Capstone project - Movie lens

Nguyen Thanh Tung

6/11/2019

Contents

1 Introduction

1.1 Describe the dataset

Movie lens dataset recorded movie rating of user on IMDB website. Each user rate one movie only once.

First, getting required packages for the challenge

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```r
library(dplyr)
```

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

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

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

```r
library(dslabs)
library(ggplot2)
```

The MovieLens dataset is automatically downloaded

- [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/
- [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

In order to predict in the most possible accurate way the movie rating of the users that haven't seen the movie yet, the he MovieLens dataset will be splitted into 2 subsets that will be the "edx", a training subset to train the algorithm, and "validation" a subset to test the movie ratings.

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

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

```
## Loading required package: tidyverse

## Registered S3 method overwritten by 'rvest':
##   method            from
##   read_xml.response xml2

## -- Attaching packages ---------------------------------------------------------------- tidyverse 1.2

## v tibble  2.1.1     v purrr   0.3.2
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```r
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")

# 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)
```

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

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

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

Algorithm development is to be carried out on the "edx" subset only, as "validation" subset will be used to test the final algorithm.

1.2 Goal of the project

Build recommendation system to predict rating of users' unrated movie.

Root mean square error (RMSE) is the metrics used to evaluate the performance of the model.RMSE describes the deviation of the prediction from the actual value, the lower the RMSE, the better the performance of the model. The evaluation criteria for this algorithm is a RMSE expected to be lower than 0.8775. The function that computes the RMSE for vectors of ratings and their corresponding predictors will be the following:

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

```
RMSE <- function(predicted_value, actual_value){
  sqrt(mean((predicted_value - actual_value)^2))
}
```

1.3 Key steps that were performed

- Data exploration: summary statistic, scatter plot, histogram
- Insights collection: movie rating is affected by user bias, movie bias, and number of rating
- Build recommendation system & report result: average method, user & movie bias method, regularization method

2 Analysis

2.1 Data exploration method & Insights collected

2.1.1 Summary statistic

- Dataset summary:

Get the first sense about the dataset, print first five row

```
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
```

Number of column, number of row in training set & test set

```
nrow(edx)
```

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

```
nrow(validation)
```

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

```
ncol(edx)
```

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

```
ncol(validation)
```

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

- Summary statistic of variables:

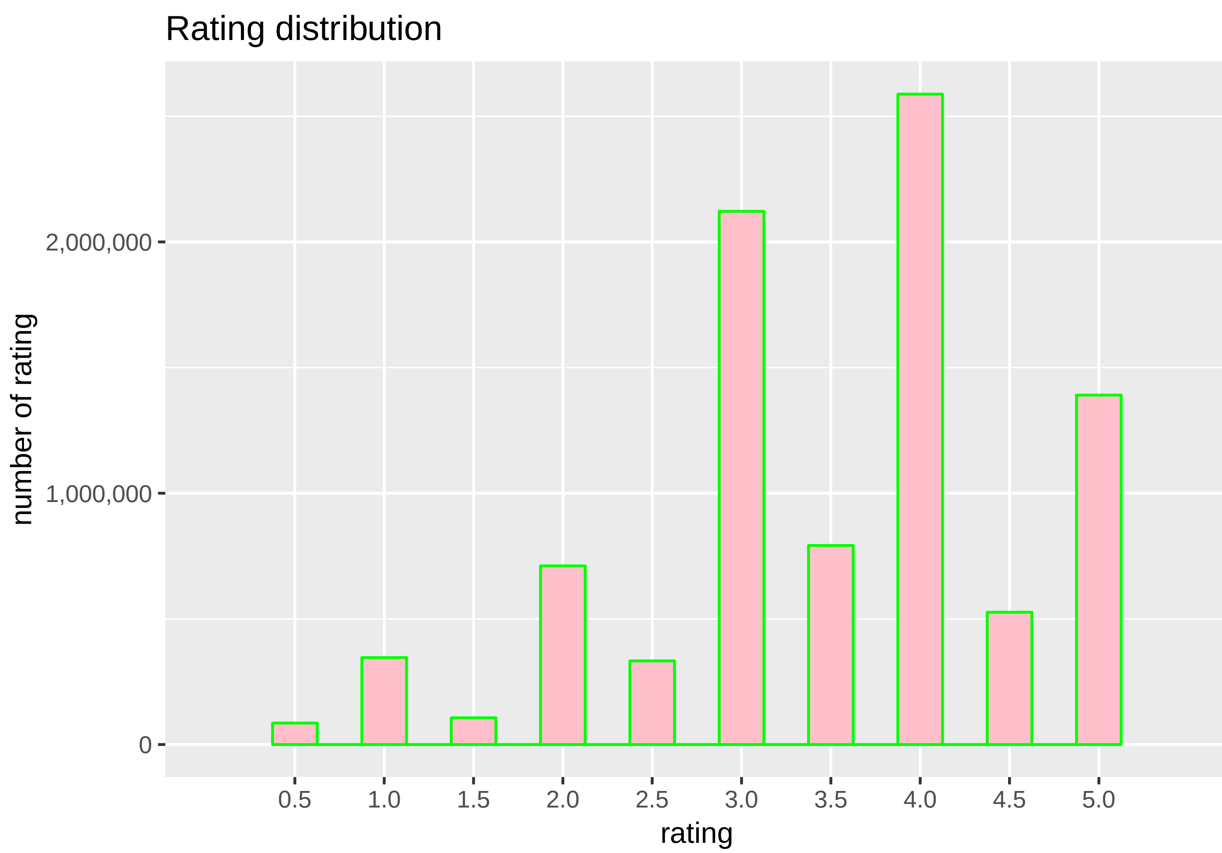
```
summary(edx)
```

