Movie Rating Prediction Project

Yudi Alvin

June 17, 2019

Overview

This project is related to the MovieLens Project of the HervardX: PH125.9x Data Science: Capstone course. The present report start with a general idea of the project and by representing its objectives.

Then the given dataset will be prepared and setup. An exploratory data analysis is carried out in order to develop a machine learning algorithm that could predict movie ratings until a final model. Results will be explained. Finally the report ends with some concluding remarks.

Introduction

Recommender systems are utilized in a variety of areas, and are most commonly recognized as playlist generators for video and music services like Netflix, YouTube and Spotify, product recommenders for services such as Amazon, or content recommenders for social media platforms such as Facebook and Twitter. In our project, we will use ratings that users have given to the movies to make specific recommendations.

Recommendation systems are one of the most used models in machine learning algorithms. For this project we will focus on create a movie recommendation system using the 10M version of MovieLens dataset, collected by GroupLens Research.

Aim of the project

The aim in this project is to train a machine learning algorithm that predicts user ratings (from 0.5 to 5 stars) using the inputs of a provided subset (edx dataset provided by the staff) to predict movie ratings in a provided validation set.

The value used to evaluate algorithm performance is the Root Mean Square Error, or RMSE. RMSE is one of the most used measure of the differences between values predicted by a model and the values observed. RMSE is a measure of accuracy, to compare forecasting errors of different models for a particular dataset, a lower RMSE is better than a higher one. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Consequently, RMSE is sensitive to outliers.

Four models that will be developed will be compared using their resulting RMSE in order to assess their quality. 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:

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

Finally, the best resulting model will be used to predict the movie ratings.

Dataset

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

```
#############################################################
# Create edx set, validation set, and submission file
#############################################################
# Note: this process could take a couple of minutes for loading required
package: tidyverse and package caret
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")
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")
```

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
# The Validation subset will be 10% of the 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 subset:
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)
```

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

Methods and Analysis

Data Analysis

The edx dataset contain six variables “userID”, “movieID”, “rating”, “timestamp”, “title”, and “genres”. Each row represents a single rating for a specific movie given by a user.

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

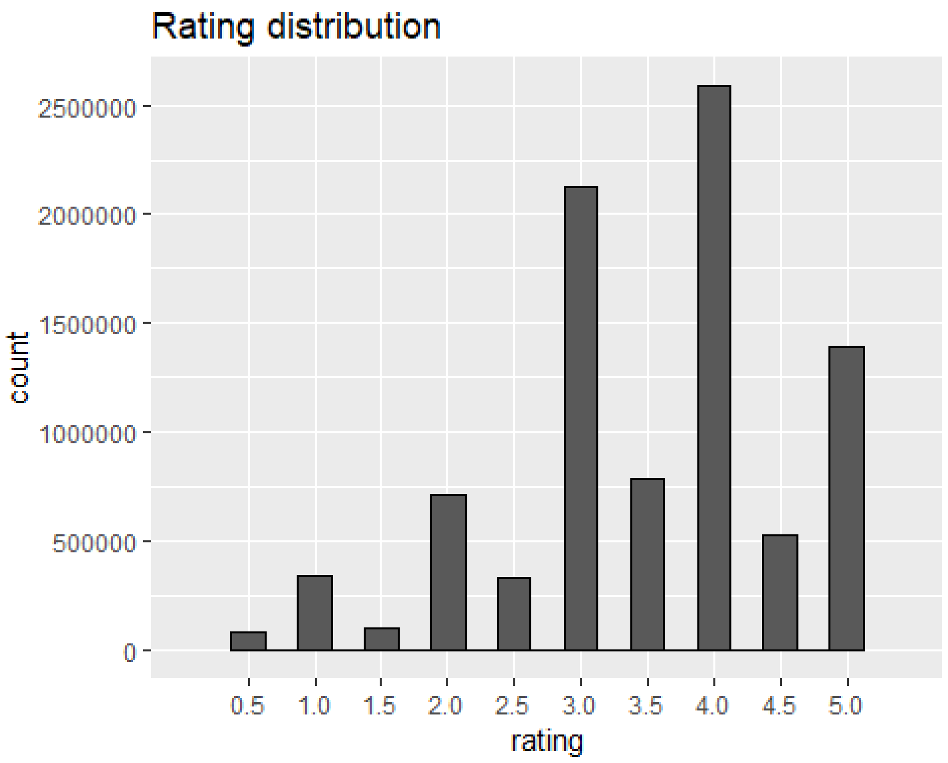
The summary of the data below confirms that there are no missing values.

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title               genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The total of unique movies and users in the edx dataset are 69878 and 10677 respectively.

