Movielens Project

Ricardo Morales

2024-06-12

1. Introduction

With the code provided, we get the **edx** data set, which contains information about the ratings of different movies, featuring also additional information such as the movie genre, title and id for both the movie rated and the user who assigned the rating. Thus, this data set has **6** variables and slightly more than **9 million rows**. We also get a data set named **final_holdout_test** which we will only use to test our model.

Our goal is to develop an **algorithm** for predicting the movie ratings assigned by the many users considered in the data set. In order to do that, we take as our main reference the approach covered in class (more precisely in **Section 6** when we learnt the basics of **Recommendation Systems**).

Therefore, we keep the following equation in mind when designing our algorithm:

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

In this equation,$Y_{u,i}$ denotes the rating given to the movie i by the user u, μ represents the average of all ratings, b_m is the *movie-specific effect* and b_u is the *user-specific effect*.

As prescribed, we will use only the **edx** data set to train our model (we will partition it in train and test sets, as recommended) and once we get confident enough with the results, we will apply the model estimated on the **final_holdout_test** data set.

2. Analysis

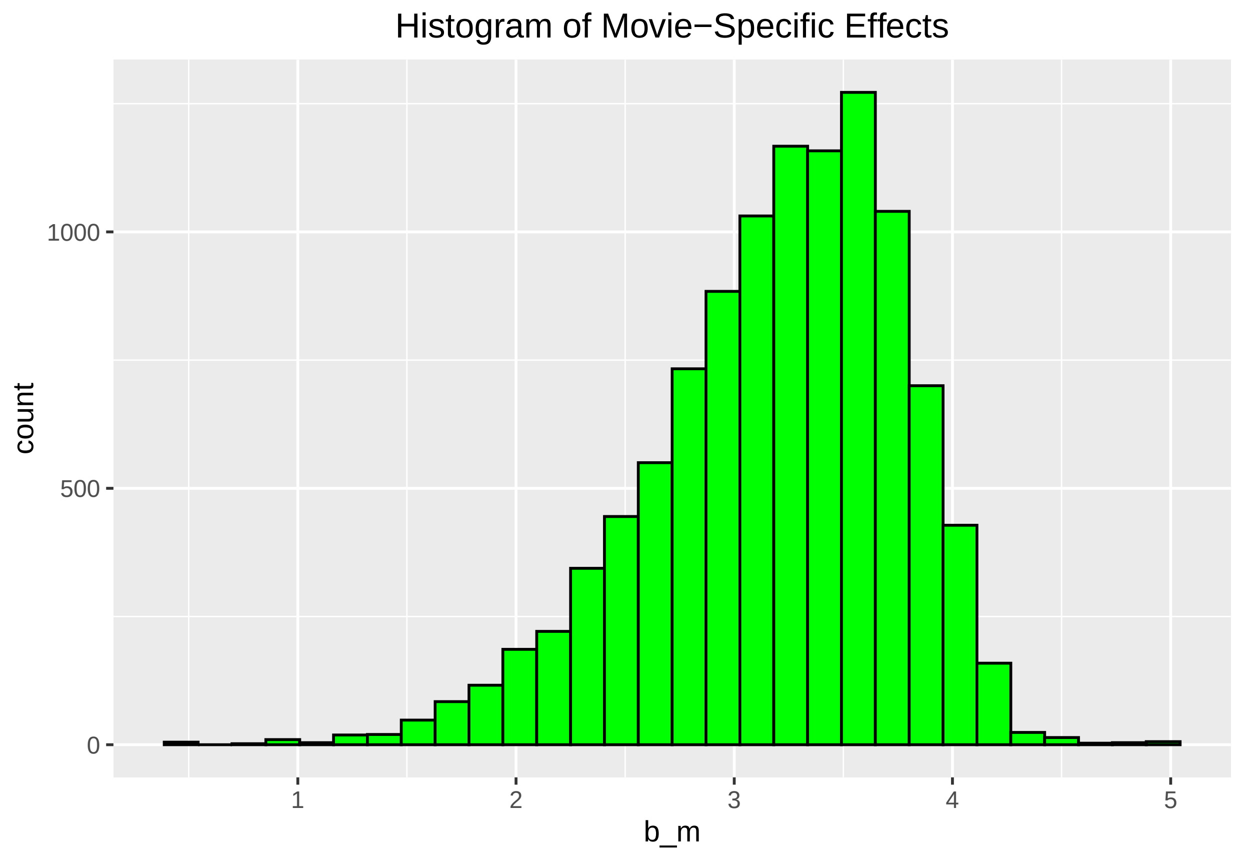
We start by partitioning the **edx** data frame into a train set (which we will use, as its name indicates, to train our algorithm) and a test set (which we will employ to make a preliminary test of the algorithm). We do this in the following way. Notice that, as usual, the test set (**edx_test**) represents 10% of the observations considered in the train set (**edx_train**).

Next, we calculate the average rating for all the movies included in the train set. We do this using the following code in R:

```r
mu <- mean(edx_train$rating)
mu
```

```
## [1] 3.512509
```

Then, we can check graphically that the ratings variability among movies and users included in the **edx** data set is significant and their distributions are skewed to the left.

