

Movielens Project

Davide Mori

30/9/2019

Contents

PH125.9X - Capstone	1
R Background	1
Goal of the project	2
Introduction to dataset	2
Methods and Analyses	3
Data Exploration Analysis	3
Modelling Approach	7
I Average movie rating model	8
II. Movie effect model	8
III. Movie and user effect model	10
IV. Regularized movie and user effect model	11
Results	14
Discussion	14
Conclusion	14

PH125.9X - Capstone

R Background

A recommender system or a recommendation system (sometimes replacing ‘system’ with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the “rating” or “preference” a user would give to an item. They are primarily used in commercial applications. 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. These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries. There are also popular recommender systems for specific topics like restaurants and online dating. Recommender systems have been developed to explore research articles and experts, collaborators, financial services, and life insurance. [Wikipedia, 2019]

Goal of the project

The aim for this project is find a machine learning algorithm that can predicts user ratings (from 0.5 to 5 stars). For this project we will have to use an edx train subset of complete movielens dataset, and a validation set to verify our results. 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:

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

The best resulting algorithm will be used to predict the movie ratings.

Introduction to dataset

The dataset is loaded from edx with the following given 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")
# 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)
edx <- rbind(edx, removed)

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

Methods and Analyses

Data Exploration Analysis

To get familiar with the dataset, we find the first 10 rows of “edx” subset as below. The subset contain the six variables “userID”, “movieID”, “rating”, “timestamp”, “title”, and “genres”. Each row represent a single rating of a user for a single movie.

	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
8	1	356	5	838983653	Forrest Gump (1994)	Comedy Drama Romance War
9	1	362	5	838984885	Jungle Book, The (1994)	Adventure Children Romance
10	1	364	5	838983707	Lion King, The (1994)	Adventure Animation Children
11	1	370	5	838984596	Naked Gun 33 1/3: The Final Insult (1994)	Action Comedy

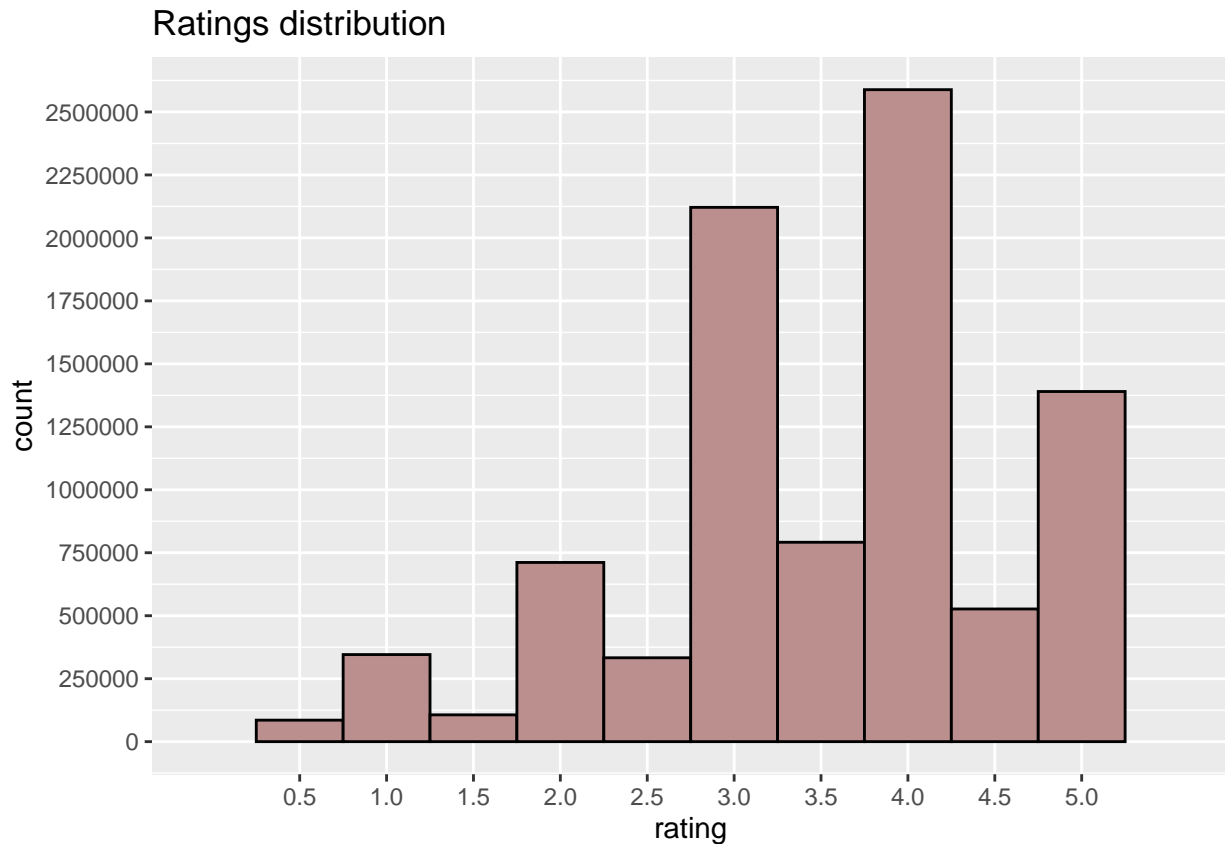
We also provided a summary of the subset, that confirms the absence of missing values.

