE is bigger than 1, is not a good result. Here is a function that computes RMSE for vectors of rating and their corresponding predictors: RMSE <- function(true_ratings, predicted_ratings){ sqrt(mean((true_ratings - predicted_ratings)^2)) }

To found the RMSE, we built the first model, which predicts rating for the movie regardless of users.

```
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_set <- edx[-test_index,]
temp <- edx[test_index,]

# make sure userId and movieId in validation set are also in edx set
test_set <- temp %>% semi_join(train_set, by = "movieId") %>% semi_join(train_set, by = "userId")

# Add rows removed from validation set back into edx set

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

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

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

rm(test_index, temp, removed)
```

This next step is an initial preparation. let's check the initial RMSE.

```
mu_hat <- mean(train_set$rating)
mu_hat
```

```
## [1] 3.512354
```

```
naive_rmse <- RMSE(test_set$rating, mu_hat)
naive_rmse
```

```
## [1] 1.059
```

```r
rmse_results <- tibble(method = "Naive RMSE", RMSE = naive_rmse)
rmse_results
```

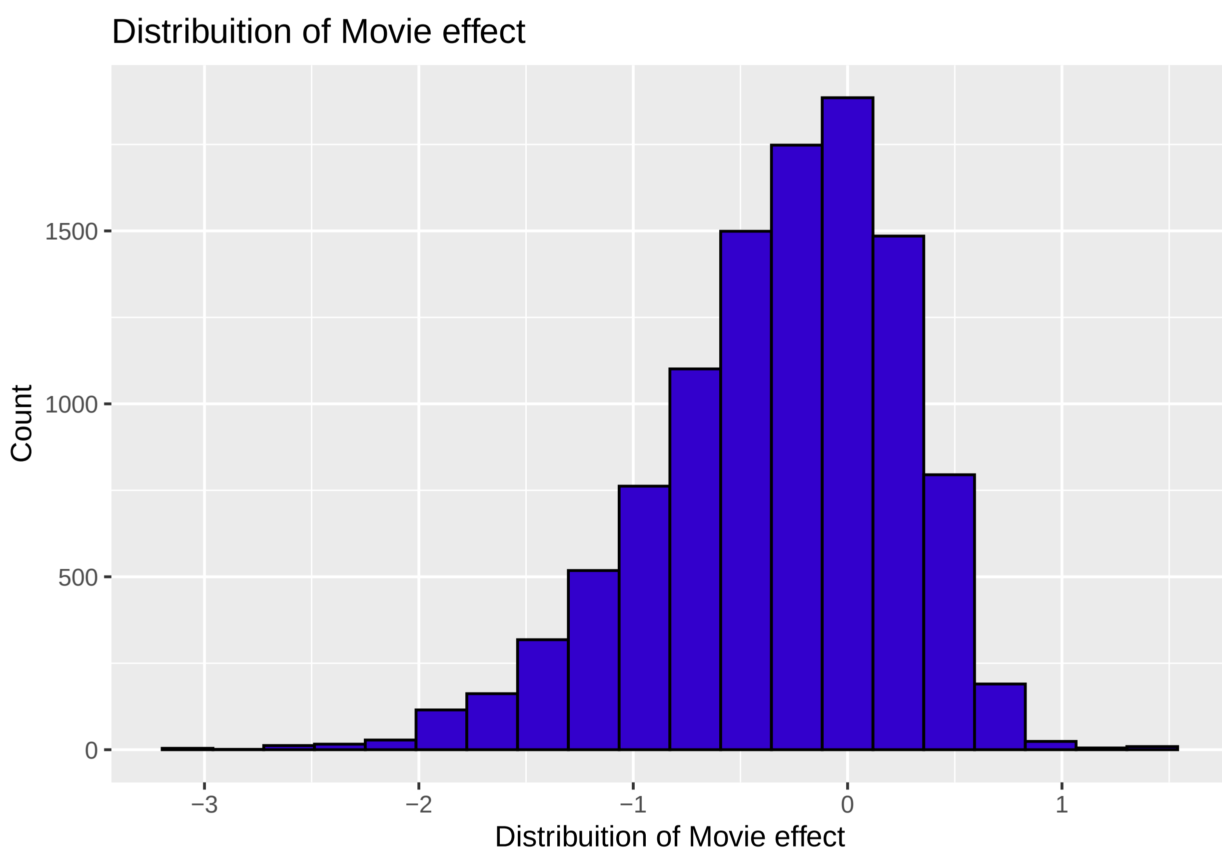
```
## # A tibble: 1 x 2
##   method      RMSE
##   <chr>      <dbl>
## 1 Naive RMSE  1.06
```

We can observe the results of RMSE is 1.060054, is not good enough. Let's include movie on train_set to see how RSME behave.

