Movielens Project

Sari Vesiluoma

1 11 2019

Introduction

The purpose of this project is to introduce a movie recommender based on the Movielens 10M data. The data preparation was done based on the code given in the exercice introduction. Ratings and movies were separately excerpted from the Movielens data, then combined. Then movielens data was split into two groups, one data frame called edx for training data and one data frame called validation for validation data including 10 % of the movielens data. Validation data was created so, that it included only rows where movieId and userId were also existing in the training set (edx). Rows which were excluded from the validation set were added back to the training set.

The dimensions of the training data (edx) and validation data are

```
dim(edx)
```

```
## [1] 9000055       6
```

```
dim(validation)
```

```
## [1] 999999      6
```

Like will be shown in the Analysis chapter, the data did not need any further cleanig, so it was possible to start creating the recommendation model, a.k.a. starting to find a solution resulting reasonably small RMSE. This was done in three steps. First to assume a model where the recommendation is the mean over all ratings in the training set, secondly to notice also that different movies get different level of ratings, and finally, in addition, noticing that different users give different level of ratings for different movies.

Analysis

The summary of the training set clearly shows that there are no NA values, so no need for pre-processing those in this data. So, this data can be used as is.

```
summary(edx)
```

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

Rows in the data look like this

```
edx %>% head()
```

```
##   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
```

