A movies recommender system using Flixster data

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

By Robson Teixeira and Nuno Gomes

M:DS – 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. ADD MAIN RESULTS HERE.

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 online 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. 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, while applying the knowledge acquired in the course of Advanced Topics in Data Science/Data Mining II over the second semestre of the Masters in Data Science.

This report is organised as follows:

Chapter 2

The Flixster data set

Flixster was an American social-networking online 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 online.[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 presented in what follows.

```
## 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
```

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

```
## # A tibble: 6 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 &#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 infinites. 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 type-integer userid, location, and lastlogin, and the type-character 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.

```
## 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

Variable	Type	Description	Tibble
rating	double	Rating of movie by user	ratings
userid	integer	User unique identifier	profiles
userid	integer	User unique identifier	ratings

Chapter 3

Exploratory data analysis and engineering

The creation of a recomendation 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.

```
movies.raw= data.frame(movies.orig)
profiles.raw= data.frame(profiles.orig)
ratings.raw= data.frame(ratings.orig)
```

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.

```
ratings.raw$date= ratings.raw$date %>% str_replace(" 00:00:00", "")
profiles.raw$memberfor= profiles.raw$memberfor %>% str_replace(" 00:00:00", "")
```

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

```
profiles.raw$age= as.integer(profiles.raw$age)
profiles.raw$profileview= as.integer(profiles.raw$profileview)
profiles.raw$gender= as.factor(profiles.raw$gender)
profiles.raw$memberfor= as.Date(profiles.raw$memberfor)
ratings.raw$date= as.Date(ratings.raw$date)
```

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

```
movies.raw$moviename= movies.raw$moviename %>%
  str_replace_all("&#233;", "é")
movies.raw$moviename= movies.raw$moviename %>%
  str_replace_all("&amp;", "&")
movies.raw$moviename= movies.raw$moviename %>%
  str_replace_all("&#\\d*;", "")
```

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

```
movies= tbl_df(movies.raw)
profiles= tbl_df(profiles.raw)
ratings= tbl_df(ratings.raw)
```

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.

```
profiles= profiles %>% select(-"profileview")
```

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

```
movies
```

```
## # 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
```

```
profiles
```

```
## # 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 Female      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
```

```
ratings
```

```
## # 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 profiles	factor	Gender of the user (*Female*, *Male*, and *Other*):w	
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

Variable	Type	Description	Tibble
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

```r
age= profiles$age
dates= ratings$date
gender= fct_explicit_na(profiles$gender, "Other")
lastlogin= profiles$lastlogin
location= profiles$location
memberfor= profiles$memberfor #
movieid.movies= movies$movieid
movieid.ratings= ratings$movieid
moviename= movies$moviename
rating= ratings$rating
userid.profiles= profiles$userid
userid.ratings= ratings$userid
```

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. 2.1.

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.

```r
summary(age) # min= 12; max= 113; NAs= 255618
```

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

```r
idx.nas.age= which(is.na(age)) # 25.5% of all age values
age= na.omit(age)
```

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

```r
var(age) # 107.5378
```

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

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

```
skewness(age) # 2.444335
```

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

We plotted the histogram and the boxplot of `ages` (fig. 3.1).