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

As shown The total of unique movies in the edx subset are about 10677 different movies and there are about 70.000 unique users:

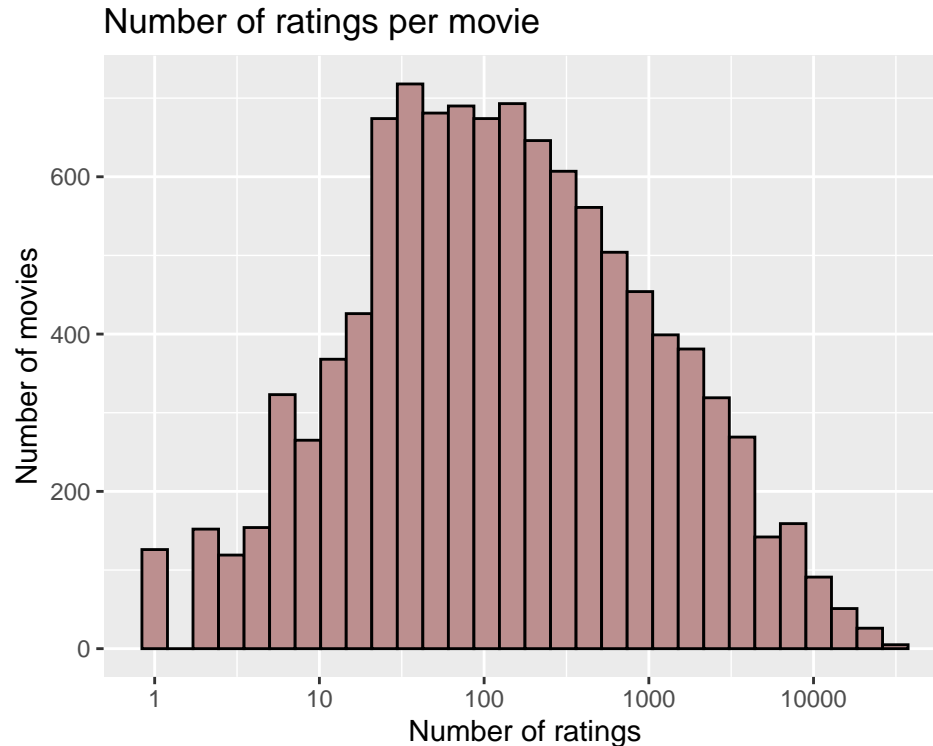
```
##      n_users n_movies n_ratings_entries n_rating_levels
## 1      69878    10677      9000055                10
```

Users have a preference to rate movies rather higher than lower as shown by the distribution of ratings below. 4 is the most common rating, followed by 3 and 5. 0.5 is the least common rating. In general, half rating are less common than whole star ratings.



After further analyses we can observe that some movies have been rated much often than other, while some have very few ratings. This will be important for our model and for our further model in terms of regularization. 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.

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



Here we show the movies with only one rating score.

