Report MovieLens-Harvard Edx

Octavio M.

4 June 2019

General Overview

This project is part of the Professional Certification of HarvardX: Data Science. The main objective of this is to minimize the root mean square error (RMSE) of the predicted ratings. First we are going to start with a short introducction, then the given dataset will be prepared and get ready for a data analysis that will be carried out to acomplish the main goal and develop a machine learning (ML) algorithm that will help us to predicts movie ratings. After that the results will be explaine and it will help to make some conclusions. *## Introduction*

Recommendation systems (RS), commonly, use ratings given by the users to give recommendations, this practice is very common among companies that sell big volume of products to a lot of different costumers and they allow to their customers give a rate to the products. With this collected data, they can use it to predict what rating an user will give to an item. Using that data the companies can recommended some similar products to that user.

Now that we know just a little bit of how RS works we can use it for our case, movies. Netflix is one companie that have a strong recommendation system.

The main core of our project is to **create a movie recommendation** system using MovieLens dataset.

Objective of the project

The objective of this project is to develope a machine learning algorithm that give us a right prediction for the user ratings, those ratings go from 0.5 to 5 stars. This will be achieved by using the inputs of a subset provided by edx staff.

The value used to evaluate algorithm performance is the Root Mean Square Error (RMSE). 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 the errors on RMSE is proportional to the size of the squared error; this make the RMSE model sensitive to outliers. Four models will be created and then compared by the resulto of RMSE model with the aim fo measure it quality. **The expected value of the RMSE model is to be lower than 0.8775**. The function for 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 model with the best result in the RMSE 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/ (https://grouplens.org/datasets/movielens/10m/)

. [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip (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 car
et
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 the most accurate way the MR of the users that haven't seen the movie, the dataset will be divided into 2 subsets that will be the training subset (edx) to train the algorithm, and the *validation* subset to test the MR.

```
# 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)
```

The data subset used to algorithm development is the *edx* one , and the *validation* subset will be used to test the final algorithm.

Analysis

Data

First of all we need to know a little bit of our data set. Down are the first rows of the *edx* subset. The subset contain six variables **userID**, **movieID**, **rating**, **timestamp**, **title**, and **genres**. Each row represent a single rating for one movie.

```
##   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
```

A summary of the subset confirms that there **are no missing values**.