```
##   n_users n_movies
## 1   69878    10677
```

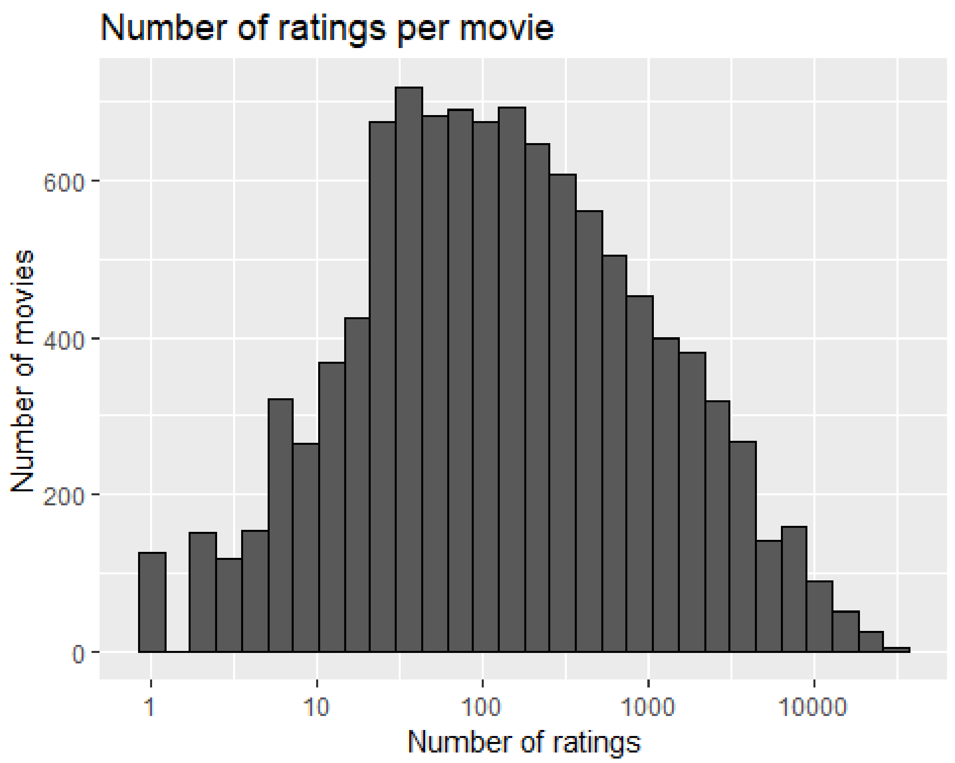
Firstly, we can create the rating distribution as shown below to analyse the on how the user rate each movie.



By analysing the rating distribution, we can interpret that users have a preference to rate movies rather higher than lower. 4 is the most common rating, followed by 3 and 5. 0.5 is the least common rating. Overall, half point ratings are less common than whole star ratings.

Next, we will also look at the number of ratings per movie.

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



we can see that there are quite a number of movies that have less than 10 ratings. Therefore, we have to take this into account as low number of ratinfs might results in less trustworhty estimate of our predictions.

We will use the regularisation method to apply a penalty term to our models in this project. Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting (the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably). Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don’t take extreme values.

Next, we can further analyse the movies that are rated only once by filtering the dataset as shown below:

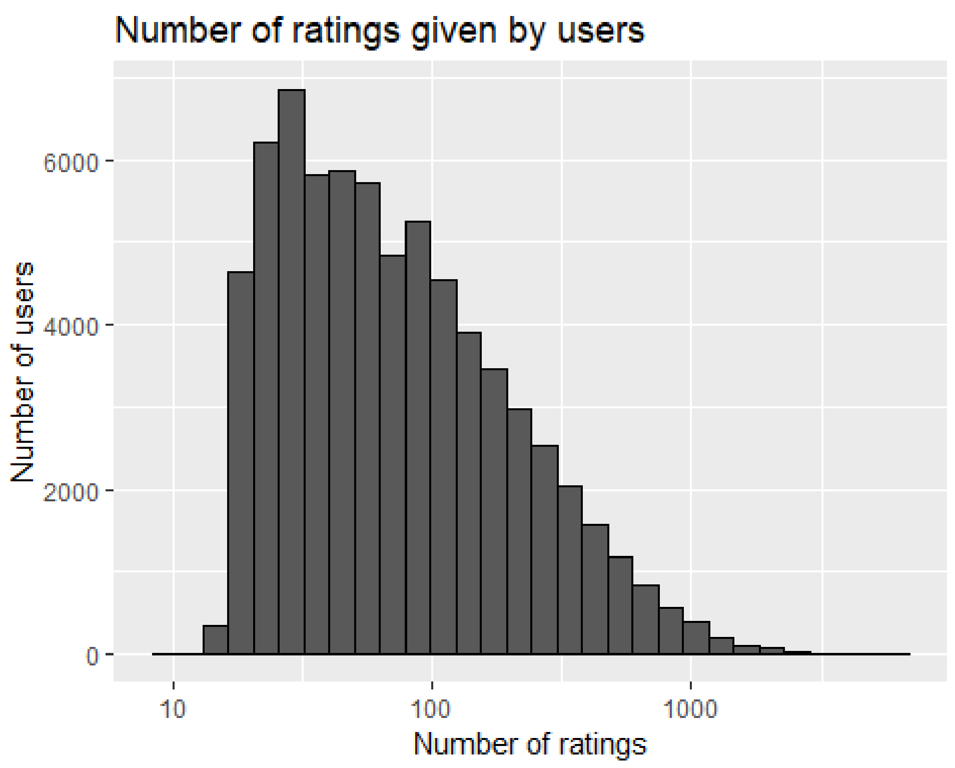
```
edx %>%
  group_by(movieId) %>%
  summarize(count = n()) %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(rating = rating, n_rating = count) %>%
  slice(1:20) %>%
  knitr::kable()
```

title	rating	n_rating
1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)	2.0	1
