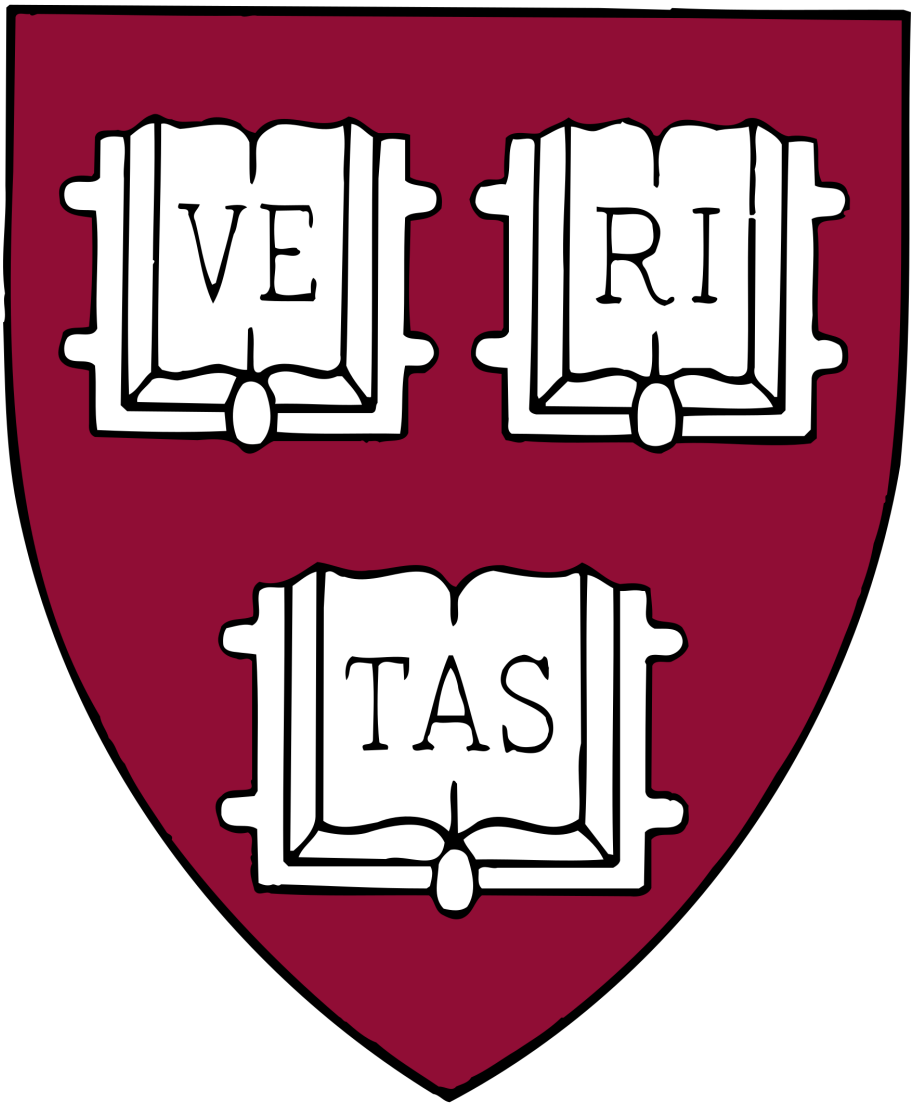


HarvardX - PH125.9x - Data Science

Movie Lens Project

Paul Nardon

2022-11-11

Contents

Abstract

This project aims to build the best predictive model – i.e., a recommendation system – to evaluate what rating a user will give to a movie using machine learning.
A recommendation system allowed us to predict how many stars a user will give a specific movie.

The goal is to minimize the Root Mean Squared Error (i.e. RMSE) of our predictive models below 0.8649. We used a small subset of a database from the GroupLens research lab that contains over 20 million ratings, 27 thousand movies and 138 thousand users.

```
library(tidyverse)
library(dslabs)
data("movielens")
```

We built three models in order to reach this objective.
The first model based on a loss function and regularization obtained a RMSE equal to 0.8646 but this model didn't succeed to explain the entire variability and connection between variables.

The second model based on parallel matrix factorization obtained a RMSE value equal to 0.8189 which is much better than the first model. However, the efficiency associated to the execution time of calculus is weak due to a sequential approach. Thereby, we had to drastically reduce the data set in order to succeed to execute calculus. Despite that, the execution time remains important, about 20 minutes.

The third model based on matrix factorization with parallel stochastic gradient descent is considered as the best model because of the RMSE value, which is equal to 0.7821, and his efficiency. In fact, the parallel stochastic gradient descent method permits to drastically reduce the execution time of calculus on the entire data set.

Introduction

Many customers are flooded with the products choices in the e-commerce activities. For example, Amazon or eBay provide plethora of items choices based on your previous purchases and previously viewed items. An important stake is how to let users efficiently find items meeting their needs. Recommender systems have been constructed to meet this goal.

As demonstrated in the Netflix competition [Bell and Koren 2007], a collaborative filter using latent factors has been evaluated as one of the best models for recommender systems. The affinity between a user and an item is determined by the product of their latent-factor vectors.

We will build three predictive algorithm models in order to suggest the best movies recommendation system for users. We will also evaluate their efficiency in term of execution time.

This report is composed of 3 parts. In the first part, we will describe the data set and we will make some reprocessing to prepare data for our predictive machine learning algorithms.
In the second part, we will analyze the distribution and characteristics of each variable from data visualization and data analysis to understand the relations between them and build different algorithms models.
In the last part of this report, we will build three models based on the analysis of the database.

The first model is based on a loss function and regularization; the second is based on parallel matrix factorization method and for this we will use the "Recommenderlab" package For the third, we will build a model based on matrix factorization with parallel stochastic gradient descent algorithm from the "Recosystem" package.

Part I - Data structure and data reprocessing

In this part one, we will start by loading the data set from which we will build our three predictive algorithm models.The data set comes from the GroupLens research lab. Then, we will transform and select the data to make it usable.

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

```
ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")
```

After loading and transform the "Movielens" data set, we will describe his structure and reprocessing their data which we will used for data analyzing and visualization (cf. part II - Data analyzing and visualization) and for building our predictive algorithm models (cf. Part III - Predictive algorithm models).

A - Analyzing the structure of the Movielens data set

The "Movielens" data set is composed of 10 000 054 observations and 6 variables which are :

- "userId" which corresponds to the users's identifier. Each user has a unique identifier;
- "movieId" which corresponds to the movie's identifier. Each movie has a unique identifier;
- "rating" which corresponds to the rating made by user. This last can give rating from 0 to 5 for each movie. 5 being the most rating value;
- "timestamp" which corresponds to the date in which user rated the movie;
- "title" which corresponds to the movie title;
- "genres" which corresponds to the movie category. Each movie can be associated to several categories.

Table 1: An overview of the Movielens data set

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	231	5	838983392	Dumb & Dumber (1994)	Comedy
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

B - Data reprocessing

After having verify there wasn't NAS's values into the "Movielens" data set, we will reprocess data of each variable and create new data sets in order to can used them for, firstly, data analysis and visualization, and, on the other hand, training and evaluate our predictive algorithm models.

B.1 - Extraction of the users' rated year

The observations contained into the "timestamp" variable are not readable. We will extract the rated year into this variable and transform it in a date format. Then, we will remove the "timestamp" variable.

Table 2: An overview of the Movielens reprocessing data set after having extracted the rated year of each movie

userId	movieId	rating	title	genres	year_rated
1	122	5	Boomerang (1992)	Comedy|Romance	1996
1	185	5	Net, The (1995)	Action|Crime|Thriller	1996
1	231	5	Dumb & Dumber (1994)	Comedy	1996
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1996
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996

B.2 - Extraction of the release year of each movie

We can observe that the movie's release year is contained into the "title" variable. We will extract it and store it into a new variable which names is "Year_release". Then, we will remove the year contained into the "title variable".

Table 3: An overview of the Movielens reprocessing data set after having extracted the release year of each movie

userId	movieId	rating	title	genres	year_rated	year_release
1	122	5	Boomerang	Comedy|Romance	1996	1992
1	185	5	Net, The	Action|Crime|Thriller	1996	1995
1	231	5	Dumb & Dumber	Comedy	1996	1994
1	292	5	Outbreak	Action|Drama|Sci-Fi|Thriller	1996	1995
1	316	5	Stargate	Action|Adventure|Sci-Fi	1996	1994
1	329	5	Star Trek: Generations	Action|Adventure|Drama|Sci-Fi	1996	1994

We will ensure that the rated year by user is less than the release year of the movie.
We can observe that there are several movies for which her release year is more than the rated year by user.
In fact, the table below show us there is 201 rows, associated to 23 movies which are concerned by this error.

Table 4: An overview of observations for which the rated year is less than the release year

userId	movieId	rating	title	genres	year_rated	year_release	nyear_after_release
785	981	3	Dangerous Ground	Drama	1996	1997	-1
1468	879	2	Relic, The	Horror|Thriller	1996	1997	-1
1583	981	1	Dangerous Ground	Drama	1996	1997	-1
1766	870	5	Gone Fishin'	Comedy	1996	1997	-1
1766	879	5	Relic, The	Horror|Thriller	1996	1997	-1
1766	981	3	Dangerous Ground	Drama	1996	1997	-1

After verifying on internet, most of the release year, contained into the "title" variable, are correct. It seemed like it was human error related to the "timestamp" variable.

Table 5: The list of the movies for which the release year is upper to the rated year

title	year_release
Dangerous Ground	1997
Relic, The	1997
Gone Fishin'	1997
One Man's Hero	1999
Yards, The	2000
'Til There Was You	1997
Shadow Conspiracy	1997
Drunks	1997
Farmer & Chase	1997
Nightwatch	1997
Talk of Angels	1998
Bliss	1997
Desperate Measures	1998
Phantoms	1998
Truman Show, The	1998
Orange County	2002
Sliding Doors	1998
Lawless Heart, The	2003
Chairman of the Board	1998
Fallen	1998
Shooter, The	1997
Men of Means	1999
Bubble	2006

We will then replace the rated year associated to these 201 rows by the release year of the movie which related to them.

Table 6: An overview of the Movielens reprocessing data set after having reprocessed the release year

userId	movieId	rating	title	genres	year_rated	year_release
1	122	5	Boomerang	Comedy|Romance	1996	1992
1	185	5	Net, The	Action|Crime|Thriller	1996	1995
1	231	5	Dumb & Dumber	Comedy	1996	1994
1	292	5	Outbreak	Action|Drama|Sci-Fi|Thriller	1996	1995
1	316	5	Stargate	Action|Adventure|Sci-Fi	1996	1994
1	329	5	Star Trek: Generations	Action|Adventure|Drama|Sci-Fi	1996	1994

We do the same control to ensure that the rated year by users is less than the release year of movie.

Table 7: An overview of observations for which the rated year is less than the release year

userId	movieId	rating	title	genres	year_rated	year_release	nyear_after_release

After having reprocessing the release year of each movie contained in the title, we will ensure that movies have only one release year.

Table 8: An overview of movies which have at least two different release years

movieId	year_release

B.3 - New variable adding : the age of the movie

We will create a new variable contained the age of each movie until today. This variable will serve us in the part II when we are looking for such correlation between variables.

Table 9: An overview the Movielens reprocessing data set after having added the age of the movie

userId	movieId	rating	title	genres	year_rated	year_release	movie_first_rating	movie_age
1	122	5	Boomerang	Comedy|Romance	1996	1992	1996	30
1	185	5	Net, The	Action|Crime|Thriller	1996	1995	1996	27
1	231	5	Dumb & Dumber	Comedy	1996	1994	1996	28
1	292	5	Outbreak	Action|Drama|Sci-Fi|Thriller	1996	1995	1996	27
1	316	5	Stargate	Action|Adventure|Sci-Fi	1996	1994	1996	28
1	329	5	Star Trek: Generations	Action|Adventure|Drama|Sci-Fi	1996	1994	1996	28

B.4 - Reprocessing the "genres" column

We will create a new data frame for later analyzing the distribution of the data associated with genres data. In fact, the "genres" variable can contain several movies categories. We will split data by adding rows in which there will be only one movie genre.

