A movies recommender system using Flixster data

Practical assignment of Advanced Topics in Data Science/Data Mining II

By Nuno Gomes and Robson Teixeira

M:DS/CC – FCUP, 17/05/2020

Contents

Abstract

We report on the methodology we adopted to create a recommendation system for films using the Flixster data set, and on the results we obtained. Our model is based on the assumption that if a group of users had liked the same films in the past, they will like similar movies in the future. Hence, if two users have a similar rating history (movies and ratings) and one of them has recently enjoyed a film that the other has not seen yet, then that movie is proposed to the latter. The recommendation system uniquely takes into account the user ratings, and does not consider the characteristics of the films. We were able to recommend films to a randomly selected user using the item-based collaborative filtering, the user-based collaborative filtering, and the popular techniques, using both a binary and a non-binary approach. We were also able to build a recommendation system based on an association-rule mining technique. Using receiver operating characteristic (ROC) curves, the area under the curves (AUC), and precision-recall curves to evaluate and compare the models, we verified that, at least with the Flixster data set, the popular method performs the best as a recommender system.

Chapter 1

Introduction

Recommender or **recommendation** systems are a branch of *Web usage mining* that aim at predicting the "preferences" or the "rating" a user would give to an item. They are widely used on the Web, mainly in on-line streaming services, such as YouTube, Amazon, or Netflix (just to name a few), and e-commerce applications (eBay, Amazon, OLX, *etc.*), in order to recommend products and services to the users. They serve three important functions: (*i*) to increase the profit of companies, (*ii*) to help users to select specific products within the available offer, by giving them personalised recommendations based on previous interactions, and (*iii*) to predict the rating for a new item. Putting it simple, recommendation systems aim at offering the right content to the right customer, at the right time, and through the right channel (Lesmeister and Chinnamgari 2019). Companies struggle for customer loyalty on a daily basis, and while doing so, they invest on recommender systems that try to increase the likelihood of purchase, by analysing customers' preferences and past interactions.

The importance of recommender systems on the success of businesses and on company-customer relationships can be inferred from the one million dollar prize that Netflix offered in 2006 to the person or team that could improve their recommendation system by at least 10 % ("The Netflix Prize" 2009). The success of companies such as Amazon or platforms like YouTube lies partly in the recommendation and marketing strategies, based on the user preferences.

During this project, we created a film recommendation system using the Flixster data set, based on binary and non binary approaches, and according to the *item-based collaborative filtering* (IBCF), *user-based collaborative filtering* (UBCF), and *popular* methods, and we built a recommendation system based on association-rules using the *Apriori* algorithm.

The remainder of this report is organised as follows: in chapter 2, we describe the Flixster data set; chapter 3 is dedicated to the exploration and engineering of the data; the IBCF, UBCF, and Popular methods, and their application to the Flixster data set are presented in chapter 4; those methods are compared in chapter 5; and in chapter 6 we describe how we built a recommendation system based on an association rule mining technique. We finalise with the main conclusions and indications for future work.

Chapter 2

The Flixster data set

Flixster was an American social-networking on-line service founded by Joe Greenstein and Sarah Chari in 2006. The platform allowed users to learn about films, to watch trailers, to share their ratings of movies, to discover new films based on their tastes and past views, and to know and meet other users with similar tastes in films. The company was bought by Fandango in 2016, and the website was shut down in February 2018 in the USA, and October 2019 internationally.

The data set used in this study was provided by the Flisxter website, and it can still be found on-line.[1] It consists of 8 196 077 ratings of 48 794 films by 147 612 users. The data are distributed in three files:

- **movie-names.txt** — a file containing a collection of 66 720 film titles and corresponding identification tags;
- **profile.txt** — a file with 1 002 796 instances, containing general information regarding the users, including user ID, gender, age, location, for how long the user has been a member, and the code tags for the last login and the profile view;
- **Ratings.time.txt** — the aforementioned data set, with 8 196 077 ratings, including the user ID, the movie ID, and the date the rating was registered.

The three data files were loaded into the variables `movies`, `profiles`, and `ratings`, respectively. A summary of the structure of each of them and a glimpse of their first rows is provided below.

```
## Rows: 66,730
## Columns: 2
## $ moviename <chr> "$ (Dollars) (The Heist)", "$5 a Day (Five Dollars a Day)...
## $ movieid   <int> 252, 253, 1, 254, 255, 256, 257, 2, 258, 3, 4, 5, 259, 26...
```

```
##    variable q_zeros p_zeros q_na p_na q_inf p_inf      type unique
## 1 moviename       0       0    0    0     0     0 character  66730
## 2   movieid       0       0    0    0     0     0   integer  66730
```

```
## # A tibble: 6 x 2
```

[1]https://sites.google.com/view/mohsenjamali/flixter-data-set

```
##   moviename                                              movieid
##   <chr>                                                    <int>
## 1 $ (Dollars) (The Heist)                                    252
## 2 $5 a Day (Five Dollars a Day)                              253
## 3 $9.99                                                        1
## 4 $windle (Swindle)                                          254
## 5 &#034;BBC2 Playhouse&#034; Caught on a Train               255
## 6 &#034;Independent Lens&#034;  Race to Execution            256
```

The previous output represents a glimpse of the `movies` *tibble*, which contains 66 730 records and two variables (`moviename` and `movieid`) of *character* and *integer* types, respectively. The table contains 66 730 unique values, with no zeros, no "not availables" (NAs), nor infinite values. In its original form, the `moviename` variable contains html code, which was subsequently removed (see section 3.1).

```
## Rows: 1,002,796
## Columns: 7
## $ userid      <int> 981904, 882359, 921220, 798641, 952904, 888, 300293, 80...
## $ gender      <chr> "Male", "Female", "Female", "Male", "Female", "Female",...
## $ location    <int> 111, 870, 993, 250, 172, 157, 221, 180, 233, 221, 282, ...
## $ memberfor   <chr> "2009-09-01 00:00:00", "2009-10-02 00:00:00", "2009-09-...
## $ lastlogin   <int> 1, 181, 124, 40, 44, 39, 91, 1, 26, 23, 490, 1097, 272,...
## $ profileview <chr> "19", "108", NA, "22", "21", "18", "13", NA, NA, "32", ...
## $ age         <chr> "19", "108", NA, "22", "21", "18", "13", NA, NA, "32", ...
```

```
##      variable q_zeros p_zeros   q_na  p_na q_inf p_inf      type  unique
## 1      userid       0    0.00      0  0.00     0     0   integer 1002796
## 2      gender       0    0.00  67529  6.73     0     0 character       2
## 3    location   57726    5.76    203  0.02     0     0   integer    1429
## 4   memberfor       0    0.00    203  0.02     0     0 character      36
## 5   lastlogin   48949    4.88  57925  5.78     0     0   integer    3099
## 6 profileview       0    0.00 255564 25.49     0     0 character     126
## 7         age       0    0.00 255564 25.49     0     0 character     126
```

```
## # A tibble: 6 x 7
##   userid gender location memberfor           lastlogin profileview age
##    <int> <chr>     <int> <chr>                   <int> <chr>       <chr>
## 1 981904 Male        111 2009-09-01 00:00:00         1 19          19
## 2 882359 Female      870 2009-10-02 00:00:00       181 108         108
## 3 921220 Female      993 2009-09-01 00:00:00       124 <NA>        <NA>
## 4 798641 Male        250 2009-11-01 00:00:00        40 22          22
## 5 952904 Female      172 2009-11-01 00:00:00        44 21          21
## 6    888 Female      157 2009-10-01 00:00:00        39 18          18
```

The output above highlights the `profiles` tibble, with 1 002 796 records and seven variables: the *integer* type `userid`, `location`, and `lastlogin`, and the *character* type `gender`, `memberfor`, `profileview`, and `age`). Zeros and NAs are present in some variables—`location` or `age`, for example—but there are no infinite values. The time tag in the `memberfor` variable was removed (see section 3.1) for being always equal to 00:00:00 and, thus, irrelevant.

```
## Rows: 8,196,077
## Columns: 4
## $ userid  <int> 882359, 882359, 882359, 882359, 882359, 882359, 882359, 882...
## $ movieid <int> 81, 926, 1349, 2270, 3065, 3522, 3583, 4216, 4871, 4917, 51...
## $ rating  <dbl> 1.5, 1.0, 2.0, 1.0, 5.0, 0.5, 0.5, 0.5, 3.5, 0.5, 1.0, 1.0,...
## $ date    <chr> "2007-10-10 00:00:00", "2007-10-10 00:00:00", "2007-10-10 0...
```

```
##   variable q_zeros p_zeros q_na p_na q_inf p_inf      type unique
## 1   userid       0       0    0    0     0     0   integer 147612
## 2  movieid       0       0    0    0     0     0   integer  48794
## 3   rating       0       0    0    0     0     0   numeric     10
## 4     date       0       0    0    0     0     0 character   1450
```

```
## # A tibble: 6 x 4
##   userid movieid rating date
##    <int>   <int>  <dbl> <chr>
## 1 882359      81    1.5 2007-10-10 00:00:00
## 2 882359     926    1   2007-10-10 00:00:00
## 3 882359    1349    2   2007-10-10 00:00:00
## 4 882359    2270    1   2007-10-10 00:00:00
## 5 882359    3065    5   2007-12-29 00:00:00
## 6 882359    3522    0.5 2007-11-13 00:00:00
```

Above, the `ratings` tibble, with 8 196 077 instances and four variables (`userid`, `movieid`, `rating`, and `date`). Similarly to the previous case, the time tag in `date` was removed (section 3.1).

Table 2.1 describes all variables contained in the three original tables of the data set. Clearly, the type of some of variables is incorrect and inconvenient for analysis—`age`, `date`, and `gender`, just to name a few—and that was taken care of in section 3.1.

Table 2.1: List of variables present in the three original files from the Flixster data set.

Variable	Type	Description	Tibble
age	character	Age of user	profiles
date	character	Date in which user rated movie	ratings
gender	character	User gender	profiles
lastlogin	integer	Last login numerical tag	profiles
location	integer	Location identifier of user	profiles
memberfor	character	Member since date	profiles
movieid	integer	Movie unique identifier	movies
movieid	integer	Movie unique identifier	ratings
moviename	character	movie name	movies
profileview	character	Numerical tag of the user	profiles
rating	double	Rating of movie by user	ratings

Variable	Type	Description	Tibble
userid	integer	User unique identifier	profiles
userid	integer	User unique identifier	ratings

In the next chapter, we describe the steps we performed to explore all relevant variables contained in the data set, the outcomes of that analysis, and the transformations we applied to some of them.

Chapter 3

Exploratory data analysis and engineering

The creation of a recommendation system requires the identification of the most important features involved in the prediction of the ratings. With that goal in mind, we cleaned the data (section 3.1) and analysed statistically the variables (section 3.2) to understand them and the relationships between them.

Since **R** does not have a base function to calculate the mode, we created one, the `getmode` function. It was useful in our statistical analysis and in the imputation of values in some of the variables.

3.1 Data cleaning

We started by removing the time tag from the variables `date` and `memberfor`, as it was always equal to 00:00:00 and, thus, irrelevant for our analysis.

Then, we converted the variables `age` and `provileview` to integers, `gender` to a factor, and `memberfor` and `date` to dates.

Several movie included strange characters in their names due to badly or poorly rendered ASCII codes. Those were identified and replaced.

Finally, for the sake of personal taste and convenience, we converted the three main tables into tibbles.

A quick inspection allowed us to conclude that the `profilesview` and `age` variables were the same. So, we decided to remove the former right away, even before a more in depth exploratory data analysis.

Below is a summary of the three tables—respectively `movies`, `profiles`, and `ratings`—as they stood after the preliminary cleaning of the variables.

```
## # A tibble: 66,730 x 2
##    moviename                                                    movieid
##    <chr>                                                          <int>
##  1 $ (Dollars) (The Heist)                                          252
##  2 $5 a Day (Five Dollars a Day)                                    253
##  3 $9.99                                                              1
##  4 $windle (Swindle)                                                254
##  5 BBC2 Playhouse Caught on a Train                                 255
##  6 Independent Lens  Race to Execution                              256
##  7 Omnibus Song of Summer                                           257
##  8 The American Experience The Battle Over Citizen Kane               2
##  9 38 (38 Home to the Realm) (38 - Vienna Before the Fall)          258
## 10 68                                                                 3
## # ... with 66,720 more rows
```

```
## # A tibble: 1,002,796 x 6
##    userid gender location memberfor  lastlogin   age
##     <int> <fct>     <int> <date>         <int> <int>
##  1 981904 Male        111 2009-09-01         1    19
##  2 882359 Female      870 2009-10-02       181   108
##  3 921220 Female      993 2009-09-01       124    NA
##  4 798641 Male        250 2009-11-01        40    22
##  5 952904 Female      172 2009-11-01        44    21
##  6    888 Female      157 2009-10-01        39    18
##  7 300293 Male        221 2009-10-02        91    13
##  8 805965 Male        180 2009-07-01         1    NA
##  9 162134 Male        233 2009-09-01        26    NA
## 10 611822 Female      221 2009-06-03        23    32
## # ... with 1,002,786 more rows
```

```
## # A tibble: 8,196,077 x 4
##    userid movieid rating date
##     <int>   <int>  <dbl> <date>
##  1 882359      81    1.5 2007-10-10
##  2 882359     926    1   2007-10-10
##  3 882359    1349    2   2007-10-10
##  4 882359    2270    1   2007-10-10
##  5 882359    3065    5   2007-12-29
##  6 882359    3522    0.5 2007-11-13
##  7 882359    3583    0.5 2007-11-13
##  8 882359    4216    0.5 2007-10-10
##  9 882359    4871    3.5 2008-07-19
## 10 882359    4917    0.5 2007-11-13
## # ... with 8,196,067 more rows
```

3.2 Statistical exploration of the variables

All features contained in the three tibbles (`movies`, `profiles`, and `ratings`) were saved in standalone variables, to facilitate their statistical analysis. Those variables are described in tab. 3.1. It is worth noting that during the creation of the `gender` variable, all NAs were assigned the category "Other".

