Movielens Capstone Project

Pierre FLINE

27/01/2022

Contents

1 Introduction

This report is produced as one of the final reports to complete the *Data Science Professional Certificate*, an online program provided by Harvard University through edx platform.

The assignment is to predict the rating of a set of movies by a set of users with the best possible accuracy. The accuracy is here measured as the Residual Mean Squared Error (RMSE), which we can calculate this way:

$$RMSE = \sqrt{\frac{1}{N}\sum^{N}(y - y')^2}$$

where:
`y`: real rating (outcome)
`y'`: predicted rating (prediction)
N: number of ratings

Two data frames are provided:
- an **edx set**: a set of known data for which we can consider that we know the outcomes and the features, and on which we a required to build a prediction algorithm.
- a **validation set**: a set of data on which we should predict the outcome, given the features. The RMSE on this set should be as low as possible.

The goal is to get a RMSE<0.86490.

2 Data creation

The original data set is downloaded from the website of the GroupLens research lab.
It is randomly split between :
- an **edx set** (90% of the observations), which will be used to create our prediction model
- a **validation set** (10% of the observations), which will be used only to assess the performance of our prediction model

We make sure that the validation set does not include any user or movie that do not appear in the edx set.

The edx set is randomly split into two sets:
- a train set: **train_set** (50% of the observations)
- a test set: **test_set** (50% of the observations)

We make sure that the test_set does not include any user or movie that do not appear in the train_set.

3 Data exploration : edx data frame

Let's explore the structure of the **edx set**.

Classes of the variables:

```
##      userId     movieId      rating   timestamp       title      genres
##   "integer"   "numeric"   "numeric"   "integer" "character" "character"
```

First values of the variables:

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                            genres
## 1:                Comedy|Romance
## 2:          Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

Number of unique users (userId):

```
## [1] 69878
```

Number of unique movies (movieId):

```
## [1] 10677
```

Timestamp distribution key values:

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 7.90e+08 9.47e+08 1.04e+09 1.03e+09 1.13e+09 1.23e+09
```

Number of unique titles:

```
## [1] 10676
```

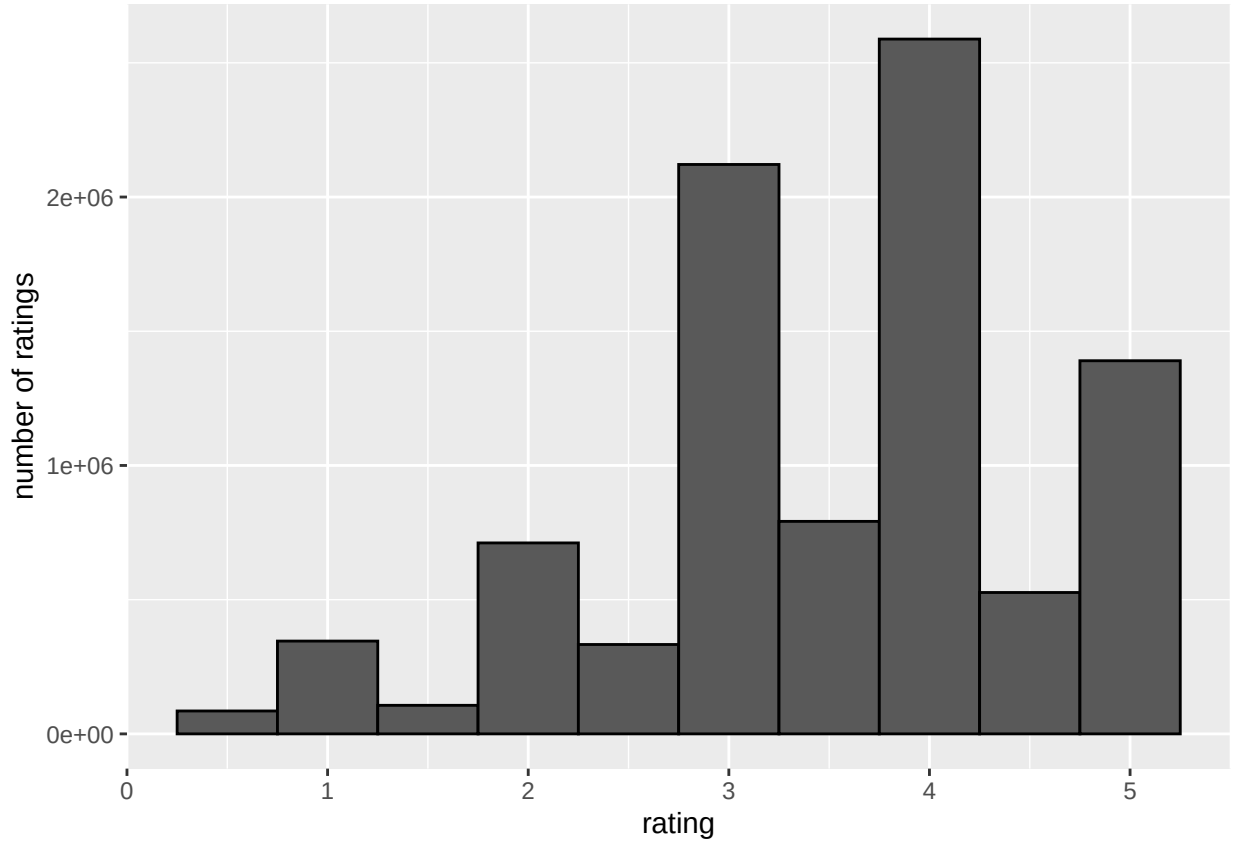
Number of unique genres (including combinations):