Table 10: An overview the data genres after reprocessing

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy
1	122	5	838985046	Boomerang (1992)	Romance
1	185	5	838983525	Net, The (1995)	Action
1	185	5	838983525	Net, The (1995)	Crime
1	185	5	838983525	Net, The (1995)	Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy

We can see that some observations do not have movies genres. We will delete them from the "data_genres" data set.

Table 11: An overview of observations which do not have movies genres

userId	movieId	rating	timestamp	title	genres
7701	8606	5.0	1190806786	Pull My Daisy (1958)	(no genres listed)
10680	8606	4.5	1171170472	Pull My Daisy (1958)	(no genres listed)
29097	8606	2.0	1089648625	Pull My Daisy (1958)	(no genres listed)
46142	8606	3.5	1226518191	Pull My Daisy (1958)	(no genres listed)
57696	8606	4.5	1230588636	Pull My Daisy (1958)	(no genres listed)
64411	8606	3.5	1096732843	Pull My Daisy (1958)	(no genres listed)
67385	8606	2.5	1188277325	Pull My Daisy (1958)	(no genres listed)

We will also create a matrix from movielens data for later analyzing the correlation between movies genres. Firstly, we will select variables "userID", "movieId", "rating", "genres" from "movielens" data and we will store them into a new data set. We will delete rows for which there is no genres mentioned.
Secondly, We will store movies genres into a list.
Afterwards, we we implement a function which goal is to create a column for each movies genres. We will mention in each column a 1 in so far as the rated movie corresponds to the genre of the column. Otherwise, we will mention a -1.
We will remove the initial genres column and transform the data set into a matrix.

Table 12: An overview the Movielens matrix data set after reprocessing

userId	movieId	rating	Action	Adventure	Comedy	Romance	Crime	Thriller	Drama	Sci-Fi	Children	Fantasy	War	Animation	Musical	Western	Mystery	Film-Noir	Horror	Documentary	IMAX
1	122	5	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	185	5	1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	231	5	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	292	5	1	-1	-1	-1	-1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	316	5	1	1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	329	5	1	1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

B.5 - Training and validation set creation

We will split the "movielens" data set into a training test and a validation test.
The training test will be used to train our predictive algorithms and select the best parameters which we will permit us to minimize the Residual Mean Square Error (RMSE). The test set will be used to evaluate our predictive algorithms.

The validation set will be 10% of the "Movielens" data. We will make sure the "userId" and "movieId" present in validation set are also in training set. Otherwise, we will add rows removed form validation set back into training set.

```
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens_reprocessing$rating, times = 1,
                                  p = 0.1, list = FALSE)
edx <- movielens_reprocessing[-test_index,]
temp <- movielens_reprocessing[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, removed)

invisible(invisible(gc())) # for cleaning memory
```

After all these reprocessing, we are ready to make data analyzing and data visualization.

Part II - Data analyzing and data visualization

First of all, we will analyse the distribution of each Movielens' data set variables in order to understand their structure and make hypothesis on parameters which could influence our predictive algorithm models. Afterwards, we will bring out correlation between variables and observations.

A - Distribution analysis

We will start by analyzing the movies distribution.

A.1 - Movies distribution

The "Movielens" data set is composed of 10 677 movies associated to 10 000 054 ratings.

The data analysis of the number of ratings by movie show us the mean is about 937 ratings while the median is equal to 135. In addition to this abyssal gap between this two variables, the distribution is not follow a normal law (cf. Table 14). As for her, the standard deviation is equal to 2 487 ratings number.
The best rated movie has obtained 34 864 ratings while the minimum of ratings number is equal to 1.

The first quantile is equal to 34 ratings. i.e. 25% of movies have got less than 34 ratings whereas 75% of movies have got less than 626 ratings (Third quantile) and 90% of them have less than 2 388 ratings (cf. Table 13).

Table 13: An overview of some quantiles and deciles of the number ratings by movie

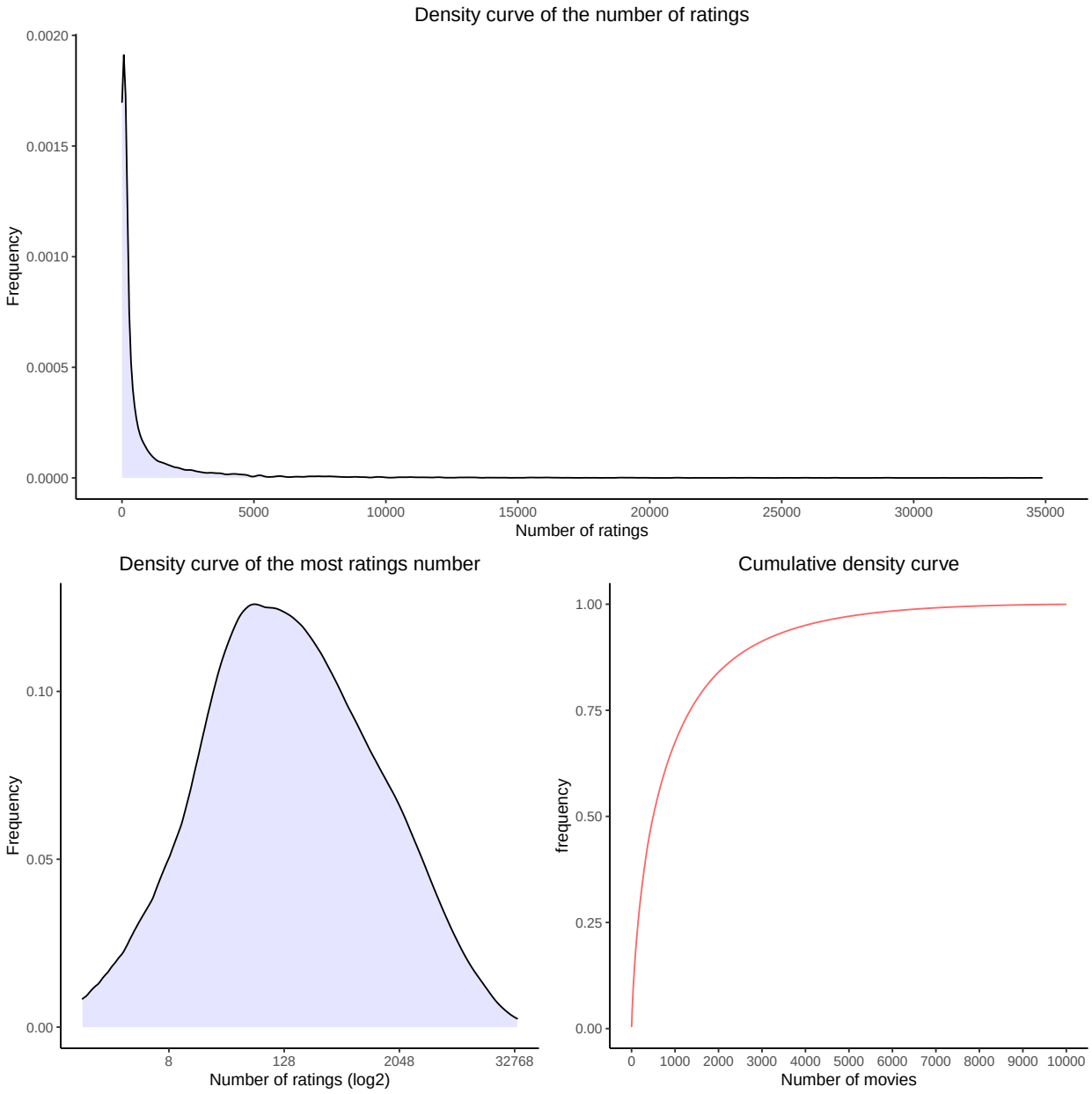
	Value
10%	11.0
25%	34.0
50%	135.0
75%	626.0
90%	2 388.0
95%	4 454.8

Table 14: An overview of the principal statistical variables of the number ratings by movie

min.	median	mean	standard_deviation	max.
1	135	936.6	2 487.33	34 864

In order to complete the analysis, we will represent a density curve of the ratings number by movies. This last one highlights the fact that the majority of movies have got a few ratings number.
We can also represent a cumulative density curve which show us that the 15% of the most rated movies (about 1 500 movies) represent 75% of the ratings number.

Fig. 1 - Density and cumulative density of the number of ratings by movie.



There is a consequence difference between the 20 most rated movies and those which are less. In fact, the 20 most rated movies have got between 23 000 and 34 000 ratings.

Table 15: The 20 most rated movies

Movie title	Ratings number
Pulp Fiction (1994)	34 864
Forrest Gump (1994)	34 457
Silence of the Lambs, The (1991)	33 668
Jurassic Park (1993)	32 631
Shawshank Redemption, The (1994)	31 126
Braveheart (1995)	29 154
Fugitive, The (1993)	28 951
Terminator 2: Judgment Day (1991)	28 948
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	28 566
Apollo 13 (1995)	27 035
Batman (1989)	26 996
Toy Story (1995)	26 449
Independence Day (a.k.a. ID4) (1996)	26 042
Dances with Wolves (1990)	25 912
Schindler's List (1993)	25 777
True Lies (1994)	25 381
Star Wars: Episode VI - Return of the Jedi (1983)	25 098
12 Monkeys (Twelve Monkeys) (1995)	24 397
Usual Suspects, The (1995)	24 037
Fargo (1996)	23 794

In contrast, the 20 least rated movies have got only one rating.