100 Feet (2008)	2.0	1
4 (2005)	2.5	1
Accused (Anklaget) (2005)	0.5	1
Ace of Hearts (2008)	2.0	1
Ace of Hearts, The (1921)	3.5	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di…) (1971)	1.5	1
Africa addio (1966)	3.0	1
Aleksandra (2007)	3.0	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Battle of Russia, The (Why We Fight, 5) (1943)	3.5	1
Bellissima (1951)	4.0	1
Big Fella (1937)	3.0	1
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3.0	1
Blind Shaft (Mang jing) (2003)	2.5	1
Blue Light, The (Das Blaue Licht) (1932)	5.0	1
Borderline (1950)	3.0	1
Brothers of the Head (2005)	2.5	1
Chapayev (1934)	1.5	1
Cold Sweat (De la part des copains) (1970)	2.5	1

From the subset, we can also observe that about 20 movies that are rated only once appear to be obscure. This means that the prediction for such type of movie will be difficult

Moving on, we also have to observe the number of ratings given by each user:

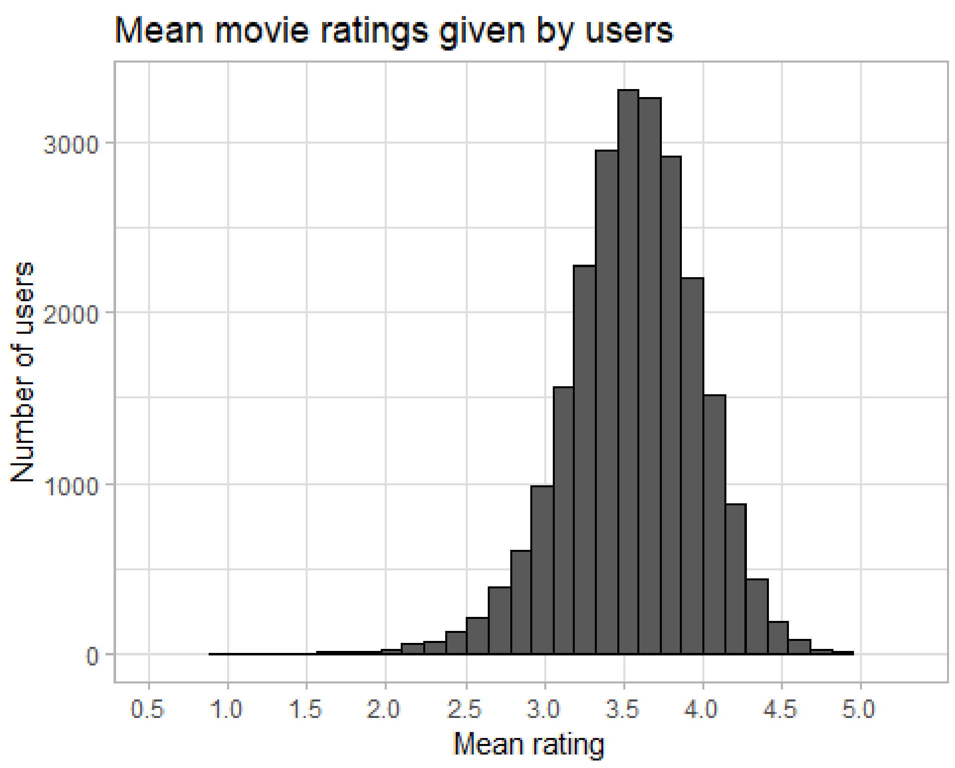
```
edx %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
ylab("Number of users") +
ggtitle("Number of ratings given by users")
```



We can observe that the majority of users have rated between 30 and 100 movies. So, a user penalty term need to be included later in our models.

Furthermore, users also differ vastly in how critical they are with their ratings. Some users tend to be more lenient and give much higher star ratings and some users tend to give lower star ratings than average. The visualization below includes only users that have rated at least 100 movies.