4.1.2 Effect on Movie (b_i)

```r
mu <- mean(train_set$rating)
b_i <- train_set %>%
group_by(movieId) %>%
summarize(b_i = mean(rating - mu))


b_i %>% ggplot(aes(b_i)) + geom_histogram(bins = 20, fill = "#3300CC", color = "black") + ggtitle("Dist
  ylab("Count")
```



We can observe the histogram show the movie has a skewed on left distribuited.

4.1.3 predict moveis on test set

```r
predicted_b_i <- mu + test_set %>%
  left_join(b_i, by='movieId') %>%
  pull(b_i)
rmse_1 <- RMSE(predicted_b_i, test_set$rating)
rmse_1
```

```
## [1] 0.9426564
```

```r
rmse_results1 <-tibble(method = "Movie effect model on test set", RMSE = rmse_1)
rmse_results1
```

```
## # A tibble: 1 x 2
##   method                          RMSE
##   <chr>                          <dbl>
## 1 Movie effect model on test set 0.943
```

```r
rmse_results1 %>% knitr::kable()
```

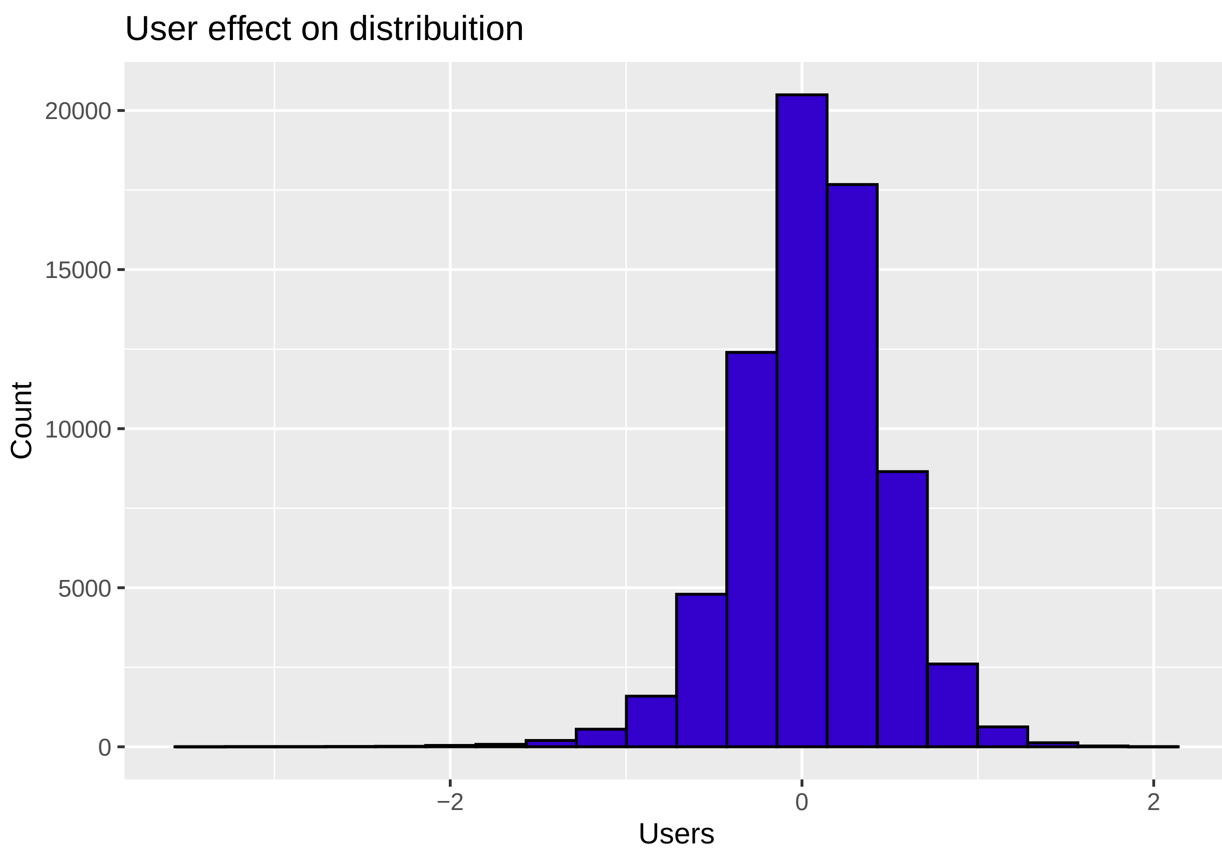
method	RMSE
Movie effect model on test set	0.9426564