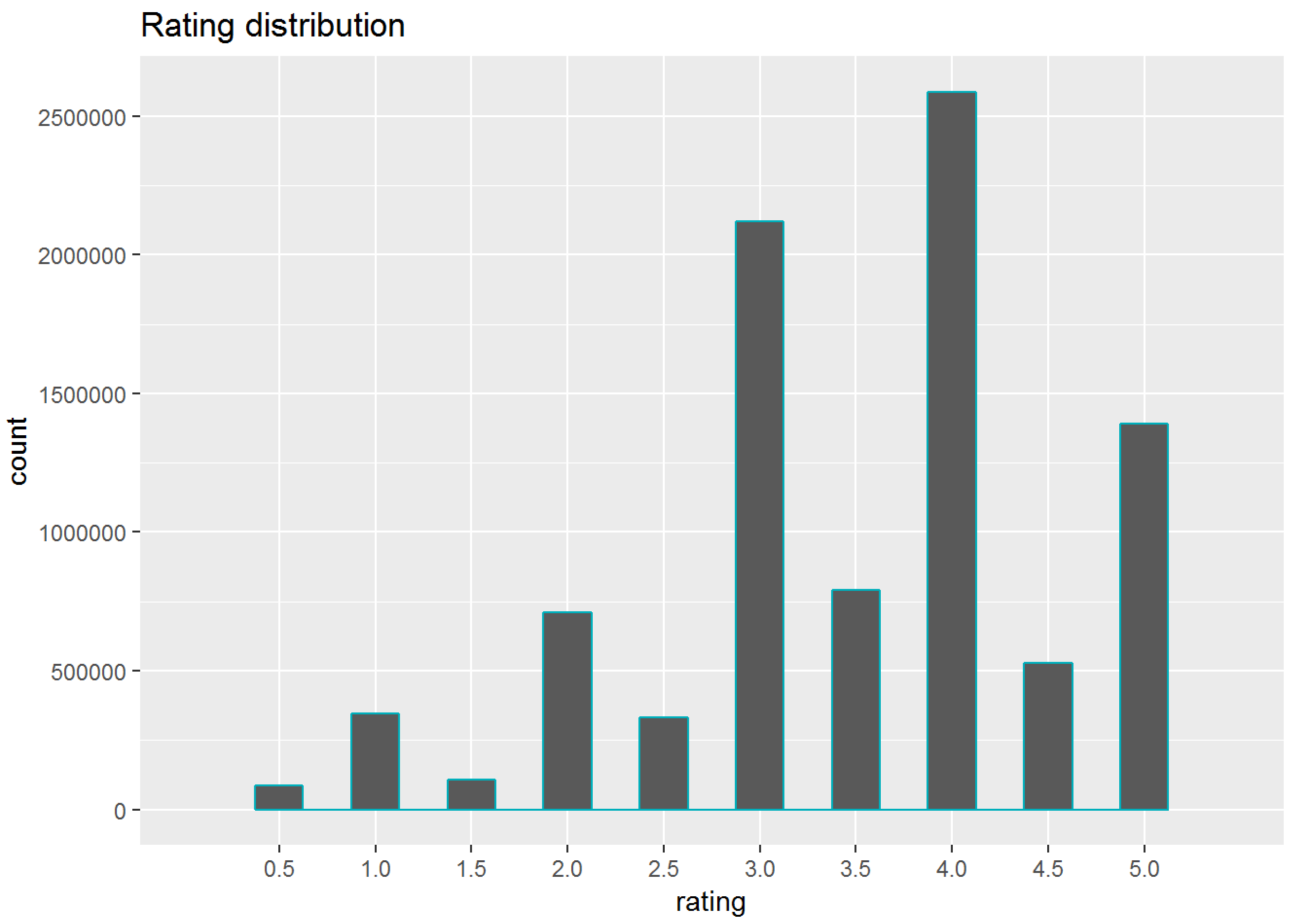
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 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
##
##
##
```

Unique movies and users in *edx* subset is **69,878** users and **10,677** different movies:

n_users <int>	n_movies <int>
69878	10677

1 row

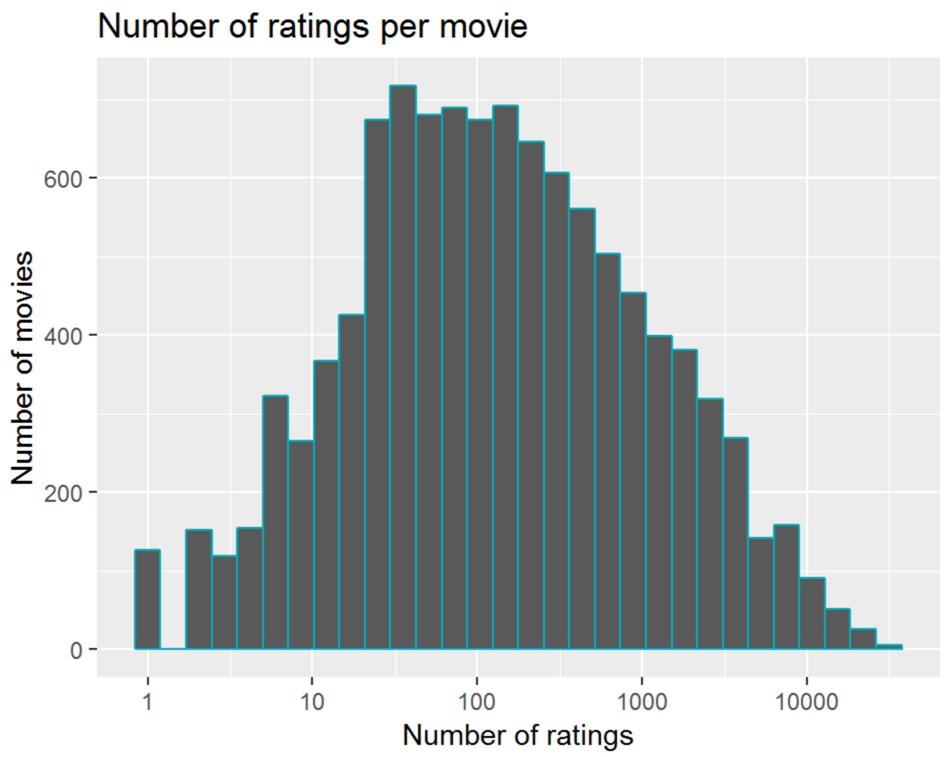
Users have a tendence to rate movies **higher than lower** as we can see below. **4 is the most common rating** and **0.5 is the least common rating**.



In the distribution we can see that some movies are rated much more times than others, while there is few movies with a very low count of ratings and in the more rare cases **only one rating**. This is vital for our model, as we said few lines ago, RMSE model sensitive to outliers. Since we have 125 movies rated only once, we need to be careful.

As we know the existence of outliers, regularisation and a penalty term will be applied to all the models used in this project. Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.