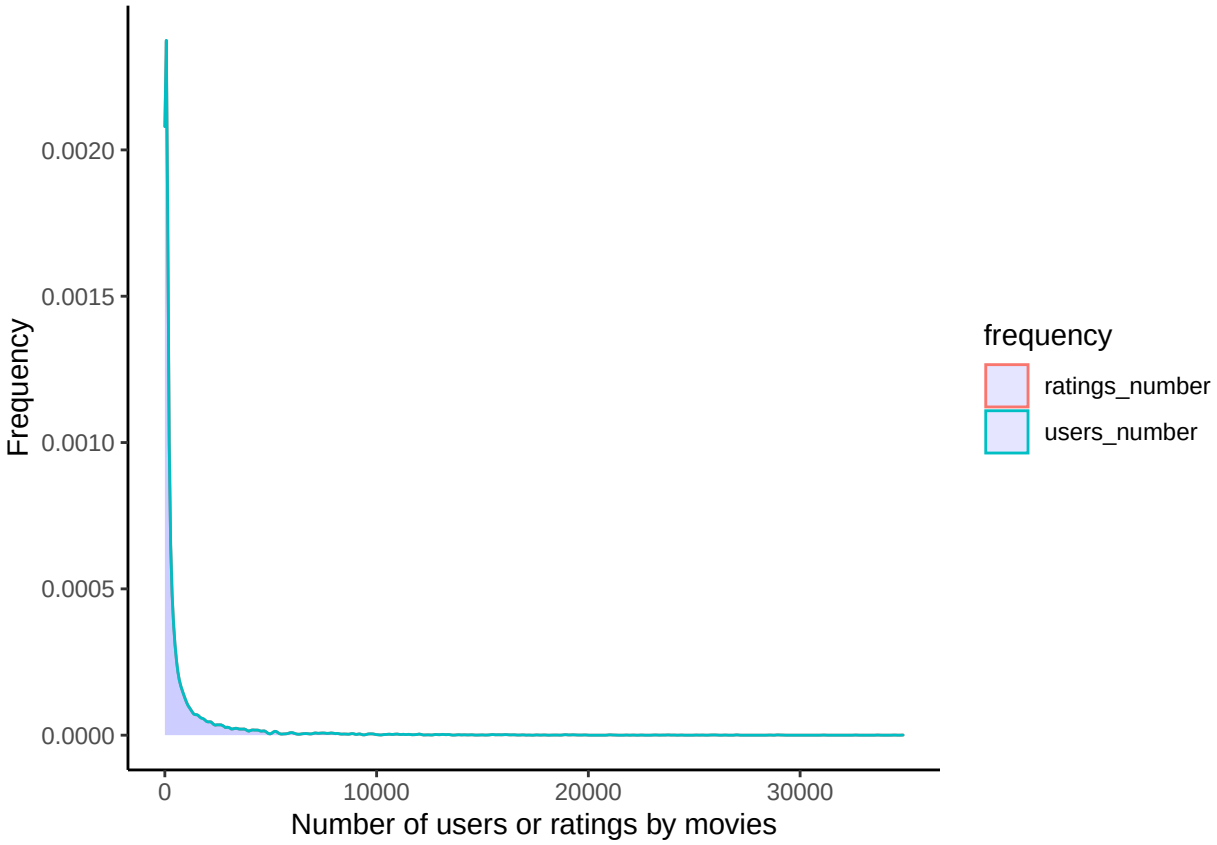
Table 16: The 20 least rated movies

Movie title	Ratings number
100 Feet	1
4	1
Accused (Anklaget)	1
Ace of Hearts	1
Ace of Hearts, The	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di...)	1
Africa addio	1
Bad Blood (Mauvais sang)	1
Battle of Russia, The (Why We Fight, 5)	1
Black Tights (1-2-3-4 ou Les Collants noirs)	1
Blind Shaft (Mang jing)	1
Blue Light, The (Das Blaue Licht)	1
Borderline	1
Brothers of the Head	1
Chapayev	1
Cold Sweat (De la part des copains)	1
Condo Painting	1
Confess	1
Cruel Story of Youth (Seishun zankoku monogatari)	1
David Holzman's Diary	1

In addition to represent the ratings number by movie, we can compare it with the users number by movies.

Let's represent a density plot of the two variables. We can see that their density plot are the same which means that a user rated only one time the same movie.

Fig. 2 - Representation of the density curve of the number of ratings and the number of users by movie.



A.2 - Users distribution

In the previous section, we have seen that the density plot of the ratings number by movie is the same that the user number by movie due to the fact that a user rate just one time the same movie. Therefore, the distribution of the ratings number by users will be the same that the distribution of the movies number by user.
Let's analyse statistical data of the ratings number by user.

The "movielens_reprocessing" data set contains 69 878 users. The mean of the number of ratings by users is about 143 while the median is equal to 69 which means the distribution do not follow a normal law. As of her, the standard deviation is about 216.
We will see that the minimum of ratings by users is equal to 20 while the maximum is equal to 7 359 (cf. table 18).

25% of users have got less than 35 ratings (i.e. first quartile) and 75% of them have got less than 156 ratings (i.e. third quartile). While 95% of users have got less than 512 ratings (cf. Table 17).

Table 17: An overview of some quantiles and deciles of the number ratings by user

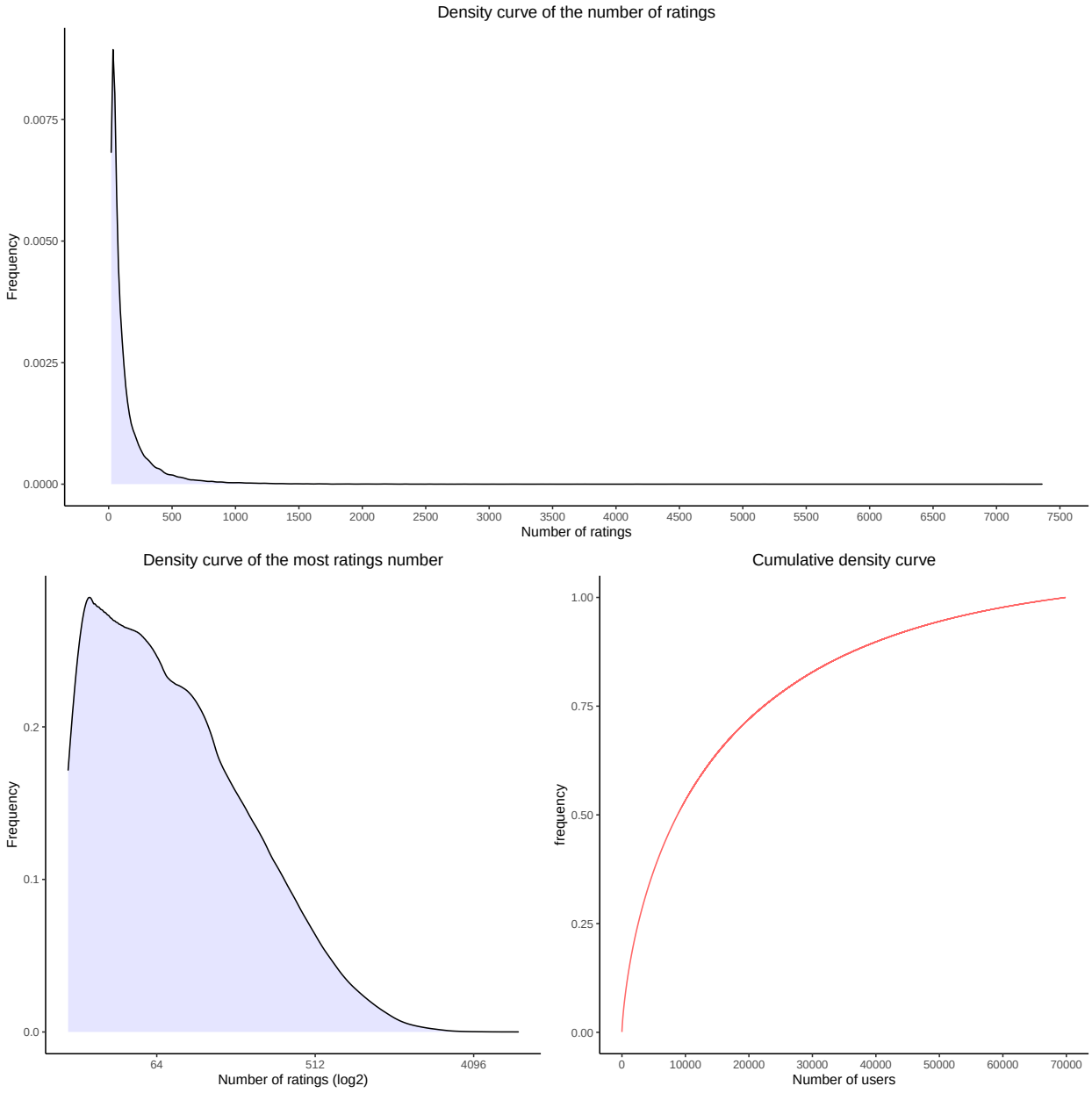
	Value
10%	24
25%	35
50%	69
75%	156
90%	335
95%	512

Table 18: An overview of the principal statistical variables of the number ratings by user

min.	median	mean	standard_deviation	max.
20	69	143.11	216.71	7 359

The density curve below shows us that the most of users have made a few ratings.
In addition, the cumulative sum of the ratings highlights the fact that about 32% of users having the most rated (about 22 300 users) represent 75% of the ratings number.

Fig. 3 - Representation of the density and cumulative density of the number of ratings by user.



A.3 - Ratings distribution

Firstly, we will analyse the distribution of ratings and the ratings number by stars (between 0 and 5), and on the other hand, we will analyse the distribution of the ratings mean by users and by movies.

The analysis of the quantiles and deciles (cf. Table 19) highlights the fact that 25% of the ratings number have got an evaluation of less than 3 stars while about 50% of them have got an evaluation between 3 and 4 stars. Besides, 25% of them have got an evaluation of more than 4 stars.

The table 20 below shows us that the ratings' mean is about 3.51 while the median is equal to 4.
The smallest evaluation is equal to 0.5 stars which is means that no one have given a zero stars to a movie.
The biggest evaluation is equal to 5 stars. To finish, the standard deviation is about 1 stars.

Table 19: An overview of some quantiles and deciles of the ratings given by users

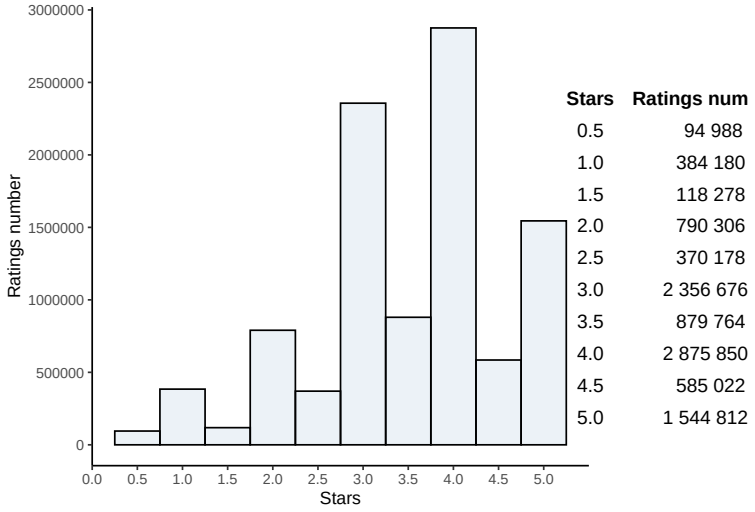
	Value
10%	2
25%	3
50%	4
75%	4
90%	5
95%	5

Table 20: An overview of the principal statistical variables of the ratings given by user

min.	median	mean	standard_deviation	max.
0.5	4	3.51	1.06	5

The histogram below represent the number of ratings by star (cf. Fig. 4). We can see that the 4, 3 and 5 stars are respectively the more used. In general, users tend to give movies high ratings.

Fig.4 - Representation of the number of rating by star



We will analyse the distribution of the ratings by user in order to find out tendencies which will be used to build our predictive algorithms.

The figure 5.1 below represents the distribution of the ratings' mean by user associated to the number of ratings by user. For example, for a rating mean equal to 3 stars, is associated several users who the best of them have rated about 3 700 movies.
It shows us that users having the most ratings number have got an average between 2.5 and 4 stars while users having the lowest ratings number are associated to the extremities of the x-axis (i.e. between 0.5 and 1.5 stars or 4.5 and 5 stars).
In fact, the figure 5.3 highlights the fact that the most of users have got a ratings' mean between 2.5 and 4.5 stars while their standard deviation is between 0.5 and 1.5 stars.

As of her, the figure 5.2 shows us a histogram of the lowest and biggest rating by user. We can see that some users rate movies more harshly or more kindly than others. In fact, each rating, between 0.5 and 5 stars, are associated to a minimum rating and a maximum rating for at least one user. For example, about 12 000 users do not give a rating below of 2 stars.

To conclude, we can make two assumptions. Firstly, some users are tougher or accommodating than others for evaluating movies. Secondly, the more users have rated and the more their ratings' mean is nearly of global ratings' mean of the data set.

Fig.5 - Distributions of the ratings by user.