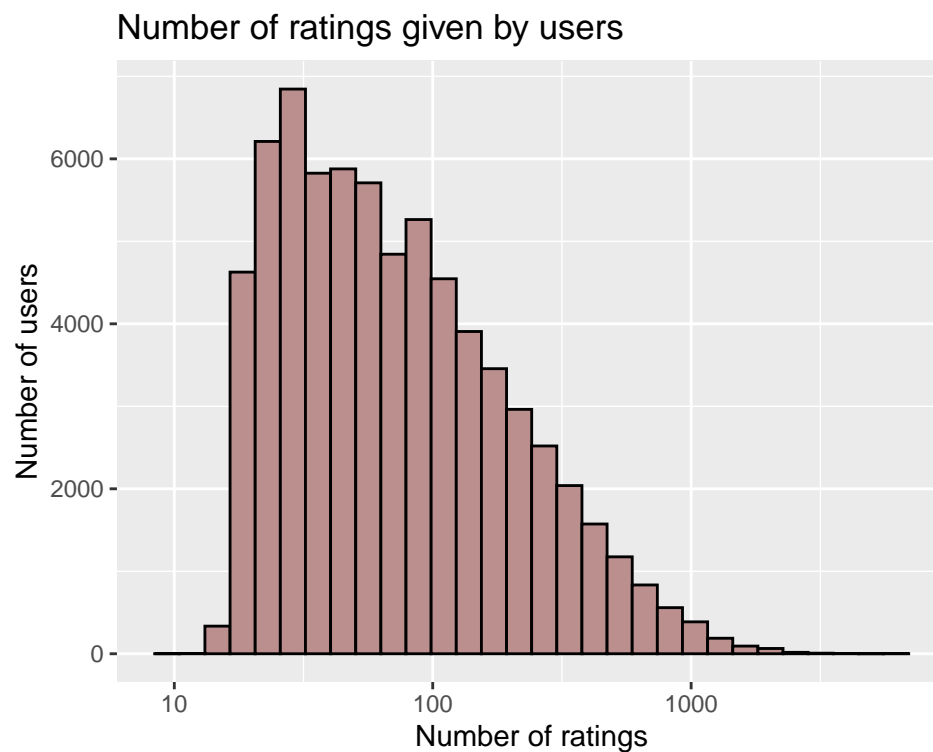
```
edx %>%
  group_by(movieId) %>%
  summarize(count = n()) %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(rating = rating, n_rating = count) %>%
  slice(1:25) %>%
  knitr::kable()
```

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

title	rating	n_rating
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
Condo Painting (2000)	3.0	1
Confess (2005)	2.5	1
Confessions of a Superhero (2007)	0.5	1
Cruel Story of Youth (Seishun zankoku monogatari) (1960)	2.5	1
David Holzman's Diary (1967)	4.0	1

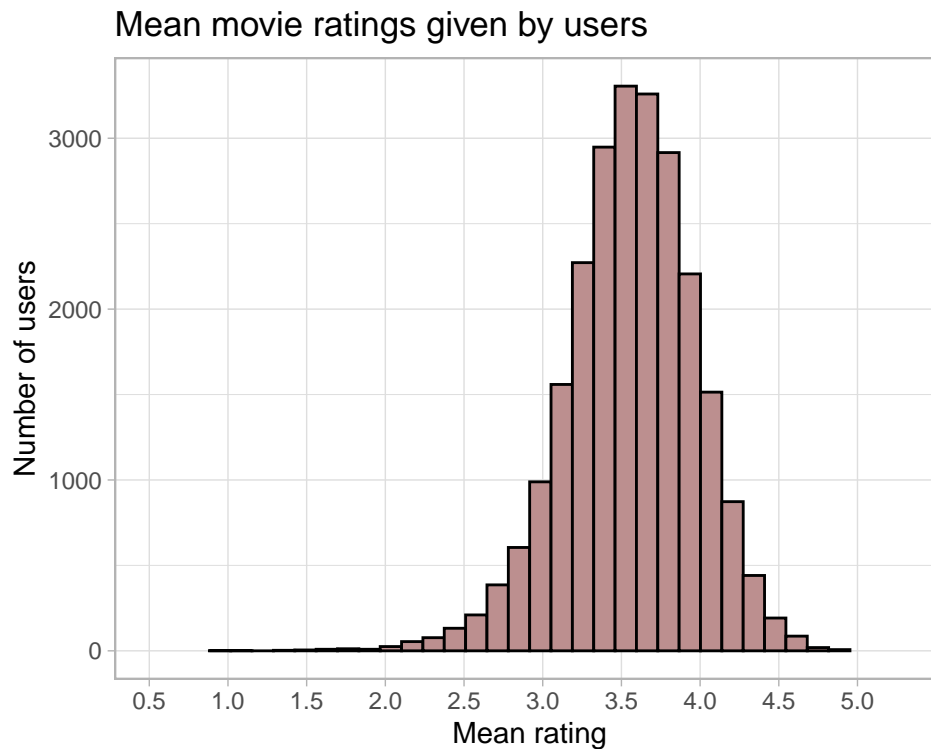
As we can observe, the majority of users have rated a number of movies between 30 and 100. So, also a user penalty term need to be included later in our models.