```
## [1] 797
```

Note that the "genres" variable in the edx set is often a combination of various genres. For example there are two observations called "Crime|Drama|Thriller" and "Comedy|Drama", both include "Drama".

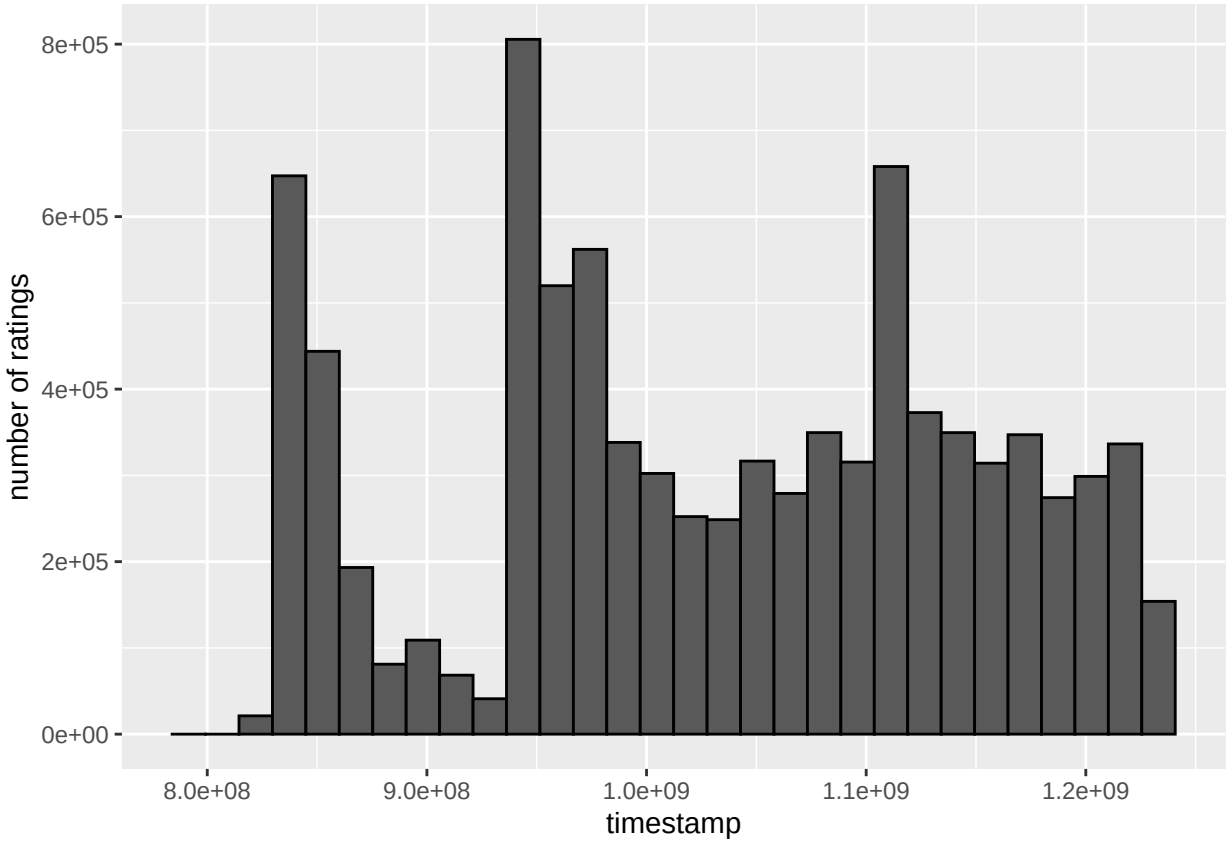
There are no NAs in the edx data frame, for any of the variables.

Ratings distribution:

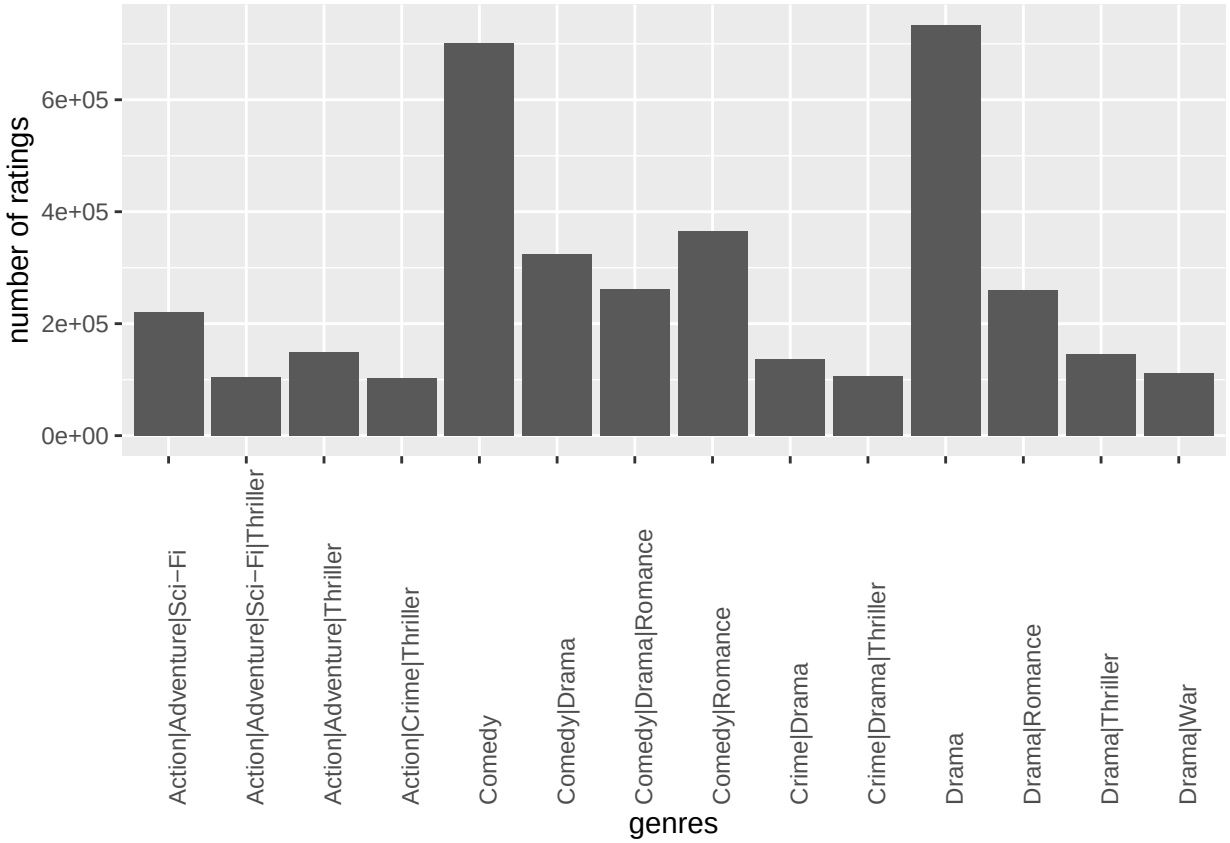


Half ratings tend to be less frequent than full ratings.

Timestamp distribution:



Genres distribution:
As there are 797 genres combinations, we filter those with more than 100 000 ratings only.
Here is the distribution:



4 Modelisation

In this part we will build various models, trained on the **train_set** and tested on the **test_set**.
The letter y is used to define the real ratings (outcomes) of a given data set.

4.1 The mean only

The distribution of the ratings showed a correlation between the ratings and the number of ratings.

Say μ is the ratings mean of the **train_set**.
Value of μ:

```
## [1] 3.5125
```

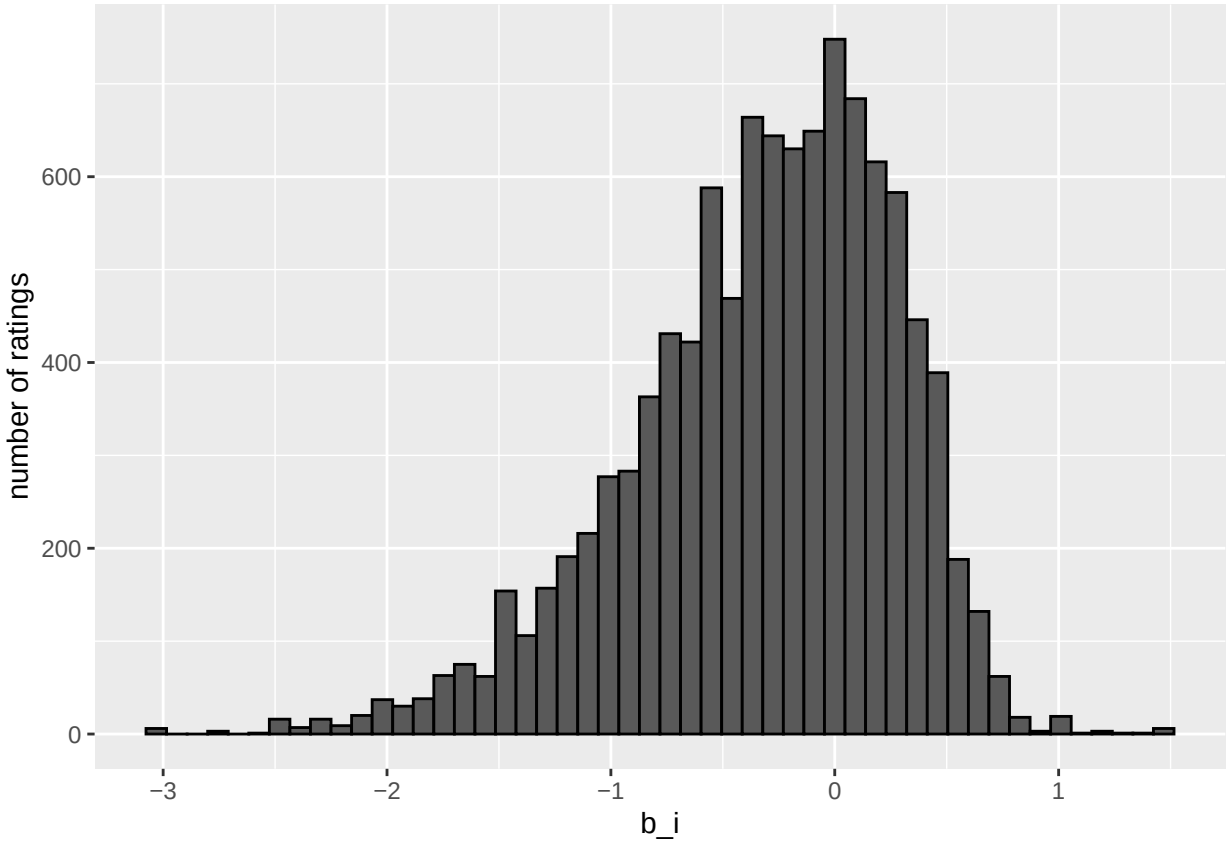
Assume each user will grade each movie as μ.
Here is the resulting RMSE on the **test_set**:

Method	RMSE
The mean only	1.06041

4.2 The mean + movie effect

Say b_i is the movie effect.
b_i is the mean of "y-μ" for each movie (movieId).

Distribution of b_i:



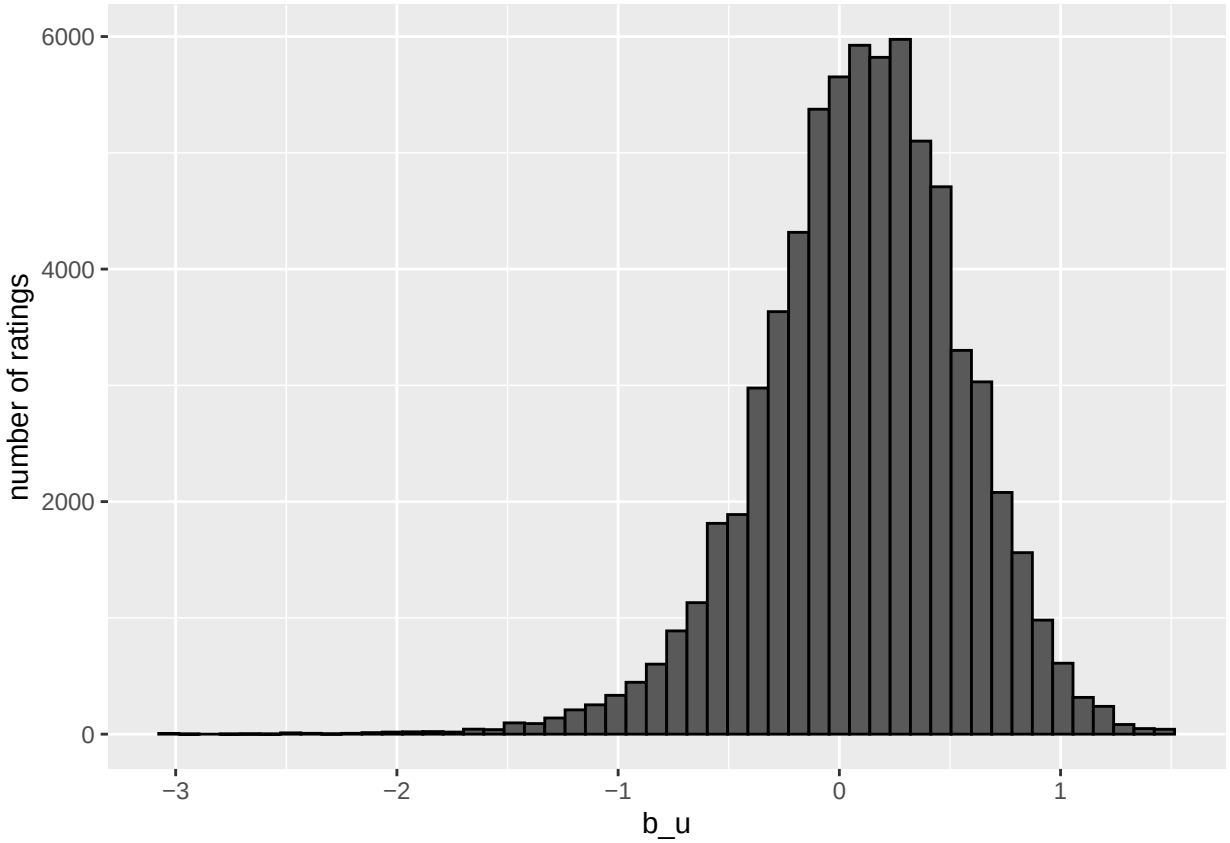
Assume each user will grade each movie as "μ+b_i".
Here is the resulting RMSE on the **test_set**:

Method	RMSE
Mean + movie effect	0.94413

4.3 The mean + user effect

Say b_u is the user effect.
b_u is the mean of "y-μ" for each user (userId).

Distribution of b_u:



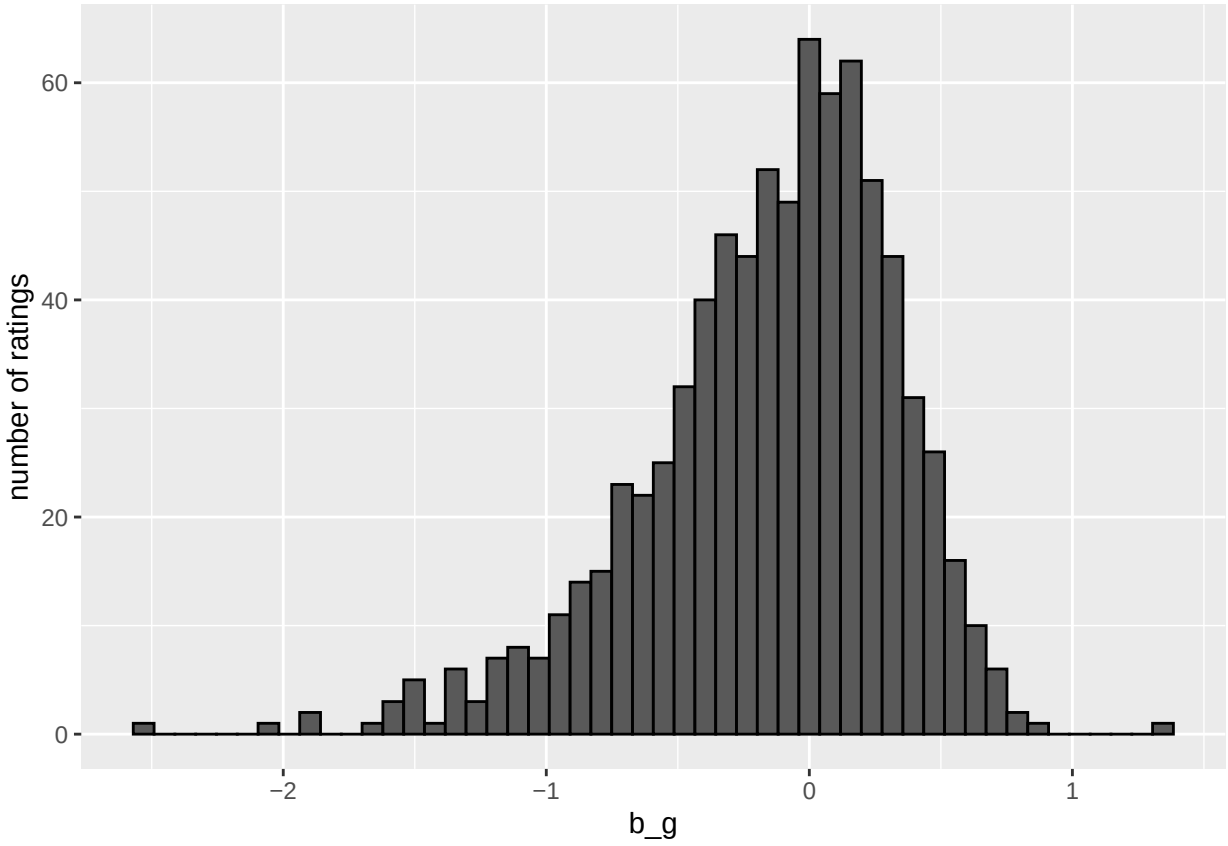
Assume each user will grade each movie as "μ+b_u".
Here is the resulting RMSE on the **test_set**:

Method	RMSE
Mean + user effect	0.98209

4.4 The mean + genre effect

Say b_g is the genre effect.
b_g is the mean of "y-μ" for each genre (genres).

Distribution of b_g:



Assume each user will grade each movie as "μ+b_g".
Here is the resulting RMSE on the **test_set**:

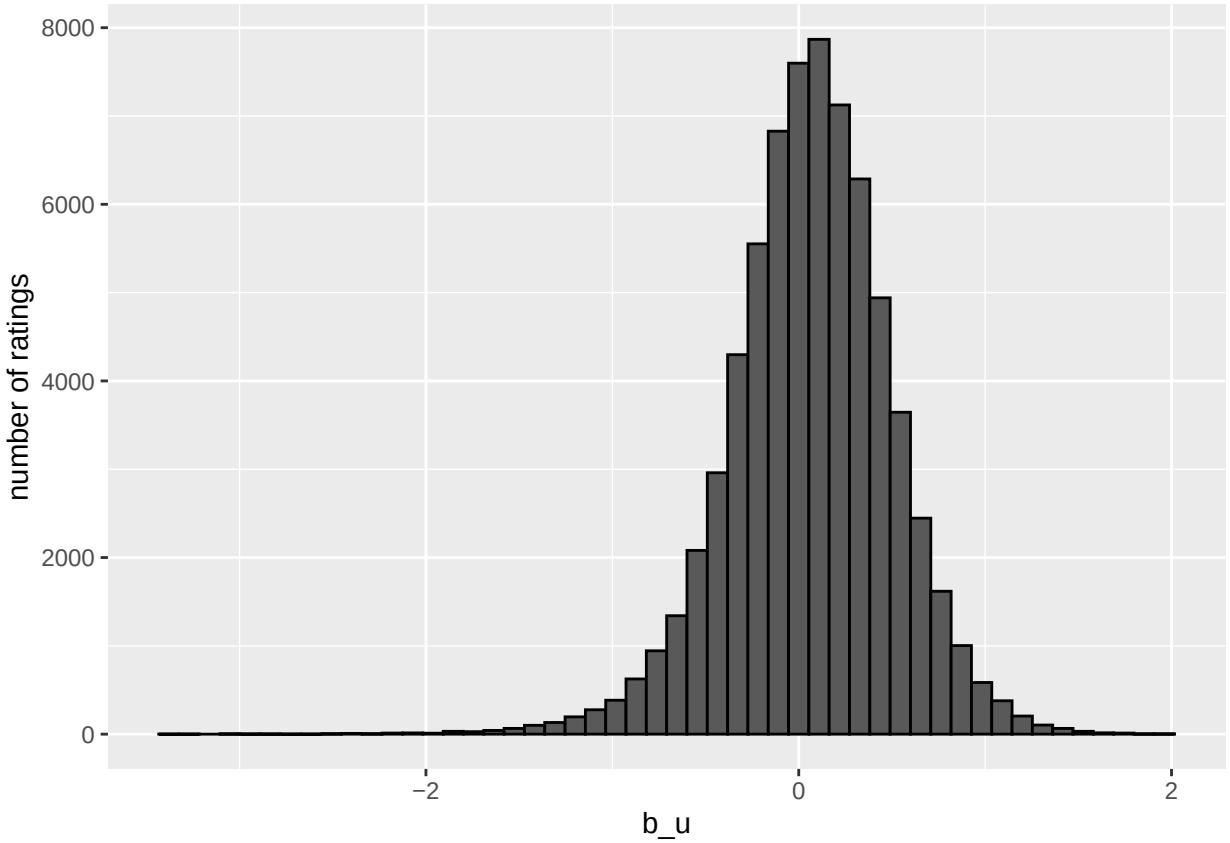
Method	RMSE
Mean + genre effect	1.01816

4.5 The mean + movie effect + user effect

