MovieLens Rating Prediction Project

José Javier Miñano Ramos

Contents

1 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 objectif.

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.

1.1 Introduction

We have to start by answering a basic question: What is a recommendation system?

For Wikipedia, a recommendation system is: "a subclass of information filtering system that provide suggestions for items that are most pertinent to a particular user. Typically, the suggestions refer to various decision-making processes, such as what product to purchase, what music to listen to, or what online news to read. Recommendation systems are particularly useful when an individual needs to choose an item from a potentially overwhelming number of items that a service may offer."

In this particular project we will apply this sort of algorithms to Netflix movies. The idea is to be capable of predicting the rating that a particular user would give to a particular movie before he or she has watched it. In a later step that we do not cover here, the high-rating movies would be recommended to every user. A precise recommendation system can, in fact, decide whether a company succeeds or fails.

For computational reasons, in this project we are not using the orginal MovieLens dataset but the smaller 10M version.

1.2 Aim of the project

Our objetive is to train a regression algorithm with several variables to predict the ratings outcome of any Netflix movie. Variables such as user effect or movie effect will be explained later in more detail. We will use techniques such as regularization to obtain better precision.

Previously, we have introduced the terms "precise recommendation system", but how do we measure precision? We need a loss function. But what is a loss function? Intuitively it's a method of evaluating how well your algorithm models your dataset. If your predictions are totally off, your loss function will output a higher number. If they're pretty good, it'll output a lower number. Mathematically, a loss function is an operator that compares each empirical observation (the actual rating of a particular movie given by a particular user) with its theorical value (the model-given prediction for a movie and an user) and gives a real number as a result. Normally, the lower the number, the better the prediction.

There are various types of loss functions, and depending on the situation it is possible to choose one or another. For this project we are going to use the Root Mean Square Error, or RMSE. The RMSE is defined as it follows:

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

Being $\hat{y}_{u,i}$ the prediction given by the model and $y_{u,i}$ the actual rating given by the user. The subscript u, i indicate that every prediction and every actual rating are dependent on user (u) and movie (i). N is the number of elements in the two vectors: predictions and actual ratings.

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

1.3 Dataset

Now we are going to download the data and prepare it for our analysis. Unfortunately, the original download code provided by EdX did not work out for me, so the technical service team of the capstone project provided me a suitable code.

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")

download.file("https://www.dropbox.com/s/nspymeso8rmmak1/edx.rds?dl=1",
              "edx.rds")

download.file("https://www.dropbox.com/s/x0s477b0kzxpl6i/validation.rds?dl=1",
              "validation.rds")

edx = readRDS("edx.rds")

validation = readRDS("validation.rds")
```

To develop, train and test our algorithms we have split the dataset into two parts: the edx part as the training set and the validation part as the test set. We will train the model in the training set and measure its performance in the test set. With this strategy we can minimize the probabilities to suffer overtraining .

2 Methods and Analysis

2.1 Data Analysis

We will start by performing the Exploratory Data Analysis (EDA). Exploratory data analysis is a key step in any data science project. It helps us to introduce the main data trends to our readers and the power of graphics and visual statistics help us to keep our audience engaged. For us, as data scientist is even more important because it helps us to familiarize with the dataset.

In the first part of our EDA we are going to explore the basic numerical properties of our dataframe such as its dimension, its variables or the statistical distribution of the variables.

As we can see below, our dataframe has 9000055 observations of 6 different variables. These variables are: userId, movieId, rating, timestamp, title and genres of the film. Rating is the dependent variable which we will try to predict.