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



Furthermore, users differ vastly in how critical they are with their ratings. Some users tend to give much lower star ratings and some users tend to give higher star ratings than average.

```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black", fill="rosybrown") +
  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

We write now the loss-function, previously anticipated, that compute the RMSE, defined as follows:

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

with N being 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))
}
```

The lower is the RMSE the better is the result of our model, as we said in the introduction.

I Average movie rating model

The first basic model that we'll show predicts the same rating for all movies, and this value is equal to the dataset's mean rating. A model based approach assumes the same rating for all movie with all differences explained by random variation :

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

with $\epsilon_{u,i}$ independent error sample from the same distribution centered at 0 and 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: The expected rating of the underlying data set is between 3 and 4.

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

```
## [1] 3.512465
```

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

```
mu_rmse <- RMSE(validation$rating, mu)
cat("Our first RMSE is", mu_rmse, "(Pretty big!!)")
```

```
## Our first RMSE is 1.061202 (Pretty big!!)
```

Here, we represent results table with the first RMSE:

```
rmse_results <- data_frame(method = "Average movie rating model", RMSE = mu_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.061202

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

In order to do better than simply predicting the average rating, we incorporate some of insights we gained during the exploratory data analysis.

II. Movie effect model

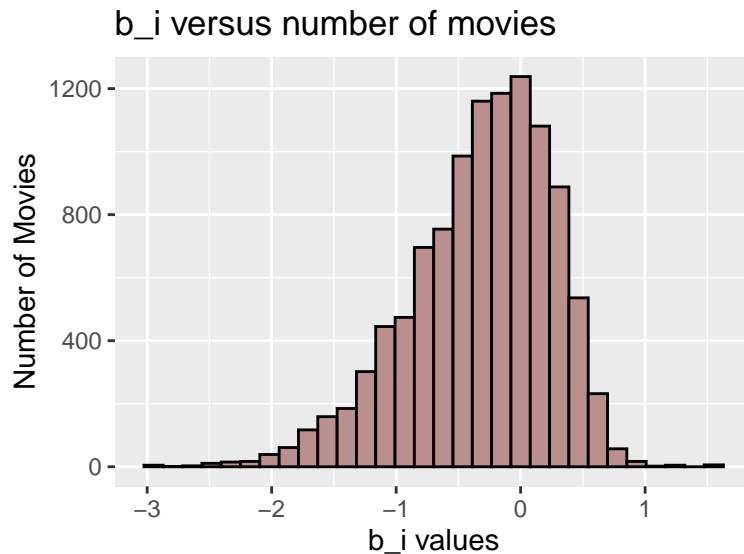
To improve above model we focus on the fact that, from experience, we know that some movies are just generally rated higher than others. Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies. We compute the estimated deviation of each movies' mean 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 :

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

The histogram is left skewed, implying that more movies have negative effects

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% ggplot(aes(b_i)) +
  geom_histogram(bins = 30, color="black", fill="rosybrown") +
  xlab("b_i values") +
  ylab("Number of Movies") +
  ggtitle("b_i versus number of movies")
```



This is called the penalty term movie effect.