```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings given by users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



Modelling Approach

Now, the loss-function that computes the RMSE is defined as follows:

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

N: the number of user/movie combinations and the sum occurring over all these combinations. The RMSE is our measure of model accuracy. We can interpret the RMSE similarly to a standard deviation: it is the typical error we make when predicting a movie rating. If its result is larger than 1, it means that our typical error is larger than one star, which is not a good result. The written function to compute the RMSE for vectors of ratings and their corresponding predictions is:

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

I. Average movie rating model

We start by building the simplest possible recommender system by predicting the same rating for all movies regardless of user who give it. A model based approach assumes the same rating for all movie with all differences explained by random variation :

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

$\epsilon_{u,i}$: independent error sample from the same distribution centered at 0 μ: the “true” rating for all movies. This very simple model makes the assumption that all differences in movie ratings are explained by random variation alone. We know that the estimate that minimize the RMSE is the least square estimate of $Y_{u,i}$, in this case, is the average of all ratings:

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

## [1] 3.512465
```

If we predict all unknown ratings with μ, we obtain the first naive RMSE:

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

## [1] 1.061202
```

Let’s represent the result on a table which will be used for our comparison across different model later.

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

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

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

method	RMSE
Naive RMSE model	1.061202

This give us our baseline RMSE to compare with next modelling approaches.

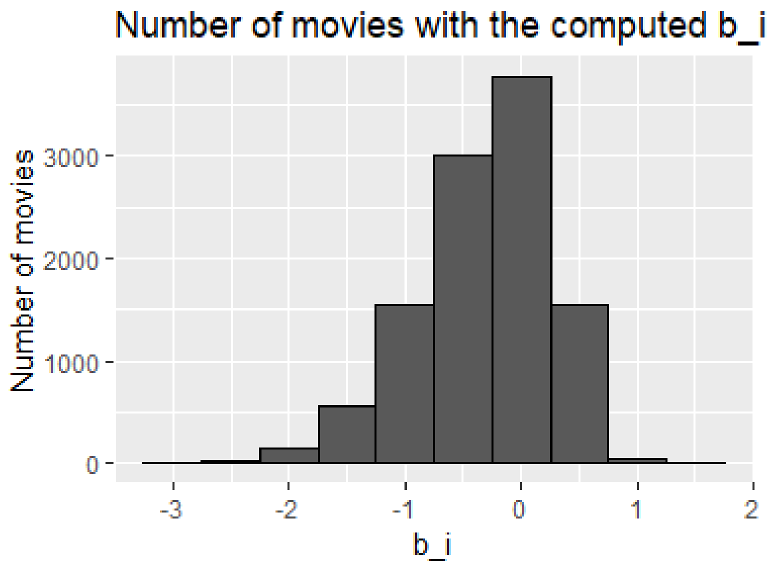
II. Movie effect model

Next, some movies are just generally rated higher than others. Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies. This movie effect can be represented by symbol b_i, which also stand for bias for each movie. Hence, our new model will be defined as follow: :

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

The least square estimate b_i is just the average of $Y_{u,i} - \mu$ for each movie i. We can plot the histogram as follow:

```
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", main = "Number of movies with the computed b_i")
```



We can observe that the histogram is left skewed, implying that more movies have negative effects. This is called the penalty term movie effect.