```
par(mfrow= c(1, 2), oma= c(0, 2, 3, 1))
hist(age,
  breaks= seq(round(min(age)) - 1, round(max(age)) + 1, by= 1),
  xlab= "Age (yrs)"
)
boxplot(age,
  yaxt= "n",
  main= "Boxplot of age"
)
axis(2, las= 2)
```

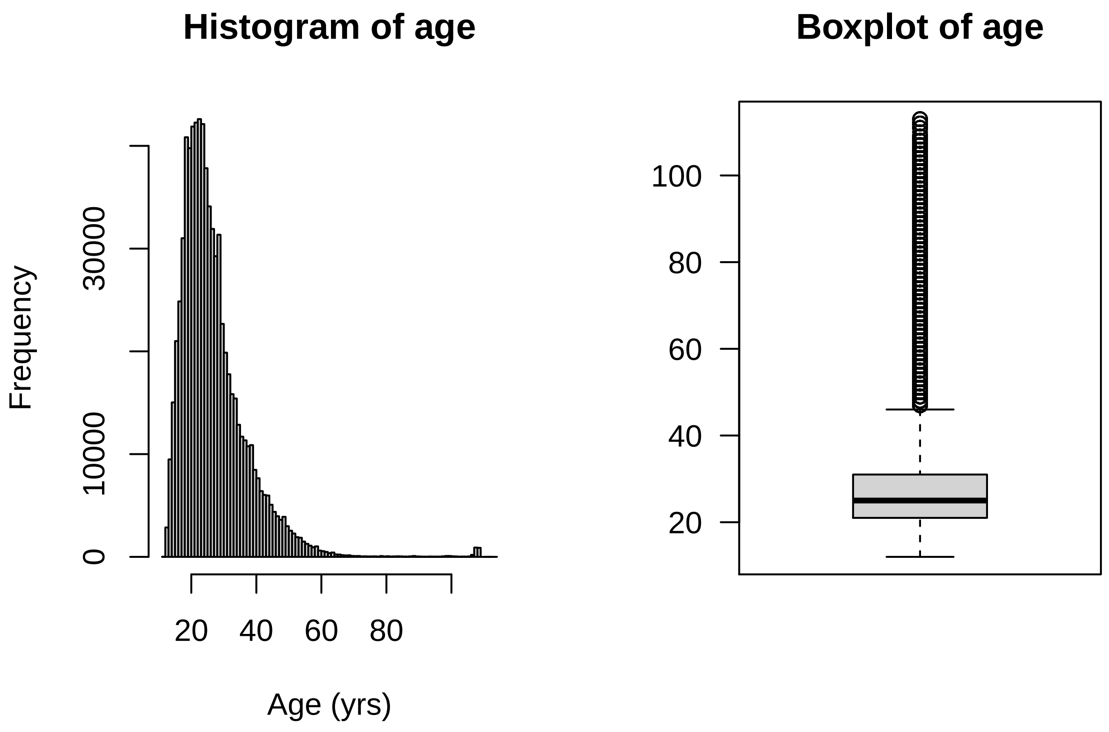

Figure 3.1: Histogram (*left*) and boxplot (*right*) of the `age` variable. The distribution is right skewed and exihibits several outliers.

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 skeweness of the distribution. This is expected, since Flixster's customers were typically young.

The set of ages above 71 years old corresponded to 0.5 % of all ages and, thus, they could be safelly considered as outliers. The histogram and boxplot of `ages` without outliers is represented in fig. 3.2.

```r
age.out= boxplot.stats(age, coef= 4)$out
age.no.out.idx= !(age %in% age.out)
age.no.out= age[age.no.out.idx]
summary(age.no.out) # max= 71 (99.5% of values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.00   21.00   25.00   27.05   31.00   71.00
```

```r
par(mfrow= c(1, 2), oma= c(0, 2, 3, 1))
hist(age.no.out,
  breaks= seq(round(min(age.no.out)) - 1, round(max(age.no.out)) + 1, by= 1),
  xlab= "Age (yrs)",
  main= "Histogram of age"
)
boxplot(age.no.out, outline= F, yaxt= "n")
axis(2, las= 2)
mtext(
  side= 3, line= 2, at= 1, cex= 1.2,
  expression(paste("Boxplot of age"))
)
mtext(side= 3, line= 1, at= 1, cex= 0.7, "Outliers removed")
```

3.2.2 Dates and Memberfor

We detected several incorrect dates in the data set, both in the `date` and in `memberfor`, respectively from the `ratings` and the `profiles` tibbles. Those dates were either before the founding of Flixster (2006/01/20) or NA values—we detected dates as early as 1900/01/01 for `memberfor` and 1941/12/07 for `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 Fixster's founding date. We ended up with two arrays with dates between 2006/01/20 and 2009/11/17 (`dates` case) and 2009/12/03 (`memberfor` case), with no NAs.