```
##      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
##
##
##
```

2.1.2 Visualization & insights collected

- Distribution of rating

```
edx %>%
  ggplot(aes(x = rating)) +
  geom_histogram(binwidth = 0.25, fill = "pink", color = "green") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  scale_y_continuous(name = "number of rating", labels = scales::comma) +
  ggtitle("Rating distribution")
```



- Average rating

```
mean(edx$rating)
```

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

- Movies with highest average rating.

```
edx %>%
  group_by(movieId, title) %>%
  summarise(avg_rating = mean(rating), n_rating = n()) %>%
  arrange(desc(avg_rating)) %>%
  head(5)
```

```
## # A tibble: 5 x 4
## # Groups:   movieId [5]
##   movieId title                                  avg_rating n_rating
##     <dbl> <chr>                                       <dbl>    <int>
## 1    3226 Hellhounds on My Trail (1999)                   5        1
## 2   33264 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)             5        2
## 3   42783 Shadows of Forgotten Ancestors (1964)           5        1
## 4   51209 Fighting Elegy (Kenka erejii) (1966)            5        1
## 5   53355 Sun Alley (Sonnenallee) (1999)                  5        1
```

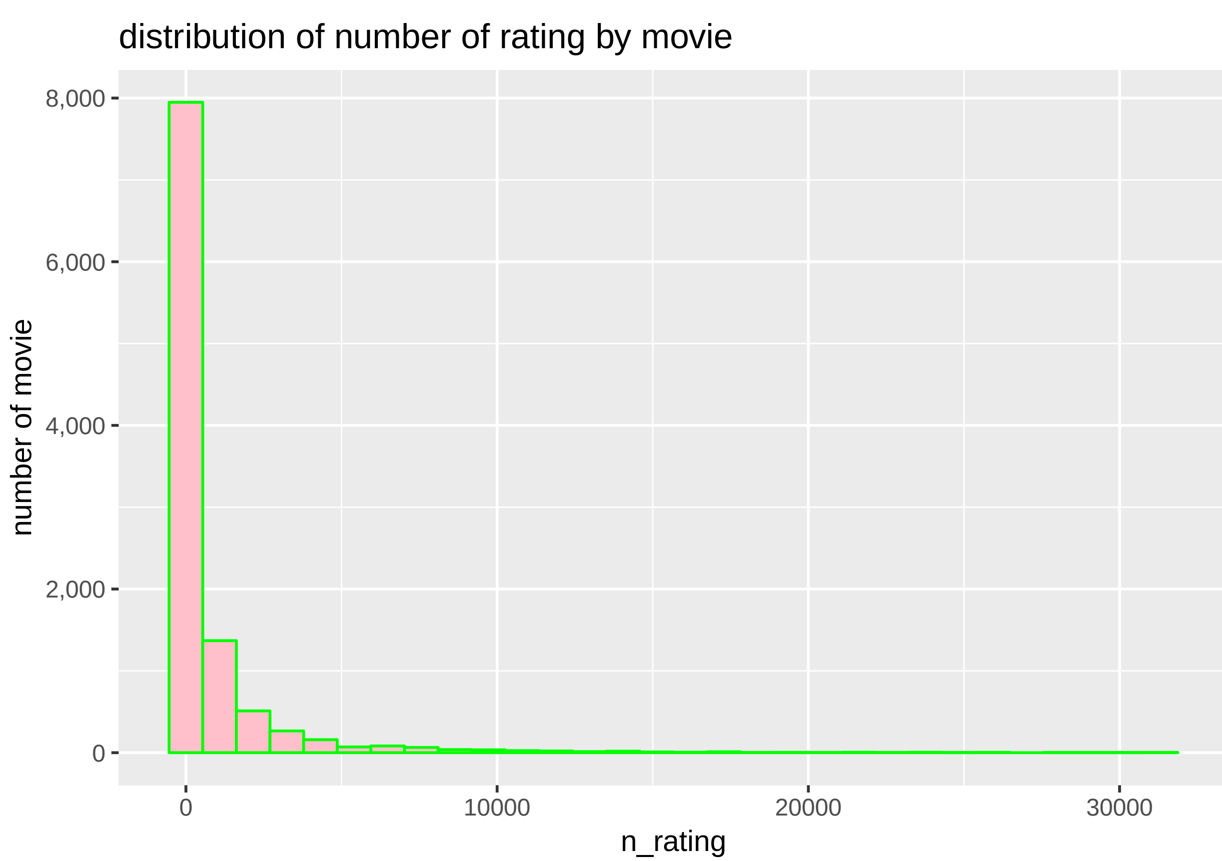
- Top 5 highest rating movie contains unpopular movie with very few number of rating. And thus, the rating of these movies should be untrustworthy. There's correlation between number of rating and rating. Here are the average number of rating of all movie and the distribution of number of rating

```
n_rating_by_movie <- edx %>%
  group_by(movieId, title) %>%
  summarise(avg_rating = mean(rating), n_rating = n()) %>%
  arrange(desc(n_rating))

