HarvardX Data Science Professional Certificate
PH125.9x Capstone 1 - MovieLens Recommendation System

Hugo Aquino

2021-09-18

Contents

1 Introduction

Data is the *new fuel* of this era in which many situations can be tracked using records that came from several sources. These data need to be cleaned and organized in a way that with the support of tools such as **R** and new techniques, the data can talk to show patterns and relationships that allow us to anticipate and therefore take better decisions.

Data Science is emerging as one of the most important knowledge areas and the **data scientists** are becoming one of the best jobs paid, this role combines computing/programming skills with statistics to analyze raw data and transforming it for its use on an industry.

Companies such as Netflix, Amazon, Spotify among others use a recommendation system (kind of system used to recommend things based on several factors) as a way to identify the adequate product and these companies are accelerating its value proposition and increasing its market share through the use of machine learning and artificial intelligence algorithms.

Therefore and as a way to continue acquiring the skills needed to become a data scientist, this project will be focused on the creation of a recommendation system using the 10M version of the MovieLens dataset through the segmentation on **train** and **validation** sets using different algorithms.

In order to compare the different algorithms, the **root mean squared error (RMSE)** will be used as the loss function, so the target is to obtain a RMSE lower than **0.86490**.

The files required for this project (pdf, rmd, r) are hosted on github.

,

2 Methods

2.1 Libraries

The libraries used for this project are:

```
# Install libraries with its dependencies
if(!require(tidyverse)) install.packages(
  "tidyverse",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(caret)) install.packages(
  "caret",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(data.table)) install.packages(
"data.table",
repos = "http://cran.us.r-project.org",
dependencies = TRUE)
if(!require(ggplot2)) install.packages(
  "ggplot2",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(ggthemes)) install.packages(
  "ggthemes",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(lubridate)) install.packages(
  "lubridate",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(corrplot)) install.packages(
  "corrplot",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(recosystem)) install.packages(
  "recosystem",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(knitr)) install.packages(
  "knitr",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(kableExtra)) install.packages(
  "kableExtra",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(tinytex)) install.packages(
  "tinytex",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(scales)) install.packages(
  "scales",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
```

```r
if(!require(tidyr)) install.packages(
  "tidyr",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(lubridate)) install.packages(
  "lubridate",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(stringr)) install.packages(
  "stringr",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(corrplot)) install.packages(
  "corrplot",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)
if(!require(recosystem)) install.packages(
  "recosystem",
  repos = "http://cran.us.r-project.org",
  dependencies = TRUE)

# Load libraries
library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(ggthemes)
library(lubridate)
library(corrplot)
library(recosystem)
library(knitr)
library(kableExtra)
library(tinytex)
library(scales)
library(tidyr)
library(lubridate)
library(stringr)
library(corrplot)
library(recosystem)
```

2.2 Data

The initial **R file** provided by the **edx team** contains the code required to download, clean and prepare **2 datasets** for the project.

The first dataset was named as **edx**, that will be used to verify the different algorithms. This dataset has **9,000,055** observations and **6** columns. The predictor mean **rating** has a value of **3.51** and a standard deviation of **1.06**.

While the second dataset was named as **validation**, that will be used only with the algorithm that has the lowest **RMSE**. This dataset has **999,999** observations and **6** columns.

Both datasets have in total **10,000,054** observations.

The edx´s dataset characteristics are:

Table 1: MovieLens dataset characteristics

Column name	Type	Characteristic	Description
userId	integer	Discrete quantitative predictor	User unique identifier
movieId	numeric	Discrete quantitative predictor	Movie unique identifier
timestamp	integer	Discrete quantitative predictor	Date and time on epoch format
title	character	Nominal qualitative predictor	Movie title that is not unique
genres	character	Nominal qualitative predictor	Movie genre classification that is not unique
rating	numeric	Outcome and that is continuous	Movie rating from 0 to 5

The first edx´s records are:

Table 2: edx dataset first records

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

2.2.1 About users

On edx dataset there are **69,878** unique users and on average every user rates approximately **129** movies. There are **50,784** users *(73%)* that rated less than **129** movies and **18,960** *(27%)* users that rated more than **129** movies.

About the amount of movies reviewed per user, only **2** have evaluated more than **5,000 movies** each one, being user **59269** the top who has evaluated **6,616** movies.

Table 3: Movies reviewed per user

Movies reviewed per user	# users	%	avg_rating
Below 129	50,784	72.675	3.644431
Between 130 and 500	16,093	23.030	3.566326
Between 501 and 1,000	2,382	3.409	3.372811
Between 1,001 and 2,000	552	0.790	3.240551
Between 2,001 and 3,000	47	0.067	3.197522
Between 3,001 and 4,000	6	0.009	3.117101
Between 4,001 and 5,000	3	0.004	3.269139
More than 5,001	2	0.003	3.231153

An histogram that displays the amount of movies reviewed by users.

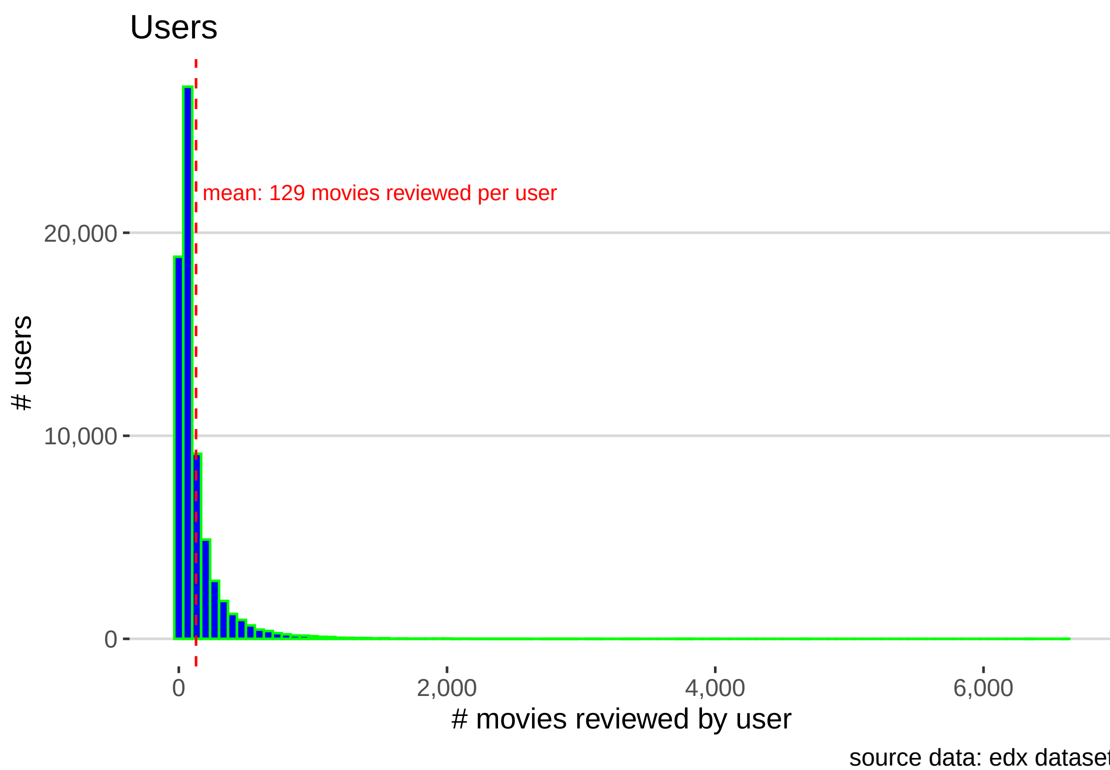

Figure 1: UserId histogram

2.2.2 About movies

There are **10,677** unique movies. **126** movies were evaluated only once by **82** users. **8,553** movies *(80%)* were rated less than **843** times by users and **2,122** *(20%)* movies were rated more than **843** times by users, being movie **296** the top which has been evaluated by **31,362** users. The **title** of this movie is **"Pulp Fiction (1994)"** and the **genres** are **"Comedy|Crime|Drama"**.