As as explain before, the RMSE result is predict on the test set. We observe the predict RMSE on test set improve the result.

4.1.3 User effects on b_u

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

b_u %>% ggplot(aes(b_u)) + geom_histogram(bins = 20, fill = "#3300CC", color = "black") + ggtitle("User
  xlab("Users") + ylab("Count")
```



The histogram show the users is normaly distribuited.

4.1.4 predict values on test set (movie and user)

```
predicted_b_u <- test_set %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

rmse_2 <- RMSE(predicted_b_u, test_set$rating)
rmse_2
```

```
## [1] 0.8646047
```

```
rmse_results2 <- tibble(method="Movie + User Effects Model", RMSE = rmse_2)
rmse_results2 %>% knitr::kable()
```

method	RMSE
Movie + User Effects Model	0.8646047

5.0 Regularisation

It has improved RMSE on movie and user effect, but it still needs to improve. For to do this, the next step is Regularisation. Regularization permits us to penalize large estimates that are formed using small sample sizes. The general idea behind regularization is to constrain the total variability of the effect sizes. Another way is the lambda parameter. When performing regularisation to reduce the variance of error prediction and overfitting, penalties are introduced on the model. Lambda purpose a good fit for training data, avoiding overfitting.

Before run regularisation, let's check the b_i effect on the movie.

5.1 Effect of movie.

```
titles <- train_set %>%
  select(movieId, title) %>%
  distinct()
titles
```

```
##        movieId                                          title
##     1:     122                               Boomerang (1992)
##     2:     185                                Net, The (1995)
##     3:     231                           Dumb & Dumber (1994)
##     4:     292                                Outbreak (1995)
##     5:     316                                Stargate (1994)
##    ---
## 10673:    6838                        Once in the Life (2000)
## 10674:    9006                     Garden of Allah, The (1936)
## 10675:   39429                                 Confess (2005)
## 10676:   56253        Symbiopsychotaxiplasm: Take One (1968)
## 10677:   62063 Dead Man's Letters (Pisma myortvogo cheloveka) (1986)
```

```
b_i %>%
  inner_join(titles, by = "movieId") %>%
  arrange(b_i) %>%
  select(title) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 1
##    title
##    <chr>
##  1 Besotted (2001)
##  2 Hi-Line, The (1999)
##  3 Grief (1993)
##  4 Accused (Anklaget) (2005)
##  5 Hip Hop Witch, Da (2000)
##  6 SuperBabies: Baby Geniuses 2 (2004)
##  7 From Justin to Kelly (2003)
##  8 Pokémon Heroes (2003)
##  9 Stacy's Knights (1982)
## 10 Dog Run (1996)
```

Here effect on b_i on movies. List 10 worst movies.