Fig. 5.1 – Distribution of the ratings mean by user

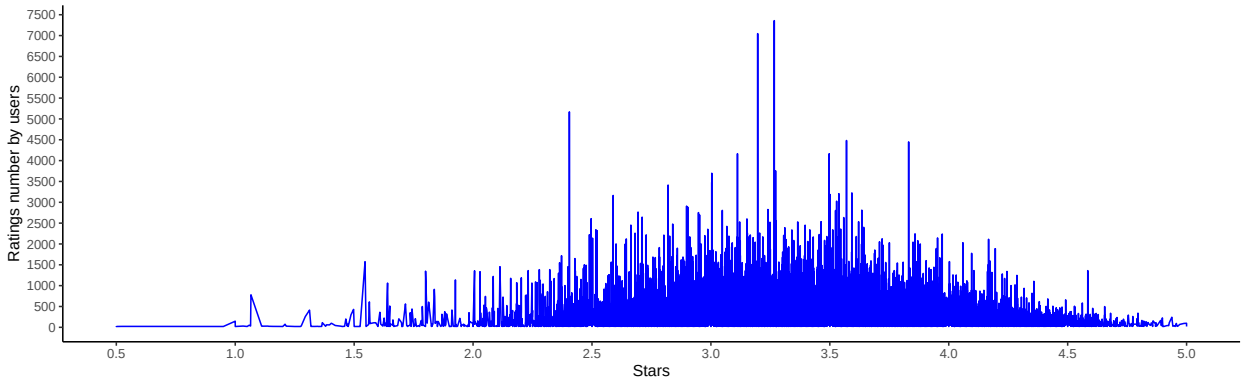
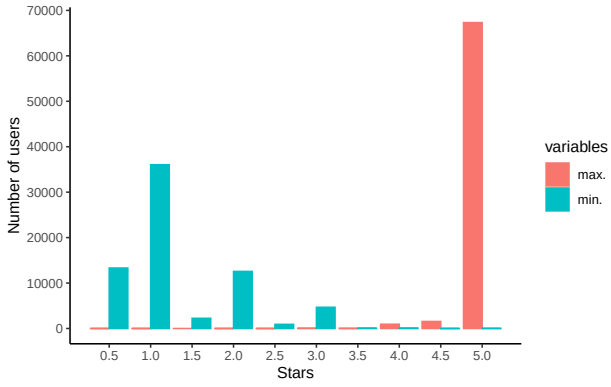
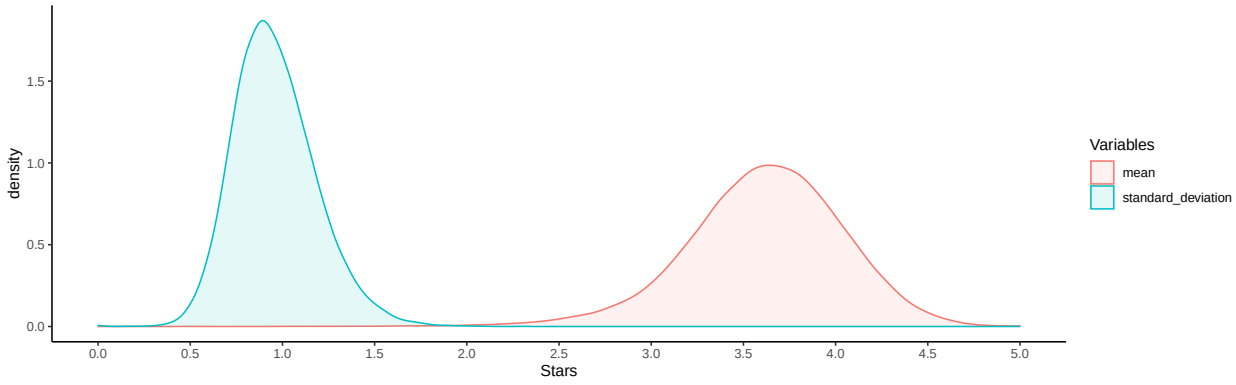


Fig 5.2 – Histogram of the minimum and the maximum ratings by user



Stars	Number of Min.	Number of Max.
0.5	13 306	4
1.0	36 037	4
1.5	2 245	1
2.0	12 563	3
2.5	912	3
3.0	4 671	47
3.5	56	19
4.0	66	937
4.5	2	1 548
5.0	20	67 312

Fig 5.3 – Density curve of the mean of ratings and of the standard deviation by user



We will do the same analyse for the distribution of the ratings by movie.

The figure 6.1, below, represents the distribution of the ratings' mean by movie associated to the number of ratings by movie. It shows us that movies having the most ratings number have got an average between

3 and 5 stars while movies having the lowest ratings number are associated to the extremities of the x-axis (i.e. between 0,5 and 2 stars or 4,5 and 5 stars). If we compare this distribution with this of the ratings' mean by user, we can see there are less movies located on the extremities of the x-axis than that users.
The figure 6.3 highlights the fact that most of movies are a ratings' mean between 1,5 and 4,5 stars. The data range is bigger than the distribution of ratings' mean by user. So it is flatter that this last one.

The figure 6.2 shows us a histogram of the lowest and biggest rating by movie. We can see that most of the movies are a minimum rating and a maximum rating respectively equal to 0,5 stars and 5 stars.

To conclude, we can make an assumption which would be the following : more the movie is rated by user more his mean rating is high.

Fig. 6 - Distributions of the ratings by user.

Fig. 6.1 − Distribution of the ratings mean by movie

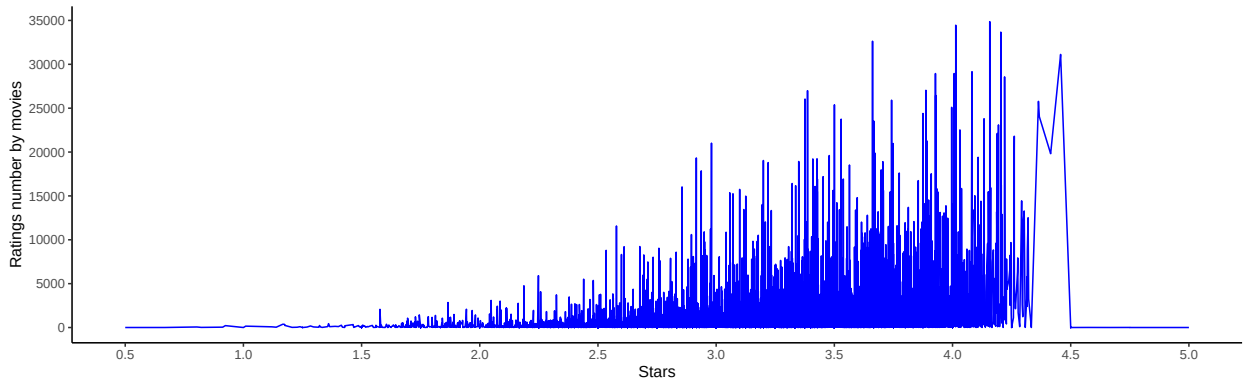
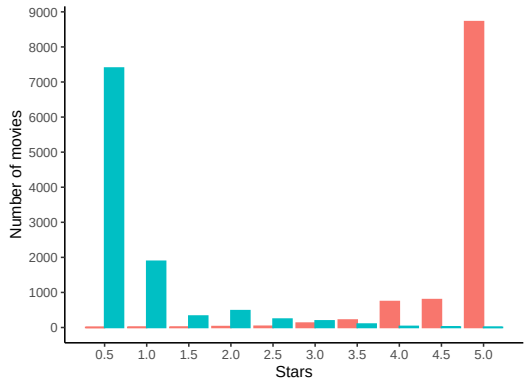
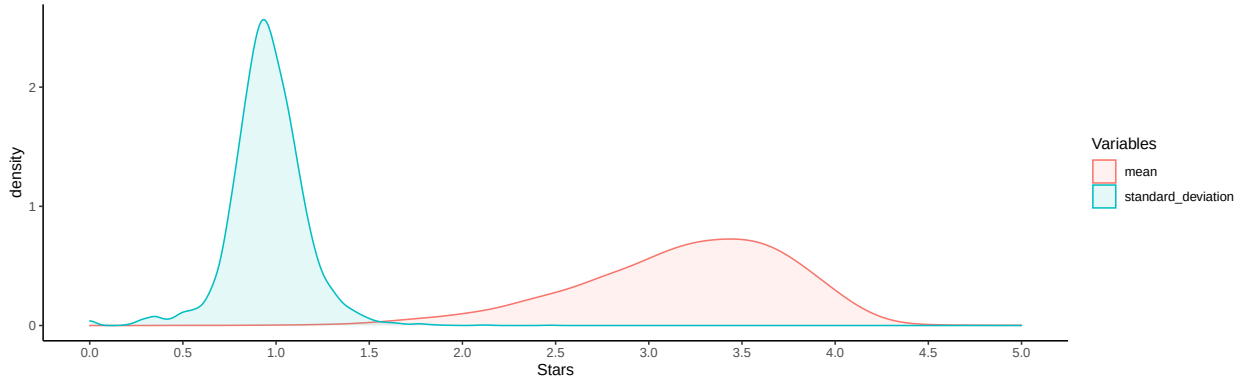


Fig. 6.2 − Histogram of the minimum and the maximum ratings by movie



Stars	Number of Min.	Number of Max.
0.5	7 399	3
1.0	1 889	6
1.5	328	7
2.0	481	27
2.5	241	36
3.0	190	126
3.5	99	214
4.0	32	741
4.5	13	796
5.0	5	8 721

Fig. 6.3 − Density curve of the mean of ratings and of the standard deviation by movie



A.4 - Movies genre distribution

Each movie is associated to a category of movie (i.e. a genre). We will analyse their distribution from the number of ratings, the number of users and movies, and from the ratings' mean.

There are 19 movie genres (cf. Fig. 7) of which each of them have a different distribution.
In fact, we can see that the number of ratings by genre is heterogeneous (cf. Fig. 7.1). The "IMAX" genre has got a very few ratings number and the genres "Documentary", "Film-Noir" and "Western" have got less than 500 000 ratings. While the most ratings number by genre have got about 4 million of ratings (i.e. "Drama" and "Comedy").

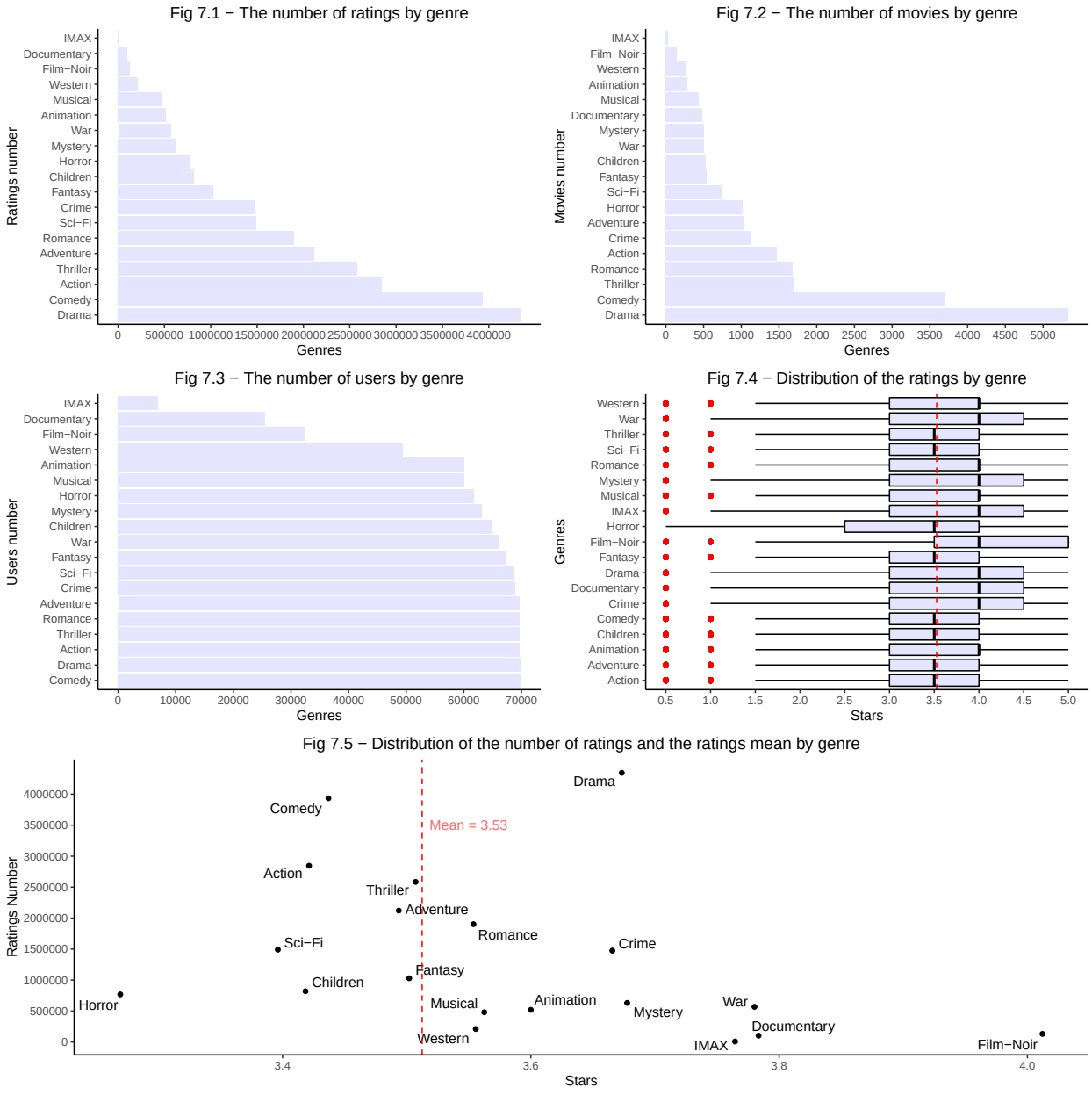
The number of movies by genre is also heterogeneous. Most of them have less or about 1 000 movies while the "Comedy" genre has got more of 3 500 movies and the "Drama" genre has got more of 5 000 movies (cf. Fig. 7.2).
On the other hand, most of genres have a number of users between 60 000 and 70 000 while "IMAX", "Documentary" and "Film-Noir" have less of the half of 70 000 users (cf. 7.3).