```
edx %>%
count(movieId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "#00AFBB") +
scale_x_log10() +
xlab("Number of ratings") +
  ylab("Number of movies") +
ggtitle("Number of ratings per movie")
```



20 movies were rated only predictions of future ratings for this movies will be difficult.

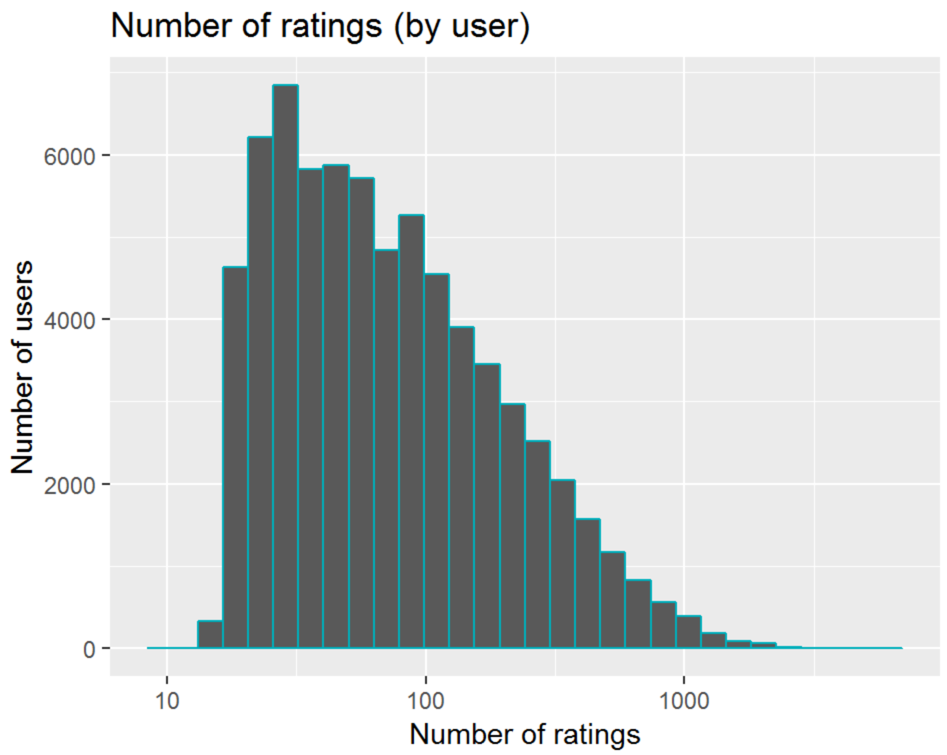
```
edx %>%
  group_by(movieId) %>%
  summarize(count = n()) %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(rating = rating, n_rating = count) %>%
  slice(1:20) %>%
  knitr::kable()
```

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

title	rating	n_rating
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

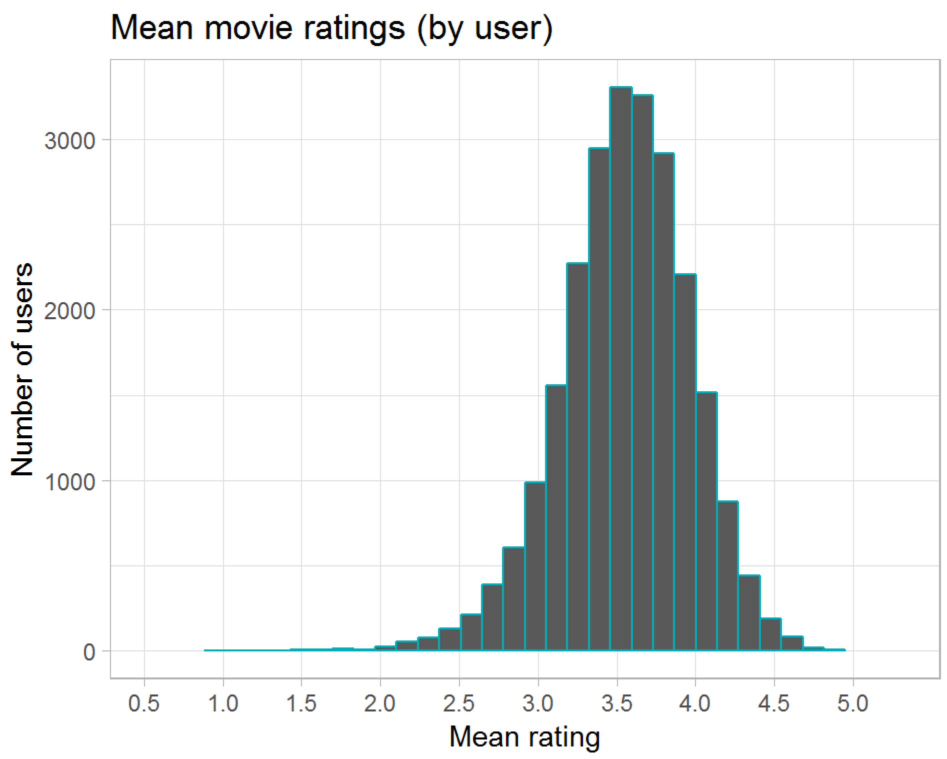
A big part of the users **give rates between 30 and 100 movies**. So, a **user penalty** term **need to be added** on our nexts models.