Table 4: Movies reviewed

Movies reviewed	# movies	%	avg_rating
Below 843	8,553	80.107	3.131255
Between 843 and 5,000	1,703	15.950	3.385821
Between 5,001 and 10,000	276	2.585	3.572059
Between 10,001 and 15,000	90	0.843	3.754865
Between 15,001 and 20,000	28	0.262	3.617725
Between 20,001 and 25,000	16	0.150	3.88607
Between 25,001 and 30,000	6	0.056	4.059805
More than 30,000	3	0.028	4.123904

An histogram that displays the number of movies reviewed by users (some of them are more rated than others) is:

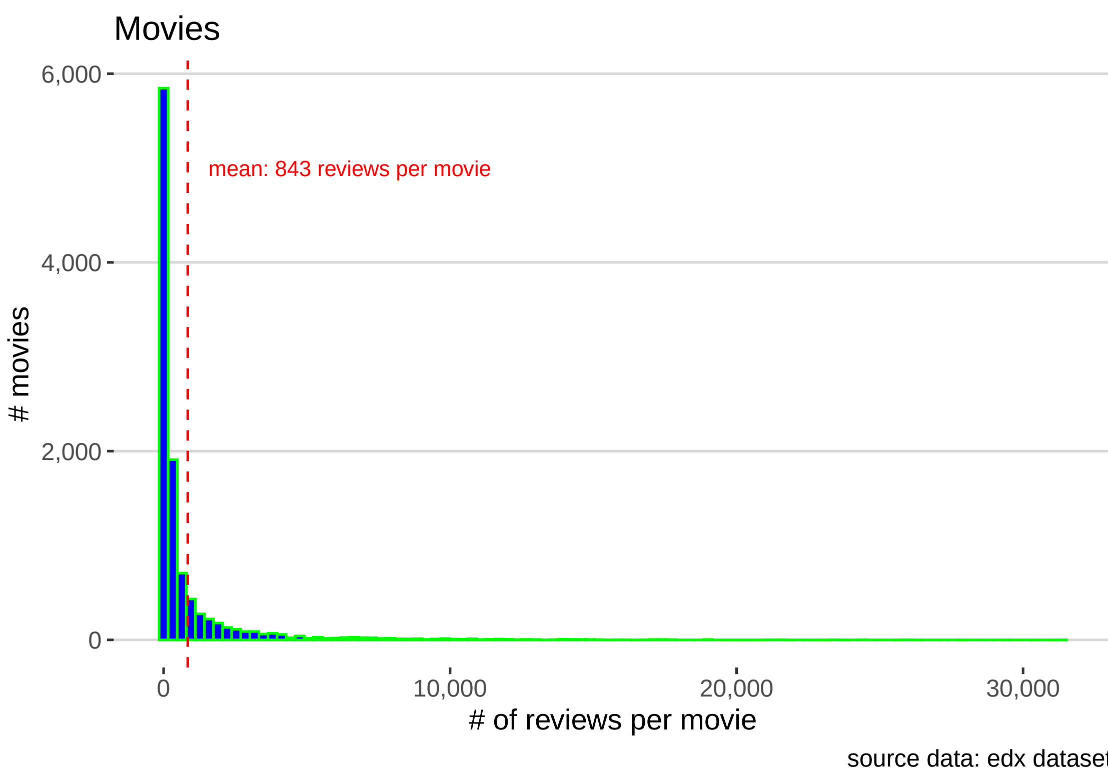

Figure 2: MovieId histogram

In order to *verify* how **sparse** is the relationship between users and movies, this graph take a sample of **500** users.

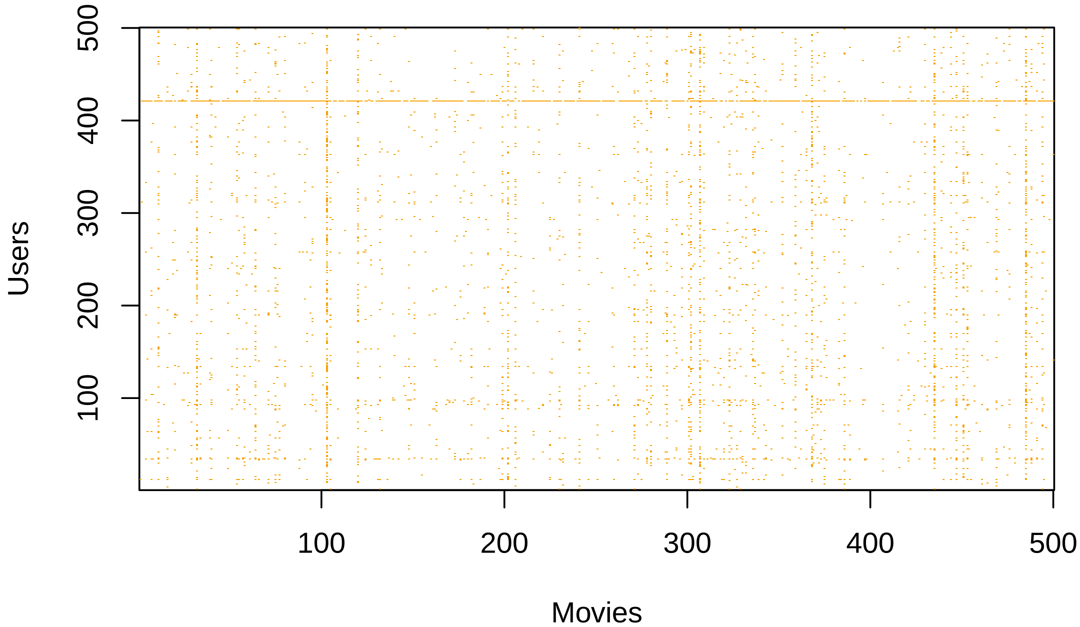

Figure 3: Sparse matrix - Movies vs Users

2.2.3 About timestamp

Timestamp is on epoch format, so it will be transformed to verify the behavior using a *scatterplot* with different time scales. The first date in which a movie was rated was **1995-01-09** and the last one is **2009-01-05**.

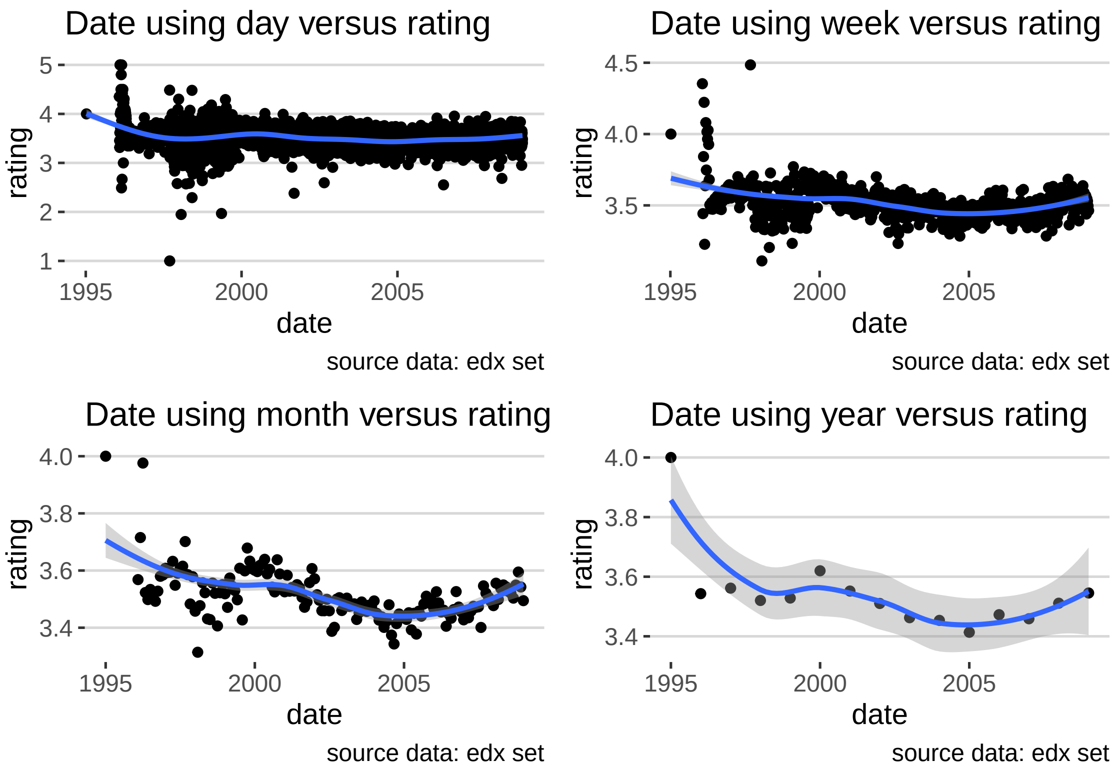

Figure 4: Timestamp versus rating

2.2.4 About title

The **top 5** and **worst 5** title´s movies based on reviews and average rating are:

Table 5: Best titles movies

title	reviews	avg_rating
Pulp Fiction (1994)	31,362	4.155
Forrest Gump (1994)	31,079	4.013
Silence of the Lambs, The (1991)	30,382	4.204
Jurassic Park (1993)	29,360	3.664
Shawshank Redemption, The (1994)	28,015	4.455

Table 6: Worst titles movies

title	reviews	avg_rating
1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)	1	2.0
100 Feet (2008)	1	2.0
4 (2005)	1	2.5
Accused (Anklaget) (2005)	1	0.5
Ace of Hearts (2008)	1	2.0