Our prediction improve once we predict using this model.

```
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
First_model_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
  data_frame(method="Movie effect (b_i) model",
    RMSE = First_model_rmse ))
# Check results after First model
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect (b_i) model	0.9439087

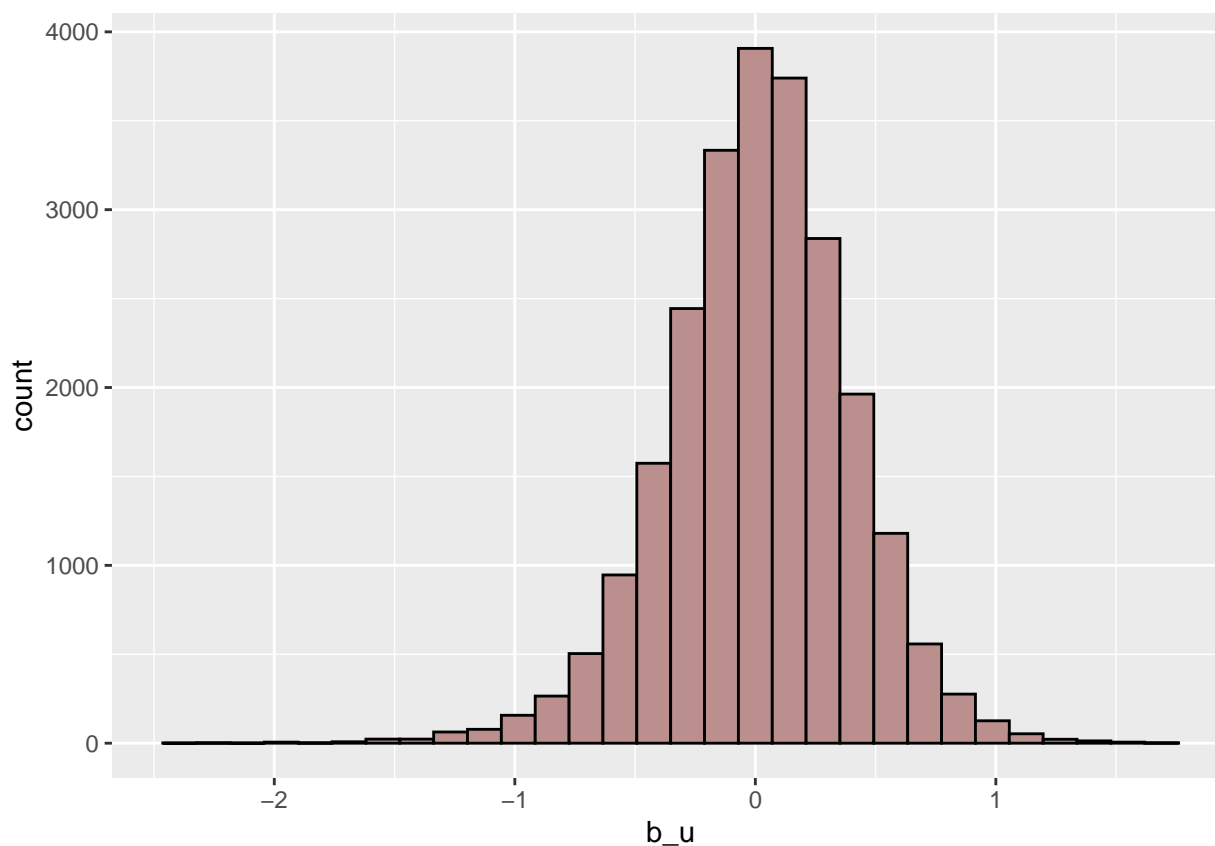
So we have predicted movie rating based on the fact that movies are rated differently by adding the computed b_i to μ . If an individual movie is on average rated worse than the average rating of all movies μ , we predict that it will be rated lower than μ by b_i , the difference of the individual movie average from the total average.

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

III. Movie and user effect model

We compute the average rating for user μ , for those that have rated over 100 movies, said penalty term user effect. In fact users affect the ratings positively or negatively.

```
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 %>% ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color="black", fill="rosybrown")
```



There is substantial variability across users as well: some users are very cranky and other love every movie. This implies that further improvement to our model may be:

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

where b_u is a 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)

Second_model_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
  data_frame(method="Movie and user (b_i + b_u) effect model",
    RMSE = Second_model_rmse))

# Check result
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect (b_i) model	0.9439087
Movie and user (b_i + b_u) effect model	0.8653488

Our rating predictions further reduced the RMSE. But we made stil 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.

Until now, we computed standard error and constructed confidence intervals to account for different levels of uncertainty. However, when making predictions, we need one number, one prediction, not an interval. For this we introduce the concept of regularization, that permits to penalize large estimates that come from small sample sizes. The general idea is to add a penalty for large values of b_i to the sum of squares equation that we minimize. So having many large b_i , make it harder to minimize. Regularization is a method used to reduce the effect of overfitting.

IV. Regularized movie and user effect model

So 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.