```r
date.flixster= as.Date("2006-01-20")
idx.memberfor.wrong= which(profiles$memberfor < date.flixster) # 57722
profiles$memberfor[idx.memberfor.wrong]= date.flixster
```

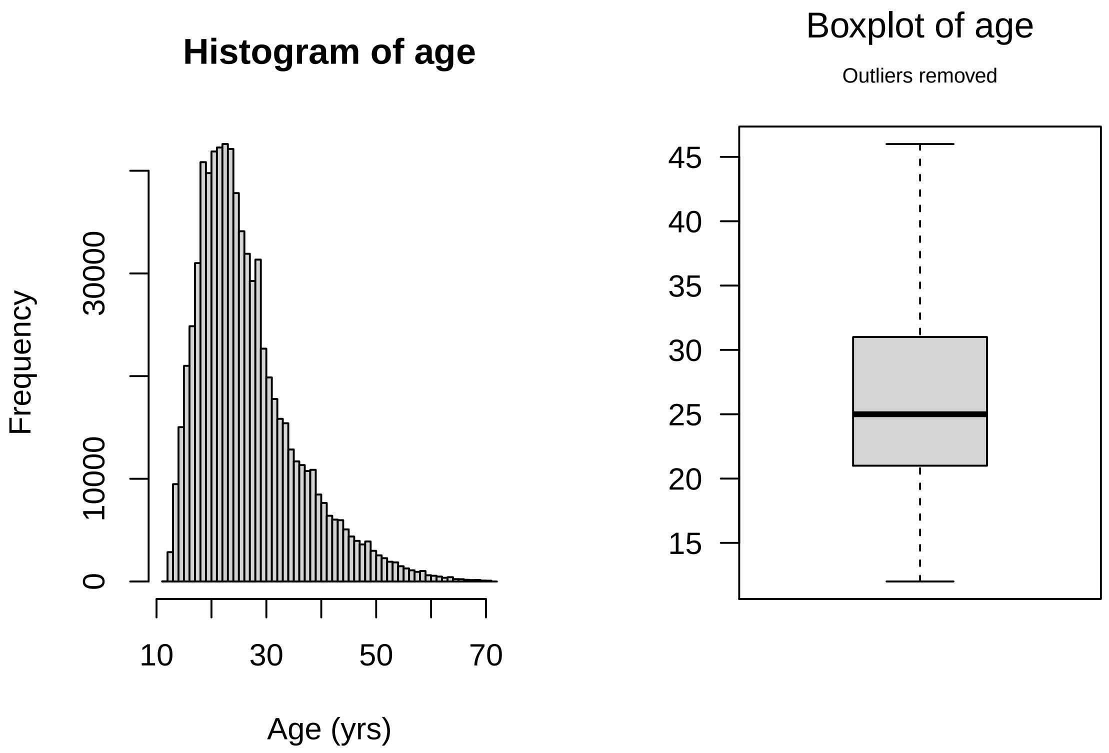

Figure 3.2: Histogram (*left*) and boxplot (*right*) of `ages` after removing the outliers.

```r
memberfor= profiles$memberfor
idx.nas.memberfor= which(is.na(memberfor)) # 203
uids.memberfor.nas= profiles$userid[idx.nas.memberfor] # 203
```

```r
# fill in NAs in memberfor with earliest rating date
# join profiles and ratings tables
users.ratings= tbl_df(merge(profiles, ratings, by= "userid"))
for (i in 1:nrow(users.ratings)) {
  if (is.na(users.ratings$memberfor[i])) {
    users.ratings$memberfor[i]= users.ratings$date[i]
  }
}
write.csv(users.ratings, file= '../data/users-ratings.csv')
users.ratings= read.csv("../data/users-ratings.csv")
memberfor= as.Date(users.ratings$memberfor)
summary(memberfor)
write.csv(memberfor, "../data/memberfor.csv")
```

```r
tmp= read.csv("../data/memberfor.csv")
memberfor= as.Date(tmp$x)
```

```r
idx.date.wrong= which(ratings$date < date.flixster) # 2998
# sum(idx.date.wrong %in% idx.memberfor.wrong) # 35
ratings$date[idx.date.wrong]= date.flixster
dates= ratings$date
```

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. **??**). The set of Flixster's users was comprised of 49 % of males, 44 % of females, and 7 % of other/unidentified genders.