```
##   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
```

Here we have a summary of each numeric variable:

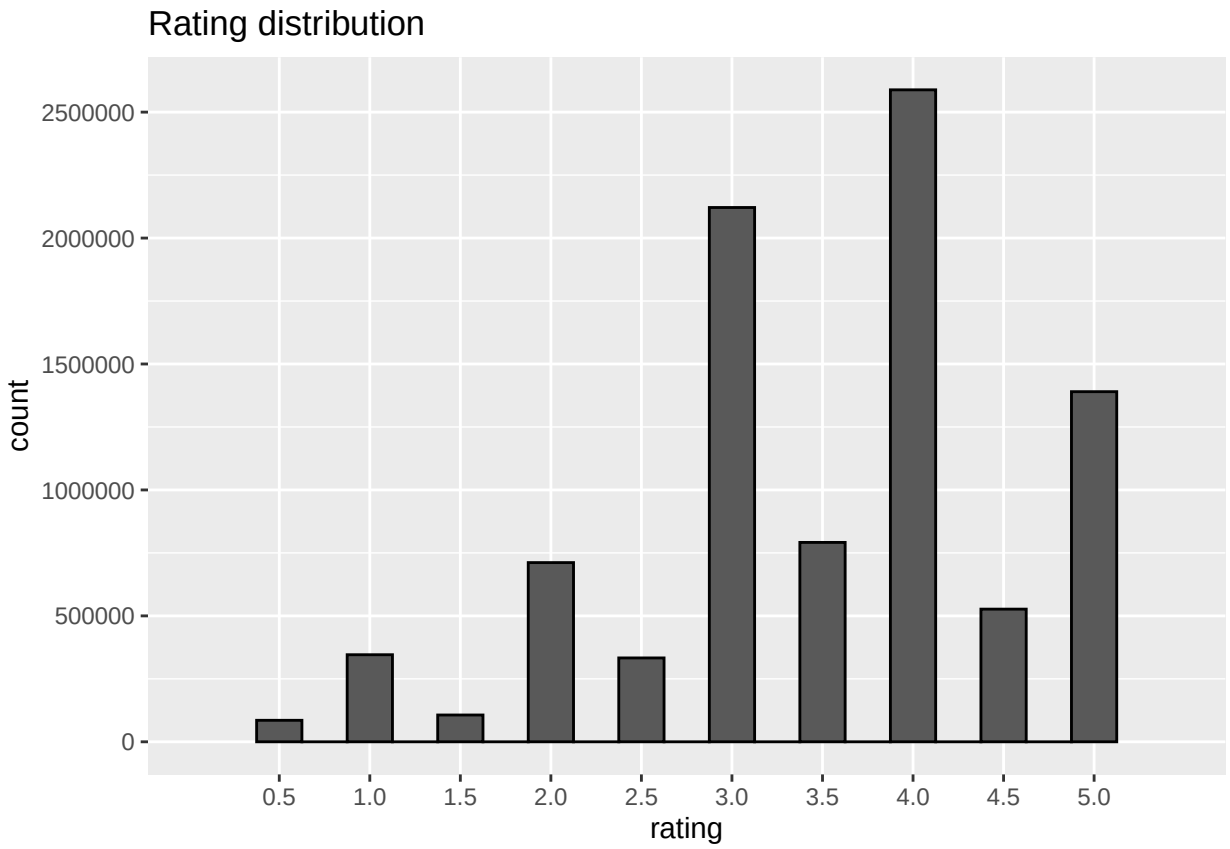
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean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

There are 69878 distinct users who has rated 10677 unique movies.

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

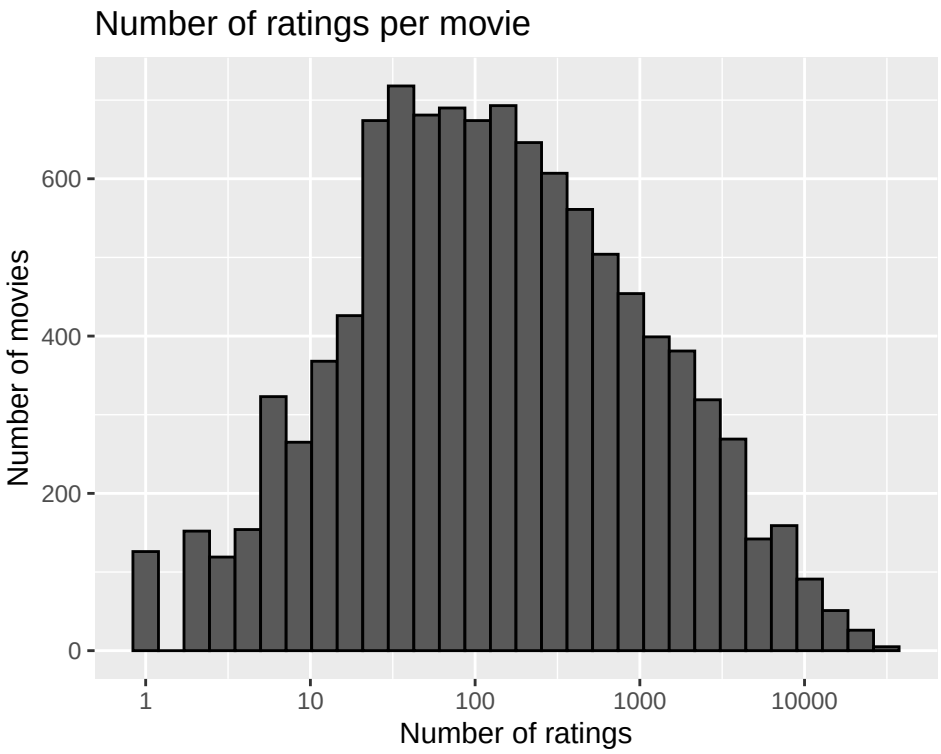
Now we are going to continue visualizing some important plots:

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```



In the ratings distribution function we can see how the whole number qualifications are more common than the decimal number qualifications. The most common rating is 4.0.