The distribution of ratings by genre highlights the fact that most of them have an interquartile between 3 and 4 stars (i.e. 50% of ratings are between 3 and 4 stars), a range between 1.5 and 5 stars and a median of about 3.5 stars (cf. Fig. 7.4).
We can see an exception with the "Film-Noir" genre which have an interquartile between 3.5 and 5 stars with a median equal to 4 stars.
The figure 7.5 show us the distribution of the ratings mean by genre. The mean of all the ratings is represented by a red dashed line. This last is equal to 3.53. We can see that the mean of each genre is between 3.4 and 3.8 stars except for "IMAX" and "Film-Noir" genres.

According to figures 7.4 and 7.5, we can make a first statement on the fact that the common genres (i.e. the most popular genres or the genres which can interested the most categories of users), which have the most ratings number, are not those for which the average of the ratings is the highest.
The figure 7.5 shows us that among the 5 genres having the most ratings number, only one (i.e. "Drama" genre) has got a ratings mean upper to the mean of all the ratings.
This statement is different to the one issued in the previous section (cf. A.3 - Ratings distribution) for which we have said that the more a movie is evaluated the more his ratings' mean is high due to the fact that the more the movie is good the more it will be watched. Nevertheless, it should be kept in mind that the fact a movie is considered as a good movie is subjective and depends on ours wishes and trends.

In fact, we can make a second statement which will be the fact that the specific genres (i.e. those which interested a specific users) are those which are the best rated due to the fact that they are of better quality and they are watched by knowledgeable and critical users.

Fig. 7 - The distribution of genres in terms of several variables



A.5 - Years distribution

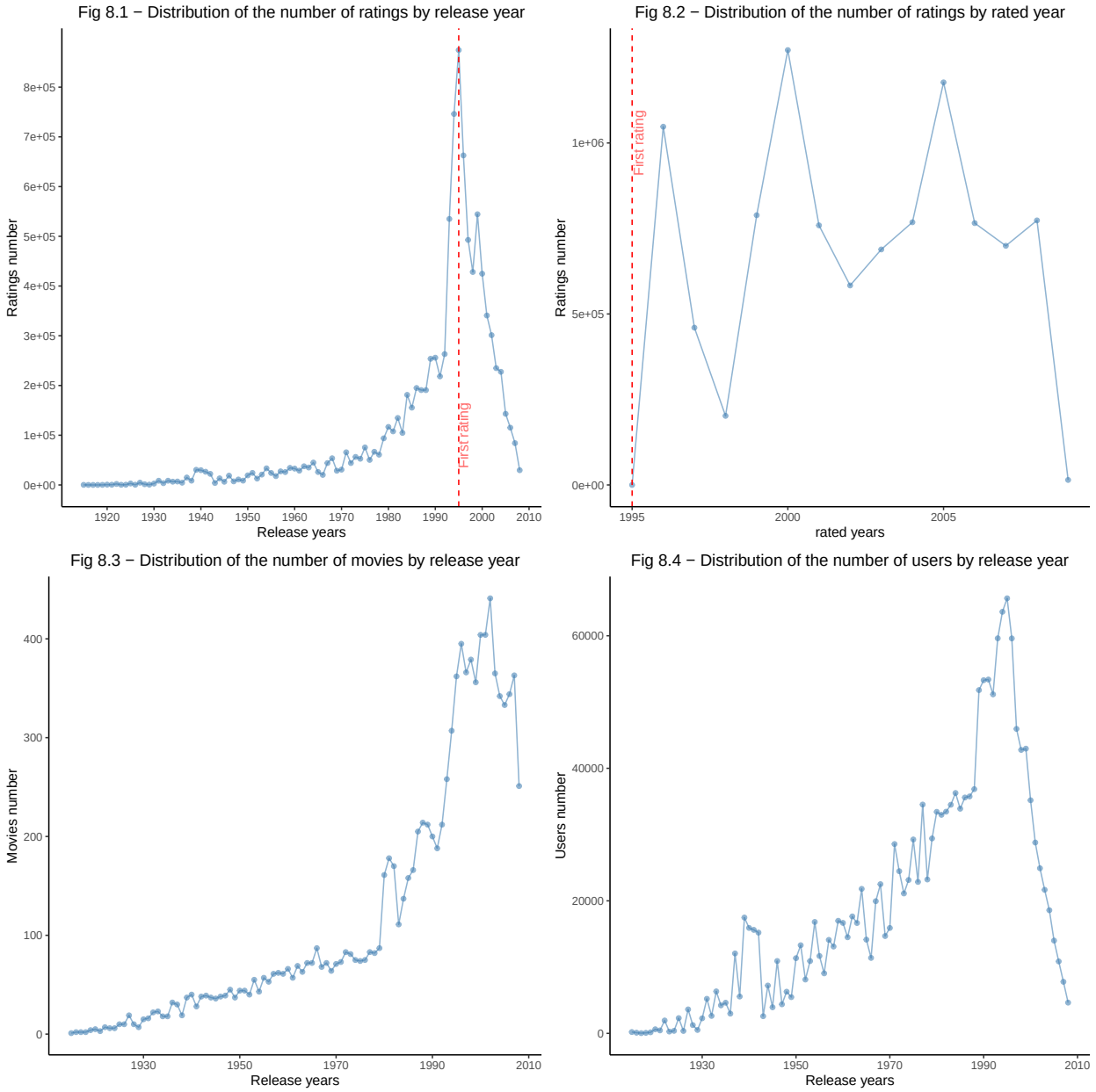
We will analyse the data distribution by release year, rated year and by the age of the movie (i.e. from today to the release year).

The figure 8.1 highlights the distribution of ratings number by release year. We can see that most of the rated movies are a release date between 1990 and 2003. This information can be compared to the date of the first rating by a user which is 1995. It's interesting to see that the year of the first rating is also the year that counts the most ratings, more than 800 000 ratings.
In general, users have rated movies which their release year is close to the year of the first rating. The movies which release year is lower than 1980 and upper 2008 have been much less ratings.

The figure 8.2 shows us the distribution of ratings number by rated year. 50% of ratings number have been carried out between 2000 and 2005 with 3 peaks which have been each more than 1 million of ratings. The last rating has been fulfilled in 2009.

The figure 8.3 represents the number of movies by release year. We can observe that the years with the most movies are between 1994 and 2007 (i.e. more than 300 movies). We can also see two others groups. Firstly, the release years between 1980 and 1993 have got a movies number between 150 and 250. Secondly, the release years lower than 1980 have got less than 100 movies. More the release year is old less the are movies.

Fig. 8 - Distribution of variables by year.



To finish, the figure 8.4 highlights the distribution of the number of users by release year. We can observe that the release year for which the number of users is the most important is 1995, the year of the first rating by a user. The release years between 1989 and 2001 have got a users number upper to 40 000. We can see that more the release year is old less there are users.

B - Analysis of the correlation between variables

We will start by analyzing the correlation between movie genres. Afterwards we will highlight correlation between others variables.

B.1 - Analysis of the correlation between "genres" variable

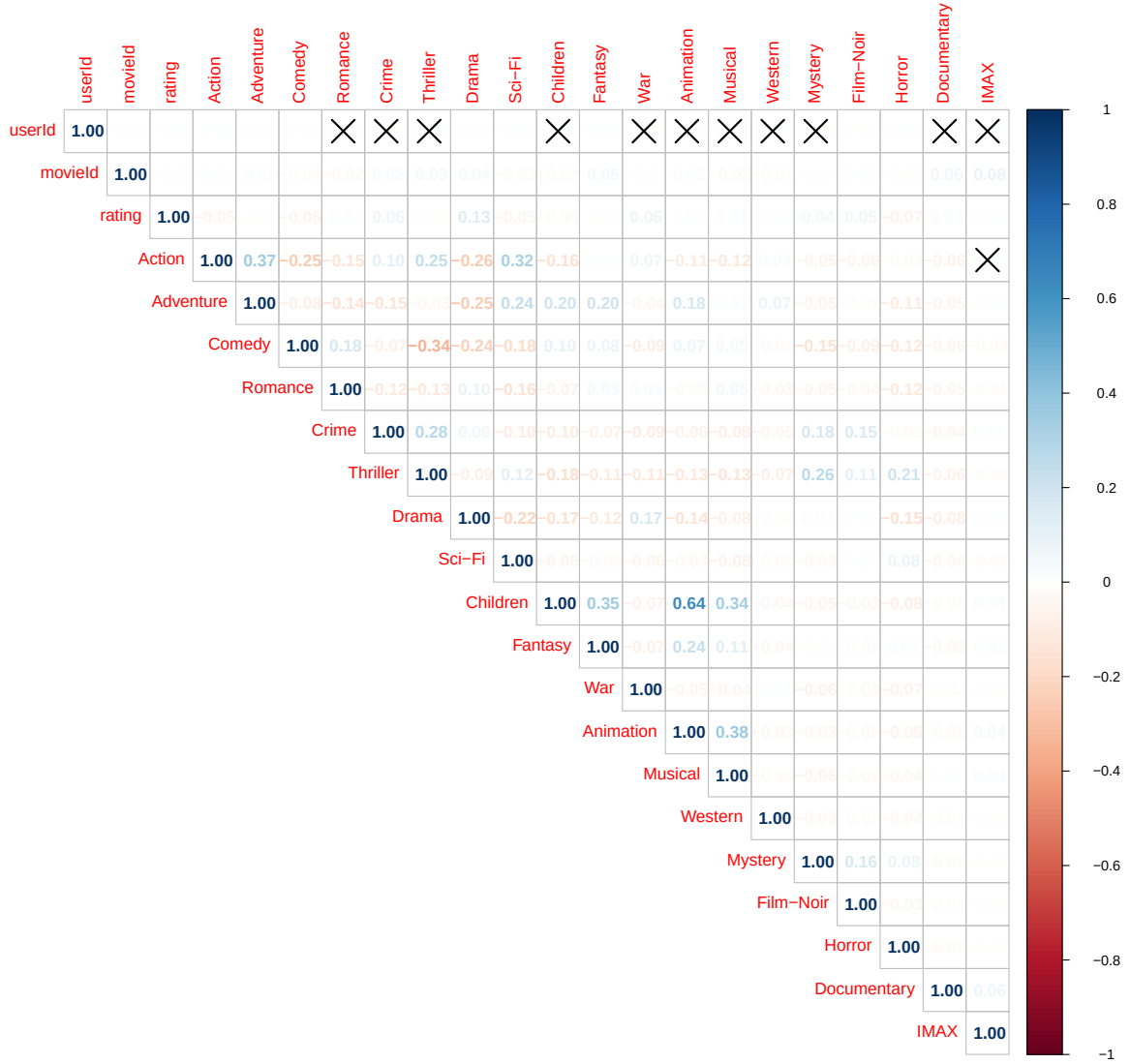
The goal is to observe if we can make groups of genres which are considered as similar by users. In fact, a user like to watch Romance movies should have more probability to watch also Drama movies than Action movies.