```
b_i %>%
  inner_join(titles, by = "movieId") %>%
  arrange(-b_i) %>%
  select(title) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 1
##    title
##    <chr>
##  1 Hellhounds on My Trail (1999)
##  2 Satan's Tango (Sátántangó) (1994)
##  3 Shadows of Forgotten Ancestors (1964)
##  4 Fighting Elegy (Kenka erejii) (1966)
##  5 Sun Alley (Sonnenallee) (1999)
##  6 Blue Light, The (Das Blaue Licht) (1932)
##  7 Hospital (1970)
##  8 Constantine's Sword (2007)
##  9 Human Condition II, The (Ningen no joken II) (1959)
## 10 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1~
```

Here Effect on b_i on 10 betters movies.

5.2 Regularisation and Lambda

Below is the regularization function to choose the best value that minimizes the RMSE.

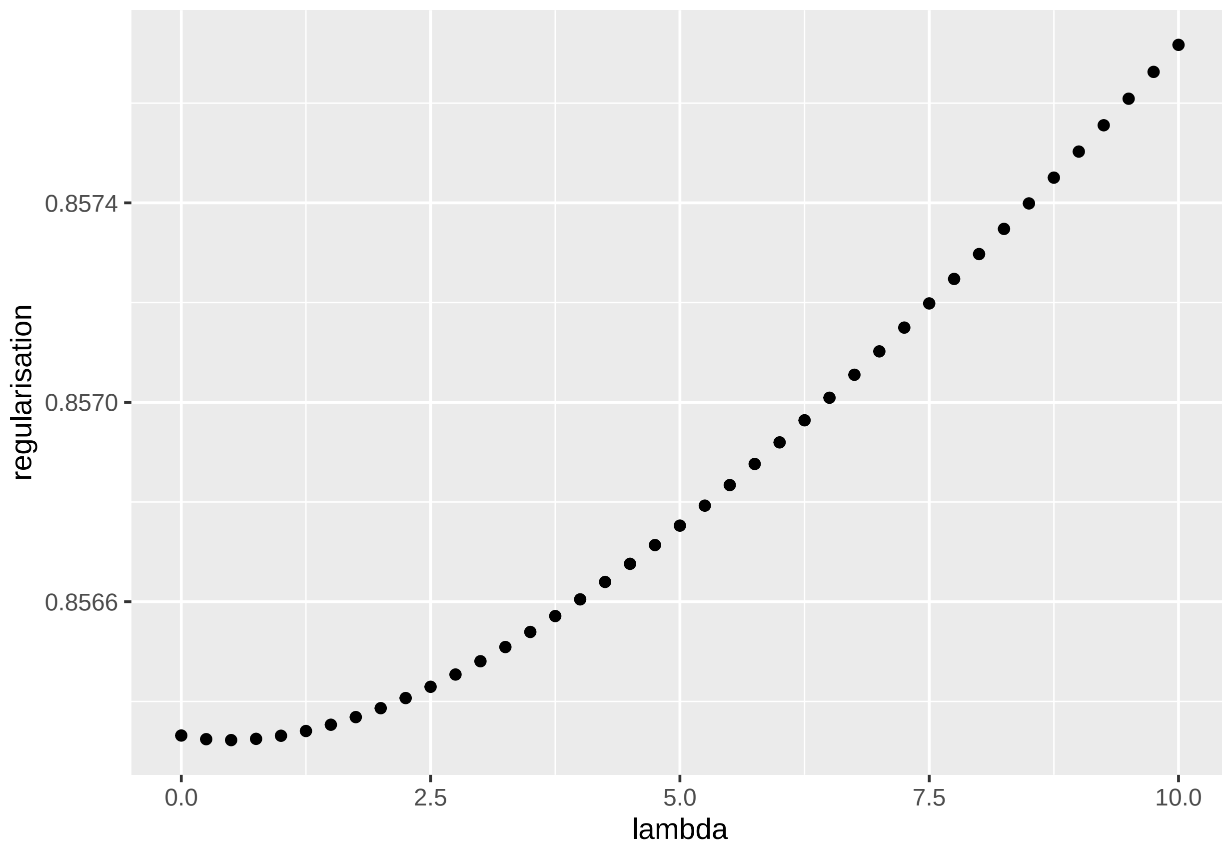
```r
lambda <- seq(0, 10, 0.25)

regularisation <- sapply(lambda, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>% group_by(movieId) %>% summarise(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>% left_join(b_i, by = "movieId") %>% group_by(userId) %>% summarise(b_u = sum(ratin
  predicted_ratings <- train_set %>% left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %
  mutate(pred = mu + b_i + b_u) %>% .$pred
  return(RMSE(train_set$rating, predicted_ratings))
})

regularisation
```

```
##  [1] 0.8563317 0.8563243 0.8563225 0.8563250 0.8563312 0.8563407 0.8563533
##  [8] 0.8563686 0.8563865 0.8564068 0.8564294 0.8564540 0.8564807 0.8565091
## [15] 0.8565394 0.8565713 0.8566047 0.8566397 0.8566760 0.8567137 0.8567526
## [22] 0.8567927 0.8568340 0.8568763 0.8569196 0.8569639 0.8570091 0.8570552
## [29] 0.8571021 0.8571498 0.8571983 0.8572474 0.8572973 0.8573478 0.8573989
## [36] 0.8574506 0.8575028 0.8575556 0.8576089 0.8576626 0.8577168
```