```
edx %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "#00AFBB") +
scale_x_log10() +
xlab("Number of ratings") +
ylab("Number of users") +
ggtitle("Number of ratings (by user)")
```



The users tend to differ too mucho in how critical are with their ratings. **Some users tend to give much lower ratings and some users tend to give higher ratings than average**. The graph below includes **only users that have rated 100 movies or more**.

```r
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "#00AFBB") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings (by user)") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



Model Approach

Here we have the loss-function, that compute the RMSE, 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. RMSE model can be interpreted as the standard deviation of a model: is the typical error made when predicting a MR. If the result is bigger than 1, it means that our **typical error is larger than one star**, **which is not very accurate**. The function in R for the RMSE for vectors of ratings and predictions is:

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

The lower difference is better for us.

I. Average movie rating model

The basic model predicts the same rating for all movies, so we need to compute the dataset mean rating. The expected rating of the underlying data set is between 3 and 4. Is necessary to start building the less complex recommender system by predicting the same rating for all movies no mater what user gave it. A model based approach assumes same rating for the movies with the differences explained by random variation :

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

$\epsilon_{u,i}$ is the independent error sample from the same distribution centered at 0 and μ the *real* rating for the movies. This model makes the assumption that the differences in movie ratings are explained only by a random variation. 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 **is between 3 and 4**.

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

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

Using μ to predict unknown ratings, the next RMSE model is obtained:

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

```
## [1] 1.061202
```

Results of the **first RMSE**:

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

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

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

method	RMSE
AMR model	1.061202

This is the first baseline RMSE that will help to compare the next models.