#average number of rating of all movie
mean(n_rating_by_movie$n_rating)
```

```
## [1] 842.9391
```

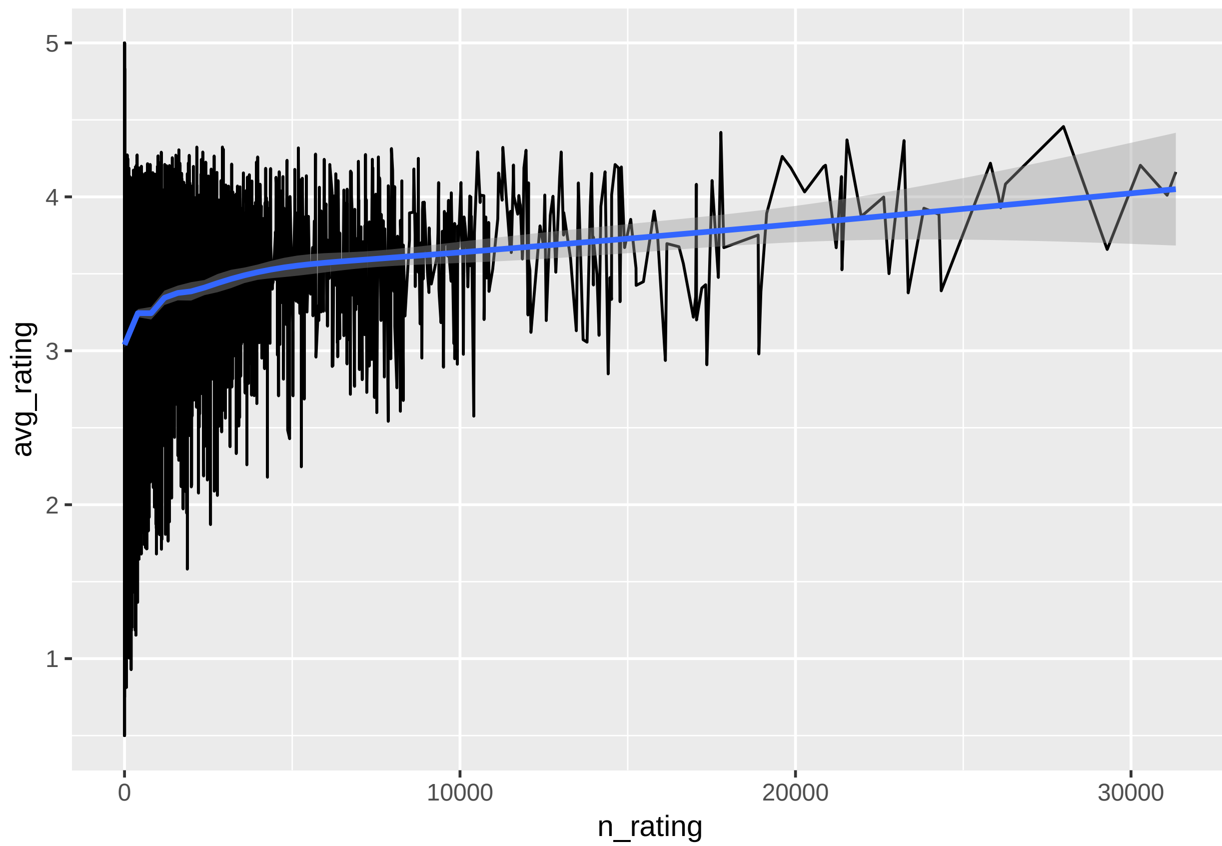
```
#distribution of number of rating by movie
n_rating_by_movie %>%
  ggplot(aes(x = n_rating)) +
  geom_histogram(fill = "pink", color = "green") +
  scale_y_continuous(name = "number of movie", labels = scales :: comma) +
  ggtitle("distribution of number of rating by movie")
```

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



```r
#correlation of number of rating and rating

n_rating_by_movie %>%
  ggplot(aes(x = n_rating, y = avg_rating)) +
  geom_line() +
  geom_smooth()
```

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



- Now, let's explore top 5 highest rating movies with number of rating > 800, we can easily see that top 5 highest rating movies are high quality movies which received many awards.

```r
edx %>%
  group_by(movieId, title) %>%
  summarise(avg_rating = mean(rating), n_rating = n()) %>%
  filter(n_rating > 800) %>%
  arrange(desc(n_rating)) %>%
  head(5)
```

```
## # A tibble: 5 x 4
## # Groups:   movieId [5]
##   movieId title                              avg_rating n_rating
##     <dbl> <chr>                                   <dbl>    <int>
## 1     296 Pulp Fiction (1994)                      4.16    31336
```

```
