MovieLens_Report

Marta Tenconi

14 May 2019

1. Introduction

This report is about the construction of a movie recommender system: taste profiles are evaluated in order to make specific recommendations to users, using already rated movies. The original data can be obtained from the GroupLens webpage: https://grouplens.org/datasets/movielens/ (https://grouplens.org/datasets/movielens/).

The following libraries and files are used for the project:
- tydiverse package
- caret package
- MovieLens 10M dataset (ml-10m)

The datasets:

The MovieLens 10M dataset (ml-10m) here used has 10000054 million ratings and 95,580 tag applications applied to 10,681 movies by 71,567 users. Users were selected at random. All users had rated at least 20 movies. Original data are contained in the following files: movies.dat and ratings.dat. (https://grouplens.org/datasets/movielens/ (https://grouplens.org/datasets/movielens/))
- the file *ratings.dat* contains all ratings. Each line of this file represents one rating applied to one movie by one user, and has the following format: UserID::MovieID::Rating::Timestamp
- the file *movies.dat* contains movie information. Each line of this file represents one movie, and has the following format: MovieID::Title::Genres

These two files are grouped in a data frame, named movielens, with 10000054 observations of 6 variables (UserID, MovieID, Rating, Timestamp, Title, Genres).
Then, the dataset 'movielens' is split into test and validation sets (*edx* = test set, *validation* = validation set)

Aim of the project:

To build a movie recommender system, using data available on the GroupLens webpage: https://grouplens.org/datasets/movielens/ (https://grouplens.org/datasets/movielens/).

Key steps:

First step is to download the data sets and to do data wrangling.
After it, we can start with the analysis: three models (A, B, C) for movie recommendation are built and for each model movie ratings are predicted.
Then, the models are evaluated by calculating the error loss, using the residual mean squared error (RMSE).
At the end the RMSEs obtained for the three models are compared. The model that provides the minor error (minor RMSE) is the best recommendation system.

2. Analysis:

For the following analysis consider y(u,i) = rating for movie i by user u; and y_hat = predicted rating.

a) Data sets download and data wrangling.

Datasets and libraries download:

The following libraries and files are used for the project:
- tydiverse package
- caret package
- MovieLens 10M dataset (ml-10m)

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------
----------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------------
----------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
```

Datasets generation and wrangling:

The downloaded files (ratings.dat and movies.dat) are grouped in one data frame called *movielens*

The data frame *movielens* is split in a test set and a validation set: (here *edx* = test set, *validation* = validation set).
The validation set is 10% of MovieLens data.
userId and movieId in validation set are also in edx set.
Rows removed from validation set are added back into edx set.

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

Test and validation set details:

Both *edx* and *validation* are object of class data.frame composed by 6 variables:
- UserID: unique ID for the user.
- MovieID: unique ID for the movie.
- Rating: a rating between 0 and 5 for the movie.
- Timestamp: date and time the rating was given, expressed as seconds since midnight UTC of January 1, 1970.
- Title: movie title.
- Genres: genres associated with the movie (Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western)
edx:

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 8
38984885 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (199
4)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller"
"Action|Adventure|Sci-Fi" ...
```

validation:

```
## 'data.frame':    999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494
1133571200 844416936 844417070 ...
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob
Roy (1995)" ...
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|D
rama|Romance|War" ...
```

edx and validation sets in tidy format:
Each row represents a rating given by one user to 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)
##                          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
```