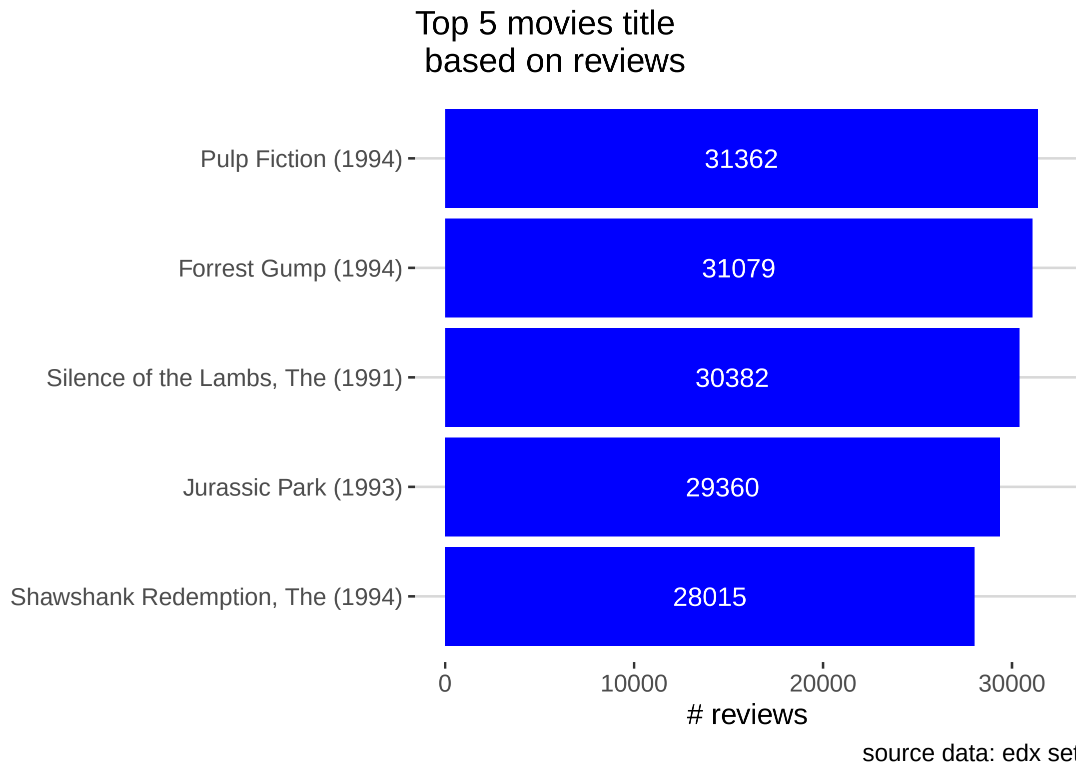

Figure 5: Movies titles most reviewed

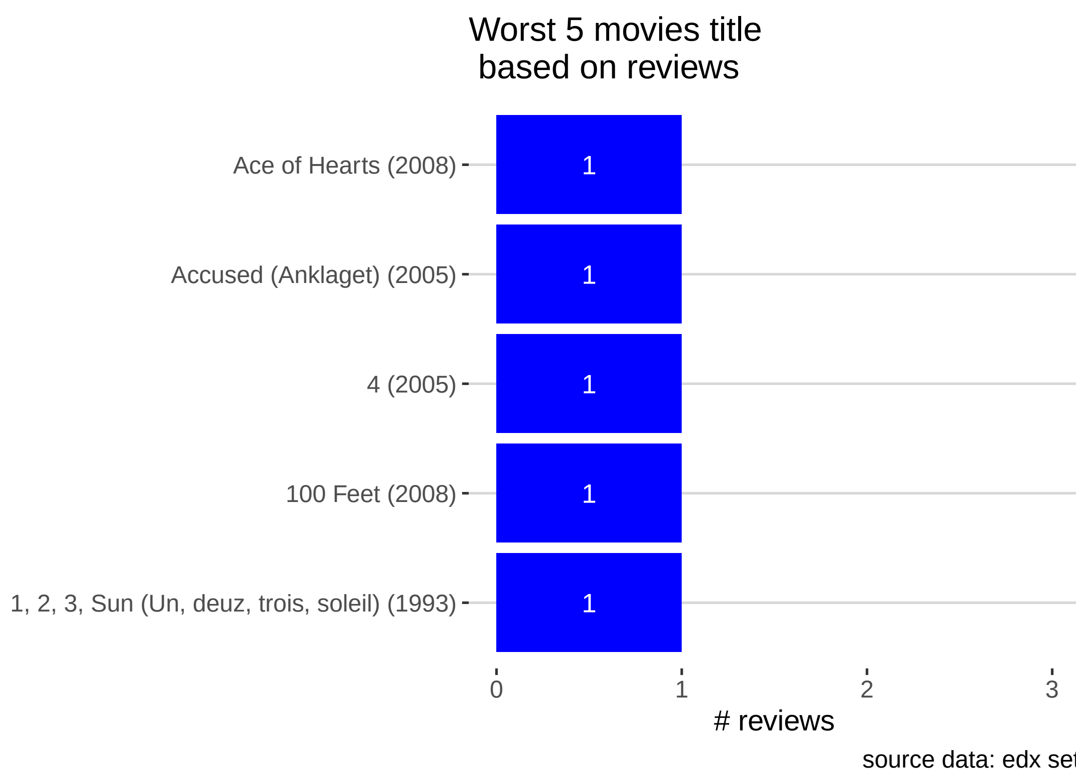

Figure 6: Movies titles less reviewed

Additionally, **title** column contains inside a *parenthesis* the year in which the movie was released, so **year_released** is a new column that contains parsed value. It´s interesting that movies around *1940* have a higher rating and for movies after *1970* the average rating has been decreasing maybe for the variety or generational change.

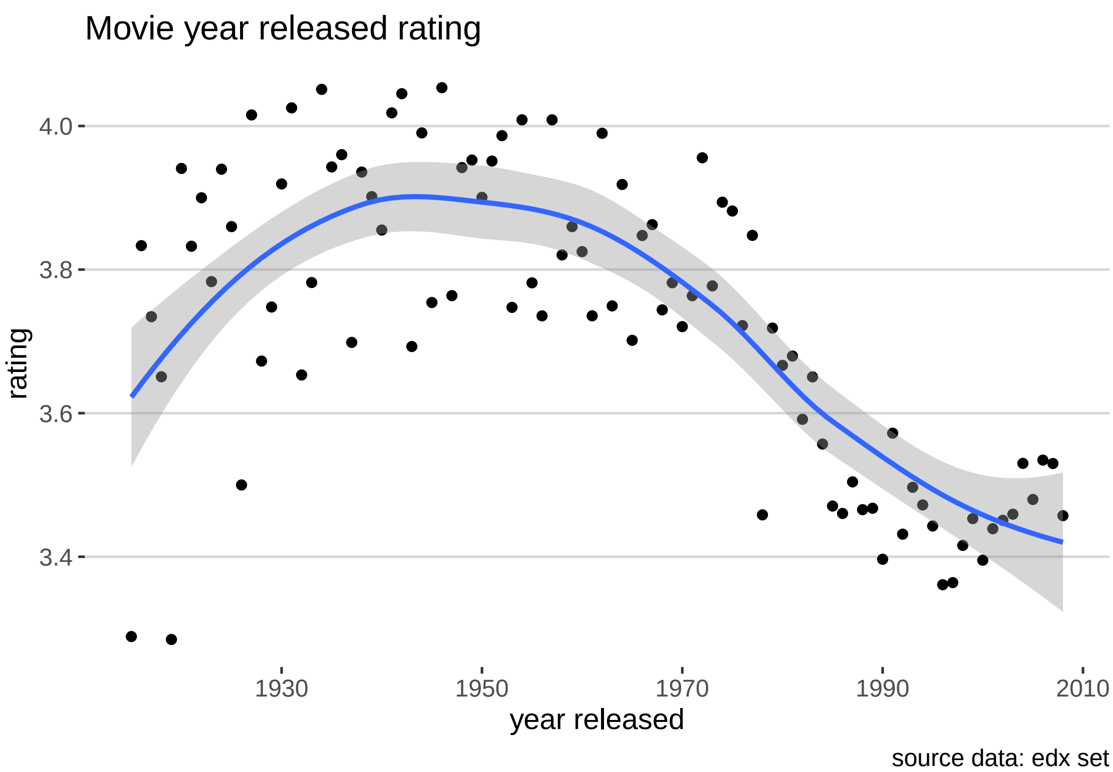

Figure 7: Year released rating

2.2.5 About genres

Several **genres** are indicated per movie. There are **797** genres being the **10** most reviewed:

Table 7: Genres reviews and rating

genres	reviews	avg_rating
Crime|Mystery|Thriller	26,892	4.199
Action|Adventure|Comedy|Fantasy|Romance	14,809	4.196
Animation|Children|Comedy|Crime	7,167	4.275
Film-Noir|Mystery	5,988	4.239
Crime|Film-Noir|Thriller	4,844	4.210
Crime|Film-Noir|Mystery	4,029	4.217
Drama|Film-Noir|Romance	2,989	4.304
Film-Noir|Romance|Thriller	2,453	4.216
Action|Crime|Drama|IMAX	2,353	4.297
Animation|IMAX|Sci-Fi	7	4.714

In addition the | *(pipe)* is used as delimiter and once parsed there are **20** unique genres with the following reviews:

Table 8: Genres reviews

genres_parsed	reviews
Action	2,560,545
Comedy	2,437,260
Drama	1,741,668
Adventure	753,650
Crime	529,521
Horror	233,074
Animation	218,123
Children	181,217
Thriller	94,718
Documentary	80,966
Sci-Fi	50,254
Mystery	30,536
Fantasy	26,080
Musical	16,264
Film-Noir	15,811
Western	15,300
Romance	12,733
War	2,314
IMAX	14
(no genres listed)	7

2.2.6 About rating

An histogram that displays the rating distribution is:

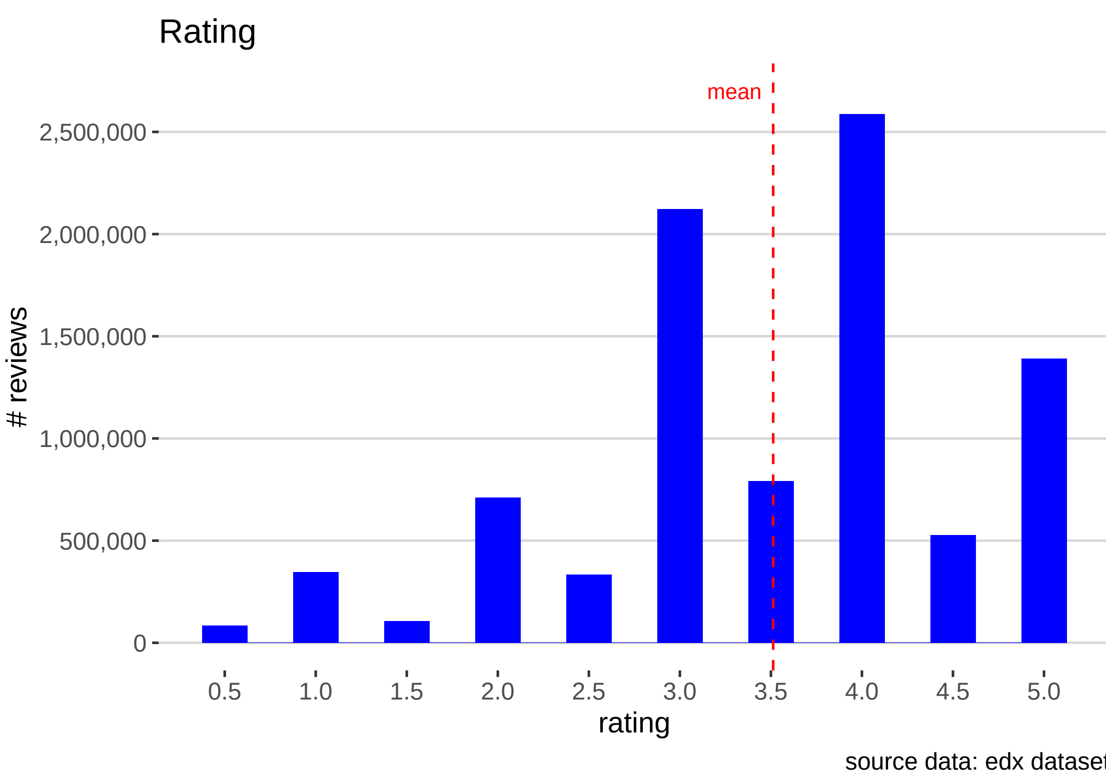

Figure 8: Rating histogram

An histogram about the ratings from the user **(59269)** with the maximum amount of reviews is:

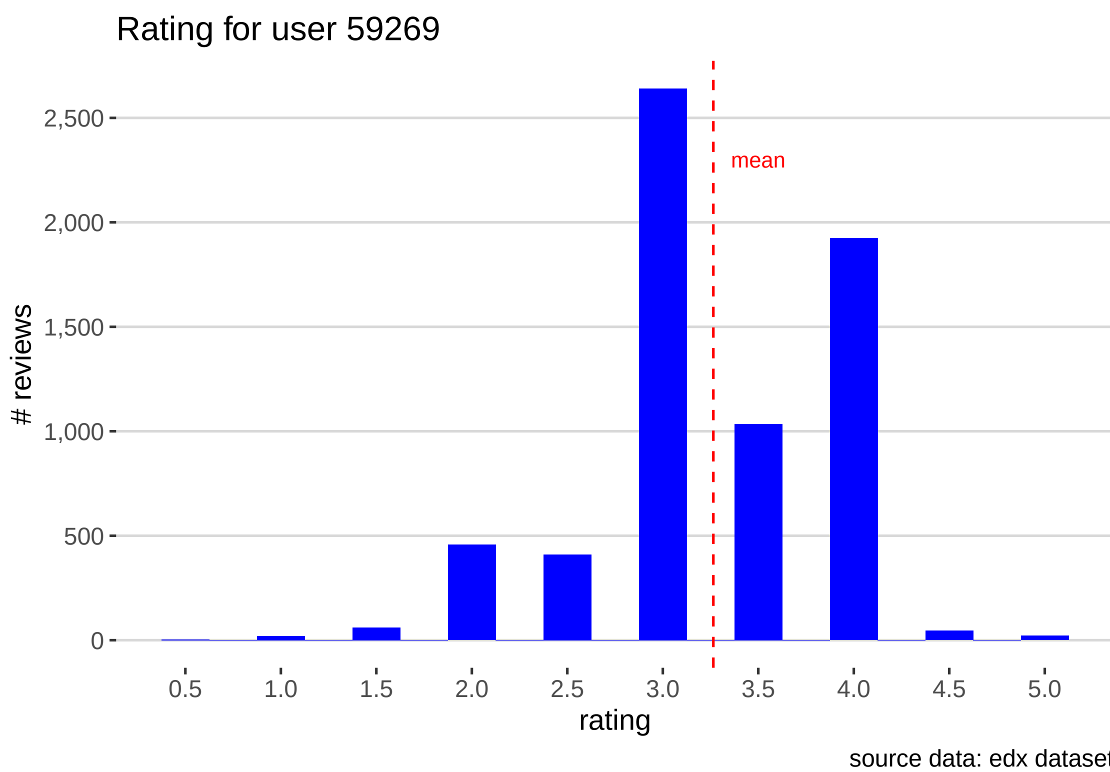

Figure 9: Rating histogram for most rated user

An histogram about the movie most rated **"Pulp Fiction (1994)"** is:

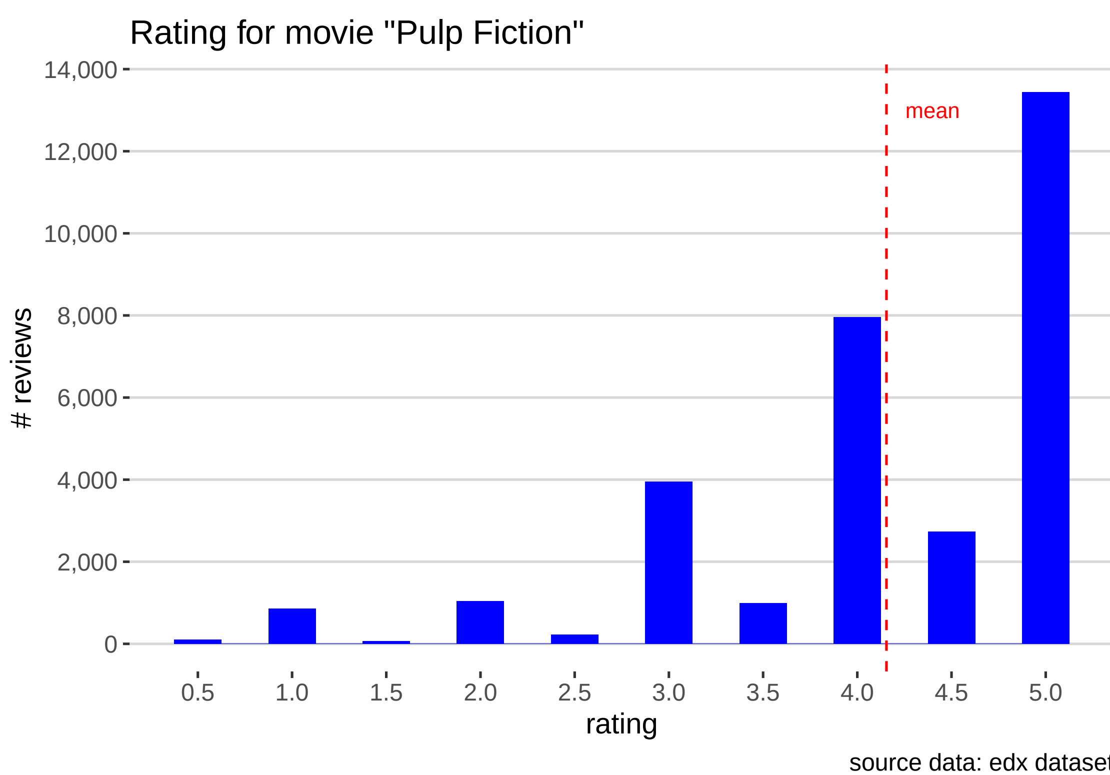

Figure 10: Rating histogram for most rated user

About the **genres** rating for the most reviewed that is **"Crime|Mystery|Thriller"** the histogram shows favorable reviews.