```
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")
```



As we can see, there are movies rated thousands of times and others that have been rated by very few users. For example, about 100 movies have been rated by a single user. We have to take in consideration this information as residual movies could be overrepresented in our model and produce unprecise estimations.

However these are less significant details that we will study in our regularization step.

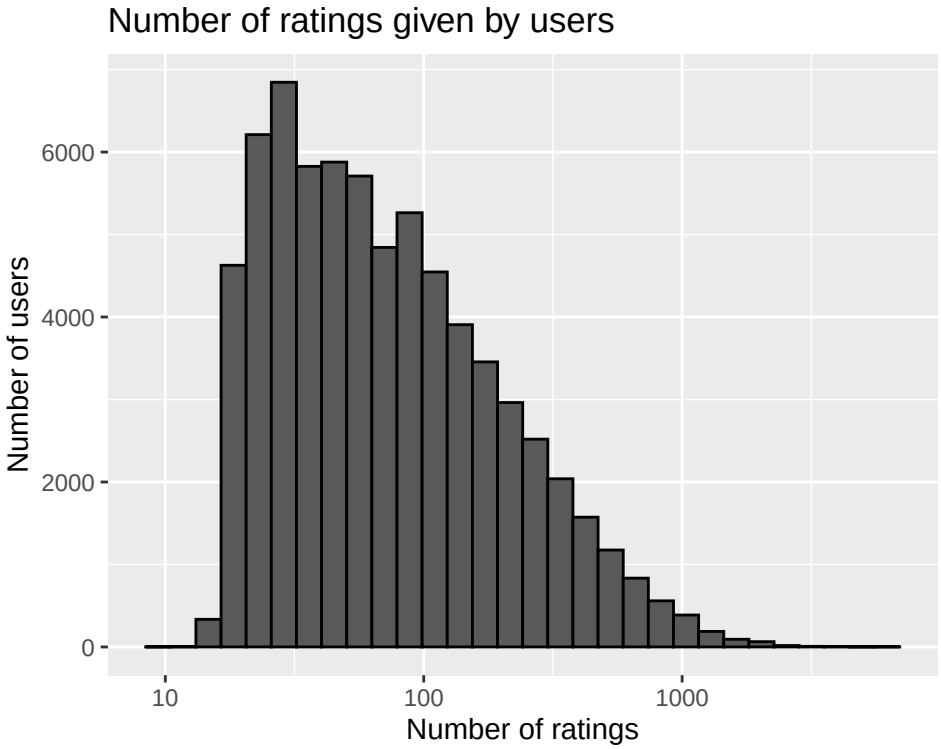
Down here we have a list of the movies that have been rated only once. We can see that most of them are "obscure" movies.