```
##   userId movieId rating timestamp                 title
## 1      1     231      5 838983392  Dumb & Dumber (1994)
## 2      1     480      5 838983653  Jurassic Park (1993)
## 3      1     586      5 838984068     Home Alone (1990)
## 4      2     151      3 868246450        Rob Roy (1995)
## 5      2     858      2 868245645 Godfather, The (1972)
##                             genres
## 1                           Comedy
## 2 Action|Adventure|Sci-Fi|Thriller
## 3                  Children|Comedy
## 4         Action|Drama|Romance|War
## 5                      Crime|Drama
```

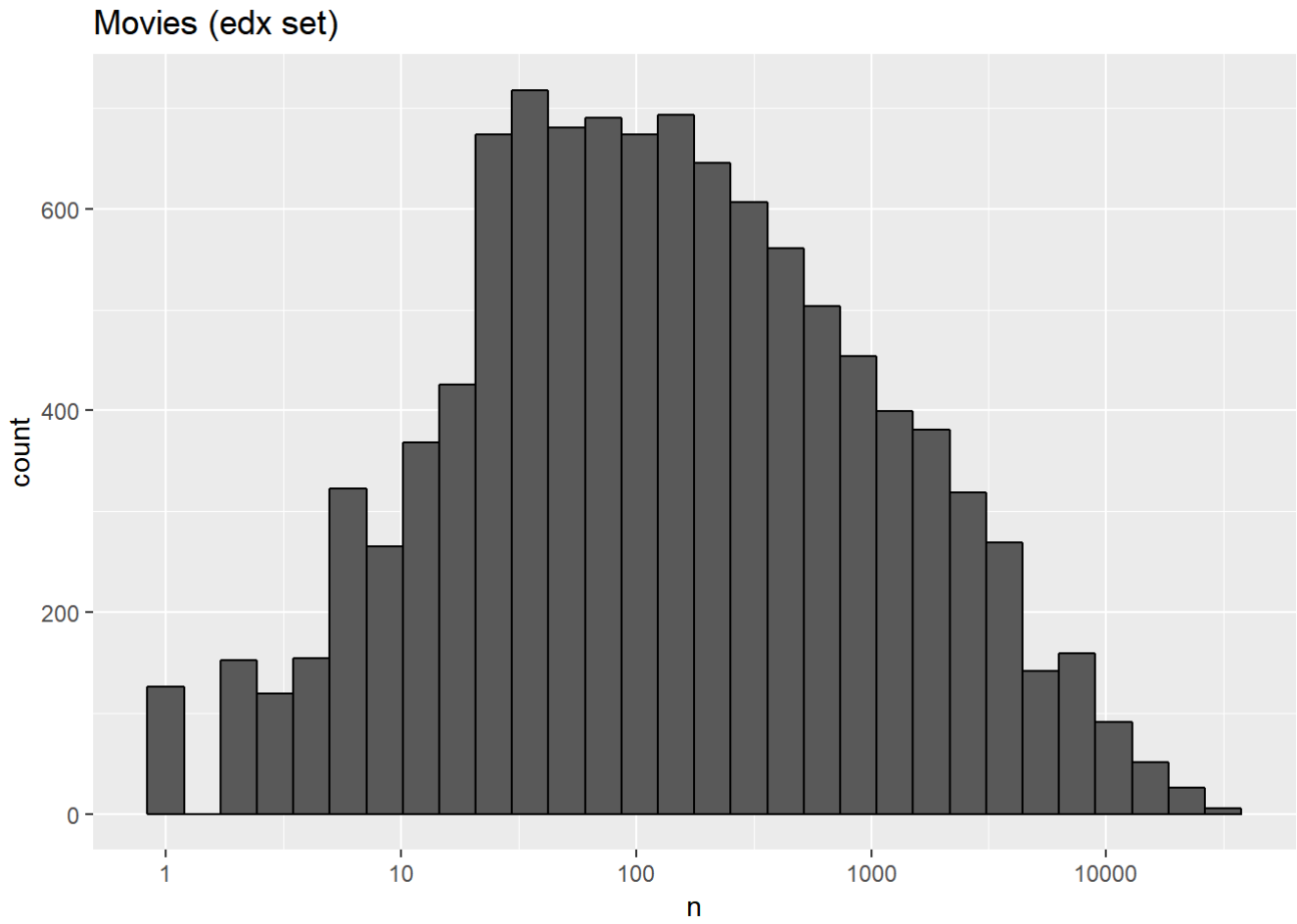
b) Construction of three models (A, B, C) for movie recommendation

Model A:

The same rating is predicted for all movies regardless of user: **y_hat_A = mu + error_u,i**
where the estimate that minimizes the RMSE is the least squares estimate of mu, corresponding to the average of all ratings:
y_hat_A = mean(edx$rating)

Model B:

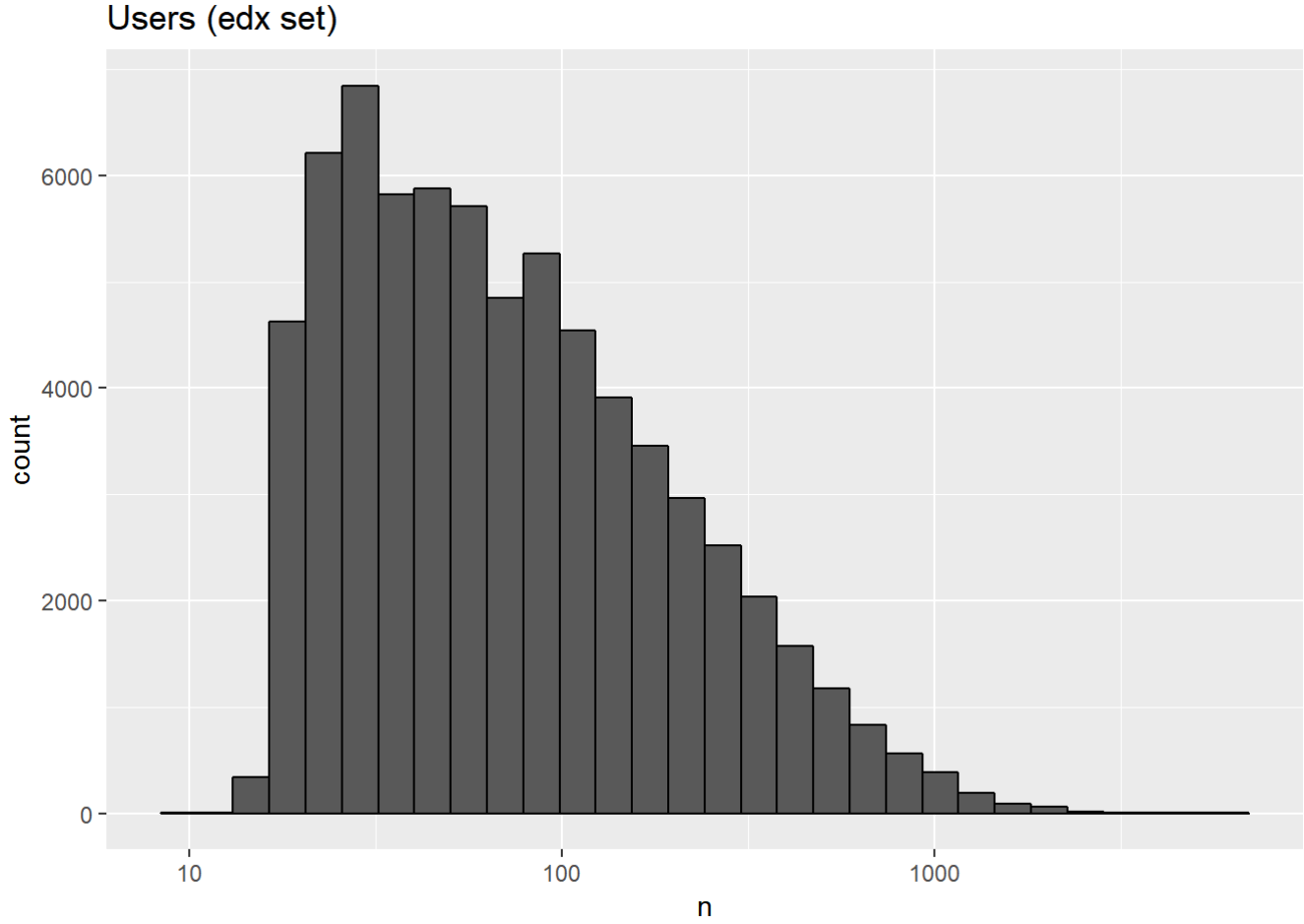
Model B takes in consideration that different movies are rated differently than others (some movies are generally rated higher or more often).



Therefore, I add a movie-specific effect (b_i) to the equation of Model A: **y_hat_B = mu + b_i + error_u,i**
Least square estimate b_i = average of (y_u,i - mu_hat) where:
y_u,i = rating;
mu = y_avg = average of all ratings.

Model C:

Model C take into consideration that some users are more active in rating than others.



Therefore, a user-specific effect (b_u) is added to the equation of Model B: **y_hat_C = (mu + b_u + b_i + error_u,i)**
b_u = the average of (y_u,i - mu_hat - b_i)

c) Residual mean squared error (RMSE):

The three models are evaluated by calculating the mean squared error (**RMSE**)

3. Results:

For the following analysis consider y(u,i) = rating for movie i by user u; and y_hat = predicted rating.

b) Construction of three models (A, B, C) for movie recommendation