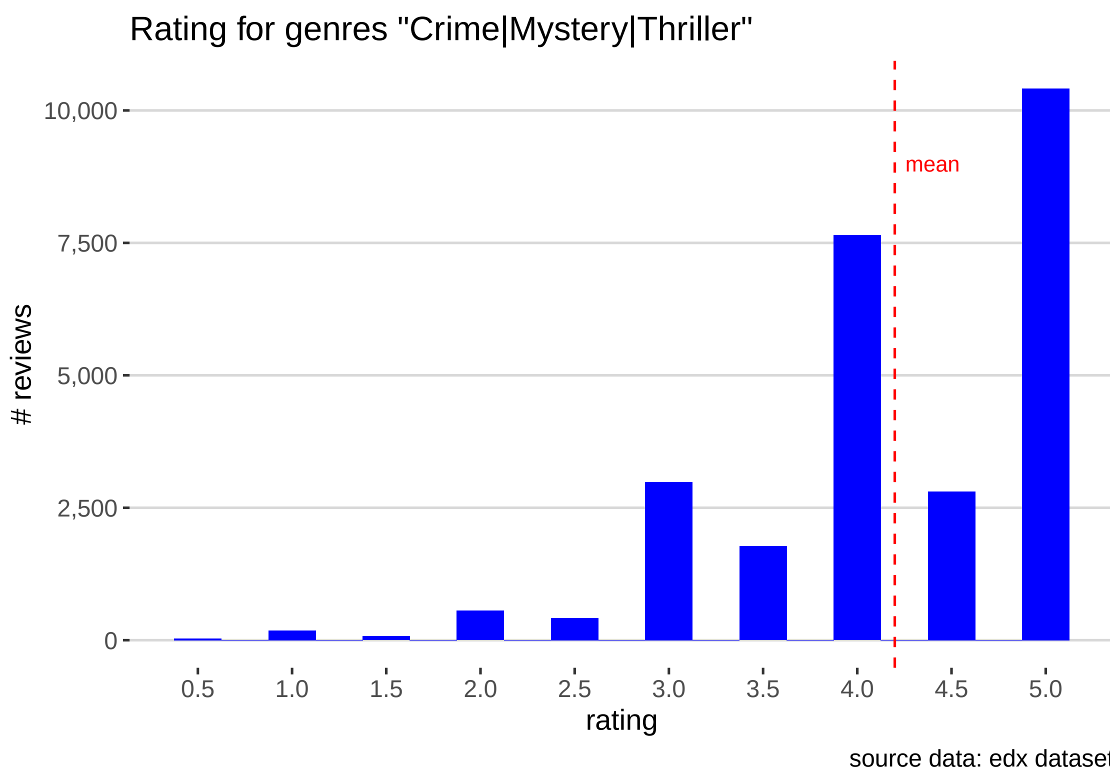

Figure 11: Rating histogram for genres "Crime|Mystery|Thriller"

3 Results

3.1 About RMSE

The formula used to obtain the loss function is:

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

$y_{u,i}$ is the rating for movie by user

$\hat{y}_{u,i}$ is the prediction

N is the user/movie combinations

3.2 About train and test datasets

edx dataset is splitted on another 2 datasets:

- **train_set** contains **8,100,048** observations that will be used with every algorithm.
- **test_set** contains **899,990** observations that will be used at the moment to obtain the respective **RMSE**.

3.3 Models

The model to be developed is **lineal** considering the mean μ that is the **true** value and the error $\epsilon_{u,i}$ (independent errors sampled from the same distribution centered at 0) the following initial formula:

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

3.4 Algorithms

3.4.1 First model - Average rating of all movies across all users

The formula used on this model is:

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

This algorithm generates the following RMSE value: **1.060054**.

3.4.2 Second model - Movie effect

The formula used on this model is:

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

This algorithm generates the following RMSE value: **0.942961**.

The following histogram shows the impact of this bias that is skewed to the left.

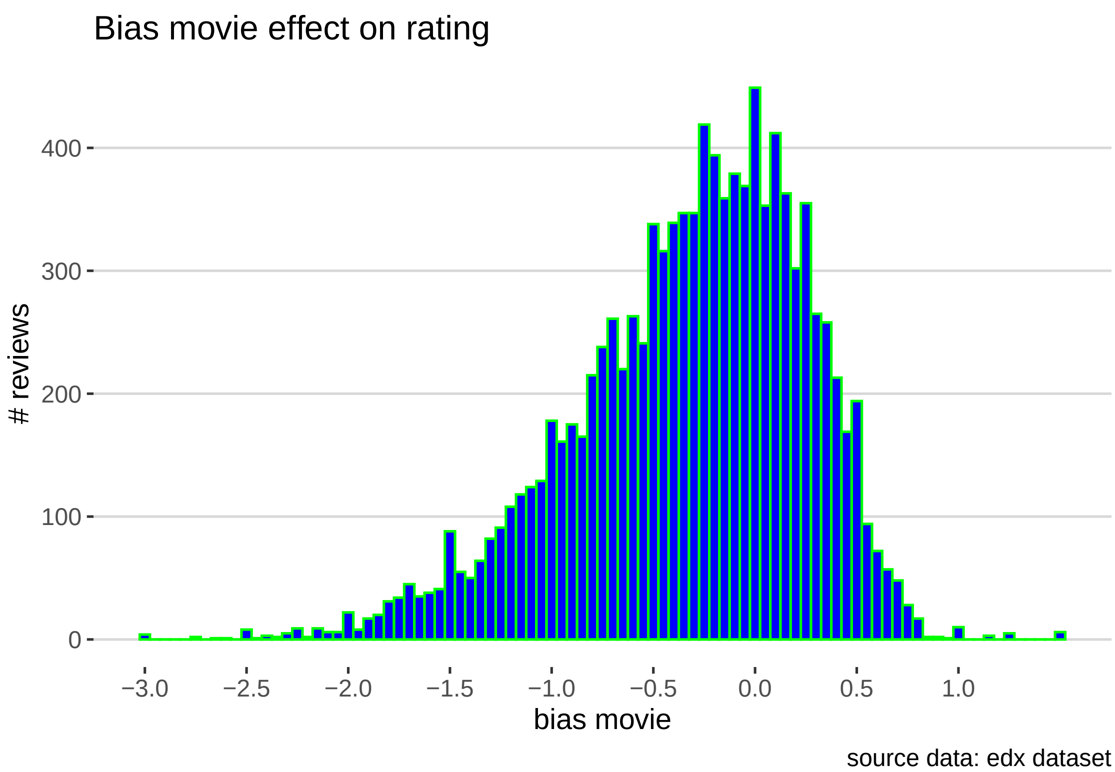

Figure 12: Bias movie

The **15 best movies** with this bias are:

Table 9: 15 best movies related to bias movie

movieId	title	users	rat_avg	bi_avg
3226	Hellhounds on My Trail (1999)	1	5.000	1.488
33264	Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1	5.000	1.488
42783	Shadows of Forgotten Ancestors (1964)	1	5.000	1.488
51209	Fighting Elegy (Kenka erejii) (1966)	1	5.000	1.488
53355	Sun Alley (Sonnenallee) (1999)	1	5.000	1.488
64275	Blue Light, The (Das Blaue Licht) (1932)	1	5.000	1.488
5194	Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	4	4.750	1.238
25975	Life of Oharu, The (Saikaku ichidai onna) (1952)	2	4.750	1.238
26048	Human Condition II, The (Ningen no joken II) (1959)	4	4.750	1.238
26073	Human Condition III, The (Ningen no joken III) (1961)	4	4.750	1.238
65001	Constantine's Sword (2007)	2	4.750	1.238
4454	More (1998)	6	4.667	1.154
5849	I'm Starting From Three (Ricomincio da Tre) (1981)	3	4.667	1.154
63808	Class, The (Entre les Murs) (2008)	3	4.667	1.154
7452	Mickey (2003)	1	4.500	0.988

The **15 worst movies** with this bias are:

Table 10: 15 worst movies related to bias movie

movieId	title	users	rat_avg	bi_avg
5805	Besotted (2001)	1	0.500	-3.012
8394	Hi-Line, The (1999)	1	0.500	-3.012
63828	Confessions of a Superhero (2007)	1	0.500	-3.012
64999	War of the Worlds 2: The Next Wave (2008)	2	0.500	-3.012
8859	SuperBabies: Baby Geniuses 2 (2004)	47	0.745	-2.768
61348	Disaster Movie (2008)	30	0.767	-2.746
6483	From Justin to Kelly (2003)	183	0.874	-2.638
7282	Hip Hop Witch, Da (2000)	11	0.909	-2.603
604	Criminals (1996)	1	1.000	-2.512
2228	Mountain Eagle, The (1926)	2	1.000	-2.512
3561	Stacy's Knights (1982)	1	1.000	-2.512
4071	Dog Run (1996)	1	1.000	-2.512
5702	When Time Ran Out... (a.k.a. The Day the World Ended) (1980)	1	1.000	-2.512
6189	Dischord (2001)	1	1.000	-2.512
8856	Roller Boogie (1979)	13	1.000	-2.512

3.4.3 Third model - User effect

The formula used on this model is:

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

This algorithm generates the following RMSE value: **0.977709**.

The following histogram shows the impact of this bias that is skewed to the left.

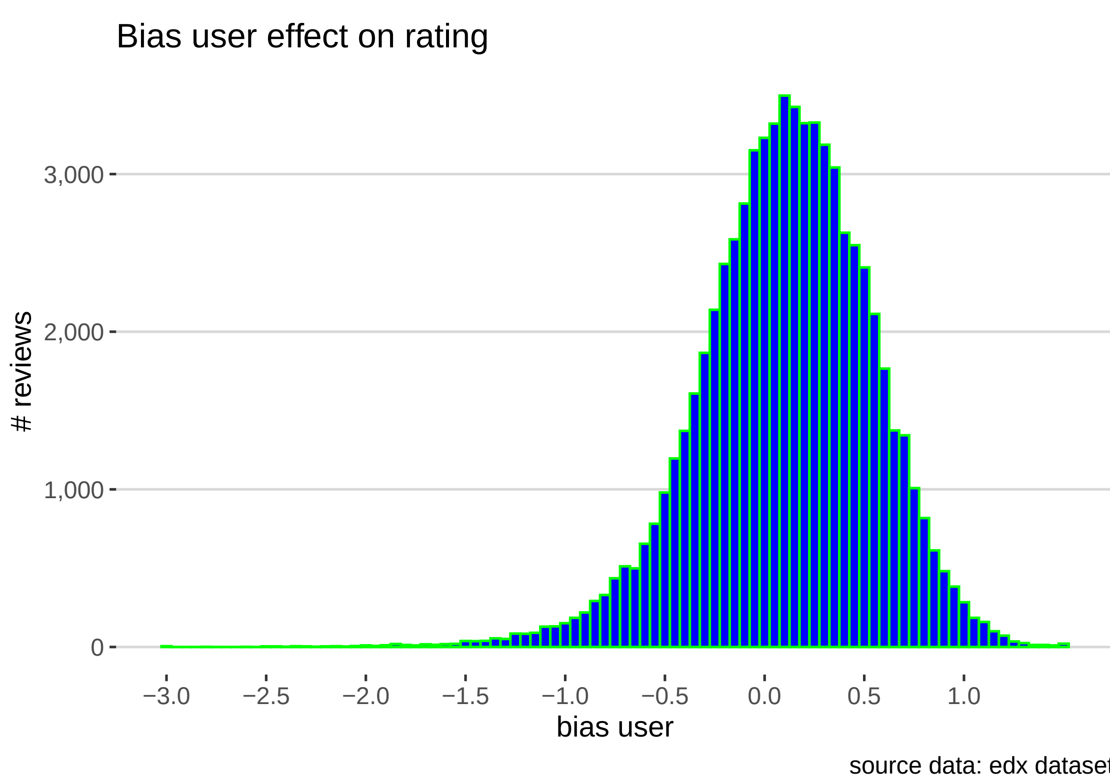

Figure 13: Bias user

The **15 best users** with this bias are:

Table 11: 15 best users related to bias user

userId	movies_rated	rating_avg	b_user_avg
1	18	5	1.488
7984	16	5	1.488
11884	18	5	1.488
13027	27	5	1.488
13513	16	5	1.488
13524	19	5	1.488
15575	25	5	1.488
18965	43	5	1.488
22045	16	5	1.488
26308	14	5	1.488
27831	17	5	1.488
30519	15	5	1.488
35184	22	5	1.488
42649	18	5	1.488
45895	16	5	1.488

The **15 worst users** with this bias are:

Table 12: 15 worst users related to bias user

userId	movies_rated	rating_avg	b_user_avg
13496	15	0.500	-3.012
48146	21	0.500	-3.012
49862	16	0.500	-3.012
62815	19	0.500	-3.012
63381	16	0.500	-3.012
6322	16	0.719	-2.794
19059	17	0.912	-2.601
3457	18	1.000	-2.512
24176	119	1.000	-2.512
24490	14	1.000	-2.512
15515	28	1.018	-2.495
59342	647	1.038	-2.475
28416	26	1.038	-2.474
43628	17	1.059	-2.454
24101	42	1.071	-2.441

3.4.4 Fourth model - Movie plus User effect

The formula used on this model is:

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

This algorithm generates the following RMSE value: **0.884399**.

3.4.5 Fifth model - Date effect

The formula used on this model is:

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

This algorithm generates the following RMSE value: **1.058253**.

The following histogram shows the impact of this bias that is not skewed.

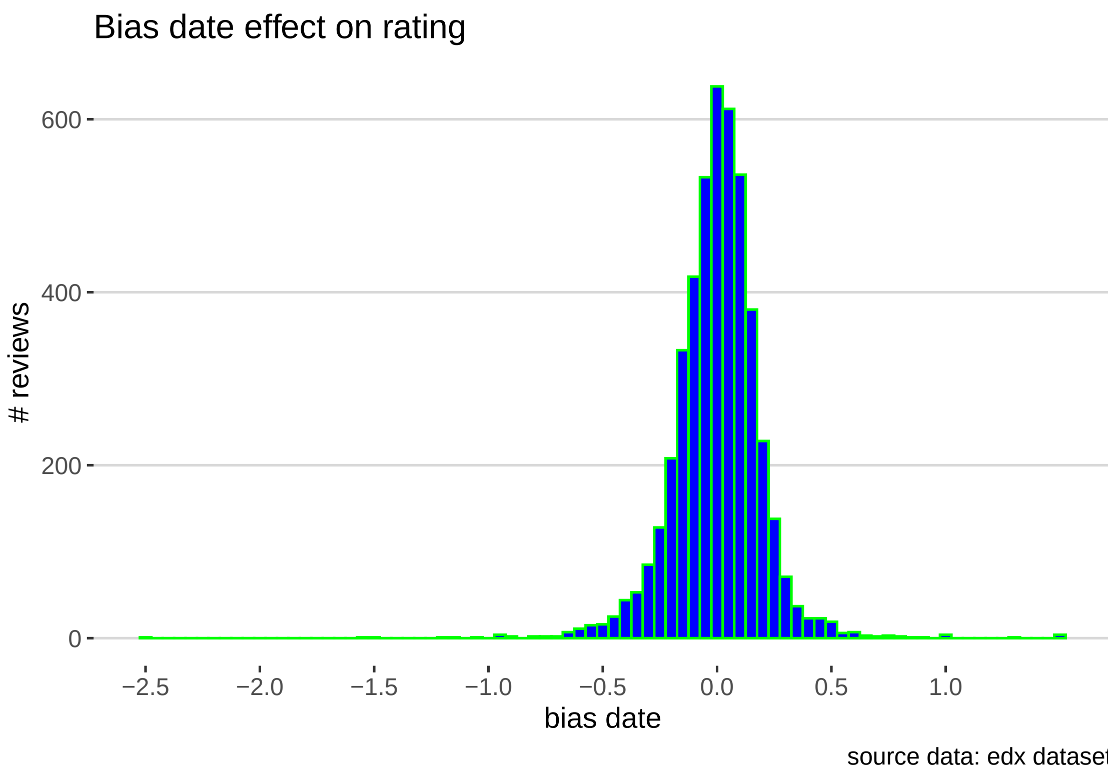

Figure 14: Bias date

3.4.6 Sixth model - Movie plus User plus Date effect

The formula used on this model is:

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

This algorithm generates the following RMSE value: **0.897684**.

3.4.7 Septh model - Genres effect

The formula used on this model is:

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

This algorithm generates the following RMSE value: **1.017501**.

The following histogram shows the impact of this bias that is skewed to the left.

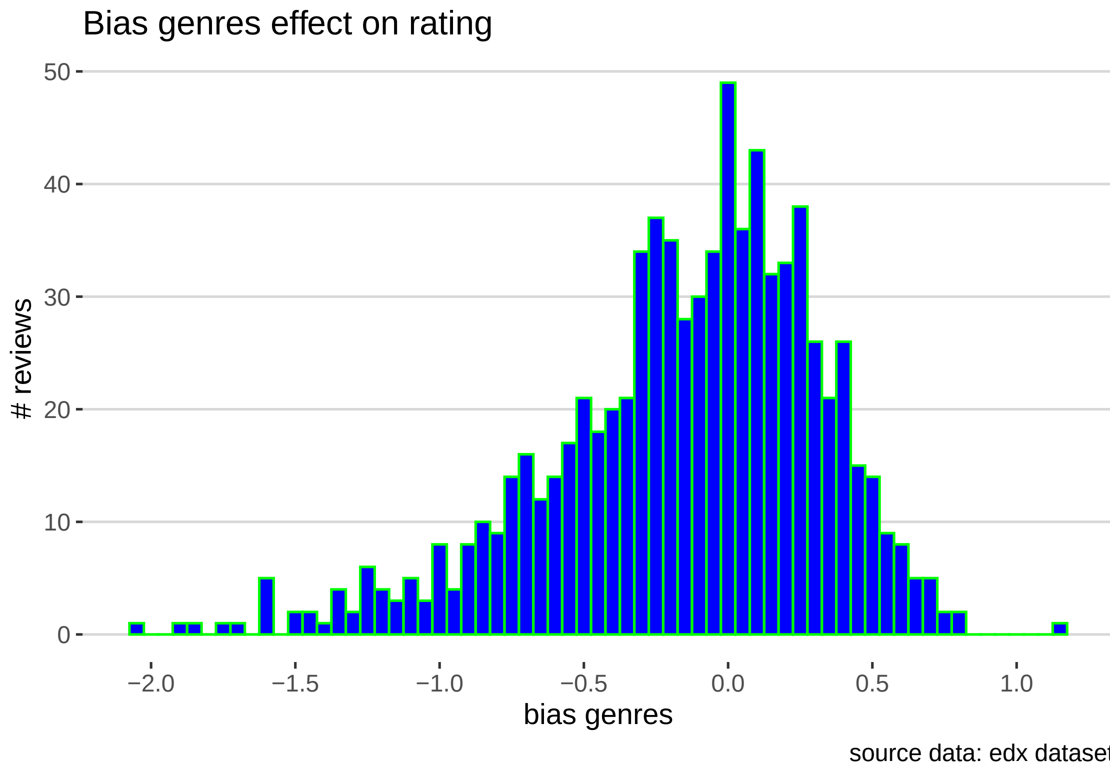

Figure 15: Bias date

3.4.8 Eighth model - Movie plus User plus Date plus Genres effect

The formula used on this model is:

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

This algorithm generates the following RMSE value: **0.957362**.

3.4.9 Correlation between predictors

The following correlogram shows the rating´s relationship between the different predictors which is higher with the combination of the **movie** and **user** effect. This can be validated with the RMSE obtained (model 4) but as it was indicated in the previous sections, there are users that rate few time the movies and also movies that are rated only once and this has an effect on the RMSE.