The amount of unique movies and users in the training data are

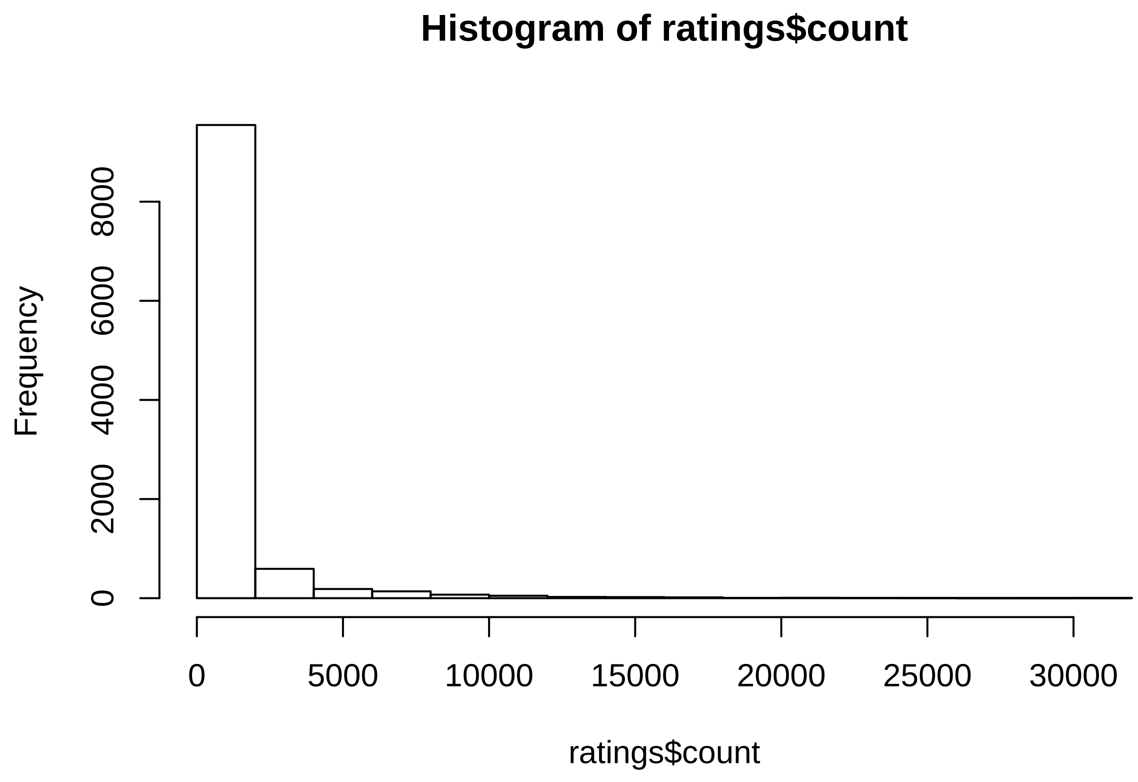
```
n_distinct(edx$movieId)
```

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

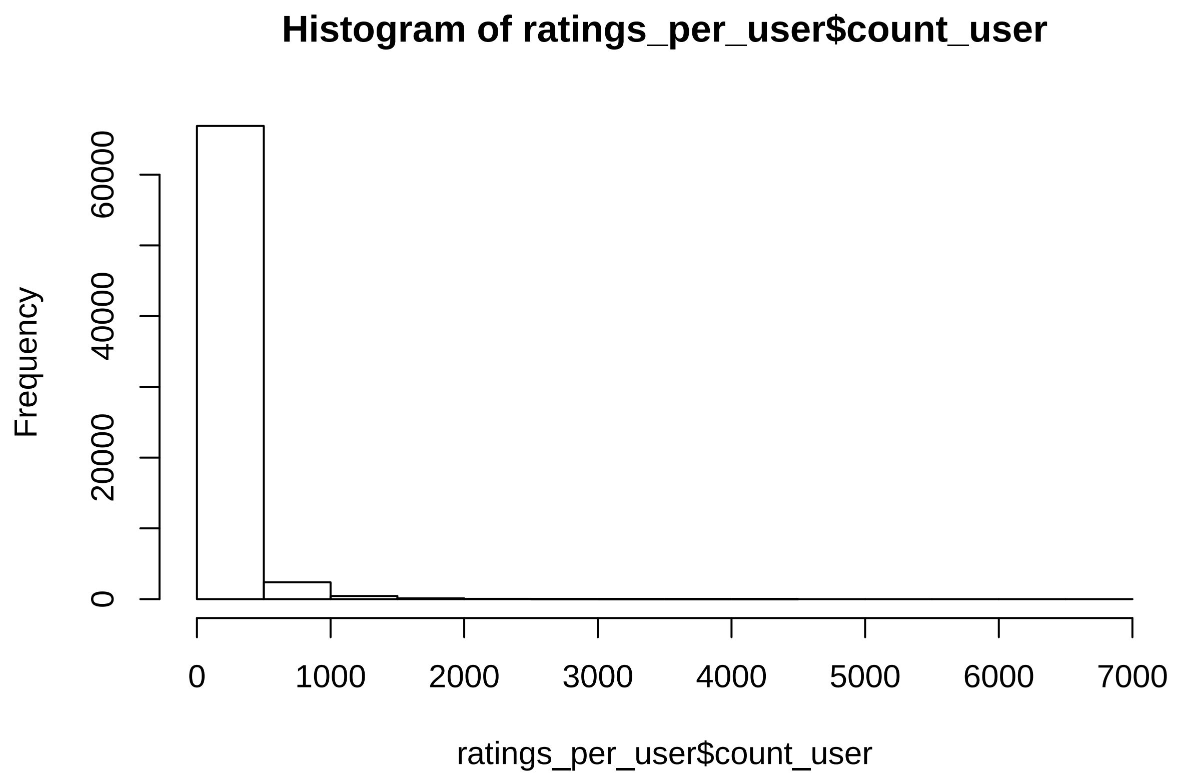
```
n_distinct(edx$userId)
```

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

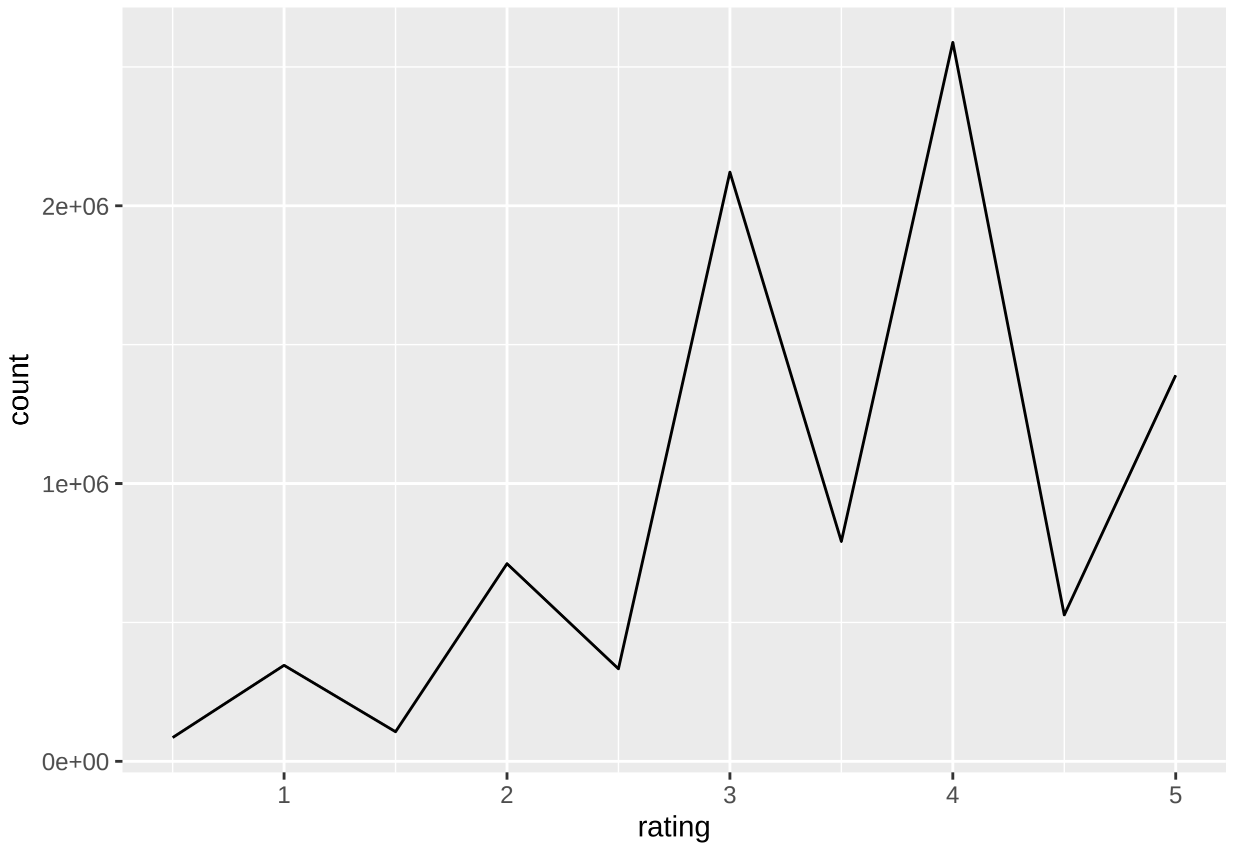
When using this data, it is important to recognize that the amount of ratings per movie vary hugely between the movies as shown in the histogram below.



Similarly, the amount of ratings per user vary really much as can be seen the next histogram.



Also, when looking the way people have used rating levels, it is clear that they have used integer ratings much more than the half (.5) ratings as can bee seen from the line chart below.



The idea here is to use some different machine learning models to test which one(s) might give the best result in recommending. The value of the result will be analyzed based on RMSE score of the algorithm.

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

To start with we will need a some kind of a baseline compared to which to try to improve the RMSE. For that purpose we will use the mean value of the ratings in the training set. The resulting RMSE value is

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

The first model to be used is to notice that some movies get higher rates than others. So, in addition to using the average, we'll notice movie specific deviation from the mean. When noticing this the value of RMSE is less than when using only the average value and it is:

```
## [1] 0.9439087
```

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

```