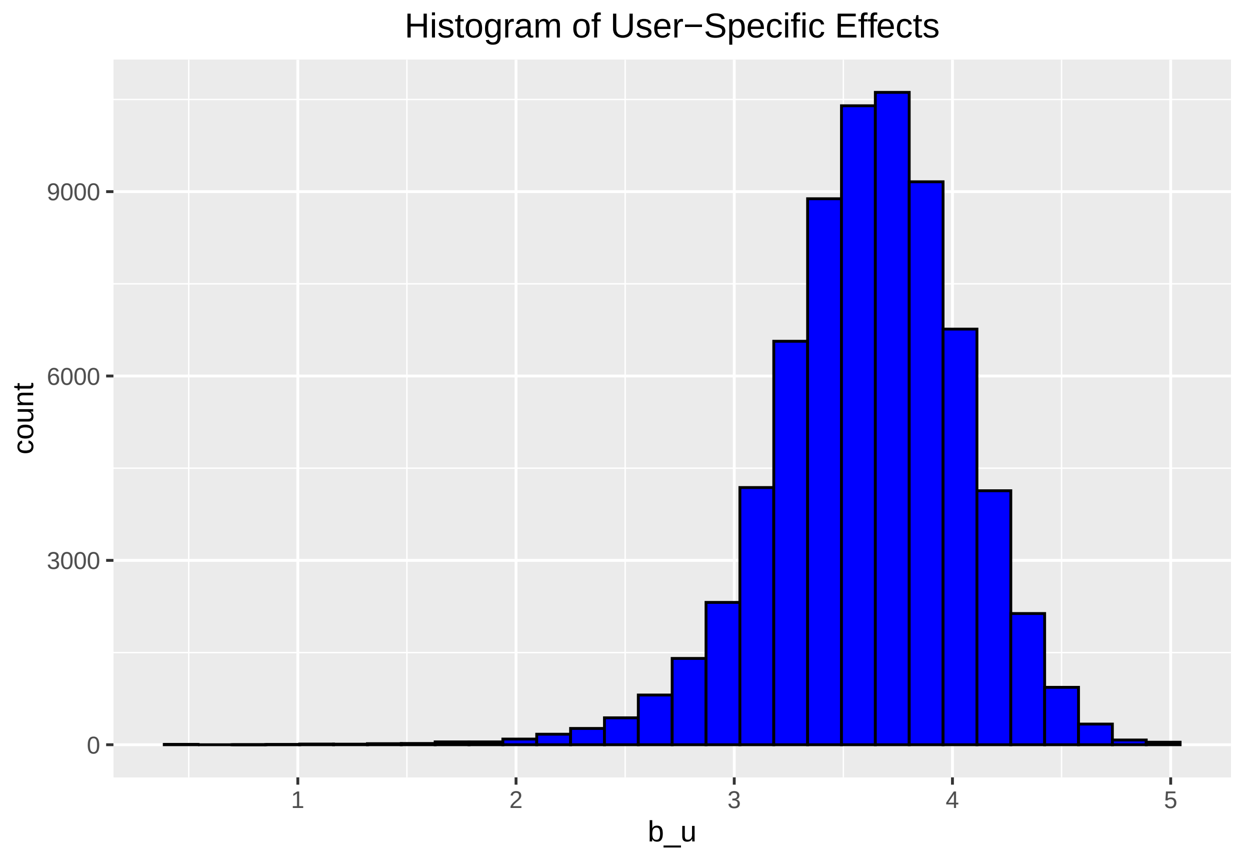


Then, we calculate the *movie-specific effects* (b_m). In order to do that, we must first group the data by movieId. Hence, we obtain these effects as the average of $Y_{u,i} - \mu$. The corresponding R code is:

```
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating-mu))
```

Next, we estimate the *user-specific effects* (b_u). These coefficients can be calculated as the average of $Y_{u,i} - \mu - b_m$. The corresponding R code is:

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



Finally, we obtain the predicted ratings according to the simple algorithm we have just developed based on the formula initially provided ($Y_{u,i} = \mu + b_m + b_u$). The respective piece of code in R is:

```
predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pr = mu + b_m + b_u) %>%
  pull(pr)
```

We run the following code in R and get a **Root-Mean-Square Error (RMSE)** equal to **0.865**.

```
model_rmse <- sqrt(mean((predicted_ratings-edx_test$rating)^2))
model_rmse
```

```
## [1] 0.8659736
```

Then, we can be confident enough to train the algorithm constructed on the entire **edx** data set and test it on the **final__holdout__test**. In other words, we replicate the procedure described before but using this larger data sets instead. The average movie rating is:

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

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

The movie-specific effects are calculated as follows:

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating-mu))
```

On the other hand, the user-specific effects are obtained as:

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

3. Results

Finally, we calculate the predicted ratings according to the algorithm developed but this time using the **final_holdout_test** data set (we do this by comparing the ratings predicted by the algorithm to the values stored in the column *rating* of the aforementioned data set).

```
predicted_ratings <- final_holdout_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pr = mu + b_m + b_u) %>%
  pull(pr)
```

Hence, we calculate the corresponding **RMSE**. We find it takes a value of **0.865**.

```
model_rmse <- sqrt(mean((predicted_ratings-final_holdout_test$rating)^2))
model_rmse
```

```
## [1] 0.8653488
```

4. Conclusion

To sum up, we are able to get a **RMSE** lower than 0.90 after constructing a simple algorithm that accounts for movie-specific and user-specific events. These effects were added to the average rating calculated on the whole **edx** data set. We could have improved our model by using techniques such as **Matrix Factorization** or **Singular Value Decomposition**, which are capable of identifying further patterns in the data.