## 2     356 Forrest Gump (1994)                    4.01    31076
## 3     593 Silence of the Lambs, The (1991)       4.21    30280
## 4     480 Jurassic Park (1993)                   3.66    29291
## 5     318 Shawshank Redemption, The (1994)       4.46    27988
```

- Now, let's explore top 5 lowest rating movie with number of rating > 800

```
edx %>%
  group_by(movieId, title) %>%
  summarise(avg_rating = mean(rating), n_rating = n()) %>%
  filter(n_rating > 800) %>%
  arrange(n_rating) %>%
  head(5)
```

```
## # A tibble: 5 x 4
## # Groups:   movieId [5]
##   movieId title                      avg_rating n_rating
##     <dbl> <chr>                           <dbl>    <int>
## 1     171 Jeffrey (1995)                   3.59      801
## 2    3802 Freejack (1992)                  2.5       802
## 3    6550 Johnny English (2003)            2.81      802
## 4    7346 Girl Next Door, The (2004)       3.34      802
## 5    2907 Superstar (1999)                 2.47      804
```

- Key insight collected:
  - Rating is affected by movie, good & popular movies tend to receive high rating across users, bad & unpopular movies tend to receive lower rating across users.
  - Rating is affected by user, some uses are more argumentative and give bad rating across movies, while other users are more easy-going and give good rating across movies.

2.2 Modelling approach

2.2.1 Average model

Fit a naive model with predicted rating equal to average rating in the training dataset. Then compare with actual rating in validation dataset to calculate RMSE.