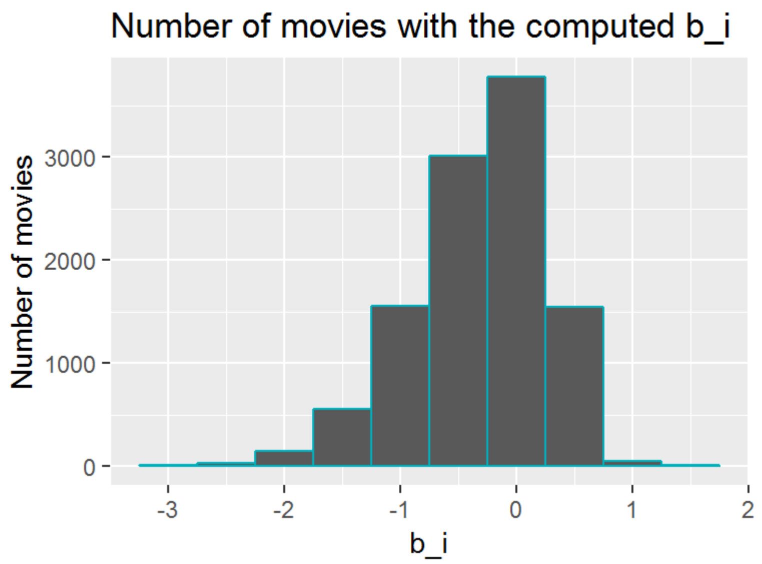
II. *Movie effect model*

Since we need to improve the quality of our model, we will focus on the fact that most movies are rated higher than others. Higher ratings are linked to more popular movies and the opposite is true for the ones with lower rates. We proceed to compute the estimated deviation of each movie mean rating from the total mean of all movies μ. The resulting variable is called "b" (as bias) for each movie "i" b_i, it represents average rank for movie i:

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

The graph have left bias, implying that more movies have negative effects

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("#00AFBB"),
ylab = "Number of movies", main = "Number of movies with the computed b_i")
```



This is the penalty term movie effect.

The prediction **show betters results using this model**.

```
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
AMR model	1.0612018
Movie effect model	0.9439087

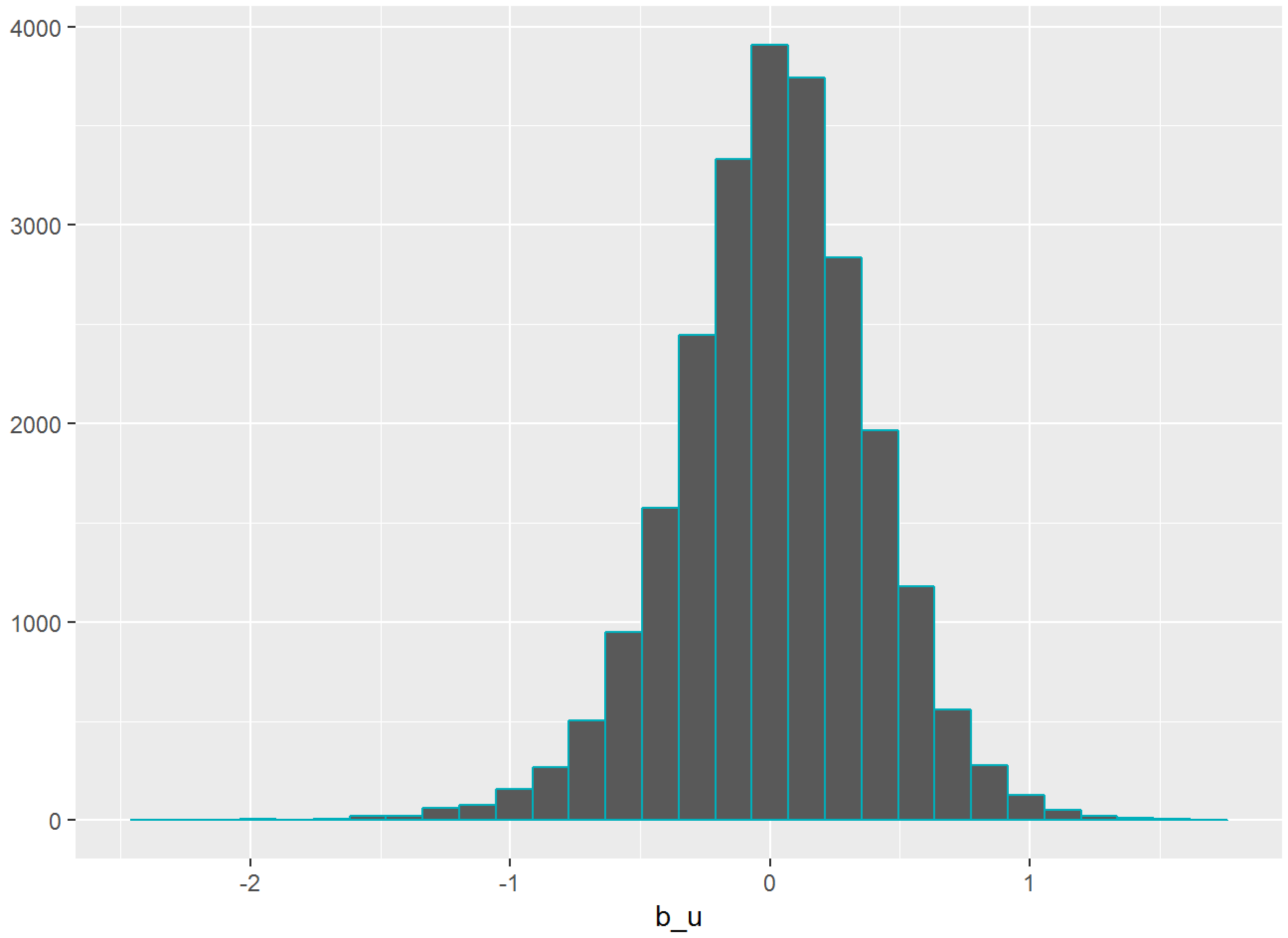
Now we have a predicted movie rating based on the fact that movies are rated differently by adding the computed b_i to μ. If an individual movie is on average rated worse that the average rating of all movies μ , we predict that it will be rated lower than μ by b_i , **the difference of the individual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