```

lambdas <- seq(0, 20, 0.25)

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

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+1))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+1))

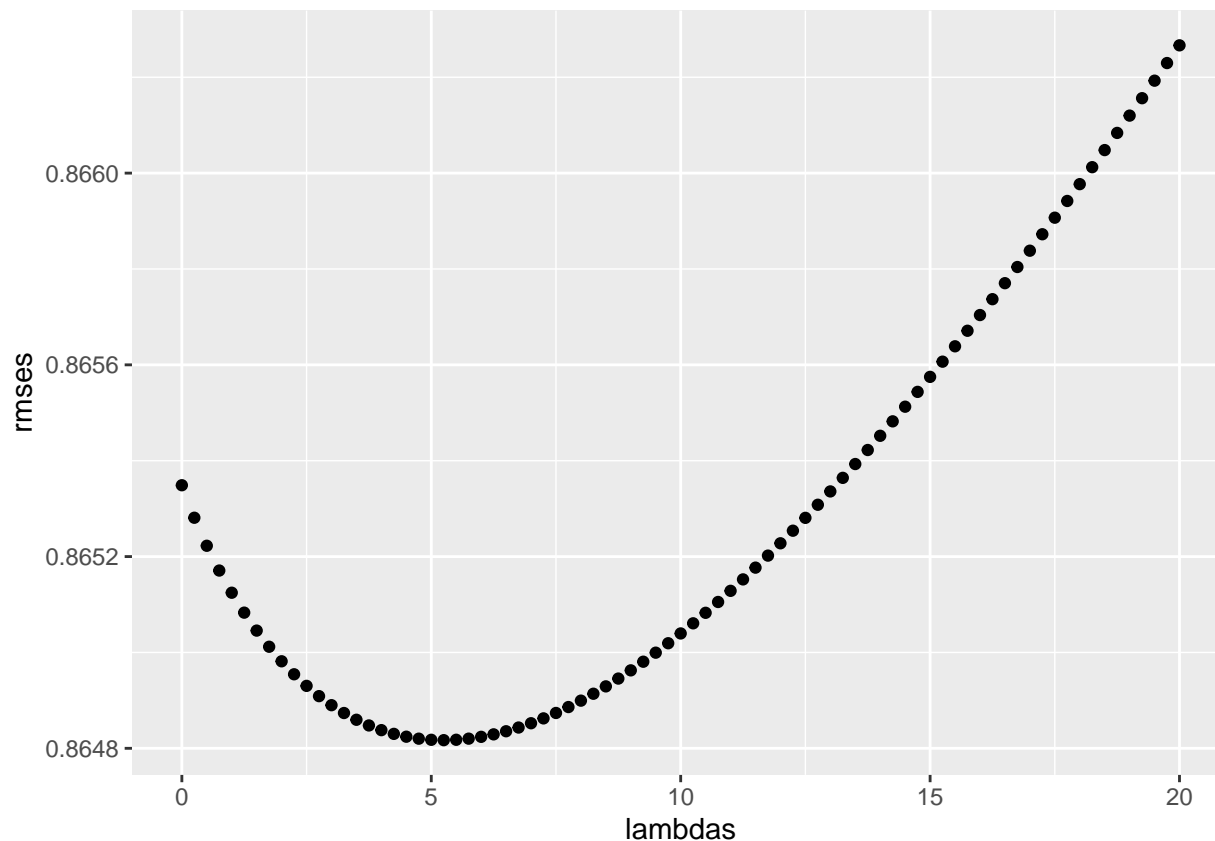
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

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

```

We plot RMSE vs lambdas to select the optimal lambda

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



For the full model, the optimal lambda is:

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

```
## [1] 5.25
```

For the full model, the optimal lambda is: 5.25

The new results will be:

```
# Test and save results
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model",
                                     RMSE = min(rmses)))

# Check result
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect (b _i) model	0.9439087
Movie and user (b _i + b _u) 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
Average movie rating model	1.0612018
Movie effect (b_i) model	0.9439087
Movie and user (b_i + b_u) 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

So we can confirm that the final model for our project is the following:

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

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

We can say to have built a machine learning algorithm to predict movie ratings with MovieLens dataset. The regularized model including the effect of user is characterized by the lower RMSE value and is hence the optimal model to use for the present project. The optimal model characterised by the lowest RMSE value (0.8648170) 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.