Our prediction improve once we predict using this model.

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

method	RMSE
Naive RMSE model	1.0612018
Movie effect model	0.9439087

Based on this model, our prediction interprets that if an individual movie is on average rated worse that the average rating μ , we predict that it will rated lower that μ by b_i.

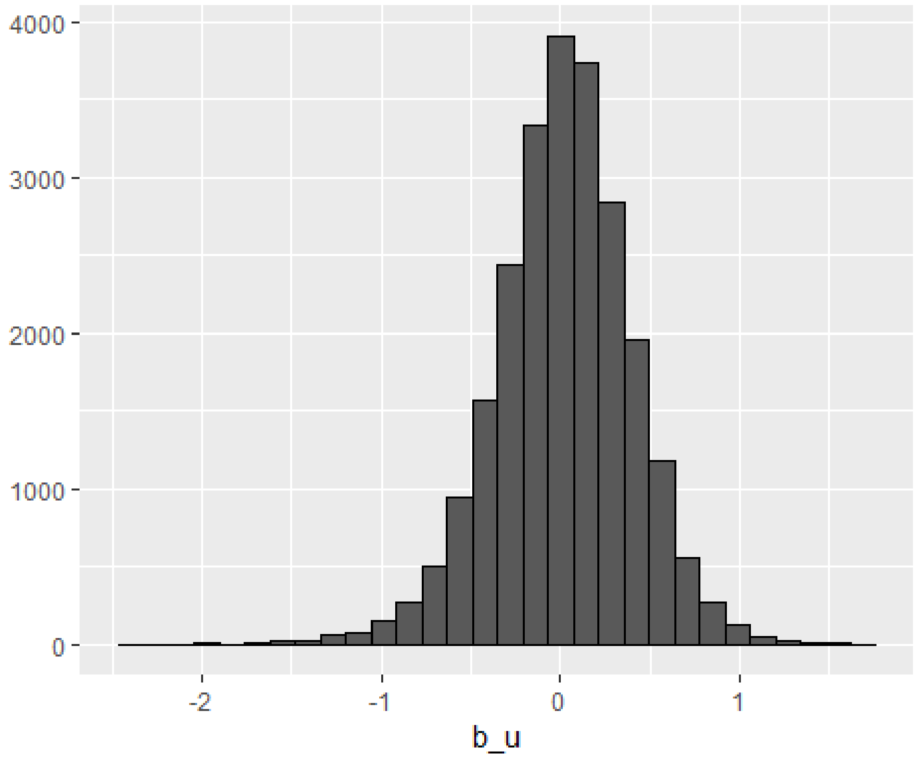
We can see an improvement but this model does not consider the individual user rating effect.

III. Movie and user effect model

Next, We compute the average rating for user μ, for those that have rated over 100 movies.

```r
user_avgs<- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

```
user_avgs%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color =
I("black"))
```



There is substantial variability across users as well: some users are very cranky and other love every movie. Hence, we can incorporate the individual user rating effect to our previous model as follow:

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

b_u: user-specific effect. If a cranky user (negative b_u rates a great movie (positive b_i), the effects counter each other and we may be able to correctly predict that this user gave this great movie a 3 rather than a 5.

We compute an approximation by computing μ and b_i, and estimating b_u, as the average of

$$Y_{u,i} - \mu - b_i$$

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

We can now construct predictors and see RMSE improves:

```
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 effect model",
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Naive RMSE model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

Our rating predictions further reduced the RMSE. But we still made mistakes on our first model (using only movies). The supposes "best " and "worst "movie were rated by few users, in most cases just one user. These movies were mostly obscure ones. This is because with a few users, we have more uncertainty. Therefore larger estimates of b_i, negative or positive, are more likely. Large errors can increase our RMSE.