```r
qplot(lambda, regularisation)
```




```
# This graph allows visualising the lambda. The minimum of lambda is 3.0.

lambdas <- lambda[which.min(regularisation)]
lambdas
```

```
## [1] 0.5
```

6.0 Build the third methodo with Regularisation

Compute movie effect with regularisation on the train set. The lambda chosen is 3.0, and the next step will see the effect on bi and bu. bi(movie+regularisation) bu(user+regularisation)

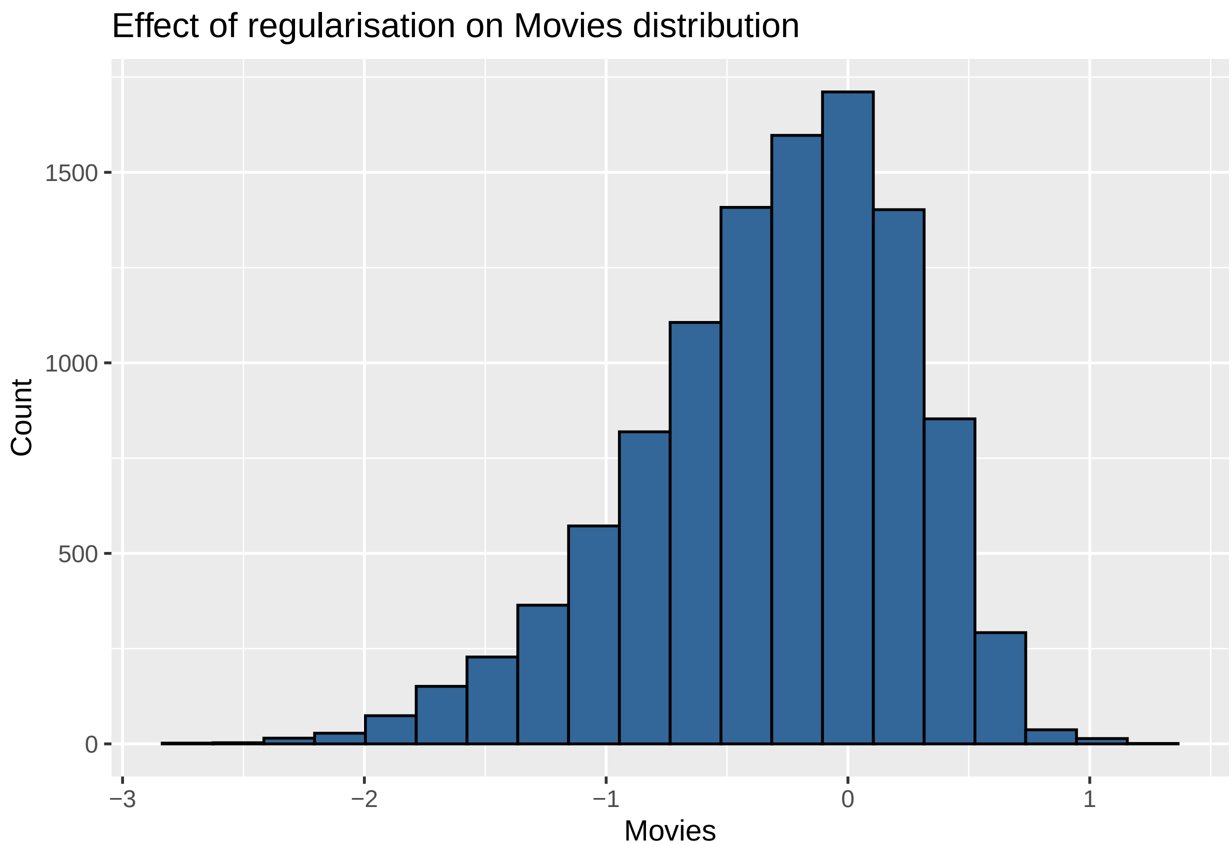
6.1 Movie (bi) + Regularisation