In so far as the correlation between movie genres is high (positively or negatively), we could make groups of users according to their preference which should be improve efficiency of our predictive algorithm.

The figure 9 highlights the fact that there are, in general, some evidence of correlation between movie genres. They is a coefficient of correlation between -0.40 and +0.40.
We can observe that the most coefficient of correlation is equal to 0.64 between "Children" and "Animation" genres which means that a user who watch children movies will have more probability to see also animation movies. Instead, a user who aim to watch comedy movies should have less probability to see also thriller movies. In fact, the coefficient of correlation between them is equal to -0.34.

We will see later if the genre effect could improve efficiency of our regularization model.

Fig. 9 - Correlation between movie genres.



B.2 - Analysis of the correlation between others variables

We will also analyse the correlation between ratings number and ratings mean (cf. figure 10.1). We can see that the more the movie was rated the higher the rating and especially the more the movie was rated the less the variability is. Movies which have the least ratings have a range of ratings between 0.5 and 5 stars. It's an important element to consider in order to optimize our regularization model.

Let's observe if the time effect could bias the ratings' mean given by user.
The figures 10.2, 10.3 and 10.4 highlight a strong time effect on the ratings mean. In fact, the coefficient of correlation between the age of movie and ratings mean is equal to 0.58 and the coefficient of correlation between release year and ratings mean is equal to -0.58. The more the movie is old, the more the ratings mean is high. Movies released between 1930 and 1970 have the best means. Then, we can see a steady

decrease of ratings mean from 1960 to 2010. There are less movies referenced in the data set from 1920 to 1990 but in general they are considered as "classics" (i.e. movies that you must watch) movies which are watched from generation to generation. In contrast, the list of movies from 1990 to today contains blockbuster movies watched by millions, artsy movies, bad movies, for which the ratings mean can be more variable.
We can also observe a strong negative correlation between rated year and ratings' mean. The ratings' mean is higher between 1995 and 2000 than others years. As we have mentioned previously, the first rating has been achieved in 1995, which correspond to the fact that there may have been a phenomenon of enthusiasm to rate movie at the beginning and this last has decreased during the following years.
Thereby, we will see later if the time effect improve our regularization model.

Fig. 10 - Correlation between others variables.

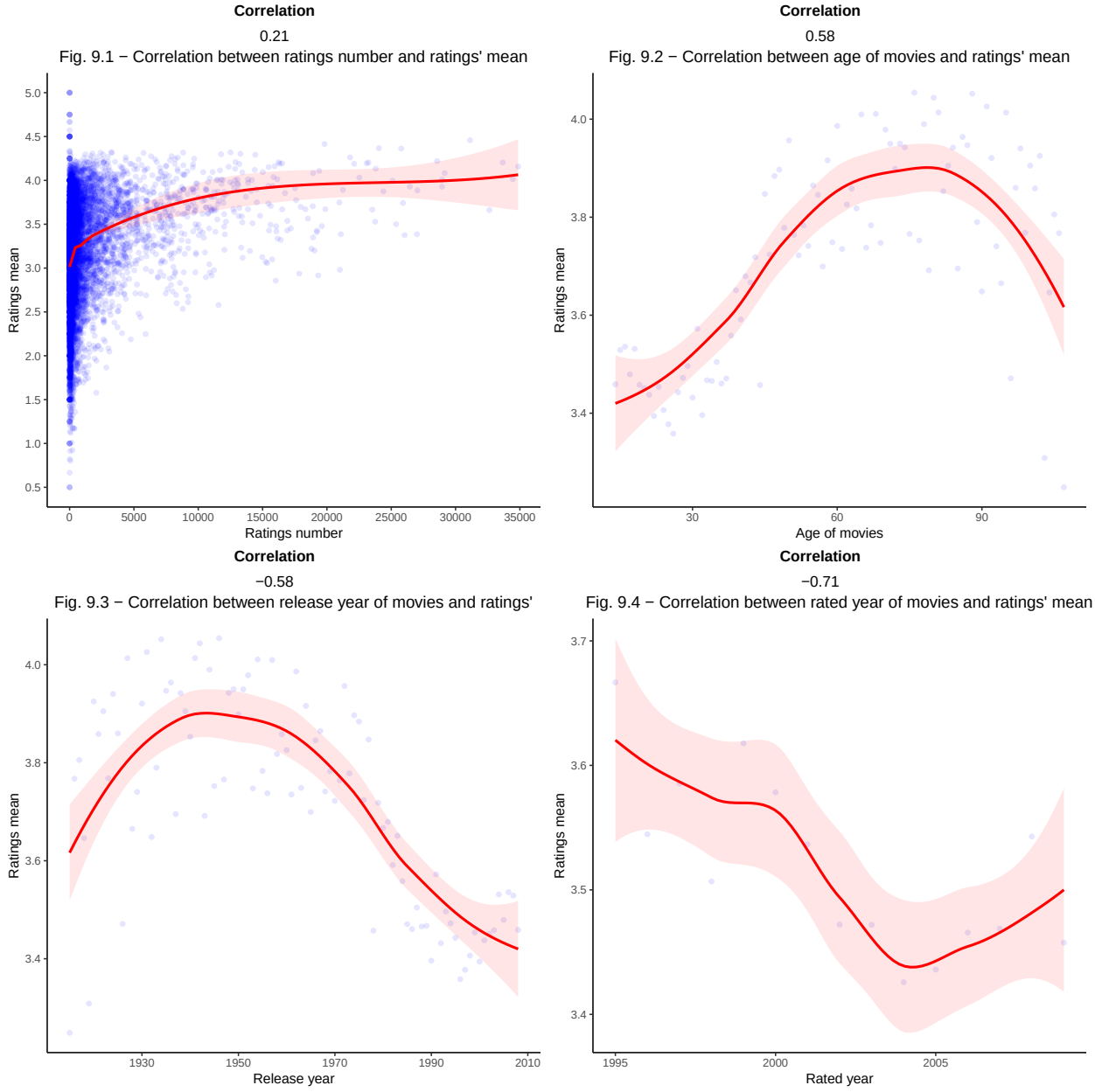


Fig. 9.1 − Correlation between ratings number and ratings' mean

Fig. 9.2 − Correlation between age of movies and ratings' mean

Fig. 9.3 − Correlation between release year of movies and ratings'

Fig. 9.4 − Correlation between rated year of movies and ratings' mean

Part III - Predictive algorithm models

We will build three predictive algorithm models. The first model is based on a loss function and regularization.
The second model will be used parallel matrix factorization from the "Recommenderlab" package while the third model will be based on matrix factorization with parallel stochastic gradient descent from the "Recosystem" package.
We will see which of these models obtains, on the one hand, the weakest Root Mean Square Error (RMSE) and, on the other hand, which is the mos efficient.

In fact, the goal is to build a model for which the RMSE is below of 0.86490.
Let's start by defining the Root Mean Square Error :

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

With :
- $y_{u,i}$ is defined as the rating for movie i by users u;
- $\hat{y}_{u,i}$ correspond to our prediction of rating for a user u and a movie i;
- N which represents the number of user/movie combinations.

We can interpret the RMSE like the standard deviation. It corresponds to the error we could make when our algorithm predicting a movie rating. Thereby, an RMSE equal to 1 means that our prediction error corresponds to one star.
We will implement this function in R as follows :

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

A - Model based on a loss function and regularization

We will build our model from these assumptions which we could make in the previous section (cf. Part II - Data analyzing and data visualization).

A.1 - Assumptions and definition of the variables of our predictive model

Assumption n°1

We will observe that some movies are generally rated higher than others which means that different movies are rated differently. Thereby, we will add a variable to our model in order to take into account a "movie effect" $\hat{b}_i$. We can estimate it given that the least square estimate is equal to the average of $Y_{u,i} - m\hat{i}u$.

We also have seen that some movies get ratings number more than others which means movies that will have a few ratings number have a more uncertainty of predictions. In order to reduce this effect, we will use regularization which permit us to penalize large estimates associated to a small sample sizes. Regularization is similarly to Bayesian approach.
In other words, we are looking for constraining the total variability due to the effect sizes of ratings number by movie.
Our model becomes :

$$\sum_{u,i}(y_{u,i} - \mu - b_i)^2 + \lambda \sum_i b_i^2$$

The first term of this equation corresponds to the sum of squares while the second is a penalty term that become larger when many b_i are consequent.

Thereby, we can estimate the value of b_i that minimize this equation as follows :

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

With n_i represents ratings number by movie i.

In other words, when n_i is very large, the prediction is considered like stable and the penalty term λ will be ignored due to the fact that $n_i + \lambda \approx n_i$.

Assumption n°2

In the previous section, we have highlighted the fact that some users are very harsh in their notation whereas that others users appreciate most of movie which they have watched and thereby give them good ratings. When our model will estimate a rating associated to a harsh user, it will consider, for example, a 3 stars as a 5 stars. This effect is to take account in our model. We Will name it "user effect".
As we can see for movies, it is right to use regularization for the user effect insofar as the more users have rated the less the variability is large. Therefore, our estimate will be more precise.

Assumption n°3

We have seen there was a correlation between movie genres which means a user who love watching action movies should also love watching adventure movies.
We can also make regularization to this effect to penalize genres which have a few ratings number.

Assumption n°4

We have observed that there was a correlation between ratings mean and, on one hand, release year and, on the other hand, rated year. We will add this too effects in our model and we will use regularization for the same reason that movie, user and genre effects.

Our model can be resume as follows :

$$\sum_{u,i}(Y_{u,i} - \mu - b_i - b_u - b_g - b_y - b_{ry})^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2 + \sum_y b_y^2 + \sum_{ry} b_{ry}^2)$$

With :
$y_{u,i}$ is defined as the rating for movie i by users u;
μ represents the ratings mean of the data set;
b_i represents the movie effect;
b_u defined as the user effect;
b_g represents the genre effect;
b_y defined as the release year effect;
b_{ry} defined as the rated year effect;
λ represents the penalty term.