Model A:

In model A the same rating is predicted for all movies regardless of user: **y_hat_A = mu + error_u,i**
where the estimate that minimizes the RMSE is the least squares estimate of mu, corresponding to the average of all ratings:

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

Model B:

Model B considers that different movies are rated differently than others (some movies are generally rated higher). Therefore, a movie-specific effect (b_i) is added to the equation of Model A: **y_hat_B = mu + b_i + error_u,i**
b_i = average of (y_u,i - mu_hat)
y_u,i = rating;
mu = y_avg = average of all ratings.

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

movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - y_avg))

y_hat_B <- y_avg + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i
```

Model C:

Model C takes into consideration that some users are more active in rating than others. Therefore, a user effect (b_u) is added to the equation of Model B: **y_hat_C = (mu + b_u + b_i + error_u,i)**
b_u = the average of (y_u,i - mu_hat - b_i)

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

y_hat_C <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = y_avg + b_i + b_u) %>%
  .$pred
```

c) Residual mean squared error (RMSE):

The three models are evaluated by calculating the mean squared errors (RMSE):

```
rmse_A
```

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

```
rmse_B
```

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

```
rmse_C
```

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

4. Conclusion

With the three approaches we get RMSE_A = 1.06, RMSE_B = 0.94, RMSE_C = 0.86.
The RMSEs decrease from approach A to C, and approach C, that applies both a user-specific and a movie-specific effect, gets the lowest RMSE = 0.86.
Therefore, **Model C is the best movie reccomnder model, with a RMSE = 0.86**.

```
rmse_C
```

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