Table 3.1: List of standalone variables created from the original features present in the three files from the Flixster data set, after type correction and preliminary data engineering.

Variable	Type	Description	Tibble
age	integer	Age of user	profiles
dates	date	Date in which user rated movie	ratings
gender	factor	Gender of the user (*Female*, *Male*, and *Other*)	profiles
lastlogin	integer	Last login numerical tag	profiles
location	integer	Location identifier of user	profiles
memberfor	date	Member since date	profiles
movieid.movies	integer	Movie unique identifier	movies
movieid.ratings	integer	Movie unique identifier	ratings
moviename	character	movie name	movies
profileview	integer	Numerical tag of the user	profiles
rating	double	Rating of movie by user	ratings
userid.profiles	integer	User unique identifier	profiles
userid.ratings	integer	User unique identifier	ratings

Understanding the structure of the data, the distribution of the variables, and the relationships between them is fundamental to build a solid model. We therefore analysed each of the features present in tab. 3.1.

3.2.1 Age

The variable `age` has a minimum of 12 years, a maximum of 113 years, and a median of 25 years. The maximum is clearly an outlier. The variable also contains more than 255 000 NAs, corresponding to approximately 25.5 % of all age values.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   12.00   21.00   25.00   27.39   31.00  113.00  255618
```

Removing the NAs, we obtain a variance of 107.5 years.

```
## [1] 107.5378
```

The skewness is positive, indicating a right-skewed distribution.

```
## [1] 2.444335
```

We plotted the histogram and the box-plot of `ages` (fig. 3.1).

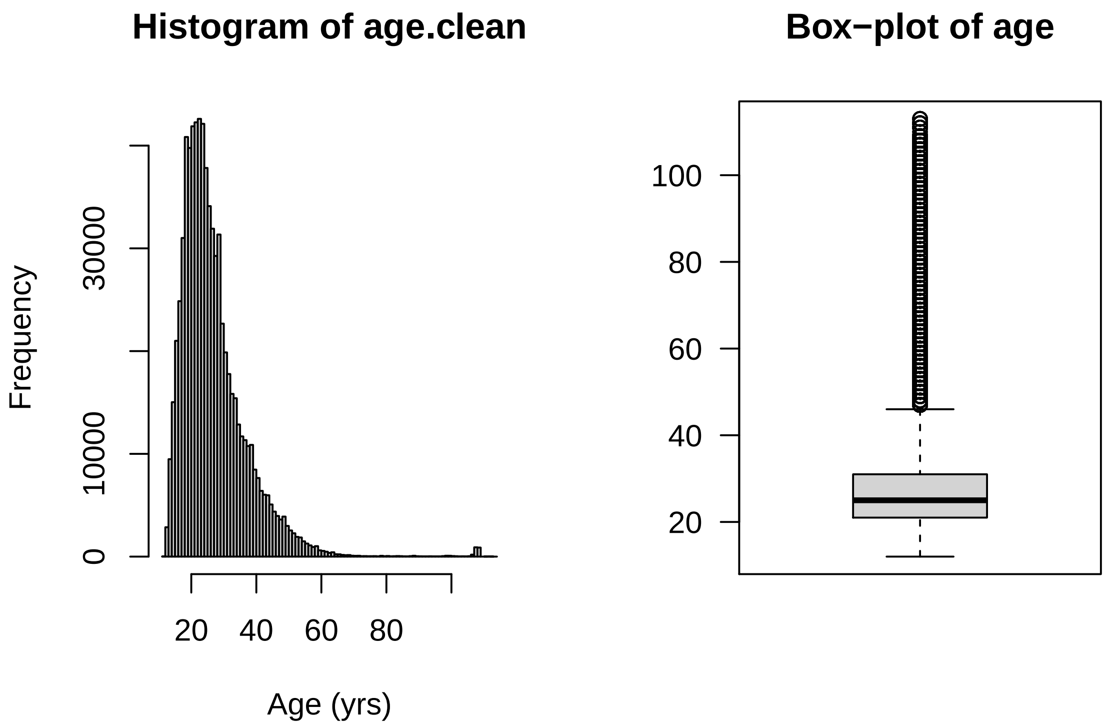


Figure 3.1: Histogram (*left*) and boxplot (*right*) of the ‘age‘ variable. The distribution is right skewed and exihibits several outliers.

Most of the individuals are between 18 and 30 years old, corresponding to more than 250 000 of Flixster's users. The histogram highlights the right skewness of the distribution. This is expected, since Flixster's customers were typically young.

We also plotted the histogram of `age` as a function of `gender` (fig. 3.2). The vertical dashed lines represent respectively the first and 99.5th percentile of the distribution of ages.

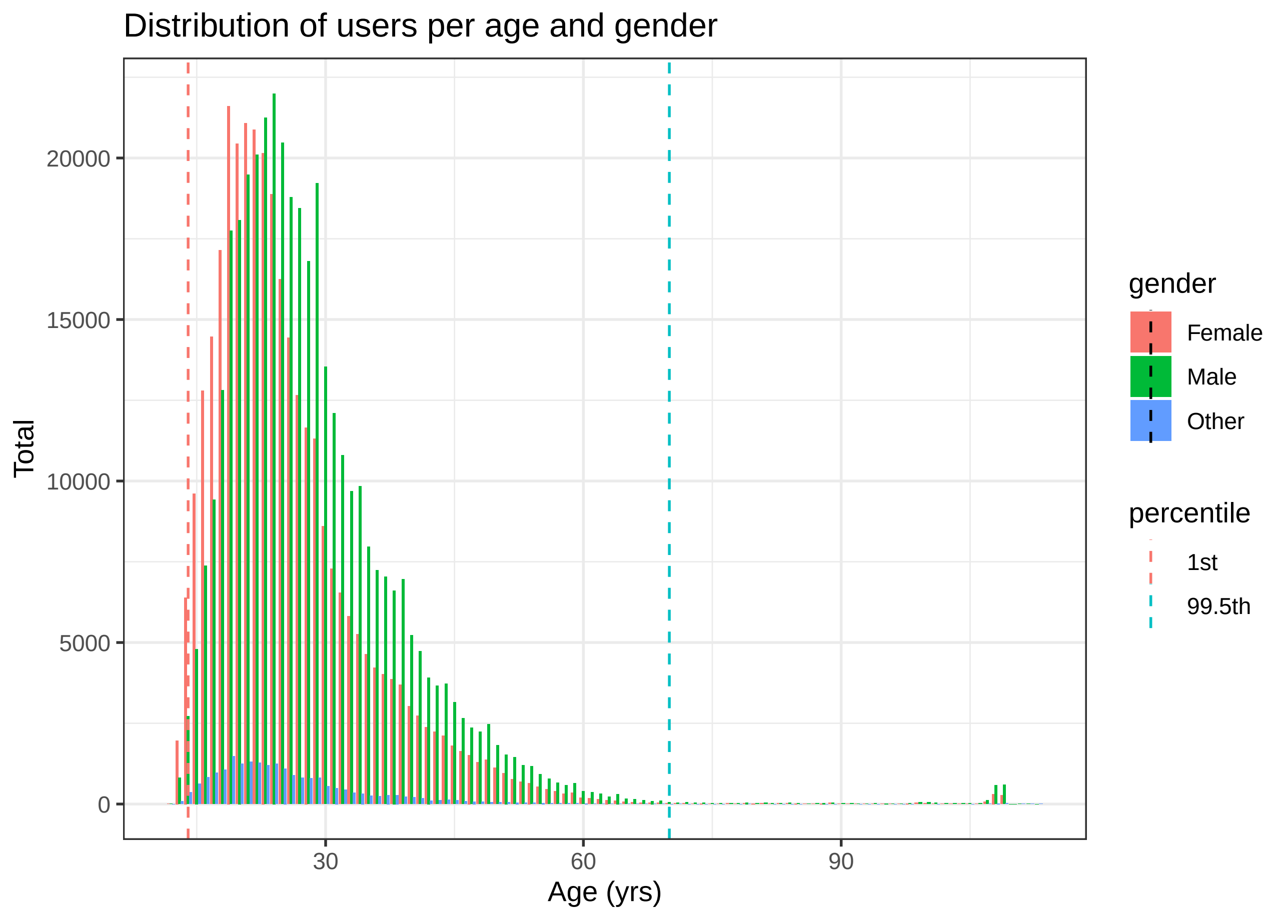


Figure 3.2: Histogram of the age of users as a function of their gender. The majority of Flixster users are women and men, with the first and 99.5th percentile between 14 and 70 years old.

Ages bellow 14 and above 70 would be safe to be considered as outliers. However, we only removed users with ages above 70 years old, the set of which corresponding to 0.5 % of all ages. The histogram and box-plot of `ages` without outliers is represented in fig. 3.3.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.00   21.00   25.00   27.05   31.00   70.00
```

3.2.2 Dates and Memberfor

We detected three types of incorrect dates in the data set, both in the `date` and `memberfor` variables (respectively from the `ratings` and the `profiles` tibbles). The first two types

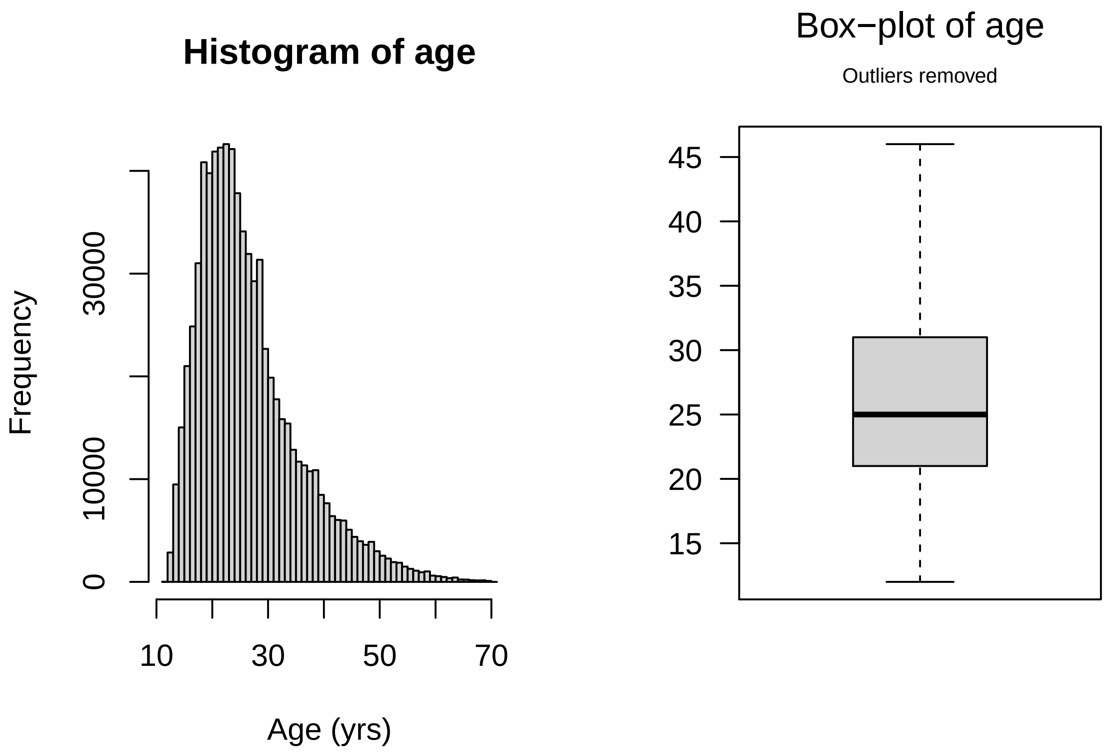


Figure 3.3: Histogram (*left*) and boxplot (*right*) of ‘ages‘ after removing the outliers.

corresponded to dates either before the founding of Flixster (2006/01/20) or NA values—we detected dates as early as 1900/01/01 in `memberfor` and 1941/12/07 in `dates`, and 203 NAs in `memberfor` (no NAs in `dates`).

Regarding the NAs in `memberfor`, we simply removed the corresponding instances since, on the one hand, the total number of NAs was small, corresponding to 0.02 % of the total number of values and, on the other hand, there was no way of estimating a reasonable date. For the remainder of the wrong dates, we performed imputation, according to the following rules: (*i*) we made `memberfor` equal to the minimum rating date every time the former was after the latter, *i.e.*, when the membership date was after the first rating, and (*ii*) we replaced all dates before 2006/01/20 by Flixster's founding date.

The third type of wrong dates corresponded to dates in `dates` which were earlier than the registration date in Flixster (the date saved in the `memberfor` variable). In those cases, we took the earliest date recorded in `dates` as the date for `memberfor`. Since there are 147 612 unique user IDs in the tibble `ratings`, we obtained a "cleaned" `memberfor` variable with that number of dates. We ended up with two arrays with dates between 2006/01/20 and 2009/11/17 (`dates` case) or 2009/11/01 (`memberfor` case), with no NAs.

3.2.3 Gender

The proportions of the three gender categories previously identified (*Female*, *Male*, and *Other*) were highlighted in a pie chart (see fig. 3.4). The set of Flixster's users was comprised of 49 % of males, 44 % of females, and 7 % of other/unidentified genders.