Say b_i is the movie effect.
b_i is the mean of "y-μ" for each movie (movieId).

Say b_u is the user effect.
b_u is the mean of "y-μ-b_i" for each pair of movie/user (movieId/userId).

Distribution of b_u:



Assume each user will grade each movie as "μ+b_i+b_u".
Here is the resulting RMSE on the **test_set**:

Method	RMSE
Mean + movie effect + user effect	0.86967

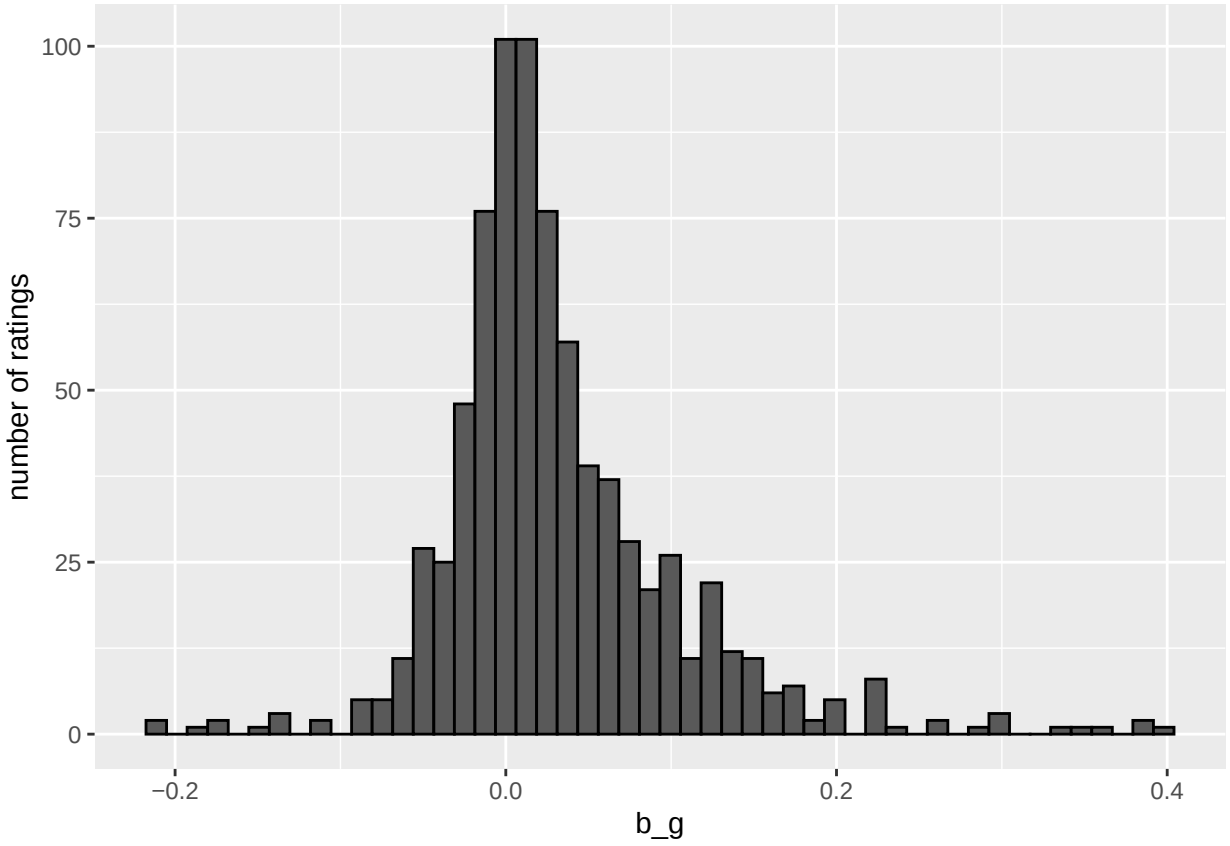
4.6 The mean + movie effect + user effect + genres effect

Say b_i is the movie effect.
b_i is the mean of "y-μ" for each movie (movieId).

Say b_u is the user effect.
b_u is the mean of "y-μ-b_i" for each pair of movie/user (movieId/userId).

Say b_g is the genre effect.
b_g is the mean of "y-μ-b_i-b_u" for each trio of movie/user/genre (movieId/userId/genres).

Distribution of b_g:



Assume each user will grade each movie as "μ+b_i+b_u+b_g".
Here is the resulting RMSE on the **test_set**:

Method	RMSE
Mean + movie effect + user effect + genre effect	0.86932

4.7 The mean + movie effect + user effect with regularization

The idea of regularization is to penalize users and/or movies with few ratings, as they are supposed to be less accurate than users and movies with lots of ratings.