```r
n.gender= length(gender)
gender.woman= gender[gender == "Female"]
n.woman= length(gender.woman)
gender.woman.pct= round(n.woman / n.gender * 100)
gender.man= gender[gender == "Male"]
n.man= length(gender.man)
gender.man.pct= round(n.man / n.gender * 100)
gender.other= gender[gender == "Other"]
n.other= length(gender.other)
gender.other.pct= round(n.other / n.gender * 100)
gender.labels= c(
  paste('Women:', gender.woman.pct),
  paste('Men:',   gender.man.pct),
  paste('Other:', gender.other.pct))
gender.labels= paste0(gender.labels, '%')
gender.colours= c("#FA9FB5", "#74A9CF", "#2ECC71")
par(mfrow= c(1, 1), oma= c(0, 0, 0, 0))
pie3D(c(n.woman, n.man, n.other), theta= pi/3,
      labels= gender.labels, labelcex= 1.5,
      col= gender.colours,
      start= pi/4, explode= 0.08)
mtext("Gender spread", side= 3, line= -4, outer= T, cex= 2)
```

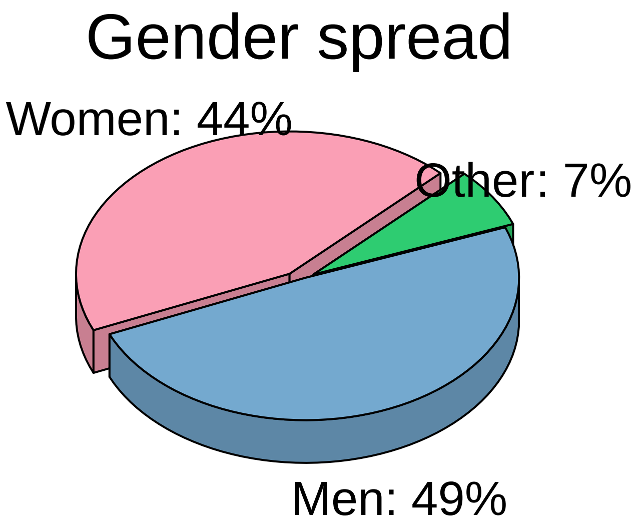

Figure 3.3: Gender spread among Flixster users.

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 boxplot of the distribution with and witout outliers are represented in figs. ?? and ??, respectively.

```
idx.nas.lastlogin= which(is.na(lastlogin)) # 5.7% of all values
lastlogin= na.omit(lastlogin)
var(lastlogin) # 102829.2
skewness(lastlogin) # 217.3447
```