3.2.4 Last login

Due to the lack of information about the data set, we do not know exactly what this variable represents. We suspect, however, it corresponds to the total number of logins during a certain period of time (during the previous month or year) per user. Assuming that is the case, most of the users had logged in to Flixster during that period of time between approximately 4 to 30 times. The minimum of `lastlogin` is 0, meaning a user never logged in, and the maximum is 177 278. There are 57 925 NAs, corresponding to 5.7 % of all values. The distribution is strongly right skewed, with several outliers lying far away from the median, *i.e.*, corresponding to number of occurrences several orders of magnitude above the median. We removed the outliers by wiping out all values with frequencies above 46 (that kept 99 % of all the values). The histogram and box-plot of the distribution with and without outliers are represented in figs. 3.5 and 3.6, respectively.

```
## Variance of `lastlogin`: 102829.2
```

```
## Skewness of `lastlogin`: 217.3447
```

Since this variable did not seem to be relevant for any of our analyses, we decided to discard it.

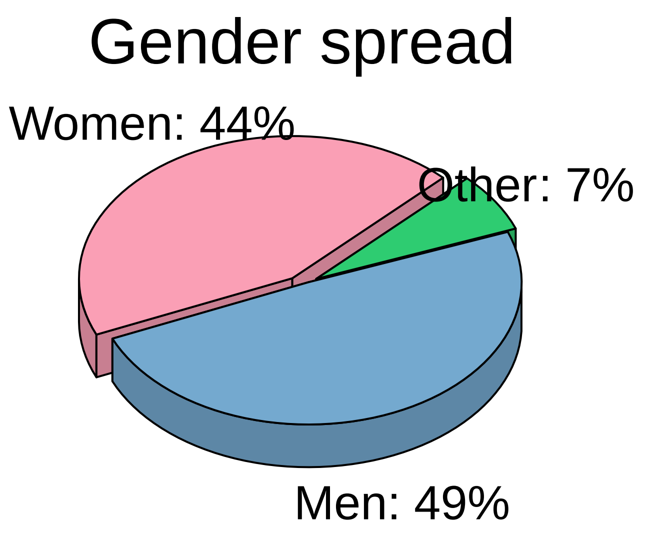


Figure 3.4: Gender spread among Flixster users.

3.2.5 Location

This feature presented several NAs, corresponding to 0.02 % of all values. The histogram (fig. 3.7) is characterised by a sharp spike at the value 0, with a difference of at least one order of magnitude to the remainder of the local maxima. This might indicate that 0 corresponds to the most typical region the users belong to, or might be just an identifier of all users for whom the location is unknown (an outlier, in that case). Since there is no information about the variables and `location` is a feature of integers, we decided to impute all NAs with the mode (the value 0). After the imputation, `location` varied between 0 and 1617.

```
## Variance of `location`: 81802.41
```

```
## Skewness of `location`: 1.036804
```

We then looked to the distribution of the `location` values without outliers. Zero and all values above 1200 were placed in this category—the remainder of the set corresponded to approximately 99 % of the original values. The new histogram and box-plot are represented in fig. 3.8.

With outliers removed, we verify that most of the values are between 180 and 300. However, similarly to what happened with `lastlogin`, the lack of information about this feature makes it useless in our analysis. Therefore, we decided to removed it from the final data set.

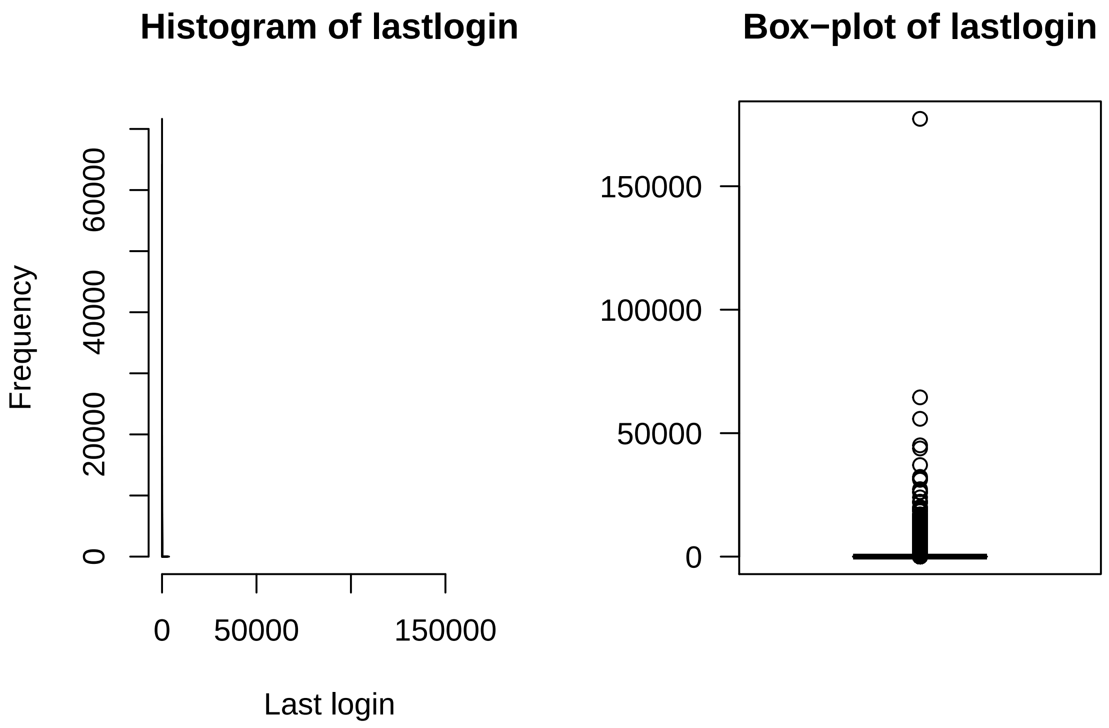


Figure 3.5: Histogram (*left*) and boxplot (*right*) of the variable ‘lastlogin‘. Some outliers lie very far away from the median.

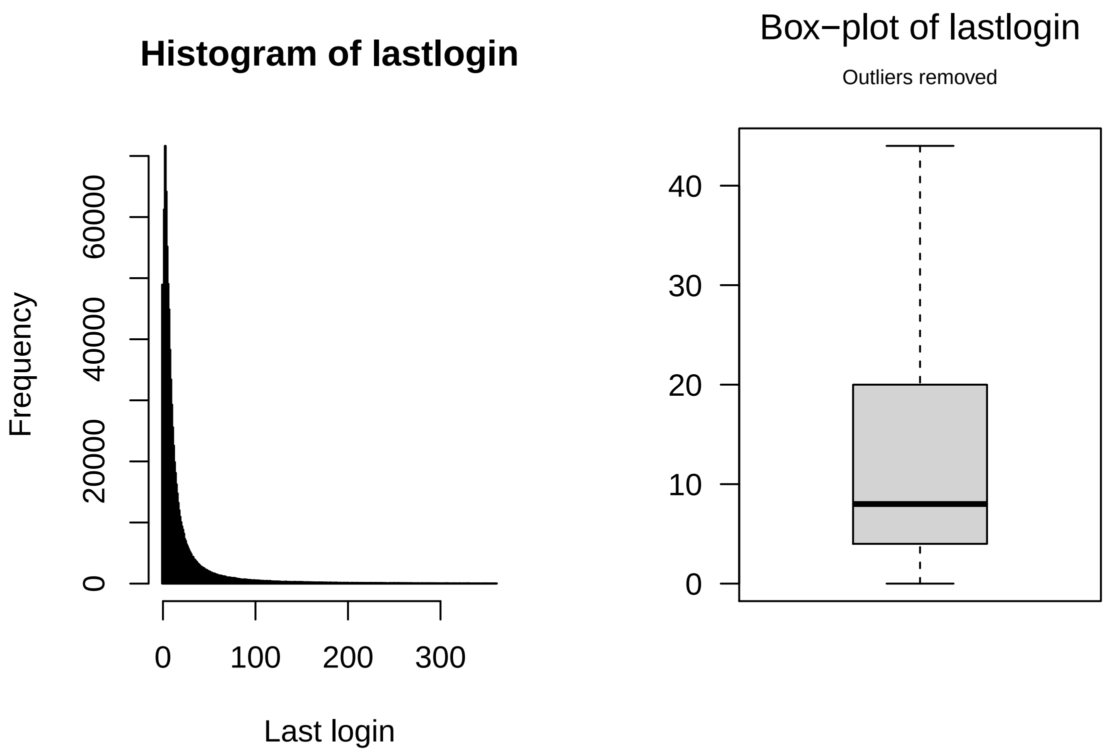


Figure 3.6: Histogram (*left*) and boxplot (*right*) of the variable ‘lastlogin‘ after removing the outliers.

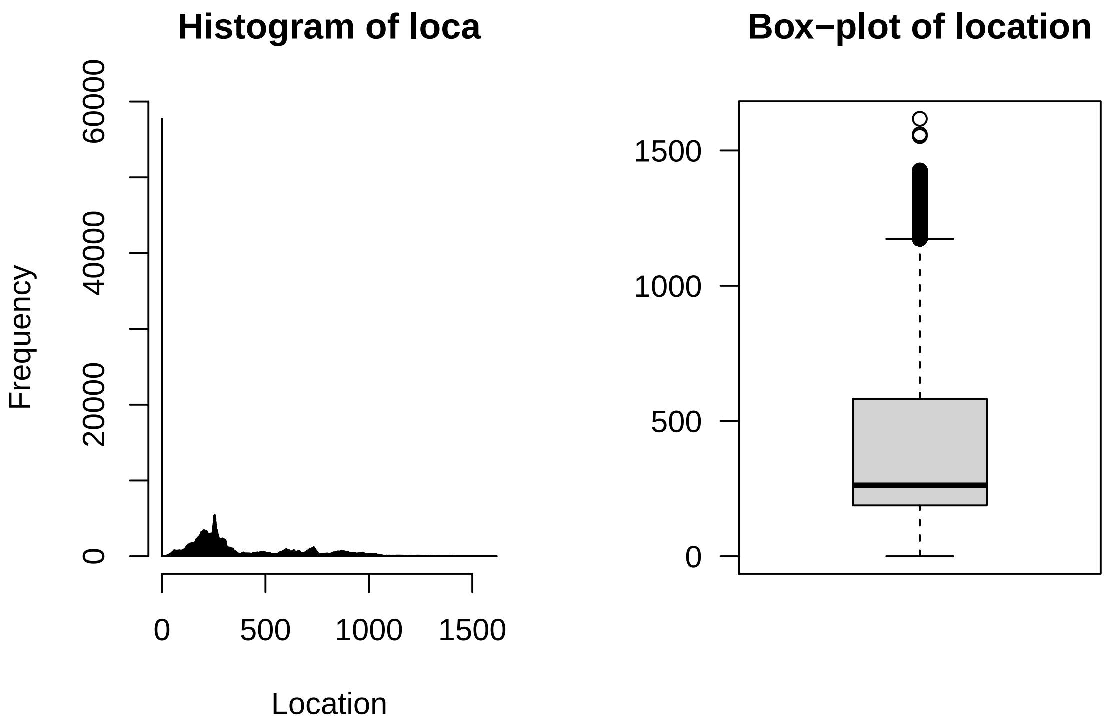
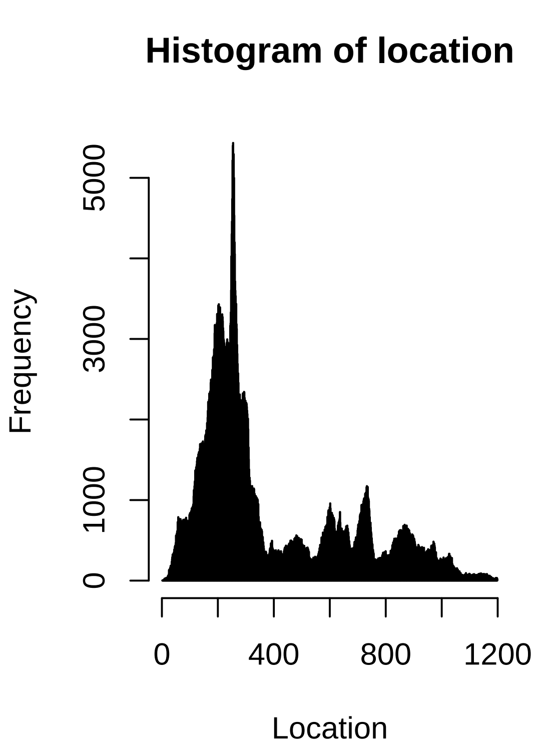


Figure 3.7: Histogram (*left*) and boxplot (*right*) of the ‘location‘ variable.



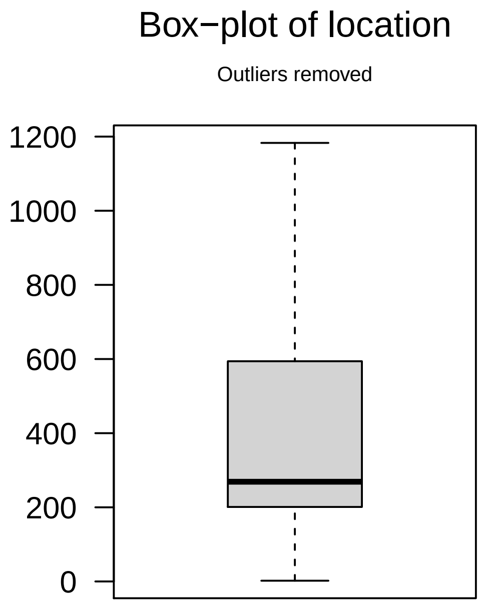


Figure 3.8: Histogram (*left*) and boxplot (*right*) of the ‘location‘ variable, after the removal of the outliers.

3.2.6 Movie ID

We can find two variables named `movieid` in the data set, one from the `movies` table, and another from the `ratings` table. The former is composed of unique entries, one per film, while the latter has several instances corresponding to the same movie, one per rating. That makes `movieid.ratings` larger than `movieid.movies` (respectively 8 196 077 against 66 730 entries). None of these variables present NAs.

Since the feature `movieid.movies` represents just the identification (ID) number of each film, there were no relevant statistics to compute. Hence, we focused our attention in `movieid.ratings` and plotted its histogram (fig. 3.9).