A.2 - Training of the predictive algorithm

Before testing our model, we will define some management rules. First of all, we will train our algorithm from edx data set for which we will split it into "train_set_reg" data set, which will be used to train our predictive algorithms, and "test_set_reg" data set which be used to select the λ parameter which minimize the RMSE.
Secondly, we will used the validation data set to estimate the RMSE and therefore the efficiency of our model.

```
# First of all, we will split edx data set into a training set and a test set in order
# to avoid over training
set.seed(1988, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_set_reg <- edx[-test_index,]
test_set_reg <- edx[test_index,]
  ## We will make sure userId and movieId in test set are also in the training data
test_set_reg <- test_set_reg %>%
  semi_join(train_set_reg, by = "movieId") %>%
  semi_join(train_set_reg, by = "userId")

rm(test_index)
```

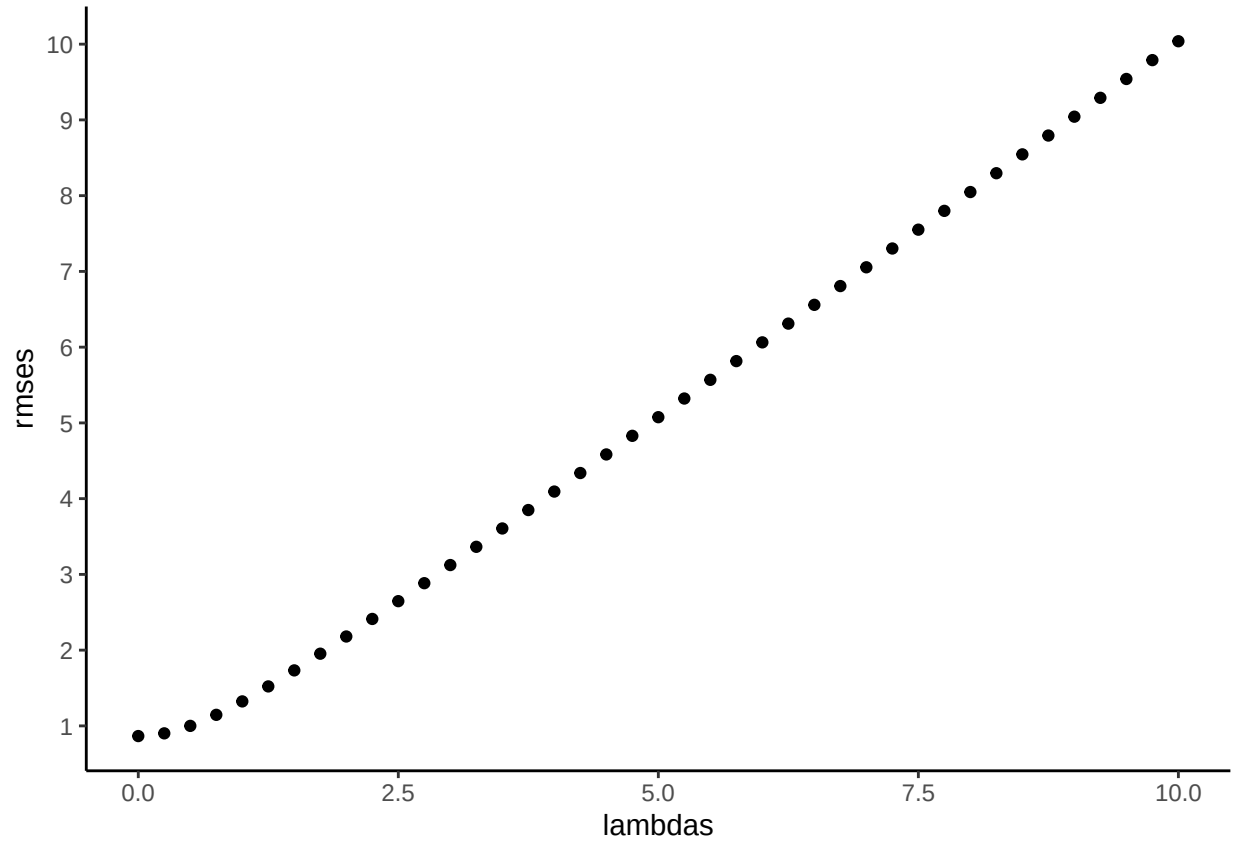
We will use the train_set_reg and test_set_reg data set to find out the value of λ that minimize the RMSE. To do that, we will use the cross-validation method as follows.

```
lambdas <- seq(0, 10, 0.25) #the lambda parameter that minimize the RMSE

rmses<- sapply(lambdas, function(l){
  # We will estimate the true rating for all movies and users
  mu_reg<- mean(train_set_reg$rating)

  # We will estimate the movies effect with a penalty term
  b_i<- train_set_reg %>% group_by(movieId) %>%
    summarize(b_i= sum(rating - mu_reg)/(n()+l))

  # We will estimate the users effect with a penalty term
```

```
  b_u<- train_set_reg %>% left_join(b_i, by="movieId") %>% group_by(userId) %>%
    summarize(b_u= sum(rating - b_i - mu_reg)/(n()+l))

  # We will estimate the release years effect with a penalty term
  b_y<- train_set_reg %>% left_join(b_i, by="movieId") %>%
    left_join(b_u, by= "userId") %>% group_by(year_release) %>%
    summarize(b_y= sum(rating - b_i - b_u - mu_reg) / (n()+l))

  # We will estimate the rated years effect with a penalty term
  b_ry<- train_set_reg %>% left_join(b_i, by="movieId") %>%
    left_join(b_u, by= "userId") %>% left_join(b_y, by= "year_release") %>%
    group_by(year_rated) %>%
    summarize(b_ry= sum(rating - b_i - b_u - b_y - mu_reg) / n()+l)

  # We will estimate the genres effect with a penalty term
  b_g<- train_set_reg %>% left_join(b_i, by="movieId") %>%
    left_join(b_u, by= "userId") %>% left_join(b_y, by= "year_release") %>%
    left_join(b_ry, by= "year_rated") %>% group_by(genres) %>%
    summarize(b_g= sum(rating - b_i - b_u - b_y - b_ry - mu_reg) / n()+l)

  # We will predict the retings
  predicted_ratings_reg<- test_set_reg %>% left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_y, by= "year_release") %>%
    left_join(b_ry, by= "year_rated") %>% left_join(b_g, by= "genres") %>%
    mutate(pred= mu_reg + b_i + b_u + b_y + b_ry + b_g) %>% .$pred

  # We will estimate the RMSE
  return(RMSE(predicted_ratings_reg, test_set_reg$rating))
})
```

We will represent values of RMSE by λ. We can see that the λ that minimize the RMSE is equal to 0.

Fig. 11 - Representation of the value of RMSE by lambda



A.3 - Evaluation of the predictive algorithm

After having define the value of the λ parameter, we can use the edx and validation data sets in order to evaluate the RMSE associated to our regularization model.

```
# We will estimate the RMSE from validation test and
#therefore we will evaluate the efficiency of our model
# the true rating for all movies and users
mu<- mean(edx$rating)

# We will estimate the movies effect with a penalty term
b_i<- edx %>% group_by(movieId) %>% summarize(b_i= sum(rating - mu)/(n()+lambda))

# We will estimate the users effect with a penalty term
b_u<- edx %>% left_join(b_i, by="movieId") %>% group_by(userId) %>%
  summarize(b_u= sum(rating - b_i - mu)/(n()+lambda))

# We will estimate the release years effect with a penalty term
b_y<- edx %>% left_join(b_i, by="movieId") %>%
  left_join(b_u, by= "userId") %>% group_by(year_release) %>%
  summarize(b_y= sum(rating - b_i - b_u - mu) / (n()+lambda))

# We will estimate the rated years effect with a penalty term
```

```
b_ry<- edx %>% left_join(b_i, by="movieId") %>%
  left_join(b_u, by= "userId") %>% left_join(b_y, by= "year_release") %>%
  group_by(year_rated) %>%
  summarize(b_ry= sum(rating - b_i - b_u - b_y - mu) / n()+lambda)

# We will estimate the genres effect with a penalty term
b_g<- edx %>% left_join(b_i, by="movieId") %>% left_join(b_u, by= "userId") %>%
  left_join(b_y, by= "year_release") %>%
  left_join(b_ry, by= "year_rated") %>% group_by(genres) %>%
  summarize(b_g= sum(rating - b_i - b_u - b_y - b_ry - mu) / n()+lambda)

# We will predict the ratings
y_hat<- validation %>% left_join(b_i, by= "movieId") %>% left_join(b_u, by= "userId") %>%
  left_join(b_y, by= "year_release") %>%
  left_join(b_ry, by= "year_rated") %>% left_join(b_g, by="genres") %>%
  mutate(y_hat= mu + b_i + b_u + b_y + b_ry + b_g) %>% .$y_hat

RMSE_reg<- RMSE(y_hat,validation$rating)
```

Table 21: Result of our predictive algorithm model

Model	RMSE
Model based on a loss function and regularization	0.8645954

The RMSE of this first predictive model based on a loss function and regularization is equal to 0.8645954 below of the target of 0.86490. However, this model didn't succeed to explain the entire variability and connection between variables.

We will develop an another model in order to catch them and to improve the RMSE result.

B- Model based on parallel matrix factorization

We will build an predictive algorithm based on the parallel matrix factorization from the "Recommenderlab" package.
Matrix factorization method [1] permits to approximate the whole rating matrix R_{m*n} from the product of two matrices of lower dimensions, P_{n*k} and Q_{n*k} such that :

$$R \approx PQ'$$

Where :
- k is the pre-specified number of latent factors;
- m represents users;
- n represents movies.

In other words, matrix factorization permits to evaluate the affinity between a user and a movie which is defined by the inner product of their latent-factor vectors.

Therefore, the rating given by user u on item v would be predicted as $p_u q'_v$ if p_u is considered as the u-th row of P and q_v is the v-th row of Q.

A solution for P and Q will be given by the model describe bellow :

$$\min_{P,Q} \sum_{(u,v)\in R} \left[f(p_u, q_v; r_{u,v}) + \mu_P ||p_u||_1 + \mu_Q ||q_v||_1 + \frac{\lambda_P}{2} ||p_u||_2^2 + \frac{\lambda_Q}{2} ||q_v||_2^2 \right]$$

With :
- (u, v) are locations of observed entries;
- $r_{u,v}$ is considered as observed rating;
- f represents the loss function and;
- μ_P, μ_Q, λ_P and λ_Q represent penalty terms in order to avoid overfitting.

We will use a recommender system to predict unknown entries in the rating matrix based on observed values. First of all, we will reprocess and transform the data set into a matrix.

B.1 - Reprocessing of the data set

Due to the size of the movielens data set and therefore execution times which could cause if we train our algorithm on it, we will select a part of him as follows.
We will select movies which have least 5 000 ratings and users who have least 20 ratings. Then, we will select only the "userId", "movieId" and "rating" variables.
We transform this selection in order to each movieId be represented in column and user be represented in row. Finally, we transform this data set into a specific format expected by the recommender system which is "realRatingMatrix".

[1] Chin, Yuan, et al. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. Journal of Machine Learning Research 17 (2016). p. 3.

```r
# selection of movies and users
movielens_recommender<- movielens %>% group_by(movieId) %>%
  filter(n() >= 5000) %>% ungroup() %>%
  group_by(userId) %>% filter(n() >= 20) %>% ungroup() %>%
  select(userId, movieId, rating)

# presentation of movies by column
movielens_recommender<- movielens_recommender %>% spread(movieId, rating)

movielens_recommender<- movielens_recommender %>% as.matrix()

# Transformation into a matrix
rownames(movielens_recommender)<- movielens_recommender[,1]
movielens_recommender<- movielens_recommender[,-1]

movielens_recommender<- as(movielens_recommender, "realRatingMatrix")
```

It is right to precise that a rating associated to a user u and an movie v is considered as a "known" value. Otherwise, when there is not a rating associated to a user u and an movie v then it consider as an "unknown" value.

B.2 - Training of the predictive algorithm

We will use the "evaluationScheme" function and cross validation to train our algorithm and select the tune parameter (i.e. given parameter) which minimize the RMSE.
This function creates three data sets. The first one, whose name is "train", permits to train the model. The second, whose name is "known" test set, is composed of known ratings which the number of ratings by user is specified by the given parameter. The third, whose name is is "unknown" test set, is used to evaluate the predictions made using "known" test set.
Let's explain the function parameters :
- the k parameter specifies the number of cross validation folds;
- the given parameter specifies the ratings number to use from the test set for evaluating the model.