```
mu <- mean(edx$rating)
mu_RMSE <- RMSE(validation$rating, mu)
naive_rmse <- RMSE(validation$rating, mu)
naive_rmse
```

```
## [1] 1.060651
```

```
rmse_results <- data_frame(method = "Naive model", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

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

method	RMSE
Naive model	1.060651

2.2.2 Movie effect model

Fit movie bias model. We compute the estimated deviation of each movies' mean rating from the total mean of all movies μ. The resulting variable is called "b" (as bias) for each movie "i" b_i, that represents average ranking for movie i:

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

```r
#data
movie_bias_data <- edx %>%
  group_by(movieId) %>%
  summarise(movie_bias = mean(rating - mu)) %>%
  mutate(mu = mu)
#model
movie_bias_model <- left_join(x = validation, y = movie_bias_data, by = "movieId") %>%
  mutate(y_hat_movie_bias = mu + movie_bias)
#movie bias RMSE
movie_bias_RMSE <- RMSE(movie_bias_model$rating, movie_bias_model$y_hat_movie_bias)
rmse_results <- rbind(rmse_results,
                      data.frame(method = "Movie rating model", RMSE = movie_bias_RMSE))
rmse_results
```

```
## # A tibble: 2 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Naive model        1.06
## 2 Movie rating model 0.944
```

2.2.3 Movie & User effect model

Fit movie bias & user bias model. In order to further lower the RMSE of the model, we add User bias factor into the model. User bias is the deviation of average rating of each user from the total mean of all movie μ plus movie bias b_i.

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

User bias b_u for user u is calculated as follow:

$$b_u = Y_{u,i} - \mu - b_i - \epsilon_{u,i}$$

```r
#user & movie bias data
user_bias_data <- edx %>%
  left_join(y = movie_bias_data, by = "movieId") %>%
  group_by(userId) %>%
  summarise(user_bias = mean(rating - mu - movie_bias))
#model
user_movie_bias_model <- movie_bias_model %>%
```

```r
  left_join(y = user_bias_data, by = "userId") %>%
  mutate(y_hat = mu + user_bias + movie_bias)

#user & movie bias RMSE
user_movie_bias_RMSE <- RMSE(user_movie_bias_model$rating, user_movie_bias_model$y_hat)

rmse_results <- rbind(rmse_results,
                      data.frame(method = "Movie & User rating model", RMSE = user_movie_bias_RMSE))
rmse_results
```

```
## # A tibble: 3 x 2
##   method                     RMSE
##   <chr>                     <dbl>
## 1 Naive model               1.06
## 2 Movie rating model        0.944
## 3 Movie & User rating model 0.866
```