```
hist(lastlogin,
  breaks= seq(round(min(lastlogin)) - 1, round(max(lastlogin)) + 1),
  xlab= "Last login"
)
```

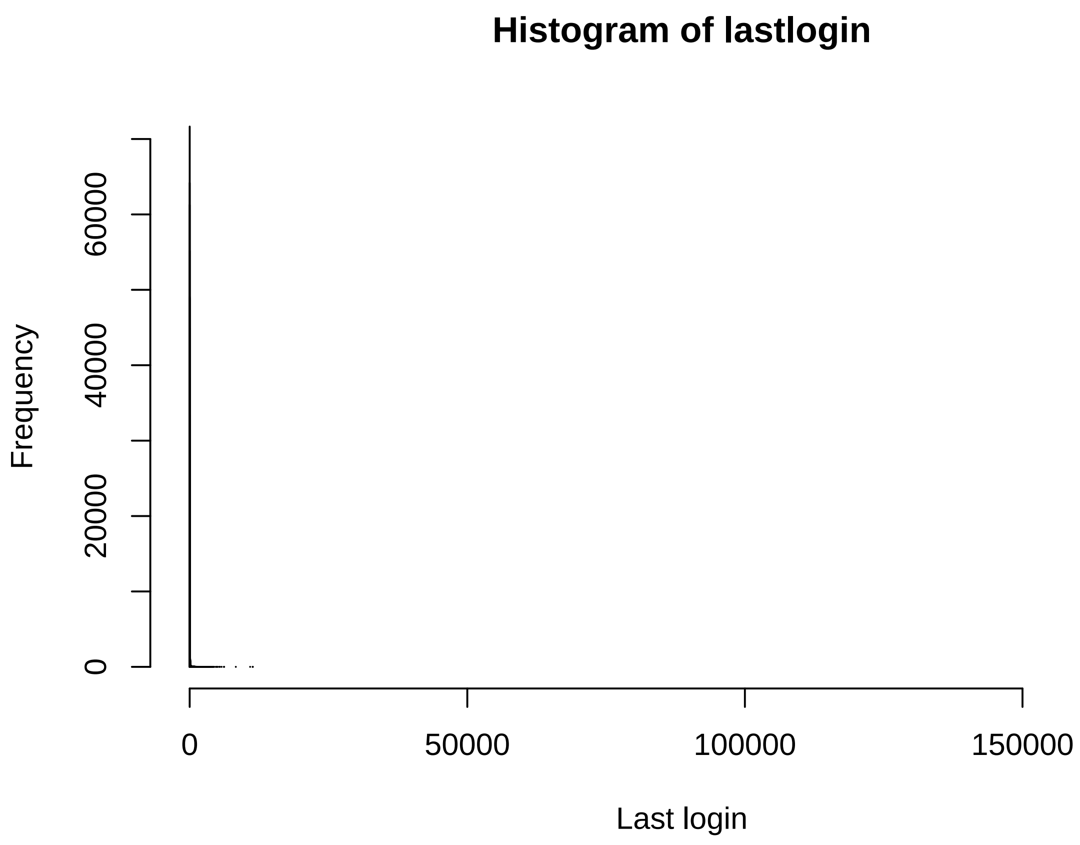

Figure 3.4: Histogram (*left*) and boxplot (*right*) of the variable `lastlogin`. Some outliers lie very far away from the median.

```
boxplot(lastlogin,
  yaxt= "n",
  main= "Boxplot of lastlogin"
)
axis(2, las= 2)
```

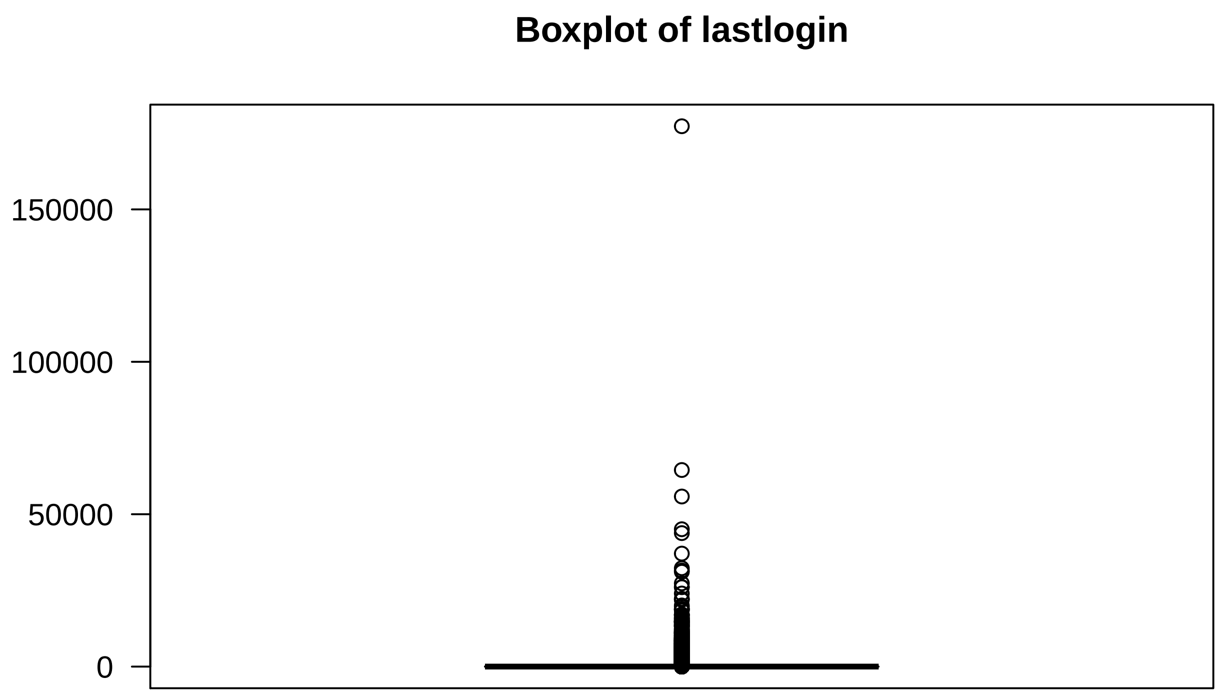

Figure 3.5: Histogram (*left*) and boxplot (*right*) of the variable `lastlogin`. Some outliers lie very far away from the median.