For this we define two parameters, λ_u and λ_i.

Say b_i is the movie effect.
b_i is defined for each movie (movieId) as follows:

$$b_i = \sum \frac{y - \mu}{n + \lambda_i}$$

where n is the number of ratings for each movie (movieId).

Say b_u is the user effect.
b_u is is defined for each pair of movie/user (movieId/userId) as follows:

$$b_u = \sum \frac{y - \mu - b_i}{n + \lambda_u}$$

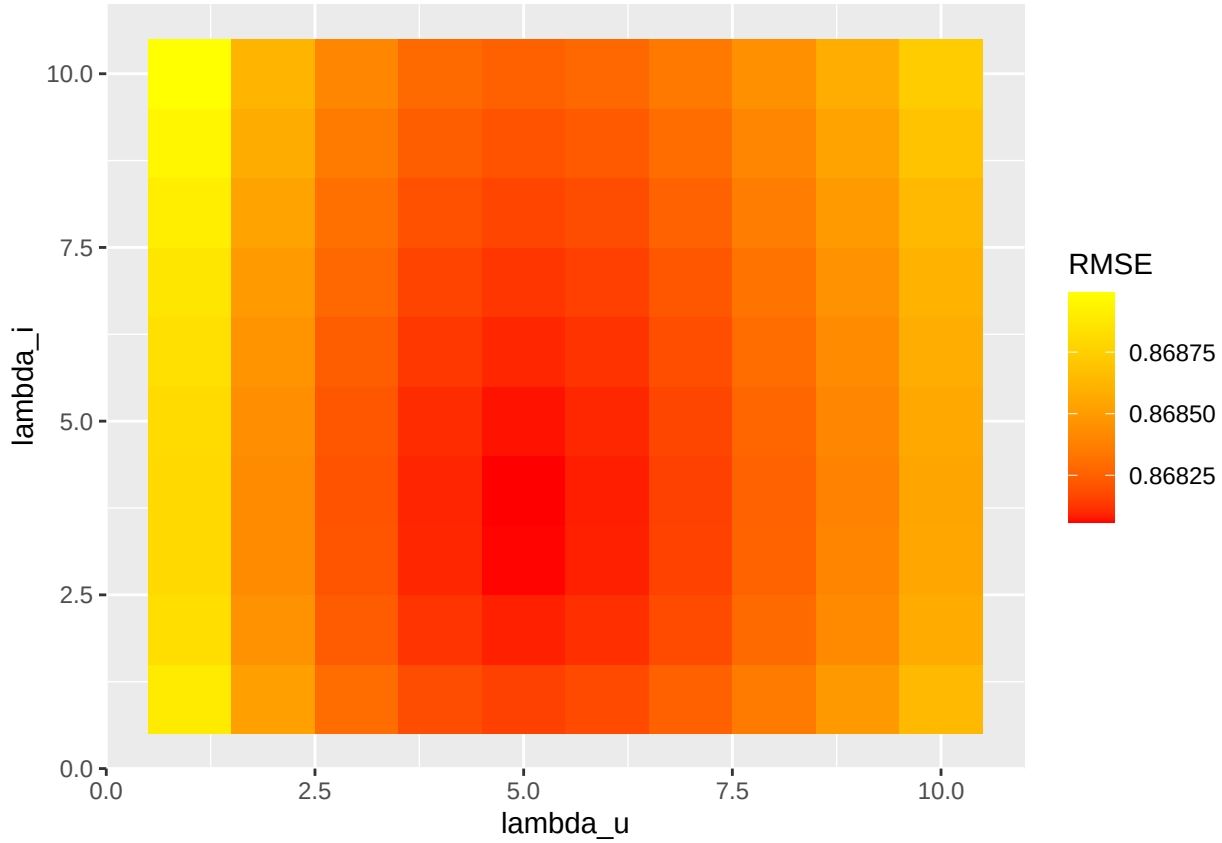
where n is the number of ratings for each pair of movie/user (movieId/userId).

Assume each user will grade each movie as "μ+b_i+b_u".
We evaluate the RMSE for λ_u and λ_i from 1 to 10, with steps of 1.

Here is the lowest RMSE obtained:

Method	RMSE
Mean + user effect regularized + movie effect regularized	0.86806

Plot of the RMSE for the various λ_u and λ_i



The lowest RMSE is obtained for the following λ_u and λ_i :

lambda_u	lambda_i
5	4

4.8 The mean + movie effect + user effect + genre effect with regularization

We perform fairly the same calculation as the previous part.

We define three parameters, λ_u, λ_i, and λ_g.