```
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

title	rating	n_rating
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

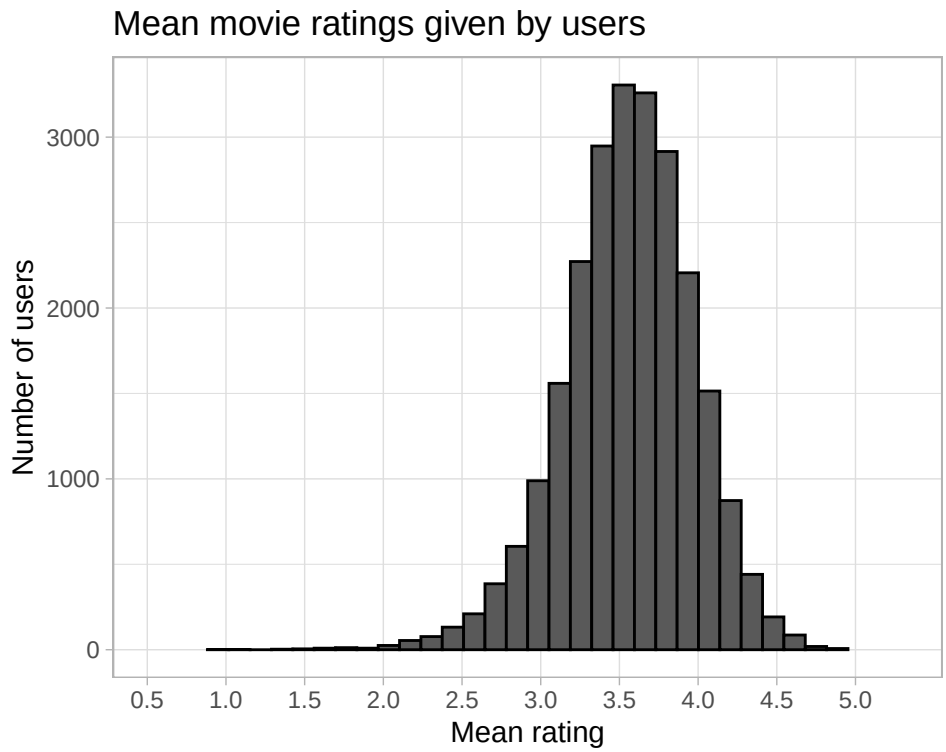
```
edx %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
ylab("Number of users") +
ggtitle("Number of ratings given by users")
```



We can observe in the plot just above that the majority of users have rated less than 100 movies. Some of them have rated less than 10 movies, this last group of users should not influence too much our algorithm as they are not reliable raters.

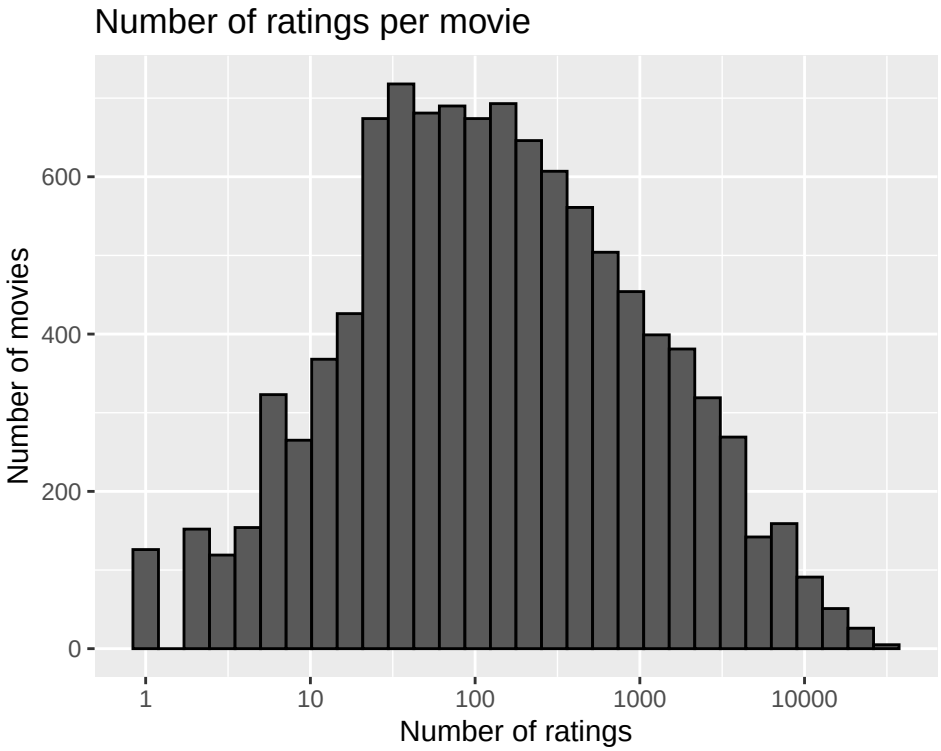
```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings given by users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5)))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

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```



The above plot represents the mean movie ratings given by users. As we can see, the majority of users usually rates movies between 3 and 4 stars.