```
### remove outliers
lastlogin.out= boxplot.stats(lastlogin, coef= 20)$out
lastlogin.no.out.idx= !(lastlogin %in% lastlogin.out)
lastlogin.no.out= lastlogin[lastlogin.no.out.idx]
hist(lastlogin.no.out,
  breaks= seq(round(min(lastlogin.no.out)) - 1, round(max(lastlogin.no.out))),
  xlab= "Last login",
  main= "Histogram of lastlogin"
)
```

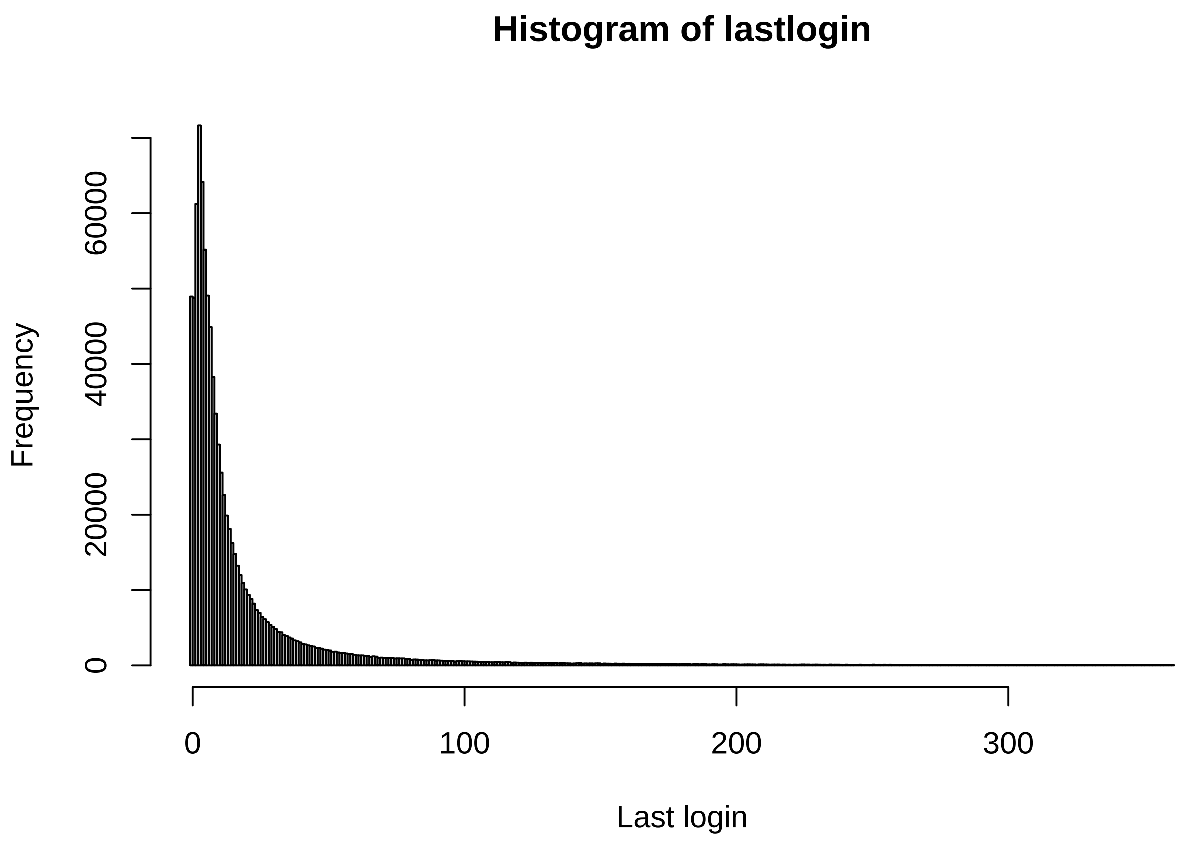

Figure 3.6: Histogram (*left*) and boxplot (*right*) of the variable `lastlogin` after removing the outliers.