We define b_i and b_u as in the previous part.

Say b_g is the genre effect.
b_g is defined for each trio of movie/user/genre (movieId/userId/genres) as follows:

$$b_g = \sum \frac{y - \mu - b_i - b_u}{n + \lambda_g}$$

where n is the number of ratings for each trio of movie/user/genre (movieId/userId/genres).
Assume each user will grade each movie as "μ+b_i+b_u+b_g".

Unfortunately we could not perform this calculation because of a lack of computer resources.

Method	RMSE
Mean + user effect regularized + movie effect regularized + genre effect regularized	NA

4.9 Matrix factorization

Another way to build our model is to do a matrix factorization, i.e. to factorize the matrix of ratings for each pair of user/movie into two matrices.

Various packages of matrix factorization are proposed: recosystem, recommenderlab, MatrixExtra, cmfrec... The recosystem is used in this report as it seems to match with our need and as it seems to be fairly simple to use.

We don't set any parameter in the $tune() function of this package and keep the default ones.

Here is the RMSE obtained:

Method	RMSE
Matrix Factorization	0.83933

5 Results

Here is a summary of the models we used and the resulting RMSE on the **test_set**:

Method	RMSE on the test_set
The mean only	1.06041
Mean + movie effect	0.94413
Mean + user effect	0.98209
Mean + genre effect	1.01816
Mean + movie effect + user effect	0.86967
Mean + movie effect + user effect + genre effect	0.86932
Mean + user effect regularized + movie effect regularized	0.86806
Mean + user effect regularized + movie effect regularized + genre effect regularized	NA
Matrix Factorization	0.83933

The best model is the one with matrix factorization.

We use the matrix factorization model to predict the results on the validation set.

Here is the resulting RMSE on the **validation set**:

Method	RMSE on the validation set
Matrix Factorization	0.8393

6 Conclusion

We reached the goal to get a RMSE<0.86490 with a Matrix Factorisation system provided by the recosystem package.

For further improvements, we could:
- explore various parameters in the $tune() function
- explore other matrix factorization packages such as recommenderlab, MatrixExtra, cmfrec...
- consider genres data more in detail
- consider timestamp data
- consider the fact that half stars are less rated than full stars