```
bi <- train_set %>% group_by(movieId) %>% summarise(bi = sum(rating - mu)/(n()+lambdas))
bi
```

```
## # A tibble: 10,677 x 2
##    movieId      bi
##      <dbl>   <dbl>
##  1       1  0.412
##  2       2 -0.312
##  3       3 -0.361
##  4       4 -0.626
##  5       5 -0.436
##  6       6  0.297
```

```
## 7        7 -0.147
## 8        8 -0.400
## 9        9 -0.514
## 10      10 -0.0869
## # ... with 10,667 more rows
```

```
bi %>% ggplot(aes(bi)) + geom_histogram(bins = 20, fill = "#336699", color = "black") + ggtitle("Effect
  xlab("Movies") + ylab("Count")
```



It can see the skewed on the right on distribution.

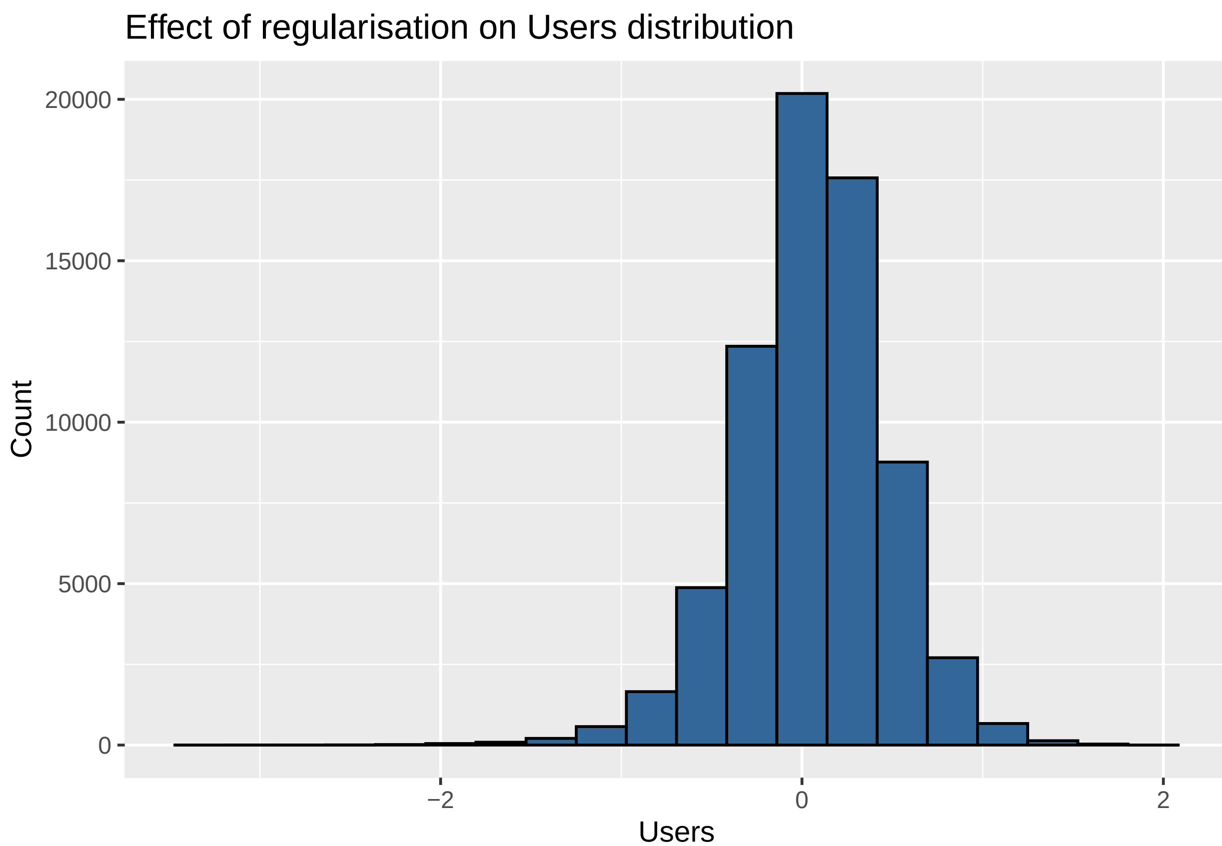
6.2 Compute user effect with regularisation on trainset