```
boxplot(lastlogin.no.out, outline= F, yaxt= "n")
axis(2, las= 2)
mtext(
  side= 3, line= 2, at= 1, cex= 1.2,
  expression(paste("Boxplot of lastlogin"))
)
mtext(side= 3, line= 1, at= 1, cex= 0.7, "Outliers removed")
```

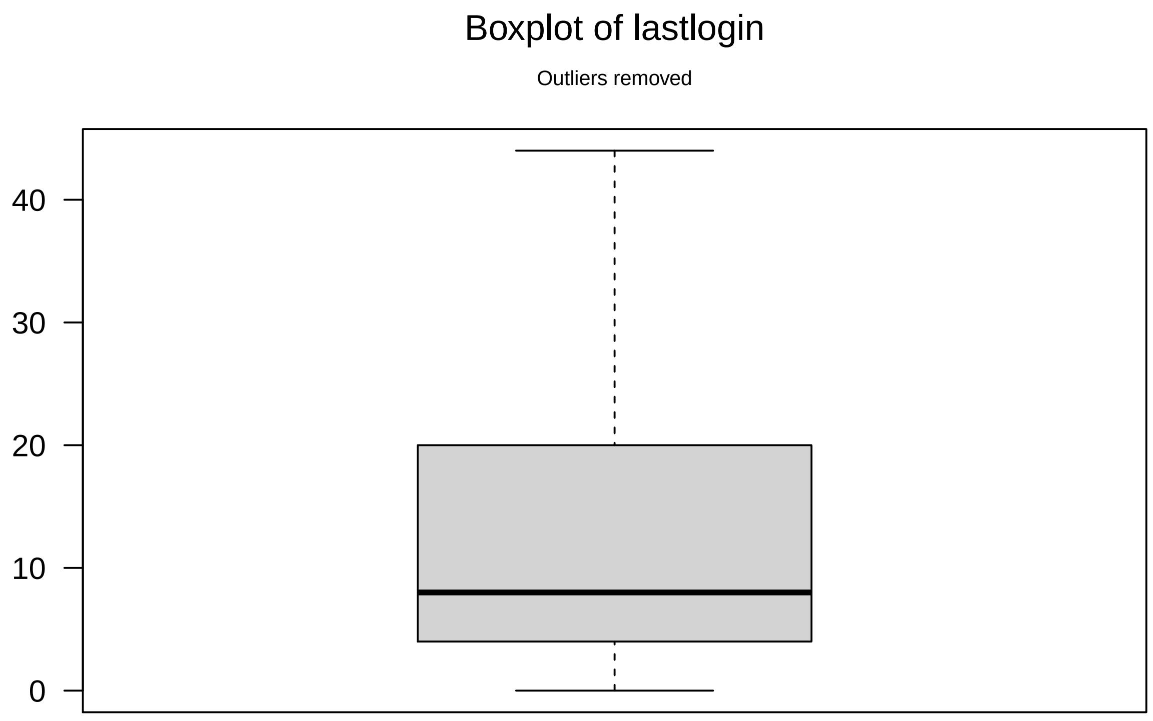

Figure 3.7: Histogram (*left*) and boxplot (*right*) of the variable `lastlogin` after removing the outliers.

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

The variables *location*, *memberfor*, *lastlogin* and *profileview* are not needed for the context this analysis, so they should be removed.

There are a significant number of *NA's* in variables *gender* (67529 or around 6.73% of total) and *age* (255618 or around 25.49% of total). We adopt the aproach the replacement the empty values into a new value to treatmet of missing values. The empty values into variable *gender* will be replaced for new value `Other`. For the empty values into *age* variable will be set the value `0`.