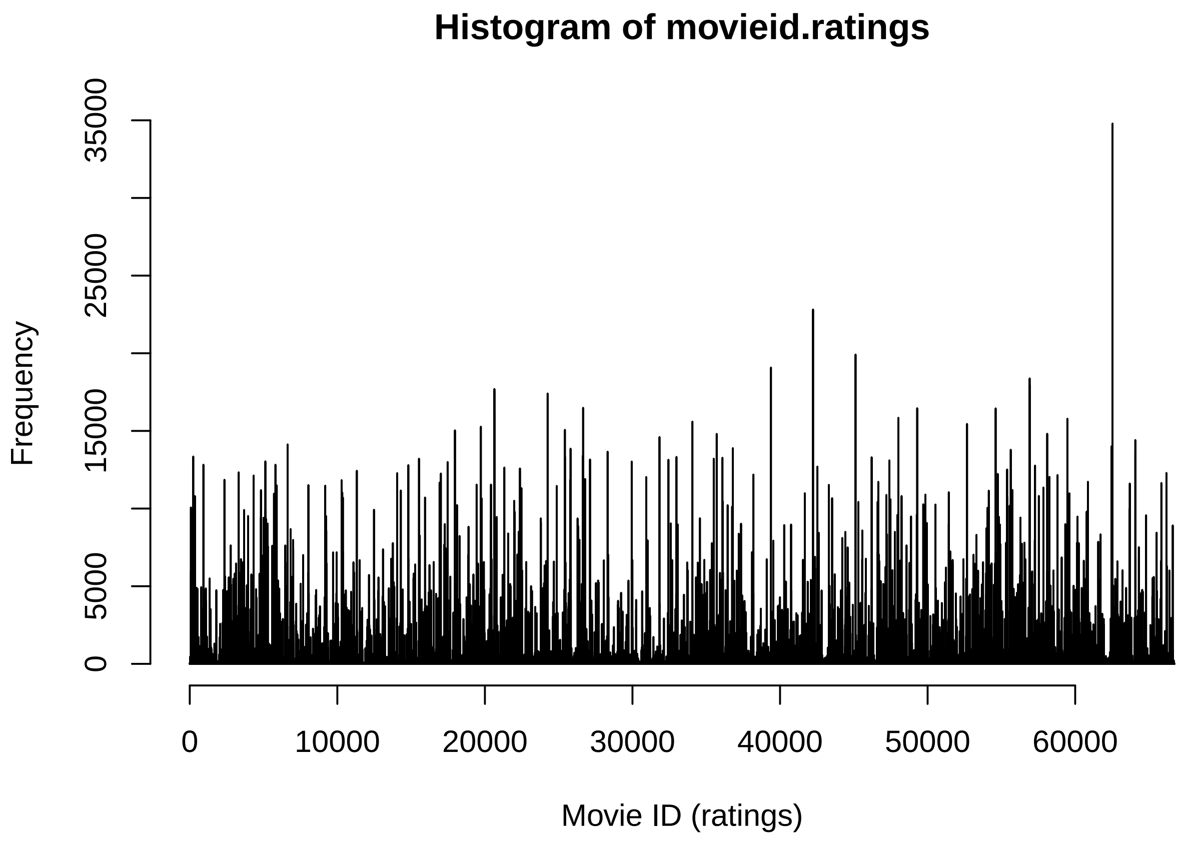


Figure 3.9: Histogram of the movie ID, contained in the ‘ratings‘ tibble. There is a particular film with ID above 60000 with the most of ratings.

From the histogram, we can see that the range of the number of ratings encompasses four orders of magnitude, existing films with much more ratings than others (naturally). In particular, there is a movie with an ID greater than 60 000, with around 35 000 ratings, that stands out.

3.2.7 Ratings

Finally, we looked at the `ratings` feature. From the summary, the histogram, and the box-plot of the variable, we verify that: (a) the ratings range between 0.5 and 5.0; (b) the most common ratings are 3.0, 3.5, and 5.0; (c) the variance is equal to 1.192 404; and (d) the distribution is left skewed ($skew = -0.7054742$). We were already expecting to find a left skewed distribution for `ratings`, since in a sufficiently large data set of this nature, most of the ratings will lie above the medium (it is expected to find a large number of users liking a substantially large subset of films and, hence, giving them high ratings). The plots of the histogram and box-plot of `ratings` are illustrated in fig. 3.2.7.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   3.500   3.613   4.500   5.000
```

```
## Variance of `ratings`: 1.192404
```

```
## Skewness of `ratings`: -0.7054742
```

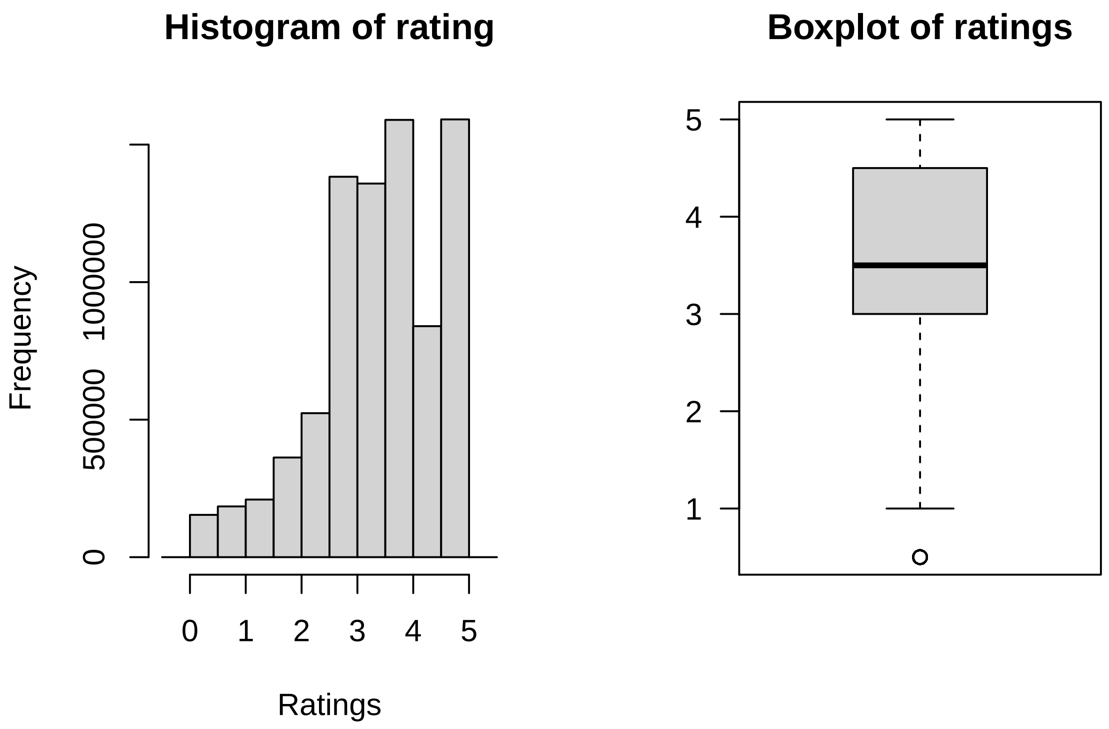


Figure 3.10: Histogram (*left*) and boxplot (*right*) of the `ratings` variable. As expected for this kind of variable (when the data set is sufficiently large), the distribution is left-skewed.

3.3 Data engineering

As mentioned previously, the variables `lastlogin` and `location` were not needed for the context of this analysis, and so they were removed (the feature `profileview` had already been removed in the preliminary data analysis).

The `gender` column in `profiles` was replaced by the updated `gender` standalone variable (including the "Other" gender), and only the instances with no NAs in the age were kept. At the end, `profiles` was reduced to a 747 178 × 3 tibble, containing no NAs and with the relevant features `userid`, `gender`, and `age`.

We then created two variables, `mean_ratings` and `total_ratings`, respectively corresponding to the average rating by each user and total number of ratings per user. Those features were added to the `profiles` tibble.

A similar approach was used with the `movies` tibble, where two new columns were created containing the average rating per film (`mean_rating`) and the total number of ratings each movie got (`total_rating`).

Finally, we joined the three tables, in sequence, into a single tibble named "flixster". In a general film recommender system, we would use *left joins* during this step, because users without ratings would still be eligible for recommendations based not on their rating history, but on similar user profiles (history of views, for example). However, in the case at hands, the only record we have of users' activity is their rating history. When it does not exist, *i.e.*, when a user has not made any rating, the instance will be useless for predictions. The same reasoning applies for the films: if a movie has not been rated, it will be pointless for predictions.

Therefore, we opted for inner joins, keeping only the common values between the tables, thus eliminating users and films without any rating.

Three tibbles were created: one containing all columns (`flixster`), a smaller version containing the variables `userid`, `gender`, `age`, `movieid`, `moviename`, `rating`, and `age` (`flixster.small`), and a tiny version, containing the features `userid`, `movieid`, `rating`, and `date` (`flixster.tiny`). The `ratings` and `profiles` tibbles were joined using the `userid` variable, to which the `movies` tibble was joined using the `movied` feature.

A preview of the `flixster` tibble is shown below.

```
## # A tibble: 6,131,346 x 12
##    userid gender   age memberfor  total.ratings.u~ mean.ratings.us~ movieid
##     <int> <fct>  <int> <date>                <int>            <dbl>   <int>
##  1    888 Female    18 2008-07-04                2              0.5   26377
##  2    888 Female    18 2008-07-04                2              0.5   53324
##  3 611822 Female    32 2008-08-13                1              5     53968
##  4  43635 Female    19 2007-06-19                5              4     16969
##  5  43635 Female    19 2007-06-19                5              4     43529
##  6  43635 Female    19 2007-06-19                5              4     46995
##  7  43635 Female    19 2007-06-19                5              4     52351
##  8  43635 Female    19 2007-06-19                5              4     60624
##  9 857015 Male      17 2009-03-02              15               4.8    4174
## 10 857015 Male      17 2009-03-02              15               4.8    7011
## # ... with 6,131,336 more rows, and 5 more variables: moviename <chr>,
## #   rating <dbl>, date <date>, mean.ratings.movie <dbl>,
## #   total.ratings.movie <int>
```

3.4 Data exploration

The `flixster` tibble contains 108 779 unique users, who have rated at least one film, during a time span of almost four years, between the 20th January 2006 and the 17th of November 2009. The majority of the users rated only a few films—68.3 % of the users rated up to 10 films—while a small number of users rated over 1000 movies—995 users in total, corresponding to approximately 0.9 % of the total users.