2.2.4 Regularized Movie & User effect model

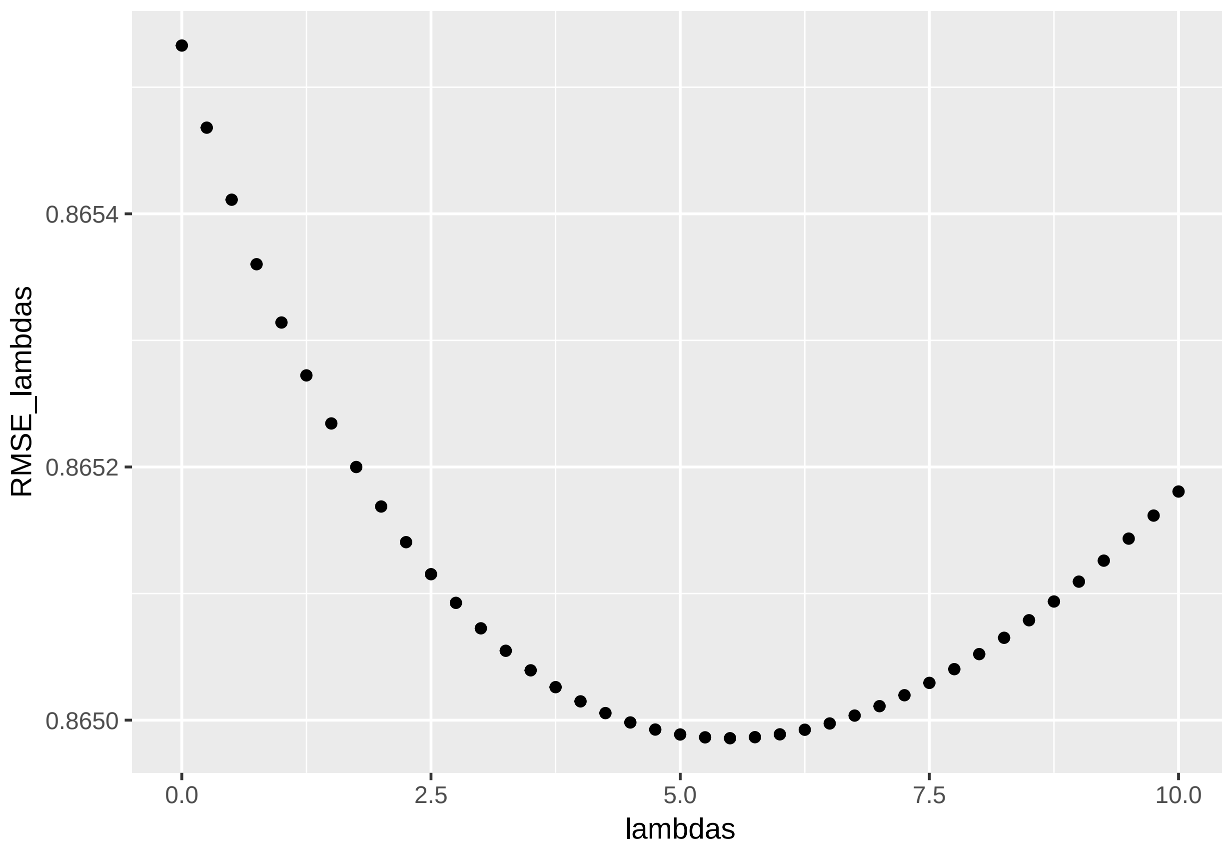
Rating of Movie with few number of ratings may be untrustworthy. Estimation of these movies may be overfitting. To prevent this, we add penalty term *lambdas* for movies with few number of ratings. If number of rating is small, then the effect of movie bias and user bias in the factor will be lowered, the prediction value will be shrink to average rating of all movies. We try different value of lambdas to find lowest RMSE.

```r
  #some movie has many few rating -> untrustworthy
edx %>% group_by(movieId) %>% summarise(n =n()) %>% arrange(n) %>% filter(n < 10)
```

```
## # A tibble: 1,046 x 2
##    movieId     n
##      <dbl> <int>
##  1    3226     1
##  2    3234     1
##  3    3356     1
##  4    3561     1
##  5    3583     1
##  6    4071     1
##  7    4075     1
##  8    4820     1
##  9    5320     1
## 10    5565     1
## # ... with 1,036 more rows
```

```r
  #regularized parameter
lambdas <- seq(0, 10, 0.25)

RMSE_lambdas <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
```

```
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

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

  return(RMSE(predicted_ratings, validation$rating))
})

result_lambdas <- data.frame(lambdas = lambdas, RMSE_lambdas = RMSE_lambdas)
result_lambdas %>% ggplot(aes( x = lambdas, y = RMSE_lambdas)) + geom_point()
```




```
lambdas[which(RMSE_lambdas == min(RMSE_lambdas))]
```

```
## [1] 5.5
```

```
rl_user_movie_bias_RMSE <- min(RMSE_lambdas)

rmse_results <- rbind(rmse_results,
                      data.frame(method = "RL Movie & User rating model", RMSE = rl_user_movie_bias_RMSE
rmse_results
```

```
## # A tibble: 4 x 2
##   method                        RMSE
##   <chr>                        <dbl>
## 1 Naive model                  1.06
## 2 Movie rating model           0.944
## 3 Movie & User rating model    0.866
## 4 RL Movie & User rating model 0.865
```

3 Result

Movie & user bias model has RMSE 0.865, which is lower than the threshold RMSE 0.875 proposed by the challenge.

4 Conclusion

The study has gone through 4 key steps: data processing, data exploration, modelling, result. The model wit best result is Movie & user bias model with regularization term to penalize movies with few number of rating, RMSE of the model is 0.864, which is lower than RMSE 0.875 required by the challenge.