## Warning in bind_rows_(x, .id): binding factor and character vector,
## coercing into character vector
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Like some movies get higher rates than others, similarly different users rate very differently. Next, in addition to using the average and movie effect, we will also use the user effect. Like you see, the resulting RMSE improves to

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

Results

The results were achived following the model that had been introduced in the lectures. The results based on the three models used are:

```
rmse_results
```

```
##                        method      RMSE
## 1            Based on average 1.0612018
## 2          Movie effect model 0.9439087
## 3 Movie + user effect model 0.8653488
```

The comparing value was created based on using the rating average only. The next value was created based on noticing that different movies are rated differently. That clearly improved the result. Finally, also it was noticed that different users rate differently, so including also the user effect resulted RMSE being `model_2_rmse`.

Conclusion

When recommending movies, it is clear, also based on this project, that in addition to an average as the basis for a recommendation, we will need to notice also that different movies are rated differently (movie effect) and that different users give rates differently (user effect). Includin both of these, the movie effect and the user effect increases the "goodness" of the recommendation remarkably.

The main limitation in this project was that only a small amount of data (10 M Movielens data) was used not to get the home laptops stuck. So, to be able to create even more convincing recommendations, more training data would be beneficial. In addition, some other algorithms could also be utilized.

As such, this was an interesting project and made me understand much better what we were doing during the lectures and why to do these this way.