This model shows an improvement but here we dont consider the individual user rating effect.

III. Movie and user effect model

The computed average rating for user μ, using those that have rated more than 100 movies, said penalty term user effect. We know users affect the ratings in both ways + or -.

```
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("#00AFBB"))
```



There is an important variability across users, some users are very hard and other very soft with movies. So, for a next improvement to our model could be:

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

where b_u is a user-specific effect. If a hard user (negative b_u rates a great movie (positive b_i), the effects counter each other and it **will be possible to make a correct prediction that this user gave this movie a 3 rather than a 5**.

Then, we proced to compute an approximation 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))
```

Then the construction of predictors and the new improved 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 effect model",
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

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

The new rating predictions' RMSE reduced, some mistakes stil on the first model (using only movies). Suppose "best" and "worst" movies were rated by few users, in most cases just one user. These movies were mostly difficult ones. This is because with a few users the uncertainty is greater. Since we have some incertainty is probably that large values appear and with them large errors **increasing our RMSE**.

When making predictions,one number is need it, to help us with that we present the concept of regularization: **it permits to penalize large estimates that come from small sample sizes**. The idea is to add a penalty for large values of b_i to the sum of squares equation that its been minimized. **Regularization is a method used to reduce the effect of overfitting**.

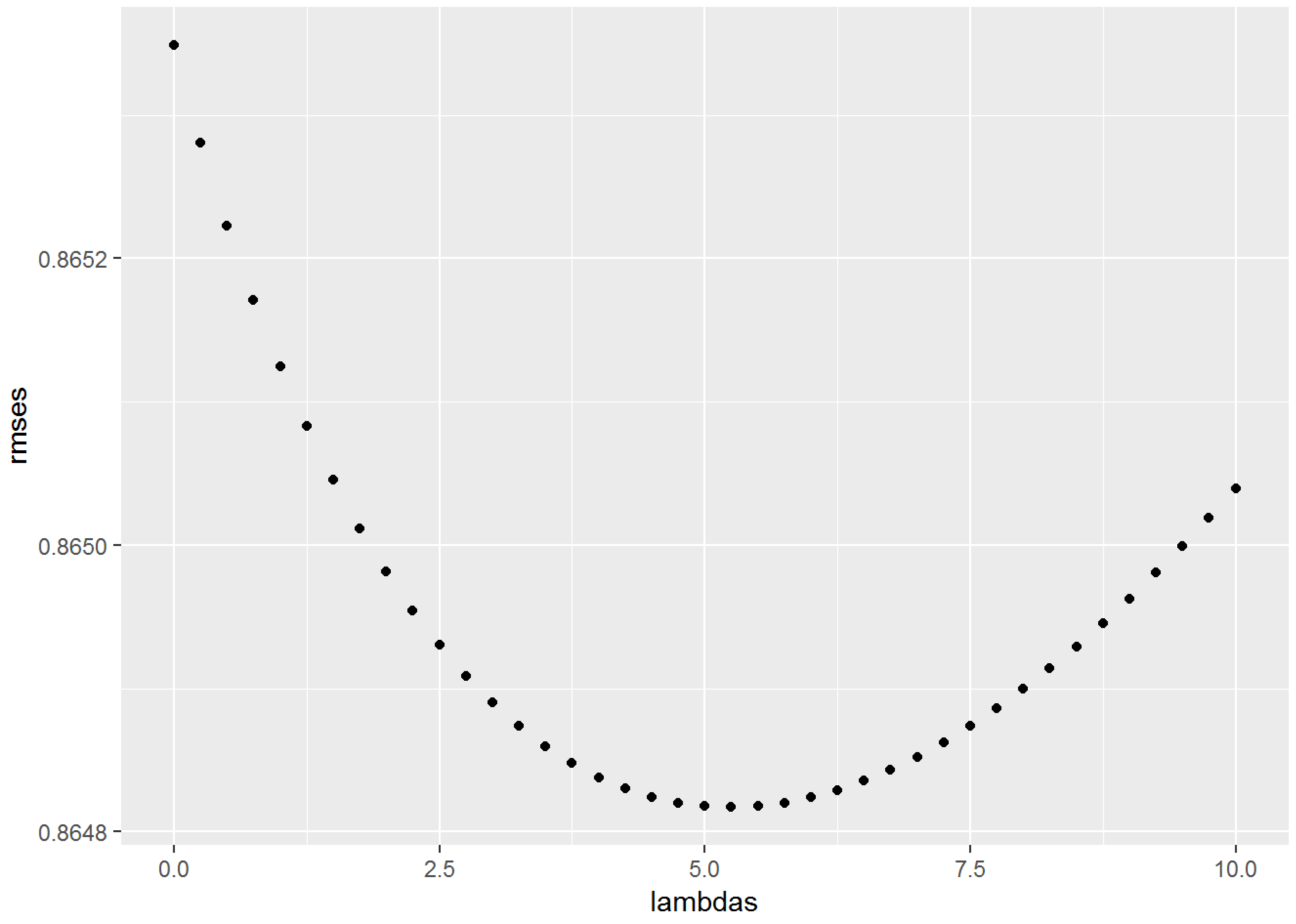
IV. Regularized movie and user effect model

The estimations of b_i and b_u are originated by movies with just a few ratings and in users that had rated a small number of movies. This can change in a big way the prediction. Using regularization allows us to penalize these aspects. First is necessary ti find the **value of lambda** (tuning parameter) that will minimize the RMSE. This reduces the b_i and b_u in the case of small number.

```
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))
})
```

Plot for lambdas compared to RMSE, this, to select the best lambda

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



The optimal lambda is:

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

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

The optimal lambda is: **5.25**

Results:

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

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

Results

RMSE values of all the models:

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

The lowest value of **RMSE is 0.8648170**.

Final model

The final model is:

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

Final considerations and further discussion

With the constructions if the MLA used to predict movie ratings with MovieLens dataset we saw a lot of factors that can give us an estimated bias, making us to give a bad prediction. The regularized model including the effect of user is the optimal model it presents the **lowest RMSE value of 0.8648170**. For further modeling its recommended to introduce new variables that can affect the results of the movie rating and raise the RMSE rate, like sex, age, etc…