```
movies= movies.orig
profiles= profiles.orig
ratings= ratings.orig

## Remove no relevant variables
profiles <- select(profiles, -c(location, memberfor, lastlogin, profileview) )

## Setting 'gender' as factors and 'age' as integer
profiles <- profiles %>% mutate(gender = as.factor(gender)) %>% mutate(age = as.integer

## Add column range_age into user data set
profiles <- profiles %>% mutate(range_age = cut_width(age, 10, boundary = 0) )

## replace NA values into gender with "unknown"
profiles <- profiles %>% mutate(gender = replace(gender, is.na(gender), "unknown"))

## replace NA values into age with "0"
profiles <- profiles %>% mutate(age = replace(age, is.na(age), 0))

flixster <- ratings %>% left_join(profiles, by = "userid") %>% left_join(movies, by = "m

remove(movies, profiles, ratings)
```

We split the dataset in two parts, the training set called `train_set` and the test set called `test_set` with 70% and 30% of the original dataset respectively.

```
# 'test_set' will be 30% of dataset
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = flixster$rating, times = 1, p = 0.3, list = FALSE)

train_set <- flixster[-test_index,]
temp <- flixster[test_index,]

# Make sure userid and movieid in 'test_set' are also in 'train_set'
test_set <- temp %>% semi_join(train_set, by = "movieid") %>% semi_join(train_set, by =
```

```r
# Add rows removed from 'test_set' set back into 'train_set' set
removed <- anti_join(temp, test_set)
```

```
## Joining, by = c("userid", "movieid", "rating", "date", "gender", "age", "range_age",
```

```r
train_set <- rbind(train_set, removed)

rm(test_index, temp, flixster, removed)
```

3.3 Data Exploration

Before start building the model, we need to understand the structure of the data, the distribution of variables and the relationship of the predictors. This information will help build a better model.

3.3.1 Profile Data Set

```r
glimpse(train_set)
```

```
## Rows: 5,757,706
## Columns: 8
## $ userid    <int> 882359, 882359, 882359, 882359, 882359, 882359, 882359, 8...
## $ movieid   <int> 1349, 2270, 3522, 3583, 4216, 4871, 5124, 9180, 10033, 11...
## $ rating    <dbl> 2.0, 1.0, 0.5, 0.5, 0.5, 3.5, 1.0, 1.0, 1.0, 2.0, 5.0, 2....
## $ date      <chr> "2007-10-10 00:00:00", "2007-10-10 00:00:00", "2007-11-13...
## $ gender    <fct> Female, Female, Female, Female, Female, Female, Female, F...
## $ age       <dbl> 108, 108, 108, 108, 108, 108, 108, 108, 108, 108, 108, 10...
## $ range_age <fct> "(100,110]", "(100,110]", "(100,110]", "(100,110]", "(100...
## $ moviename <chr> "A Nightmare on Elm Street 3 - Dream Warriors", "Aladdin"...
```

From this initial exploration, we can see that `train_set` consists of 5757622 observations spread across 8 variables. The user information are stored in userid, gender, age, range_age columns; the movie information are stored in movieid and moviename columns; the rating information are stored in rating and date columns.

References

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