```
## # A tibble: 6 x 2
##   userid     n
##    <int> <int>
## 1      6     1
## 2      7     1
## 3      9     1
## 4     28     1
## 5     56     1
## 6     70     1
```

Figure 3.11 depicts the histogram of the number of ratings per user.

We created a heat-map of users *vs.* movies for a sample of 100 unique users (see 3.12). As expected, the user-movie matrix is sparse, being the majority of cells empty. Some films had more ratings than others, and some users were more active than others.

Table 3.2 highlights the top 10 of most active users, *i.e.*, the top 10 users with the most ratings. The most active user rated over 30 000 movies since December 2007, and the second most active user rated over 24 600 films since August 2007—these are two examples of users whose registration date in the platform (the `memberfor` variable) was posterior to the date of the first rating and, thus, that had to be corrected (see section 3.2.2). The first woman appearing in this top 10 occupies the fifth position in the table with 7356 ratings since July 2009. In terms of gender spread, this top 10 is perfectly balanced, with a 1-to-1 ratio of women to men.

Table 3.2: List of the top 10 active users, in terms of ratings..

User ID	Gender	Age	Member for	Total number of ratings	
1	103006	Male	27	2007-12-05	30977
2	211247	Male	36	2007-08-09	24622
3	182127	Male	51	2009-06-29	9974
4	458966	Male	22	2009-01-03	9134
5	170822	Female	63	2009-07-14	7356

User ID	Gender	Age	Member for	Total number of ratings	
6	13466	Female	32	2009-07-23	6973
7	338577	Female	41	2008-09-15	6148
8	733571	Female	35	2006-01-20	5216
9	120742	Male	30	2008-10-02	5039
10	41389	Female	35	2009-07-01	5013

The sample distribution of ratings per number of movies is represented by the histogram of fig. 3.13. As expected, most of the films received a low number of ratings—typically up to around 100 ratings.

The top 10 most rated movies is presented in tab. 3.3. The most rated film in Flixster was "Transformers: Revenge of the Fallen", with 34 791 ratings. "Shrek", an animated film, occupies the third position, with 19 916 classifications.

Table 3.3: List of the top 10 most rated movies.

Movie ID	Movie name	Total number of ratings	
1	62530	Transformers: Revenge of the Fallen	34791
2	42237	Pirates of the Caribbean: Dead Mans Chest	22817
3	45119	Shrek	19916
4	39384	Pirates of the Caribbean: At Worlds End	19079
5	56915	The Lord of the Rings - The Fellowship of the Ring	18381
6	56916	The Lord of the Rings - The Two Towers	17950
7	20644	Harry Potter and the Prisoner of Azkaban	17698
8	20645	Harry Potter and the Philosopher's Stone	17612
9	24251	Harry Potter and the Chamber of Secrets	17410
10	26656	Ice Age	16487

The histogram of ratings per date is plotted in fig. 3.16.

The year with the most activity in Flixster was 2007, followed by 2009. We cannot identify seasonality in the data. On the one hand, the peak of activity in 2007 coincide with the first half of the year, during and right after the two most important film awards happen (The Golden Globes in early January, and the Oscars in late February). However, on the other hand, 2008 clearly does not follow that pattern. In 2009, the number of ratings increased steadily until June, but started to decline rapidly after that, until the end of the data set.

Table 3.4 highlights the dates with more ratings. On the 15th of April 2008, "Star Wars: episode IV – A New Hope" became the film most rated on a single day in Flixster, with 418 ratings.

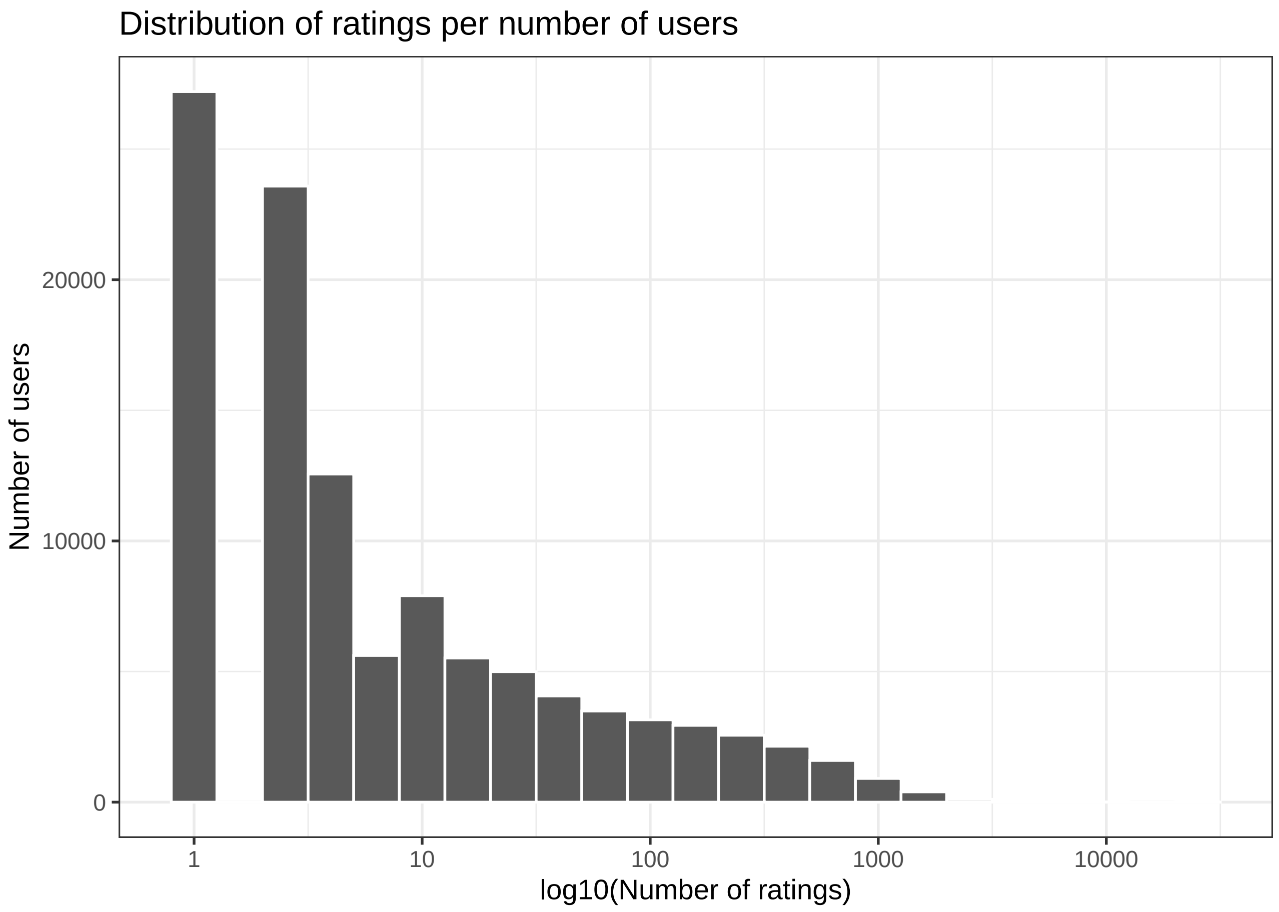