IV. Regularized movie and user effect model

Some estimates of b_i and b_u are caused by movies with very few ratings and in some users that only rated a very small number of movies. Hence this can strongly influence the prediction. The use of the regularization permits to penalize these aspects. We should find the value of lambda (that is a tuning parameter) that will minimize the RMSE. This shrinks the b_i and b_u in case of small number of ratings.The general idea behind regularization is to constrain the total variability of the effect sizes. Specifically, instead of minimizing the least square equation, we minimize an equation that adds a penalty:

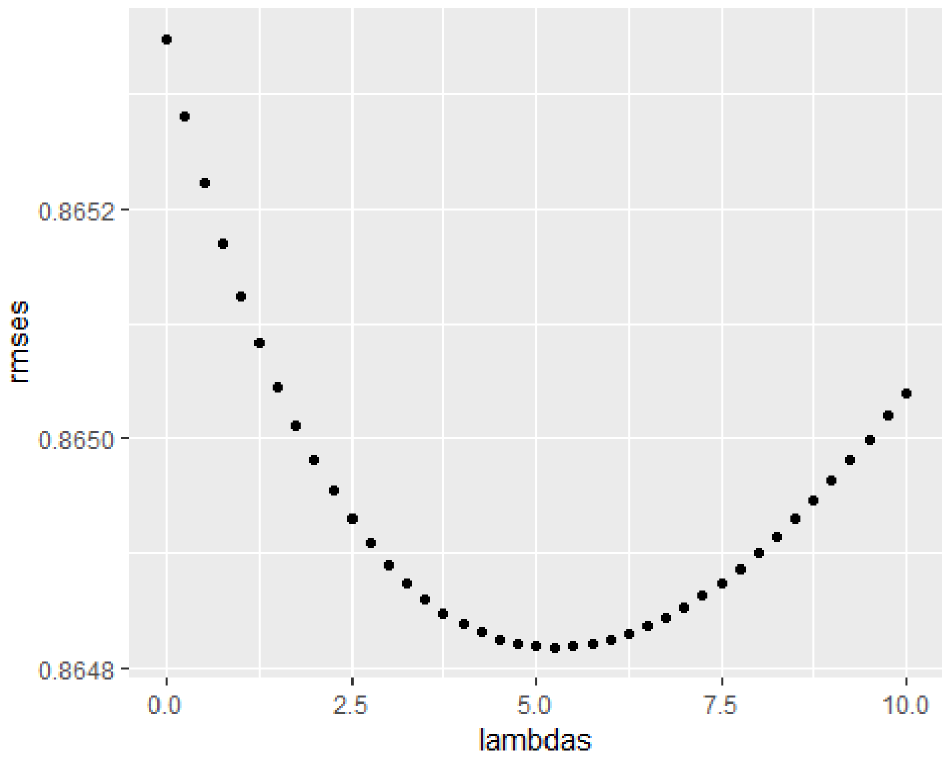
#

$$1/N\sum(Y_{u,i} - \mu - b_i)^2 + \lambda \sum_{b_i^2}$$

```
lambdas <- seq(0, 10, 0.25)

rmses <- 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))
})
```

RMSE vs lambdas

```
qplot(lambdas, rmses)
```



The optimal lambda is:

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

## [1] 5.25
```

The new results will be:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user
effect model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Naive RMSE model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

Results

The RMSE values of all the represented models are the following:

method	RMSE
Naive RMSE model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

We therefore found the lowest value of RMSE that is 0.8648170.

Discussion

The final model for our project is:

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

This model work well if the average user doesn't rate a particularly good/popular movie with a large positive b_i, by disliking a particular movie.

Conclusion

This optimal model characterised by the lowest RMSE value (0.8648170) is lower than the initial evaluation criteria (0.8775) given by the goal of the present project. We could also affirm that improvements in the RMSE could be achieved by adding other effect (genre, year, age,..). Other different machine learning models could also improve the results further, but hardware limitations, as the RAM, are a constraint.

Appendix - Enviroment

```
print("Operating System:")

## [1] "Operating System:"
```

```
version

##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          5.3
## year           2019
## month          03
## day            11
## svn rev        76217
## language       R
## version.string R version 3.5.3 (2019-03-11)
## nickname       Great Truth
```