```
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")
```



2.2 Modelling Approach

As we explained earlier, to measure our model's performance we will use a loss function. It exists a huge variety of loss functions, but for this project we are using the RMSE:

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

N is the number of given ratings, $\hat{y}$ represents the predictions and y represents the actual rating. To compute the RMSE we will use this function:

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

2.2.1 I. Average movie rating model

To start developing a model we need a baseline to start from. We will start by a constant model which we will call M0, this model constantly predicts the same rating for every movie. As we have studied through the course, the mean minimizes the RMSE so we will start constantly predicting the mean rating of the movies in the edx dataset:

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

$\epsilon_{u,i}$ represents the random variations of the model. Mathematically this is called "white noise". This is, a sucession of independent and identically distributed random variables centered at zero with standard deviation equal to one.

Calculating the mean...

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

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

The RMSE of this model is:

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

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

Now, we create a table that groups the results:

```
rmse_results <- data_frame(method = "Average movie rating model (M0)", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

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

method	RMSE
Average movie rating model (M0)	1.061202

From now on, we will improve our model to add some of the information that we noticed during our exploratory data analysis.

2.2.2 II. Movie effect model (M1)

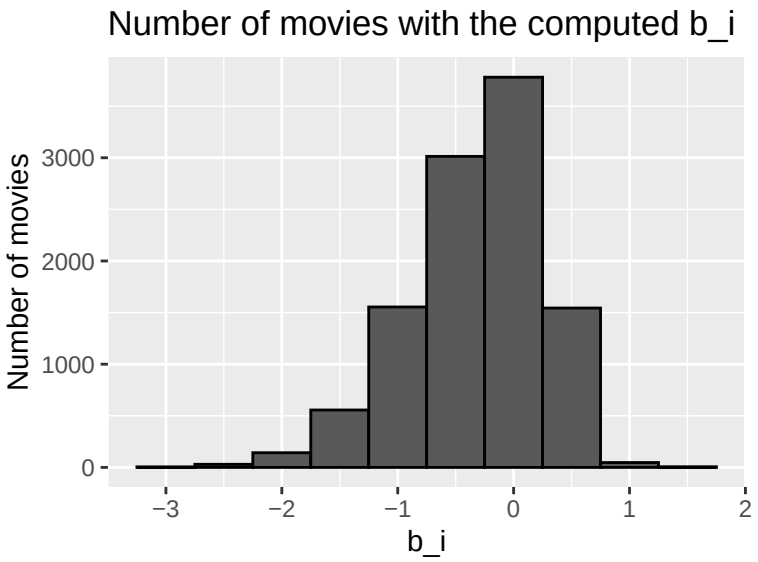
In this chapter, we are going to add a new variable to our model: the movie effect. We naturally know that some movies are "better" than others, so obviously, some movies receive higher ratings than others. In fact, as we have seen in the rating distribution plot, it seems like ratings are normally distributed. Now, we are going to compute the approximate effect that every single movie makes in its own rating.

Knowing the standard deviation of every movie's rating from its mean rating will give us some understanding of this movie effect of movie bias. So we have the next model:

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

Where b_i represent the the bias for movie i.

```
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")
```



Coding the 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 (M1)",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model (M0)	1.0612018
Movie effect model (M1)	0.9439087

Basically we have repeated the process for M0 but improving its capabilities and as we can see, we have obtained a lower RMSE, so a better prediction.

However, we can keep improving.