The recommender function permits to build a recomender model based on matrix factorization (i.e. the "LIBMF" parameter) from the "train" data set.
Afterwards, we will use predict function in order to make prediction of ratings from the "known" data set. The "calcPredictionAccuracy" function permits to evaluate the prediction error for a recommendation system from the "unknown" test data set which is compared to the predictions made.

```r
# choose the best given parameter
given<- c(-10,-1,5,10,15)
best<- sapply(given, function(g){
  eval<- evaluationScheme(movielens_recommender, method= "cross", k= 10, given= g)
  rec <- Recommender(getData(eval, "train"), "LIBMF")
  predictions<- predict(rec, getData(eval, "known"), type="ratings")
  acc<- calcPredictionAccuracy(predictions, getData(eval, "unknown"))
})

# Value of given that minimize the RMSE
best_given<- given[which.min(best[1,])]
```

Table 22: Results of RMSE in terms of given parameter

	-10	-1	5	10	15
RMSE	0.8405638	0.8186752	0.9548999	0.8935996	0.8695487
MSE	0.7065475	0.6702292	0.9118338	0.7985202	0.7561150
MAE	0.6485322	0.6347779	0.7361392	0.6894309	0.6702099

The tuning given parameter that minimize the RMSE is equal to -1. We will implement it in our model in order to evaluate the RMSE.

B.3 - Evaluation of the predictive algorithm

We will evaluate the RMSE of our model from the best tune parameter that we have evaluate in the previous step.
We repeat the same process, step by step.

```
eval<- evaluationScheme(movielens_recommender, method= "cross", k= 10, given= best_given)
rec <- Recommender(getData(eval, "train"), "LIBMF")
predictions<- predict(rec, getData(eval, "known"), type="ratings")
acc<- calcPredictionAccuracy(predictions, getData(eval, "unknown"))
```

On the other hand, we can see that the model based on parallel matrix factorization permits to reduce the RMSE compared to that associated to the regularization model. On the other hand, the execution time is important since then the data set is consequent due to a sequential approach. Let's see if we can reduce the RMSE and the execution time using a a model based on matrix factorization with parallel stochastic gradient descent algorithm from the "Recosystem" package.

Table 23: Results of our predictive algorithm models

Model	RMSE
Model based on a loss function and regularization	0.8645954
Model based on parallel matrix factorization	0.8189507

To conclude, we can also use the predict function to make movie recommendations to specific users. We will analyse the top 10 movies recommendations to the first 3 users.

```
# We will make movie recommendations
LIBMF_reco<- rec %>% predict(getData(eval, "known"), n= 10)
reco_users<- as(LIBMF_reco, "list") %>% head(3)
```

Table 24: Movies recomendations for the first user

title	genres
Office Space	Comedy|Crime
Pulp Fiction	Comedy|Crime|Drama
Goodfellas	Crime|Drama
Shawshank Redemption, The	Drama
Full Metal Jacket	Drama|War
Good, the Bad and the Ugly, The (Buono, il brutto, il cattivo, Il)	Action|Adventure|Western
Devil's Advocate, The	Drama|Horror|Thriller
True Romance	Action|Crime|Romance
Sin City	Action|Crime|Film-Noir|Thriller
Batman Begins	Action|Crime

Table 25: Movies recomendations for the second users

title	genres
Terminator 2: Judgment Day	Action|Sci-Fi
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars)	Action|Adventure|Sci-Fi
Godfather, The	Crime|Drama
Toy Story	Adventure|Animation|Children|Comedy|Fantasy
Schindler's List	Drama|War
Wizard of Oz, The	Adventure|Children|Fantasy|Musical
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)	Action|Adventure
Terminator, The	Action|Sci-Fi|Thriller
Die Hard	Action|Crime|Thriller
Shawshank Redemption, The	Drama

Table 26: Movies recomendations for the third users

title	genres
Princess Bride, The	Action|Adventure|Comedy|Fantasy|Romance
Saving Private Ryan	Action|Drama|War
Sixth Sense, The	Drama|Mystery|Thriller
Good Will Hunting	Drama|Romance
Rain Man	Drama
American History X	Crime|Drama
Green Mile, The	Crime|Drama|Fantasy
Wallace & Gromit: A Close Shave	Animation|Children|Comedy
Amelie (Fabuleux destin d'AmÃ©lie Poulain, Le)	Comedy|Romance
Life Is Beautiful (La Vita Ã¨ bella)	Comedy|Drama|Romance|War

C - Model based on matrix factorization with parallel stochastic gradient descent

This model also use matrix factorization [2] which finds out two matrices $P \in \mathbb{R}^{m*n}$ and $Q \in \mathbb{R}^{k*n}$ such that $r_{u,v} \simeq p_u^T q_v, \forall u, v \in \Omega$, given an incomplete matrix $R \in \mathbb{R}^{m*n}$.

Where :
- Ω means the indices of the existing elements;
- $r_{u,v}$ represents the element at the uth row and the vth column;
- $p_u \in \mathbb{R}^k$ represents th uth column of p;
- $q_v \in \mathbb{R}^k$ represents th vth column of q;
- k represents the pre-specified number of latent features.

$$\min_{P,Q} \sum_{(u,v)\in\Omega} (r_{u,v} - p_u^T q_v)^2 + \lambda(||p_u||^2 + ||q_v||^2)$$

With :
- λ is a regularization model for avoiding over-fitting;
- $||.||$ is the Euclidean norm;
- $(u, v) \in \mathbb{R}$ indicates that rating $r_{u,v}$ is available.

It is right to note that the process to solve P and Q is referred to as the training process. In order to evaluate the efficiency of the used solver, we can use some known elements as missing in the training process and collect them as the test set. Since P and Q are found out, the RMSE on the test set will be used to evaluate the model. It is defined as :

$$\sqrt{\frac{1}{\Omega_{test}} \sum_{(u,v)\in\Omega test} e_{u,v}^2}, e_{u,v} = r_{u,v} - p_u^T q_v$$

with Ω_{test} represents the indices of the elements associated to the test set.

Stochastic gradient method is an iterative process used to solve :

$$\min_{P,Q} \sum_{(u,v)\in\Omega} (r_{u,v} - p_u^T q_v)^2 + \lambda(||p_u||^2 + ||q_v||^2)$$

At each step, a single element $r_{u,v}$ is sampled to obtain the following equation :

$$(r_{u,v} - p_u^T q_v)^2 + \lambda(||p_u||^2 + ||q_v||^2$$

The gradient of the equation above is :

$$g_u = \frac{1}{2}(-e_{u,v} q_v + \lambda p_u), h_v = \frac{1}{2}(-e_{u,v} p_u + \lambda q_v)$$

Afterwards, the model is updated along the negative direction of the sampled gradient :

$$p_u \leftarrow p_u - \eta g_u, q_v \leftarrow q_v - \eta h_v$$

with η is the learning rate.

[2] Chin, Yuan, et al. A Learning-rate Schedule for Stochastic Gradient Methods to Matrix Factorization. Department of Computer Science, National Taiwan University, Taipei, Taiwan. pp. 1 - 2 (2015).

The stochastic gradient descent method permits to fix the learning rate as a constant while some schedules dynamically adjust η in the training process for faster convergence. This process should be accelerate the training from matrix factorization.

C.1 - Reprocessing of the data set

We will create a train and test data set from, respectively, edx data set and validation data set. However, we will select only "userId","movieId" and "rating" variables.

```
# We will set a train and a test set
train_set <- edx %>% select(userId, movieId,rating) %>% as.matrix()
test_set <- validation %>% select(userId, movieId,rating) %>% as.matrix()

# We will save them on hard disk
write.table(train_set , file = "train_set.txt" , sep = " " , row.names = FALSE,
            col.names = FALSE)
write.table(test_set, file = "test_set.txt" , sep = " " , row.names = FALSE,
            col.names = FALSE)

invisible(invisible(gc())) # for cleaning memory


train_set <- data_file( "train_set.txt")
test_set <- data_file( "test_set.txt")
```

C.2 - Training of the predictive algorithm

We will start by create a model object which contains all the model data.

```
r = Reco()
```

Afterwards, we will use tune function, which is based on cross validation, in order to select the best tuning parameter before training our predictive algorithm, with :
- "dim" represents the latent factors number;
- "lrate" which represents the learning rate. As we seen it before, it can be thought of as the step size in gradient descent;
- "costp_l1" which is a regularization term for user;
- "costq_l1" represents a regularization term for movie;
- "nthread" wich is the number of threads for parallel computing;
- "niter" represents the number of iterations.

```
opts = r$tune(train_set, opts = list(dim = c(10, 20, 30),
                                     lrate = c(0.1, 0.2), costp_l1 = 0,
                                     costq_l1 = 0, nthread = 1, niter = 10))
```

We can observe in the table below the parameters that minimize the loss function.

Table 27: Best tuning parameters

dim	costp_l1	costp_l2	costq_l1	costq_l2	lrate	loss_fun
30	0	0.01	0	0.1	0.1	0.7976836

We will train our recommendation model from train data set and the best tuning parameters which we have identified above.

```
r$train(train_set, opts = c(opts$min, nthread = 1, niter = 20),
        out_model = file.path(getwd(), "model.txt"))

invisible(invisible(gc())) # for cleaning memory
```

We will use predict function to predict unknown ratings from the test set.

```
r$predict(test_set, file.path(getwd(), "pred.txt"))
```

Then, we will save the ratings prediction, found out with the predict function, as a vector which name is "pred_ratings". We do the same for the real ratings from the "test_set" data set. The real ratings will be saved into a vector which name is "real_ratings".

```
real_ratings <- read.table("test_set.txt", header = FALSE, sep = " ")$V3
pred_ratings <- scan("pred.txt")
```

C.3 - Evaluation of the predictive algorithm

To finish, we can use the RMSE function, which we have built later, to evaluate the efficiency of our model, i.e. the RMSE.

```
result_recosystem <- RMSE(real_ratings,pred_ratings)
```

We note that the RMSE is equal to 0.78 and the efficiency of the model permit to drastically reduce the execution time of calculus. In fact, we could train and evaluate the model from the entire Movielens data set whereas we had to reduce the data set to train and evaluate the model based on parallel matrix factorization. The parallel stochastic gradient descent permits to be more efficient.

Table 28: Results of our predictive algorithm models

Model	RMSE
Model based on a loss function and regularization	0.8645954
Model based on parallel matrix factorization	0.8189507
Model based on matrix factorization with parallel stochastic gradient descent	0.7820620

Conclusion

The construction of each of these models allowed us to test different methods and principles. They all obtained a RMSE value below than the objective of 0.8649. The fist model based on a loss function and regularization obtained a RMSE equal to 0.8646 but this model didn't succeed to explain the entire variability and connection between variables.

The second model based on parallel matrix factorization obtained a RMSE value equal to 0.8189 which is much better than the first model. However, the efficiency associated to the execution time of calculus is weak. Thereby, we had to drastically reduce the data set in order to succeed to execute calculus. Despite that, the execution time remains important, about 20 minutes.

The third model based on matrix factorization with parallel stochastic gradient descent is considered as the best model because of the RMSE value, which is equal to 0.7821, and his efficiency. In fact, the parallel stochastic gradient descent method permits to drastically reduce the execution time of calculus.

References

Chin, Yuan, et al. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. Journal of Machine Learning Research 17 (2016). p. 3.

Chin, Yuan, et al. A Learning-rate Schedule for Stochastic Gradient Methods to Matrix Factorization. Department of Computer Science, National Taiwan University, Taipei, Taiwan. pp. 1 - 2 (2015).

Robert M. Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. ACM SIGKDD Explorations Newsletter 9, 2 (2007), 75–79.

Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. 2008. Coordinate Descent Method for Large-scale L2-loss Linear SVM. Journal of Machine Learning Research 9 (2008), 1369–1398.