```
bu <- train_set %>% left_join(bi, by = "movieId") %>% group_by(userId) %>% summarise(bu = sum(rating - b
bu
```

```
## # A tibble: 69,878 x 2
##    userId      bu
##     <int>   <dbl>
## 1       1  1.70
## 2       2 -0.375
## 3       3  0.421
## 4       4  0.701
## 5       5  0.0841
## 6       6  0.326
## 7       7  0.0463
```

```
## 8       8  0.206
## 9       9  0.0480
## 10     10  0.0220
## # ... with 69,868 more rows
```

```r
bu %>% ggplot(aes(bu)) + geom_histogram(bins = 20, fill = "#336699", color = "black") + ggtitle("Effect
  xlab("Users") + ylab("Count")
```



It can see the distribution on user is normal.

6.3 Compute predicted value on teste set

```r
predict_bi_bu <- test_set %>% left_join(bi, by = "movieId") %>% left_join(bu, by = "userId") %>%
  mutate(pred = mu + bi + bu) %>% .$pred


rmse_3 <- RMSE(test_set$rating, predict_bi_bu)
rmse_3
```

```
## [1] 0.8644622
```

```r
rmse_results_3 <- tibble(method = "Regularisation movie and user effect model on test set", RMSE = rmse_
rmse_results_3
```

```
## # A tibble: 1 x 2
```

```
##   method                                               RMSE
##   <chr>                                               <dbl>
## 1 Regularisation movie and user effect model on test set 0.864
```

It's observed a decrease of RMSE on the regularisation model.

7.0 Final Validation

Along with the training and test, we can see the improvement of the value on RMSE. This step is the final validation on the validation set.

```
mu_edx <- mean(edx$rating)
mu_edx
```

```
## [1] 3.512464
```

7.1 Movie effect (bi)

```
bi_edx <- edx %>%
  group_by(movieId) %>%
  summarize(bi = sum(rating - mu_edx)/(n()+lambdas))
```

7.2 User effect (bu)

```
bu_edx <- edx %>%
  left_join(bi_edx, by="movieId") %>%
  group_by(userId) %>%
  summarize(bu = sum(rating - bi - mu_edx)/(n()+lambdas))
```

7.3 Prediction on validation set

```
prediction_edx <- validation %>%
  left_join(bi_edx, by = "movieId") %>%
  left_join(bu_edx, by = "userId") %>%
  mutate(pred = mu_edx + bi + bu) %>%.$pred

rmse_4 <- RMSE(validation$rating, prediction_edx)
rmse_4
```

```
## [1] 0.8654111
```

```
rmse_results4 <- tibble(method = "Final regularisation, edx vs validation", RMSE = rmse_4)
rmse_results4
```

```
## # A tibble: 1 x 2
##   method                                  RMSE
##   <chr>                                  <dbl>
## 1 Final regularisation, edx vs validation 0.865
```

8.0 Conclusions

Following this course, we started preparing the data, split into train and test set. Before running RMSE, we analysed the data and the predicted, and it observed a little high value. For better results, we started developing RMSE and used two predictors, movie and user, without exploring other predictors. When running RMSE and regularisation, we found better results rather than the initial, around 0,8652226 is quite near the goal project < 0.86490. But still predict movies to users by ratings. But some hypothesis it can arise. During the development of the project, we tried two packages: recommenderlab and recosystem. However, during the development of these packages, it wasted a lot of time and ran out of the computer. Probably the effect of these two packages can give a lower RMSE on validation data.

9.0 References

https://cran.r-project.org/web/packages/recommenderlab/index.html

https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html

https://www.diva-portal.org/smash/get/diva2:927356/FULLTEXT01.pdf

Recommendation systems: Principles, methods and evaluation https://doi.org/10.1016/j.eij.2015.06.005

Zumel Nina, Mount Jhon: Practical Data Science with R. ed; Manning, 2019. book