2.2.3 III. Movie and user effect model (M2)

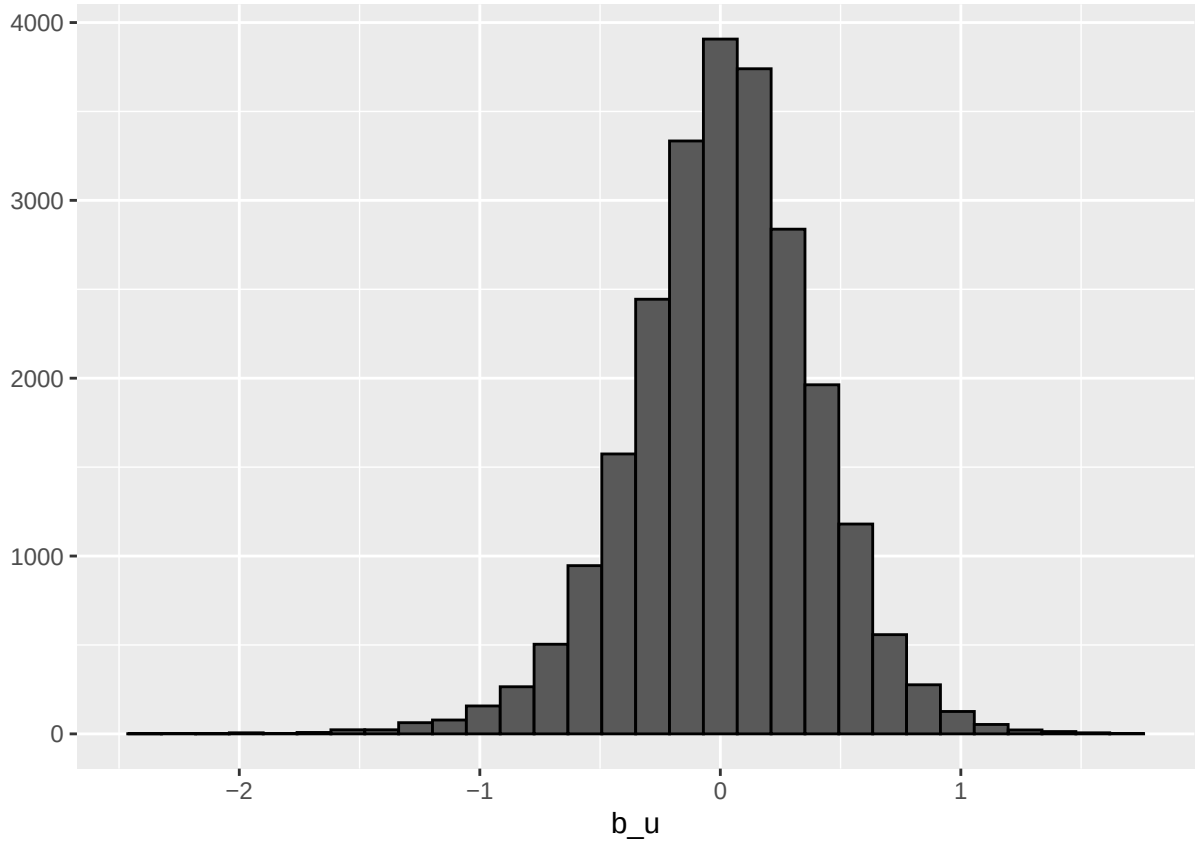
In this step we are going to add the user effect. The idea is very similar to movie effect: Some users love every movie and some others hate every movie. Between them there exists a vast amount of users that have different opinions on different movies, our objective is to calculate this effect. In fact, to successfully apply a recommendation system this may be the most important part, as our objective is to propose personal suggestions to every user. To avoid unrealistic predictions we avoid users who have rated less than 100 movies.

Our model will look like this:

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

Where b_u is represents user effect.

```
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"))
```



We compute the user effect as the average of

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

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

Now we will calculate the RMSE and see how our model 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 + user effect model (M2)",
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model (M0)	1.0612018
Movie effect model (M1)	0.9439087
Movie + user effect model (M2)	0.8653488

Even though we have improved a lot our model and the RMSE has gone down, we have a problem: low representative films have higher ratings that expected. It is not correct that a movie rated only once with 5 stars has a 5 stars rating prediction, if it was so good that movie should have been watched more times. To correct this mistake we are going to regularize: Impose a penalty to unpopular (in terms of how many times were rated) movies so they do not influence the model. That is the next step.