Figure 3.11: Histogram of the number of ratings. Approximatelly 68.3 percent of the users rated 10 films or less. The x-axis is in a log10 scale.

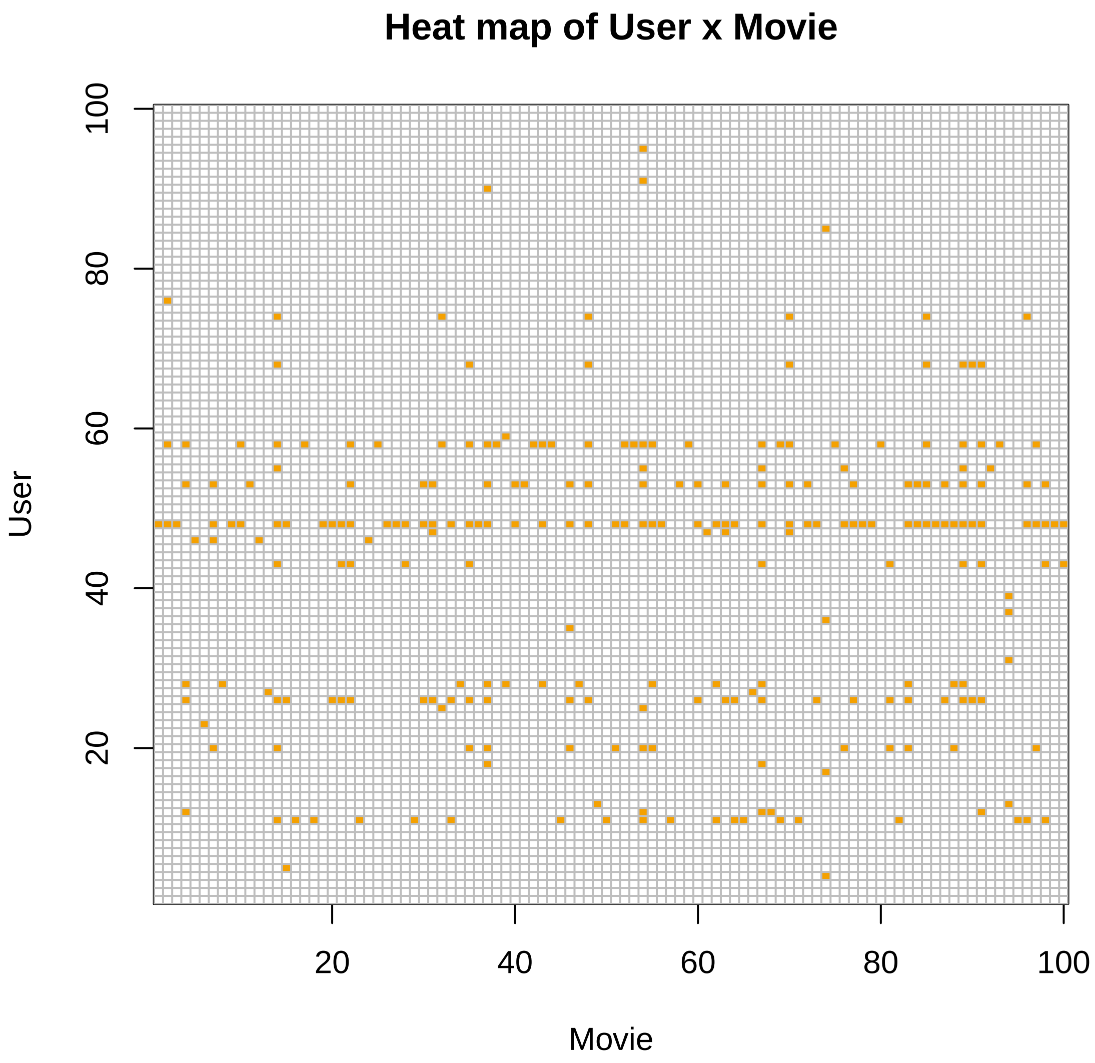


Figure 3.12: Heatmap of user *vs* movie created from a sample of unique user IDs. The matrix is sparse, with several empty cells. Some movies present more ratings, and some users are more active.

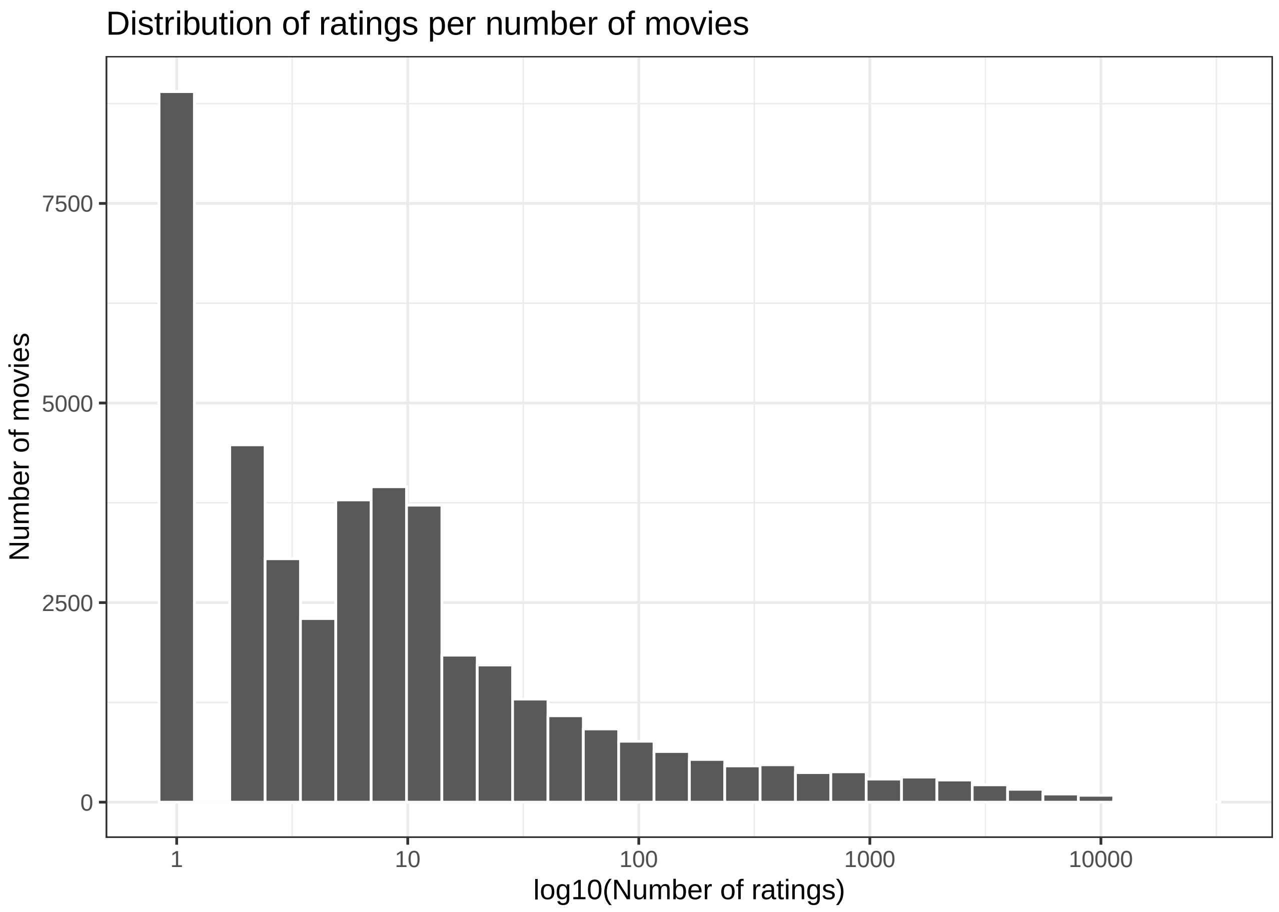


Figure 3.13: Histogram of ratings per number of movies. The x-axis is in a log10 scale. Most of the films received approximately up to 100 ratings.

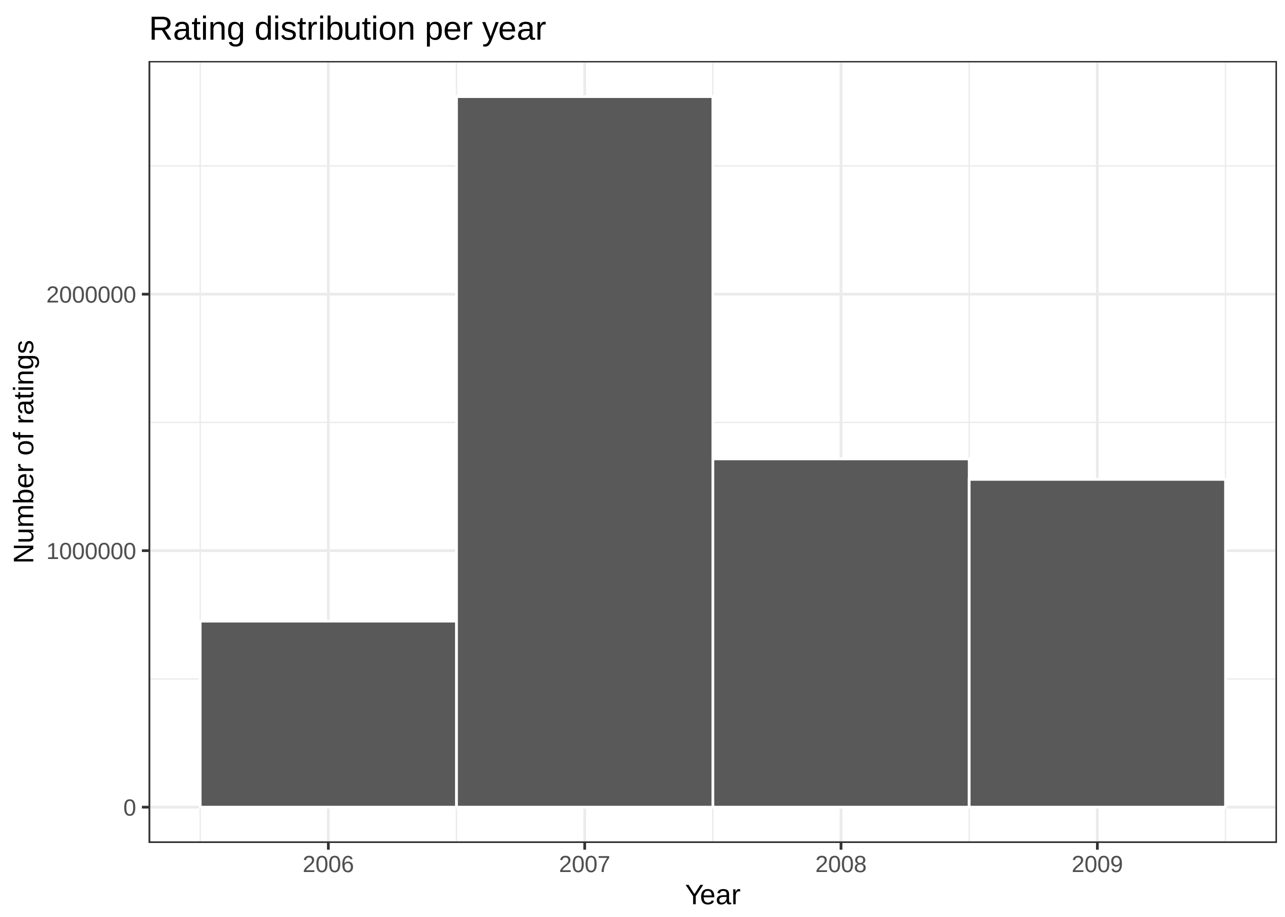


Figure 3.14: Histogram of ratings per year. The year with the most ratings was 2007, with over 2,500,000 ratings.

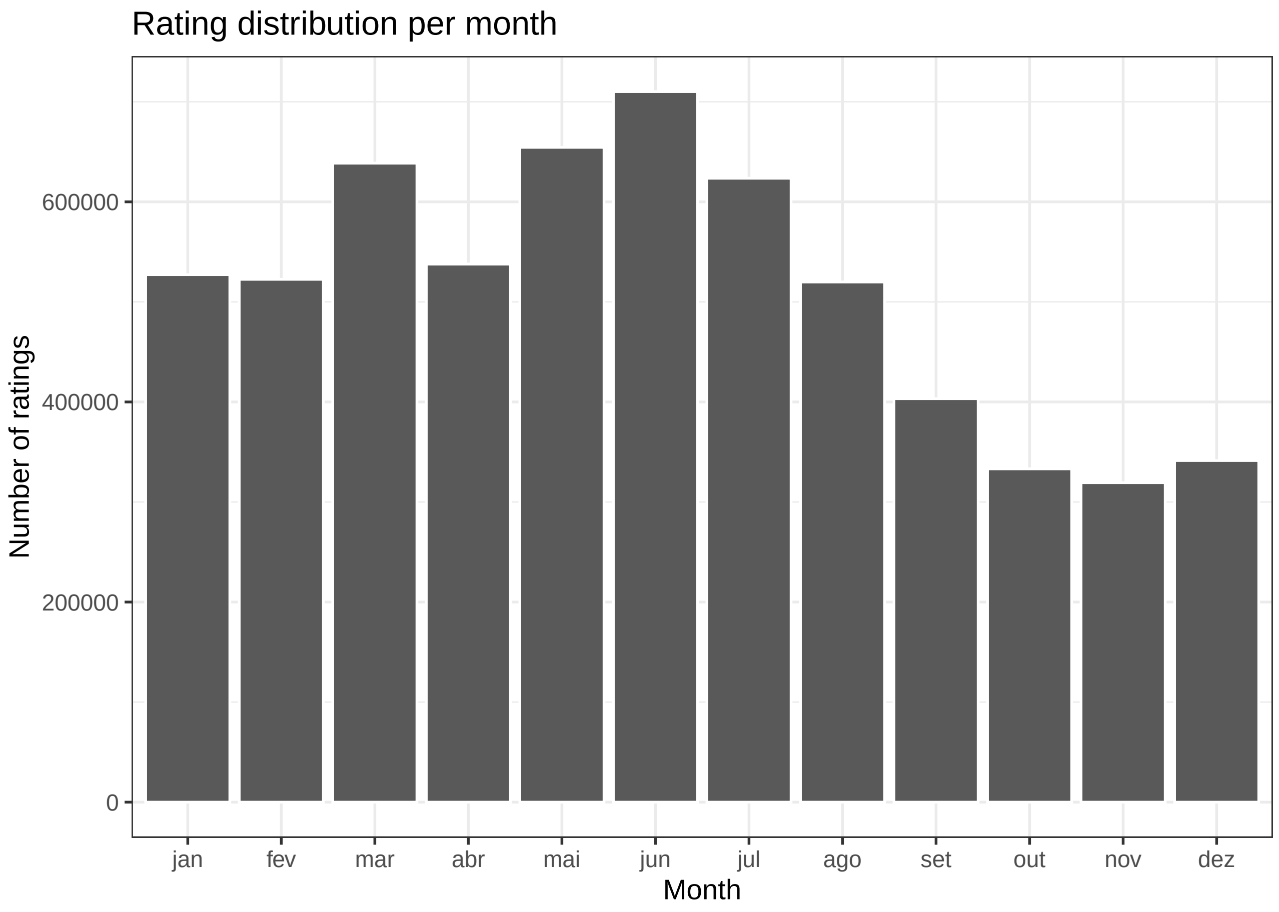


Figure 3.15: Histogram of ratings per month. The month with the overall most ratings was June, with over 700,000 ratings.

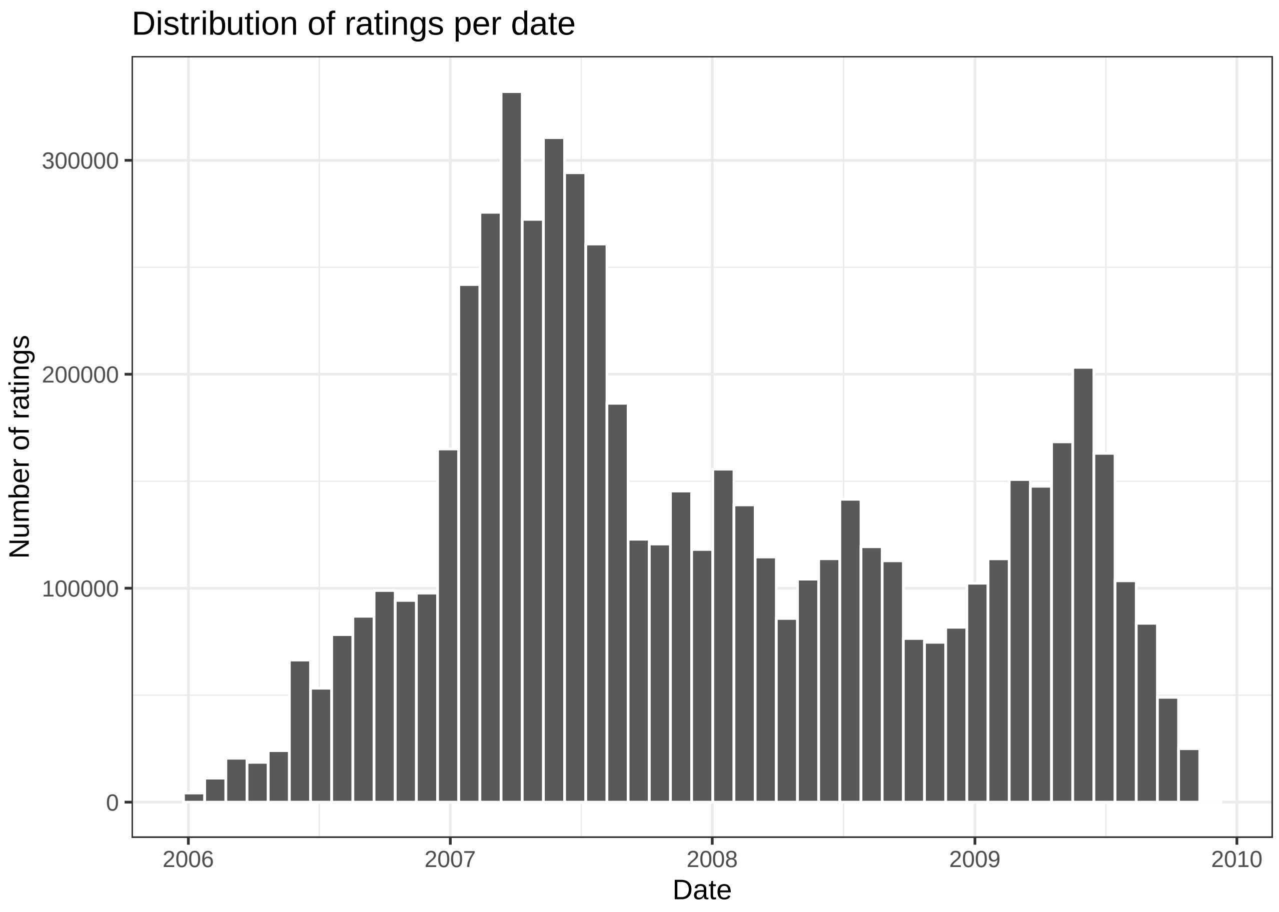


Figure 3.16: Histogram of ratings per date. The x-axis is in a log10 scale. Most of the films received approximately up to 100 ratings.

Table 3.4: List of film titles, sorted by date and by total number of ratings, in descending order.

n	Date	Movie name	Count
1	2008-04-15	Star Wars: Episode IV - A New Hope	418
2	2009-06-25	Transformers: Revenge of the Fallen	273
3	2008-08-08	Jackie Chans Who Am I? (Wo shi shei)	259
4	2009-02-25	Transformers: Revenge of the Fallen	258
5	2009-06-28	Transformers: Revenge of the Fallen	258
6	2009-02-23	Transformers: Revenge of the Fallen	257
7	2009-04-06	Slumdog Millionaire	257
8	2009-04-27	Transformers: Revenge of the Fallen	251
9	2009-06-24	Transformers: Revenge of the Fallen	250
10	2009-02-24	Transformers: Revenge of the Fallen	249

Figure 3.17 depicts the sample distribution of Flixster's ratings. The most used classifications were 3 and above.

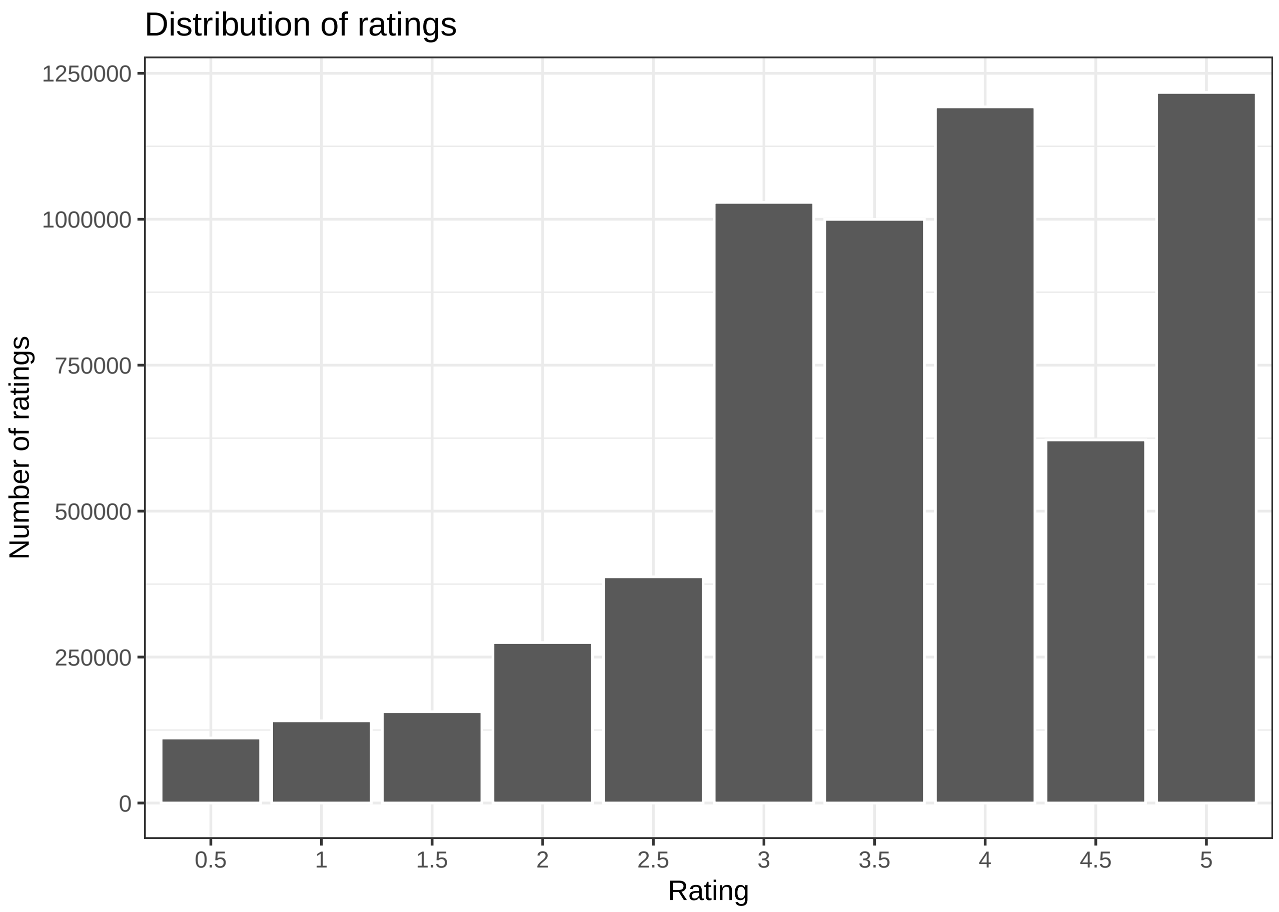


Figure 3.17: Histogram of ratings. Classifications 3 and above were the most popular.

3.5 Data preparation

The data was further prepared for the recommender model. The process consisted of the five following steps:

1. Selection of relevant data;
2. Building of the recommendation model;
3. Normalisation of the data;
4. Binarisation of the data;
5. Train/test splitting of the data.

3.5.1 Selection of relevant data

Due to time and mainly computer resources constraints, we decided to use a subset of the `flixster` tibble, which initially contained over eight million instances. The subset was created using two criteria to select the most relevant data: the most active users, and the most rated films.

We started by considering users with 10 or more ratings, and films which had received classifications 100 or more times. However, when creating the model based on the user-based collaborative filtering (IBCF), **RStudio** would repeatedly consume all the RAM of our computers and shut down. Consequently, we increased the thresholds, so that users were classified as active if they had produced 50 or more ratings, and movies with less than 500 classifications were disregarded. We ended up with a table of 4 752 926 rows, corresponding to a reductio of about 58 % from the original `ratings` tibble.

Such selection of data yielded a data set with 34 464 users and 5292 films, corresponding to a reduction of approximately 9 % in the total number of instances from the previous selection.

3.5.2 Building of the recommendation model

We followed two approaches in order to create the recommender model: using *binary* and *non-binary* information on the users' ratings. That came down to run two **R** instructions from the *recommenderlab* package:

```r
# binary
rec.mod.bin= recommenderRegistry$get_entries(
  dataType= "binaryRatingMatrix"
)

# non-binary
rec.mod= recommenderRegistry$get_entries(
  dataType= "realRatingMatrix"
)
```

We further instantiate the recommender rating matrices.

```r
load("../data/flixster.u50m500.rdata")
FLIXST= flixster.u50m500

ratingmat= dcast(
  FLIXST,
  userid ~ movieid,
  value.var= "rating"
  )

ratingmat= as.matrix(ratingmat)
```

3.5.3 Normalisation of the data

Normalising the ratings matrix is very important when preparing the data for the recommender system, since it allows to remove the rating bias and, therefore, to account for those users who systematically grade movies with high or low ratings. We started to produce recommendations without normalising the data, that we planned to compare with similar recommendations obtained with normalised rating matrices. However, we did not manage to run the code with the normalised data in the due time, so the results presented in the following section will contain only the recommendations obtained initially (*i.e.*, without normalisation).

The normalisation of the ratings matrix would have consisted in applying the `normalize` function of the *recommenderlab* package, as is summarised in the following instruction:

3.5.4 Binarisation of the data

Using the "realRatingMatrix" class from the *recommenderlab* package, we created a real-valued sparse rating matrix, from which we generated a binary version of it using the `binarize` instruction from the same package. Basically, `binarize` created a matrix belonging to the "binaryRatingMatrix" class by setting all ratings equal to or larger than 0.5 to 1, and all others to 0.[1]

3.5.5 Train/test splitting of the data

We split the data set into two pieces, in a 7-to-3 ratio: the *train* set, containing 70 % of the data, and the *test* set, with the remainder 30 %. The former was used to train the models, while the latter was used to simulate "fresh" data to evaluate the performance of the models.

Both the training and the testing sets were generated from the rating matrices, so we ended up with four sets: one train-test pair for the binary approach, and another train-test pair for the no-binary approach. The split resulted in *recomenderlab*'s structures of class "binaryRatingMatrix" (binary case) and class "realRatingMatrix" (no binary case). The train sets consisted of

[1]Perhaps a more reasonable approach would have been to make all ratings bellow 2 or 3 as 0, and above it equal to 1, but since this is an exercise, this choice does not impact on the type of outcomes.

$12\,122 \times 2540$ matrices with $3\,335\,518$ ratings, and the test sets to 5210×2540 rating matrices with $1\,434\,740$ ratings.

Chapter 4

Recommendation engines

The goal of this section was to implement systems that would deliver relevant results, *i.e*, effective recommendations to Flixster's users.

Three recommender methods were applied to both the binary and non-binary approaches: *item-based collaborative filtering* (IBCF), *user-based collaborative filtering* (UBCF), and the *popular* method.

The IBCF and UBCF methods take into account the information about the users and the inherent collaboration between them to obtain recommendations of items. While in the former the recommendations are based on the similarity between items, in the latter the recommendations are performed on a neighbourhood of users that share similar tastes. We used the Cosine distance function (default in the *recommenderlab* package) for both.

The popular method simply recommends the most popular films, based on the number of ratings they have received.

In the following sections, we will present the code implemented for all of those methods applied in both the binary and non-binary cases, and the outputs of their application on the recommendation of 10 films to the first user of the `flixster.u50m500` table.

4.1 Binary approach

4.1.1 IBCF

The recommendations obtained for the first user are presented in tab. 4.1. A common characteristic to all the films is the age group for which they are intended, typically at younger ages.

n	Movie name

Table 4.1: List of the 10 recommended films to user #1, obtained with the IBCF method using a binary approach.

n	Movie name
1	“The Lion King”
2	“Men in Black”
3	“Mrs. Doubtfire”
4	“Home Alone”
5	“The Mask”
6	“Jumanji”
7	“Jurassic Park”
8	“Toy Story 2”
9	“A Bugs Life”
10	“Dr. Dolittle”

4.1.2 UBCF

The ten recommended movies are indicated in tab. 4.2. These ten films are a bit different from the ones in 4.1, withonly one movie in common. That is expected, since the results in tab. 4.2 were obtained taking into account a neighbourhood of users sharing similar tastes.

Table 4.2: List of the 10 recommended films to user #1, obtained with the UBCF method using a binary approach.

n	Movie name
1	“The Lion King”
2	“Spider-Man 3”
3	“Forrest Gump”
4	“Halloween”
5	“The Brave One”
6	“Sydney White”
7	“Mr. Beans Holiday”
8	“War (Rogue Assassin)”
9	“Lucky You”
10	“The Ex”

4.2 Popular

The ten movies recommended for the first user by this method are compiled in tab. 4.3. This table presents films in common to the two previous tables. The titles were selected according to the user and the most popular movies.

Table 4.3: List of the 10 recommended films to user #1, obtained with the Popular method using a binary approach.

n	Movie name
1	"The Lion King"
2	"Men in Black"
3	"Home Alone"
4	"The Mask"
5	"Mrs. Doubtfire"
6	"Jurassic Park"
7	"Forrest Gump"
8	"X-Men"
9	"The Fast and the Furious"
10	"Jumanji"

4.3 Non-binary approach

In the non-binary approach, the code used to obtain the results was similar to the one used in the binary approach. In what follows, we just refer the resulting tables with the recommendations for each method.

4.3.1 IBCF

The films recommended for the first user are indicated in tab. 4.4. These recommendations are very different from the equivalent ones obtained with the binary approach. However, they are apparently consistent with each other.[1]

Table 4.4: List of the 10 recommended films to user #1, obtained with the IBCF method using a non-binary approach.

n	Movie name
1	"batteries not included"
2	"88 Minutes"
3	"8 Seconds"
4	"A Nightmare on Elm Street 2 - Freddys Revenge"
5	"A Walk in the Clouds"
6	"Above the Law"
7	"Air Bud Spikes Back"
8	"Adaptation"

[1]We have to bare in mind that we do not have a genre feature in the data set, so our analysis cannot fall on this characteristic.

n	Movie name
9	“Adventures in Babysitting”
10	“Aliens vs. Predator: Requiem (AVP 2)”

4.3.2 UBCF

The table with the recommendations for the first user in the `flixster.u50m500` tibble are presented in tab. 4.5.

Table 4.5: List of the 10 recommended films to user #1, obtained with the UBCF method using a non-binary approach.

n	Movie name
1	“101 Dalmatians”
2	“102 Dalmatians”
3	“12 Angry Men (Twelve Angry Men)”
4	“15 Minutes”
5	“21 Grams”
6	“28 Weeks Later. . . ”
7	“3 Ninjas”
8	"*batteries not included"
9	“101 Dalmatians (One Hundred and One Dalmatians)”
10	“12 Rounds”

4.3.3 Popular

Finally, the movies recommendations for the first user of the data set are compiled in tab. 4.6. Since the *popular* technique relies on the popularity of the films, it is normal to get movies of different genres—although we recall that we do not have access to a genre feature in the data set.

Table 4.6: List of the 10 recommended films to user #1, obtained with the Popular method using a non-binary approach.

n	Movie name
1	“Charlottes Web (1973)”
2	“Hana”
3	“13 Ghosts (1960)”
4	“Gone in 60 Seconds (1974)”
5	“Inglorious Bastards (Deadly Mission) (Counterfeit Commandos)”
6	“Casper (1995)”
7	“Reindeer Games (Deception)”
8	“Iron Man (Ironman)”
9	“Beauty and the Beast (1992)”

n	Movie name
10	“Barb Wire”

At the end, all models were saved for future reference.

Chapter 5

Evaluation of the recommendation engines

We evaluated the performance of the methods implemented in the previous section by means of *k-fold cross-validation* (kCF). In this approach, the data is split into k blocks, one of which is set aside as the test set and used to compute the accuracy. The process is repeated k times, using a different fold as the test data in each run. We used a 4-fold CV, thus obtaining four sets of the same size (12 630 ratings in both the binary and non-binary approaches).

The code for the binary and the non-binary approaches is similar. We started by creating a list of the models, and to define a set of parameters:

- `TOP.N`: the top N recommendations;
- `ITEMS`: number of films to generate recommendations;
- `RAT.THRESH`: minimum rating to be considered as a good option;
- `N.FOLDS`: number of folds/samples to run the evaluation to (for the cross-validation).

Then, we selected the most relevant data using the same criteria as before: users with more than 50 ratings, and films which have been rated more than 500 times. The resulting selection yields two $16\,841 \times 2066$ rating matrices respectively of class “binaryRatingMatrix” and “realRatingMatrix”, with 4 559 716 ratings each.

We applied the 4-fold CV, and evaluated the list of recommender models. We got four sets of size 12 630.

We collected the training data for the runs, the known ratings used for the predictions in the test data, and the ratings used for evaluation of the test data, and plotted the histogram of the training and testing data corresponding to the distribution of ratings per user (fig. 5.1).

As expected, the number of users decrease with the number of ratings, *i.e.*, less and less users rate many films.

Then, using the `getConfusionMatrix` from the *recommenderlab* package, we created confusion matrices for each method, and summed up the indices of all metrics in order to have an overall picture of all the folds.

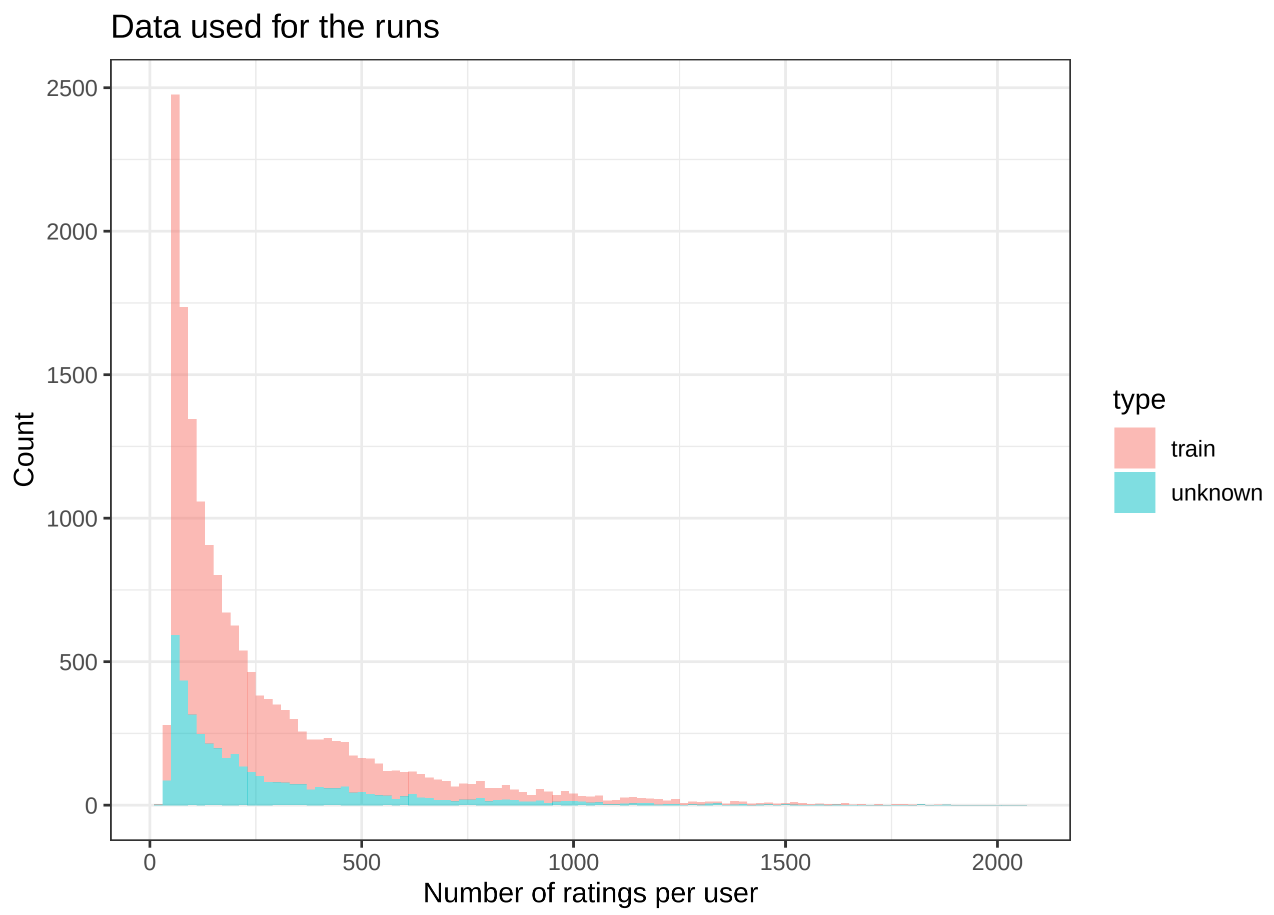


Figure 5.1: Distribution of the number of ratings per user in the train and test data sets. As expected, the number of users rating films decreases with the amount of ratings per movie.