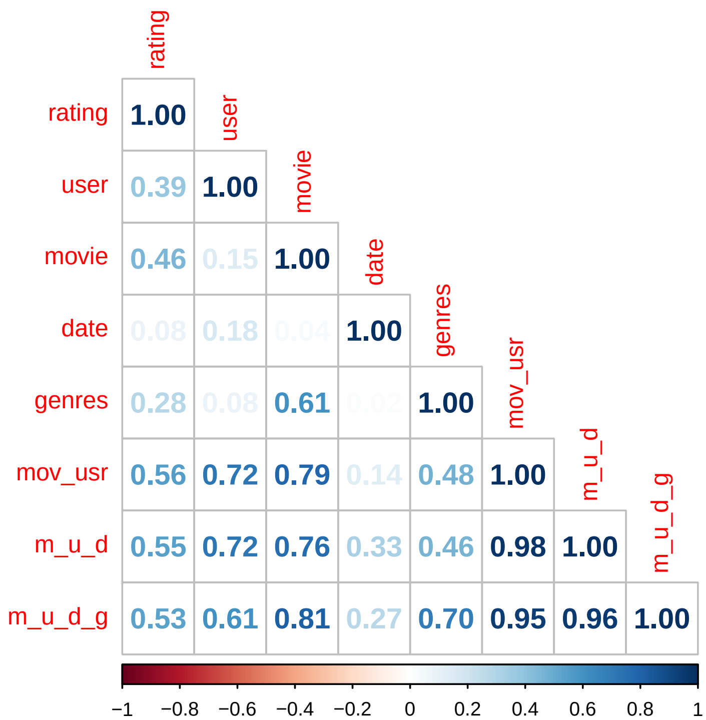

Figure 16: Correlogram

3.4.10 Regularization

By now the results obtained are far from the target, so another techniques need to be applied and one is the regularization which constrains the total variability of the effect sizes by penalizing large estimates that come from small sample sizes.

So **lambda** calculation will be done to obtain the minimal value that generates a lower RMSE.

3.4.10.1 Nineth model - Regularized Movie effect For this model is required to obtain a **penalty** term λ as indicated in the following graph, being **1.5** the value obtained.

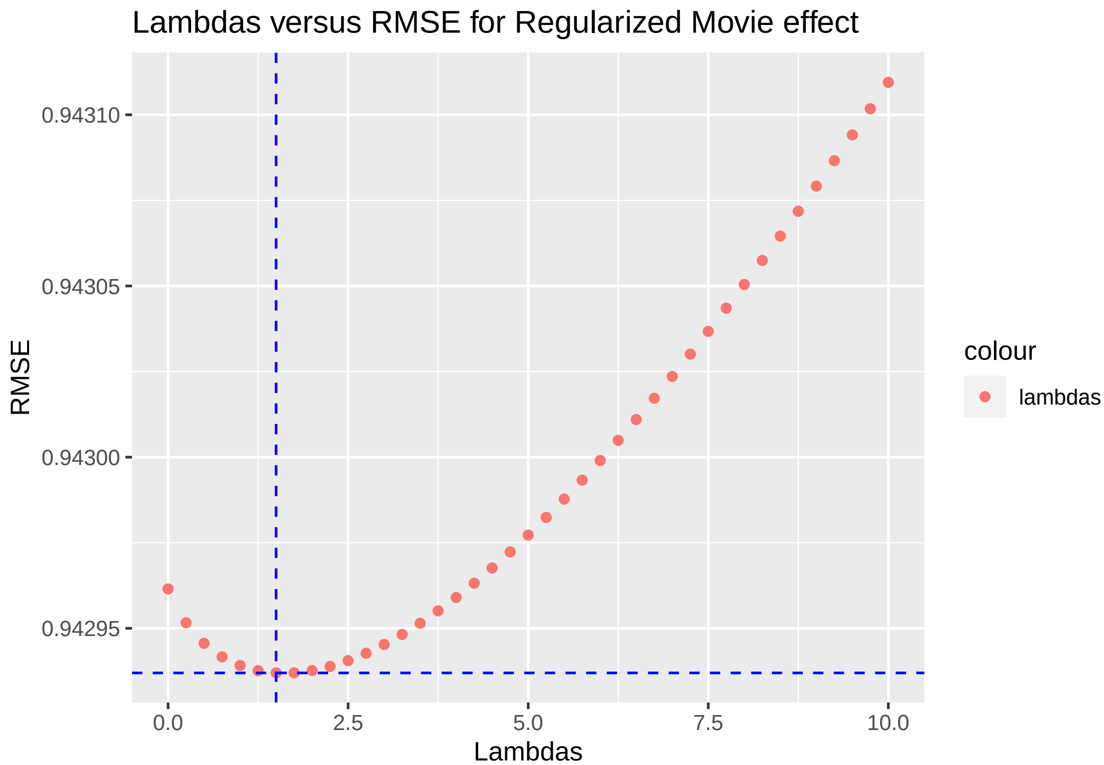

Figure 17: Lambda versus RMSE for regularized movie effect

The formula used on this model is:

$$\frac{1}{N}\sum_{u,i}\left(y_{u,i}-\mu-b_i\right)^2+\lambda\left(\sum_i b_i^2\right)$$

The term $\frac{1}{N}\sum_{u,i}\left(y_{u,i}-\mu-b_i\right)^2$ is used to obtain b_i and regularized term $\lambda\left(\sum_i b_i^2\right)$ avoids over fitting by penalizing the magnitudes of the parameters.

By using a cross-validation the $\hat{b}_i$ using the adequate λ can be found:

$$\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}\left(Y_{u,i}-\hat{\mu}\right)^2$$

This algorithm generates the following RMSE value: **0.942937**.

3.4.10.2 Tenth model - Regularized User effect For this model is required to obtain a **penalty** term λ as indicated in the following graph, being **5.25** the value obtained.

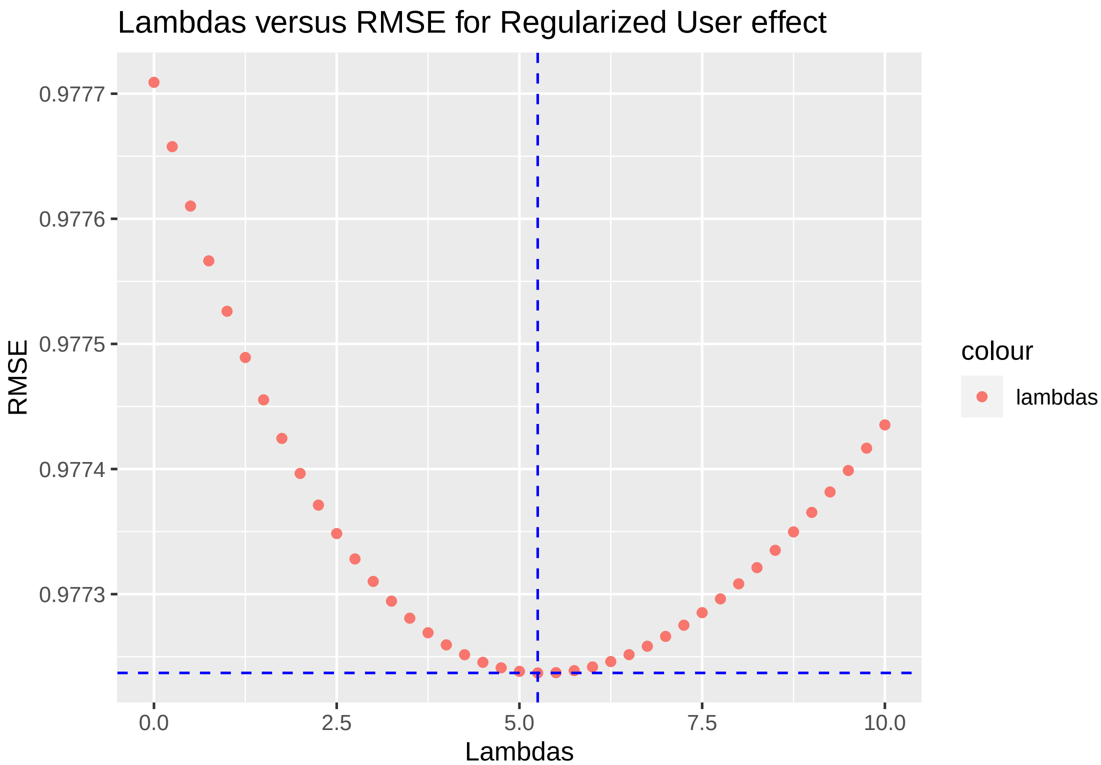

Figure 18: Lambda versus RMSE for regularized user effect

The formula used on this model is:

$$\frac{1}{N}\sum_{u,i}\left(y_{u,i}-\mu-b_u\right)^2+\lambda\left(\sum_{u}b_u^2\right)$$

The term $\frac{1}{N}\sum_{u,i}\left(y_{u,i}-\mu-b_u\right)^2$ is used to obtain b_u and regularized term $\lambda\left(\sum_{u}b_u^2\right)$ avoids over fitting by penalizing the magnitudes of the parameters.

By using a cross-validation the $\hat{b}_u$ using the adequate λ can be found:

$$\hat{b}_u(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}\left(Y_{u,i}-\hat{\mu}\right)^2$$

This algorithm generates the following RMSE value: **0.977237**.

3.4.10.3 Eleventh model - Regularized Movie plus Regularized User effect

For this model is required to obtain a **penalty** term λ as indicated in the following graph, being **5** the value obtained.