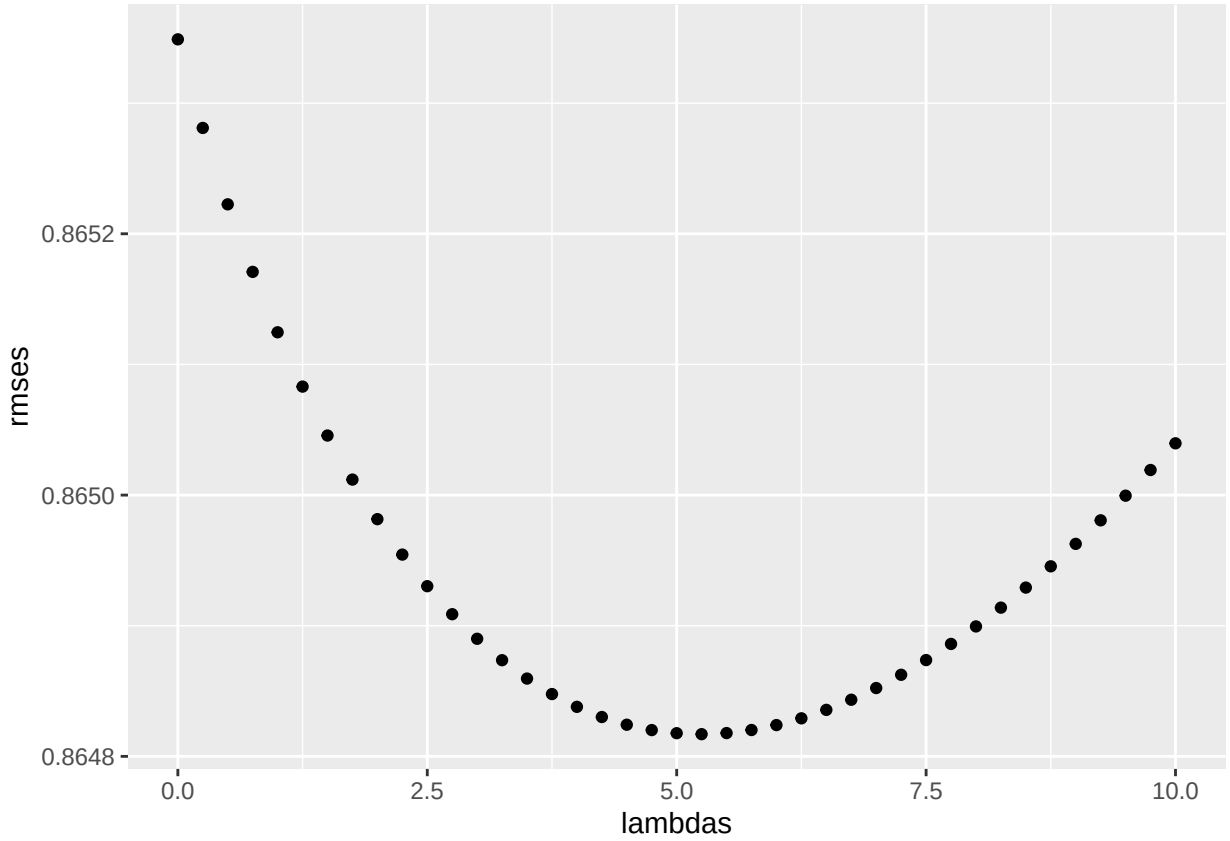
2.2.4 IV. Regularized movie and user effect model (M3)

With this code, we are looking for the lambda value that optimizes our model and regularize predictions.

```
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))
})
```

The optimal lambda is near 5.0.

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



More precisely:

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

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

The optimal lambda is: 5.25

So our final model is this one below:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model (M3)",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model (M0)	1.0612018
Movie effect model (M1)	0.9439087
Movie + user effect model (M2)	0.8653488
Regularized movie and user effect model (M3)	0.8648170

3 Results

Down here we have a brief summary of our algorithms

method	RMSE
Average movie rating model (M0)	1.0612018
Movie effect model (M1)	0.9439087
Movie + user effect model (M2)	0.8653488
Regularized movie and user effect model (M3)	0.8648170

We achieved our lowest RMSE with a value of 0.8648170 using our last model (M3):

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

4 Conclusion

We have improved our model from the simplest model (M0), to the last, regularized model (M3) by about 25%. We could have had even better results including variables such as genre, year of the film, age of the user, duration... However this requires more computational power.