```r
cm.ibcf.bin= getConfusionMatrix(results.bin$ibcf)
cm.ubcf.bin= getConfusionMatrix(results.bin$ubcf)
cm.pop.bin=  getConfusionMatrix(results.bin$pop)

# condense results of all folds
columns.to.sum= c("TP", "FP", "FN", "TN", "precision", "recall", "TPR", "FPR")
cm.sum.ibcf.bin= Reduce(
  "+",
  getConfusionMatrix(results.bin$ibcf)
)[, columns.to.sum]
cm.sum.ubcf.bin= Reduce(
  "+",
  getConfusionMatrix(results.bin$ubcf)
)[, columns.to.sum]
cm.sum.pop.bin= Reduce(
  "+",
  getConfusionMatrix(results.bin$pop)
)[, columns.to.sum]
```

Finally, we plotted the ROC curves and the precision/recall graphs (fig. 5.2)

From the ROC curve and the corresponding *area under the curves* (AUCs), and from the precision-recall curve, we conclude that the best performing method for recommendation of Flixster's films is Popular, since this method attains the highest values for ROC, AUC, and precision-recall.

We applied the same procedure for the non-binary approach, and we obtained similar results, having the Popular method attained the best performance out of the three methods.

The time needed to run the code for each of the approaches (binary and non-binary) was large in our machines. For instance, the evaluation of the recommendation engines took approximately three hours to complete.

In an attempt to reduce the time of execution, we repeated the whole process for a smaller data set, containing uniquely the variables `userid`, `movieid`, `moviename`, and `rating`—we do not reproduce the code here, as it is virtually the same as the one we presented in the previous sections, and the results are analogous.

However, we realised that the time of execution was similar to the previous approaches. Therefore, the solution would be to remove even more rows from the data set, using stronger restrictions (for example, keeping only users with 500 or more ratings, and films with over 1000 ratings).

Another approach would be to apply stratified sampling, in order to extract a sample of records with the highest statistical significance possible, keeping the original multivariate histogram as faithfull as possible. This was left as future work, since we had already run out of time.

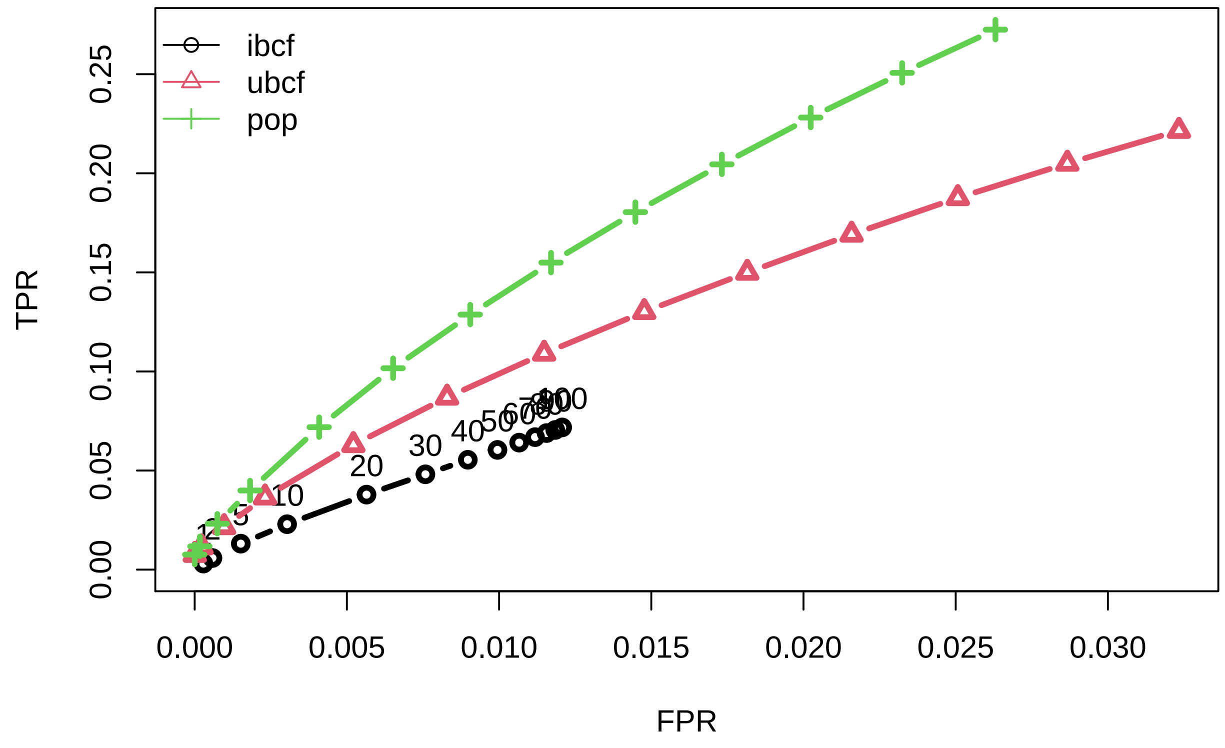


Figure 5.2: ROC curves for the methods IBCF, UBCF, and Popular. The latter is the method with the best performance.

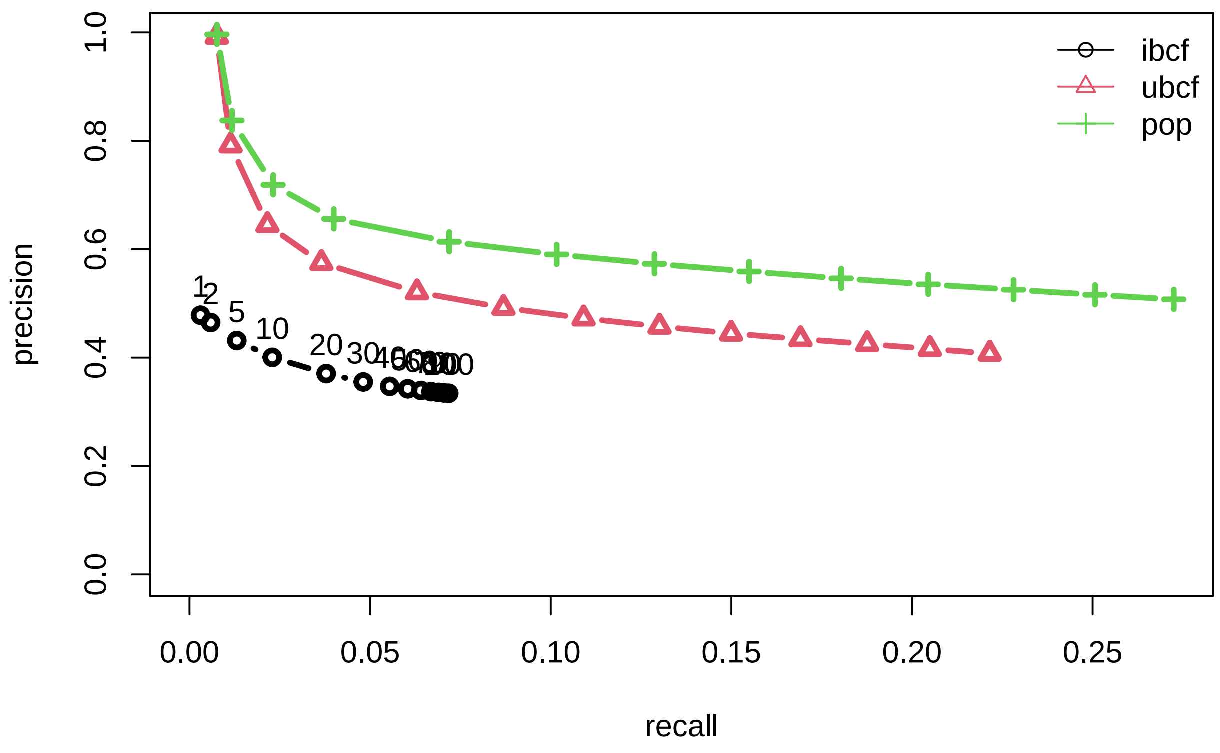


Figure 5.3: Precision-Recall curve for the methods IBCF, UBCF, and Popular. The latter is the method with the best performance.

Chapter 6

Association rules

We tried to build a recommendation system based on association-rule mining techniques, by applying the *Apriori* algorithm. Given the data set, the goal was to find the films that are associated with each other. This would give us the opportunity to understand associations between the movies.

We did two main attempts to get association rules out of the Flixster data set. In the first one, before applying the Apriori algorithm, we transformed the data set to binary values, where 1 represents a positive rating, and 0 represents a negative classification or no rating at all. The rating threshold was 3, *i.e.*, all ratings bellow 3 were classified as 0, and all equal or above 3 were classified as 1. We used the `binarize()` function from the *recommenderlab* package for that task.

Since the Apriori algorithm expects a matrix as input rather than an object of class "binaryRatingMatrix", we converted the previous object to the matrix format.

The output was a matrix of booleans, which we converted to a matrix of ones and zeros:

We passed the parameters `support` and `confidence` to the algorithm. The *support* tells us the fraction of transactions that contain an itemset. It measures how often a rule should occur in the data set. The *confidence* measures the strength of the rule. We initially tried a support of 0.5 and a confidence of 0.8 in order to keep only meaningul rules. We obtained 13 rules, but they were not meaningful, since some of them were empty, while in others the *right-hand-side* was always equal to "{userid}"—this might indicate a flaw in our algorithm. As the number of returned rules was small and not meaningul, we relaxed the support and confidence thresholds to 0.3 and 0.7, respectively.

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.7    0.1    1 none FALSE            TRUE       5     0.3      1
##  maxlen target  ext
##      10  rules TRUE
```

```
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 5199
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[2540 item(s), 17332 transaction(s)] done [1.32s].
## sorting and recoding items ... [147 item(s)] done [0.03s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.15s].
## writing ... [1408 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

The `rules` object has all the film associations that were extracted and mined from the data set. With these parameters, 1408 movie association rules were extracted, in total. A subset of those is indicated in tab. 6.1.

Table 6.1: Sample of the inspection of the rules generated with the *Apriori* algorithm on the Flixster data set. Some of the rules are not meaningul, as they are empty or point to "userid", instead of a value of `movieid` (this is an indication of a flaw in our algorithm).

n	lhs		rhs	support	confidence	coverage	lift	count
1	{}	=>	{userid}	1.00000000	1.000000000	1.00000000	1.000000	17332
2	{19787}	=>	{userid}	0.3122548	1.0000000	0.3122548	1.000000	5412
21	{20642}	=>	{24251}	0.3071775	0.7397527	0.4152435	1.460462	5324
22	{20642}	=>	{20645}	0.3103508	0.7473947	0.4152435	1.468523	5379
125	{30936}	=>	{45119}	0.3231595	0.7933428	0.4073390	1.305690	5601

Chapter 7

Conclusions and prospects

We analysed the Flixster data set and created recommender systems based on a binary and non-binary approach, applying the *item-based confident frequency*, the *user-based confident frequency*, and the *popular* methods, and association rules.

The number of records in the original data set was reduced by approximately 58 %, by applying restrictions on the number classifications produced by users and on the number of films' ratings. This was an attempt to, on the one hand, to make the data set more relevant and, on the other hand, to reduce the code's execution time. While the relevance of the data was indeed increased, the time of excecution did not change, and other approaches needed to be tried, such as using stronger restrictions on the ratings per user and per film, or applying stratified sampling—it would be interesting to compare the recommendations obtained with any of these approaches with the ones we presented in this report.

For each of the first three aforementioned methods, we managed to recommend 10 films for a randomly chosen user (we picked the first in the data set) in both the binary and non-binary approaches. Out of the three, the method which attained the best score in the ROC curves, the AUC, and in the precision-recall plots was the *popular* recommender system. Since this method simply recommends films based on their ratings, this result becomes interesting in the context of human behaviour. In fact, it is indicative that people tend to search for and watch the most popular films, instead of searching for titles more similar to previous movies they have seen and enjoyed—this would be an appealing subject for further analysis.

We were able to produce association rules out of the data set, by setting values for the support and the confidence. However, out of the hundreds generated, not all of them were meaningful, as by inspection we could spot empty fields and senseless right-hand sides.

This ended up to be a very challenging assignment, not so much because of the nature of the problem, but mostly because of the limitations of our computers. The code took too long to finish, specially when generating non-binary UBCF recommendations and when comparing the models. We feel we could have done a better job, and some of the tasks initially programmed were not fulfilled, such as obtaining recommendations using a normalised data set, or trying hybrid recommender systems. These are fields we are keen to explore in a future opportunity.

References

In addition to the references highlighted within the report, we got inspiration from other sources, such as:

- Ivamoto, V., 2019, *Movie Recommendation System in R*: for the EDA;
- Novikova, J., 2016, *Building a Movie Recommendation System*: for the evaluation and comparison of the models;
- Liu, B., 2011, *Web Data Mining*, Springer: for main concepts;
- Ribeiro, R., 2020, *Notes on Data Mining II / Advanced Topics in Data Science*: for concepts, ideas, and general guidance.

Lesmeister, Cory, and Sunil Kumar Chinnamgari. 2019. *Advanced Machine Learning with R.* Packt Publishing Ltd.

"The Netflix Prize." 2009. https://www.netflixprize.com/.