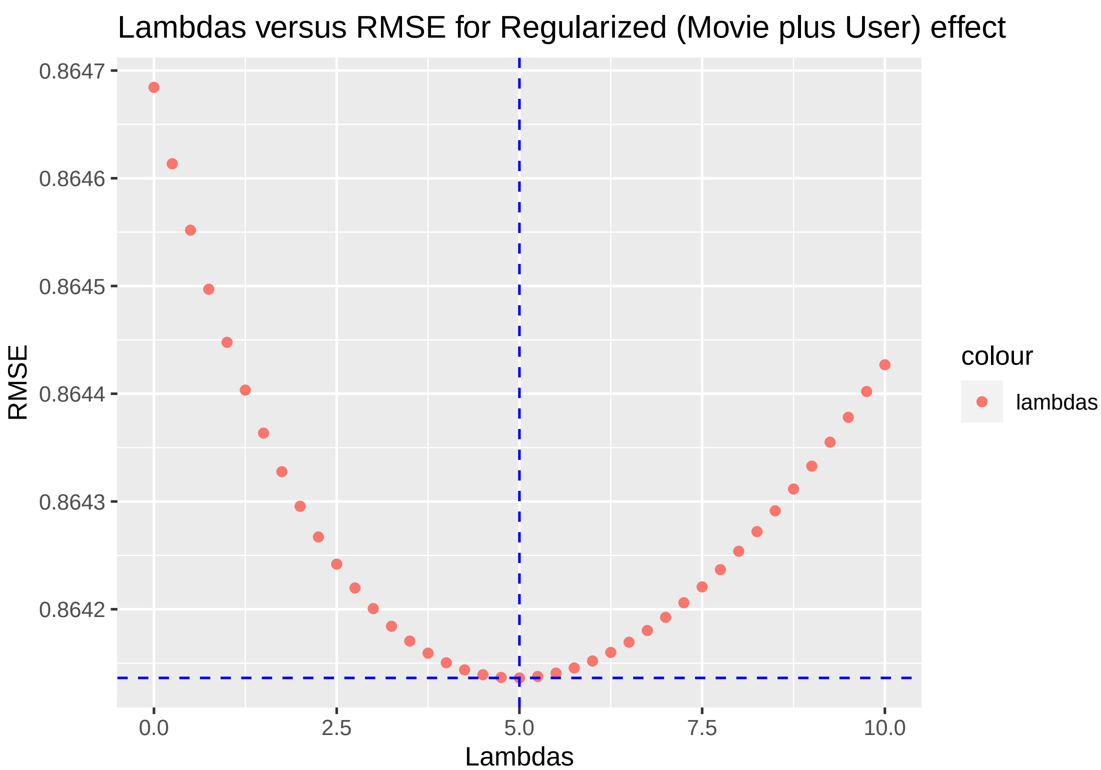

Figure 19: Lambdas versus RMSE for Regularized (Movie plus User) effect

The formula used on this model is:

$$\frac{1}{N}\sum_{u,i}\left(y_{u,i}-\mu-b_u-b_i\right)^2+\lambda\left(\sum_u b_u^2+\sum_u b_i^2\right)$$

The term $\frac{1}{N}\sum_{u,i}\left(y_{u,i}-\mu-b_u-b_i\right)^2$ is used to obtain b_i and b_u and regularized term $\lambda\left(\sum_u b_u^2+\sum_u b_i^2\right)$) avoids over fitting by penalizing the magnitudes of the parameters.

The regularized term $\lambda\left(\sum_u b_u^2+\sum_u b_i^2\right)$ avoids over fitting by penalizing the magnitudes of the parameters.

By using a cross-validation the $\hat{b}_u$ and $\hat{b}_i$ using the adequate λ can be found:

$$\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}\left(Y_{u,i}-\hat{\mu}\right)^2$$

$$\hat{b}_u(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}\left(Y_{u,i}-\hat{\mu}-\hat{b}_i\right)^2$$

This algorithm generates the following RMSE value: **0.864136**.

3.4.11 Matrix factorization

Matrix factorization method is used to solve a recommendation system. The idea is to approximate the whole rating matriz $R_{m\times n}$ by the product of two matrics of lower dimensions $P_{k\times m}$ and $Q_{k\times n}$, such that

$$R\approx P'Q$$

The process of solving the matrices **P** and **Q** is referred to as *model training*, and the selection of **penalty** parameters is called *parameter tuning*.

There is an open source library called **recosystem** that can be used using parallel marix factorization (Chin, Yuan, et al. 2015) that have the following steps:

Table 13: Recosystem steps

Step	Input	Output
Model training	Training data set	-
Parameter tuning	Training data set	-
Exporting model	-	User matrix P, item matrix Q
Prediction	Testing data set	Predicted values

In our case **DataSource** was created using **data_memory()**

The usage of **recosystem** is quite simple, mainly consisting of the following steps:

1. Create a model object (a Reference Class object in R) by calling **Reco()**.
2. (Optionally) call the **tune()** method to select best tuning parameters along a set of candidate values.
3. Train the model by calling the **train()** method. A number of parameters can be set inside the function, possibly coming from the result of **tune()**.
4. (Optionally) export the model via $output(), i.e. write the factorization matrices **P** and **Q** into files or return them as R objects.
5. Use the **predict()** method to compute predicted values.

3.4.11.1 Twelveth model - Matrix factorization

In this model the following data will be used:

- Using **data_memory** for both **train_set** and **test_set** datasets:
  - As **user_index** the predictor **usedId**
  - As **item_index** the predictor **movieId**
  - As **rating** the outcome **rating**
- In the **tuning** parameters only was changed **nthread** from **1** to **6**, the rest continued the same.
- Using **out_memory** for the predicted values.

The results of 20 iterations are:

```
## iter      tr_rmse          obj
##    0       0.9726   1.0833e+07
##    1       0.8772   8.9950e+06
##    2       0.8458   8.3770e+06
##    3       0.8256   8.0010e+06
##    4       0.8110   7.7538e+06
##    5       0.8000   7.5728e+06
##    6       0.7914   7.4413e+06
##    7       0.7843   7.3347e+06
##    8       0.7779   7.2431e+06
##    9       0.7724   7.1703e+06
##   10       0.7675   7.1069e+06
##   11       0.7631   7.0510e+06
##   12       0.7593   7.0059e+06
##   13       0.7559   6.9627e+06
##   14       0.7530   6.9278e+06
##   15       0.7503   6.8952e+06
##   16       0.7481   6.8705e+06
```

```
## 17 0.7460 6.8453e+06
## 18 0.7440 6.8223e+06
## 19 0.7424 6.8027e+06
```

This algorithm generates the following RMSE value: **0.790498**, being the **lowest** value obtained.

3.4.11.2 Thirteenth model - Matrix factorization using validation dataset

The following data will be used:

- Using **data_memory** for both **validation** dataset:
  - As **user_index** the predictor **usedId**
  - As **item_index** the predictor **movieId**
  - As **rating** the outcome **rating**
- Using **out_memory** for the predicted values.

The RMSE obtained is: **0.791136**.

3.4.12 Resume

In the next table are indicated the **RMSEs** obtained on every algorithm being **matrix factorization (best performance)** the one that generated a **0.7905** that is below the proposed target.

Table 14: RMSEs obtained - Target < 0.86490

Algorithm	RMSE
Model #1 - Average rating movie	1.060054
Model #2 - Movie effect	0.942961
Model #3 - User effect	0.977709
Model #4 - Movie plus User effect	0.884399
Model #5 - Date effect	1.058253
Model #6 - Movie plus User plus Date effect	0.897684
Model #7 - Genres effect	1.017501
Model #8 - Movie plus User plus Date plus Genres effect	0.957362
Model #9 - Regularized Movie effect	0.942937
Model #10 - Regularized User effect	0.977237
Model #11 - Regularized (Movie plus User) effect	0.864136
Model #12 - Matrix factorization using recosystem	0.790498
Model #13 - Matrix factorization using recosystem on validation dataset	0.791136

The models **#11** and **#12** took **20** and **70 minutes** respectively to run being the ones that could obtain a RMSE below the target.

4 Conclusion

The algorithms that took less time to be executed obtained a higher **RMSE** in some cases very similar to the standard deviation and the ones that took more time on being executed obtained a lower **RMSE**.

The results obtained by **factorization model** (models #12 and #13) are **8.99%** more efficient than the target proposed with the constraint about the computing resources needed to execute the algorithms (for this project a end-user device with **8 GiB RAM**, **2 virtual cores**, **2.90 GHz Intel** processor speed running **Windows 10**). In the other side models such as **user (#2)**, **date effect (#5)** and **genres (#7)** had a lower performance by **8.64%**, **20.11%**, and **16.21%** respectively.

Now based on this experience, the high computing capabilities are needed to generate value as soon as possible, because for the case of a **recommendation system** a strategic decision can be supported based on the results obtained and **cloud computing** can be used with the consideration of the costs involved.

As a future work techniques more advanced such as **neural networks** and **deep learning** can be explored as a way in which an organization will be interested on generate the best customer experience possible in a era of commodities and substitute products using parameters´ relationship where the most important in uncertain times is to increase business value.

5 References

- Introduction to Data Science. Rafael A. Irizarry. https://rafalab.github.io/dsbook/
- Create Awesome LaTeX Table with knitr::kable and kableExtra. Hao Zhu. https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_pdf.pdf
- recosystem: Recommender System Using Parallel Matrix Factorization. Yixuan Qiu. https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html