MovieLens Report

Dani Beltrán

February 4, 2019

Introduction

MovieLens is a web-based recommender system: it is a virtual community that recommends movies for its users to watch. The MovieLens dataset contains about 11 million ratings. Each rating is followed by the rater's user ID, the time when the rate was done, the rated movie ID and additional information from this movie, such as the title, year and genres.

The main goal of this project is to analyze the MovieLens dataset and then develop an algorithm to predict ratings. The evaluation criteria for this algorithm is the RMSE (Root Mean Square Error), which is expected to be lower than 0.8775.

In first place, all variables are explored. They are analyzed in order to determine if they are related or not with the rating. After that, a first approach is made with the mean rate of each movie and the mean trend of each user to rate over or under the mean. This model results in a satisfying RMSE.

Methods

Dataset creation and partition

First of all, the dataset is downloaded and split in 2 sections: the train set (edx) and the test set (validation). They contain around the 90% and the 10% of the total data set respectively. In addition, test set ratings with a movie ID or user ID which is not found in the train set are removed.

The following code is provided by edx in the web page. It's placed in the "Data Science: Capstone" course, at the "Create Test and Validation Sets" section.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(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(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

```r
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

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

Ratings and timestamp exploration

A first preview with a random sample of 1000 ratings over the timestamp is made.

```r
figure1 <- edx[sample(nrow(edx), 1000),] %>% ggplot(aes(timestamp, rating)) + geom_point() +
  xlab("Timestamp") + ylab("Rating")
```

The time when ".5" ratings start to be possible is defined as "h_time".

```r
h_time <- edx %>% filter(rating %in% c(4.5,3.5,2.5,1.5,0.5)) %>% summarize(t=min(timestamp)) %>% .$t
```

The mean of the rates over time is represented by splitting the data in 1000 groups, which are defined according to timestamp ranges. Also a red vertical line is added in the h_time.

```r
figure2 <- edx %>% group_by(gr=cut(timestamp, breaks= seq(min(timestamp),max(timestamp), length.out=100
    summarise(Time = min(timestamp), Mean_Rate = mean(rating)) %>% ggplot(aes(Time, Mean_Rate)) +
   geom_point() + geom_vline(xintercept=h_time, colour="red") + xlab("Timestamp") + ylab("Mean rate by
```

The rates distribution before and after the h_time are plotted.

```r
figure3 <- edx %>% filter(timestamp < h_time) %>% ggplot(aes(rating)) + geom_histogram(binwidth=.5) +
  xlab("Rating") + ylab("Number of rates")
figure4 <- edx %>% filter(timestamp >= h_time) %>% ggplot(aes(rating)) + geom_histogram(binwidth=.5) +
  xlab("Rating") + ylab("Number of rates")
```

Movies data analysis

Data is grouped by movieId. Then, the title, genres, number of rates and mean rate of each movie is saved as "movieInfo".

```r
movieInfo <- edx %>% group_by(movieId) %>%
  summarise(title=title[1], genres=genres[1], number=length(rating), meanmovierate=mean(rating))
```

Repeated movies are searched

```r
r1 <- movieInfo %>% filter(title %in% title[duplicated(title)])
```

The possibility that some users voted for the repeated movie 2 times is checked

```r
r2 <- edx %>% filter(title == "War of the Worlds (2005)") %>%
  filter(userId %in% userId[duplicated(userId)]) %>% select(-timestamp,-genres)
```

Ratings from the duplicated movie (see results section) are set as ratings from the same movie. The staying data is from movieId 34048 rates, because the genres are more descriptive.

```
edx[edx$movieId==64997,] <- edx[edx$movieId==64997,] %>% mutate(movieId=34048, genres="Action|Adventure
```

Ratings which are repeated twice in the same userId are found and one of them is eliminated. Rates between repeats were equal or very similar, so the error produced by eliminating one of them at random instead of calculating the mean is negligible.

```
edx <- edx %>%
   anti_join(edx[edx$title == "War of the Worlds (2005)",][duplicated(edx[edx$title == "War of the Worl
```

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

After correcting the repeated data, movieInfo is recalculated.

```
movieInfo <- edx %>% group_by(movieId) %>%
  summarise(title=title[1], genres=genres[1], number=length(rating), meanmovierate=mean(rating))
```

A preview of the mean rates distribution is made.

```
figure5 <- movieInfo %>% ggplot(aes(movieId, meanmovierate)) + geom_point() +
  xlab("Movie Id") + ylab("Movie mean rate")
```

The distribution of the number of rates per movie is plotted.

```
figure6 <- movieInfo %>% ggplot(aes(number)) + geom_histogram() + scale_x_log10() + scale_y_log10() +
  xlab("Number of rates per movie") + ylab("Number of movies")
```

Mean rates are compared to the number of rates per movie, which may be a reasonably good indicator of how popular are the movies.

```
figure7 <- movieInfo %>% ggplot(aes(number, meanmovierate)) + geom_point() +
  xlab("Number of rates per movie") + ylab("Movie mean rate")
```

The year is found in the title column, as a string. The years are extracted and their parenthesis are removed. Also, they are set as numeric and bound to the movieInfo data set.

```
year <- str_extract(movieInfo$title, "[(]\\d{4}[)]$")
year <- as.numeric(str_extract(year, "\\d{4}"))
movieInfo <- movieInfo %>% cbind(year)
```

A preview of the distribution of movies by year is plotted

```
figure8 <- movieInfo %>% ggplot(aes(year)) + geom_histogram(bins = length(unique(year))) +
  xlab("Year") + ylab("Number of movies per year")
```

Movies are grouped by year and the means of their mean rates are calculated. Then the result is plotted.

```
figure9 <- movieInfo %>% group_by(year) %>% summarise(meanrate = mean(meanmovierate)) %>%
    ggplot(aes(year, meanrate)) + geom_point() + xlab("Year") + ylab("Year mean rate")
```

Also the antiquity of the movie when the rate is done is taken in count. Timestamps are converted to dates and the year is saved. Then the "antiquity" is calculated as the year of the rate minus the year of the movie.

```
ant_edx <- edx %>% mutate(
           rateyear = as.numeric(format(as.POSIXct(edx$timestamp, origin="1970-01-01", tz="GMT"), "%Y")),
           movieyear = as.numeric(str_extract(str_extract(edx$title, "[(]\\d{4}[)]$"), "\\d{4}")),
           antiquity = rateyear - movieyear)
```

Data is grouped by antiquity and the mean rate of each group is calculated. Antiquities lower than 0 are considered as 0. Then the result is plotted.

```
figure10 <- ant_edx %>% mutate(antiquity=ifelse(antiquity>= 0,antiquity,0)) %>%
    group_by(antiquity) %>% summarise(meanrate=mean(rating)) %>%
    ggplot(aes(antiquity, meanrate)) + geom_point() + xlab("Antiquity") + ylab("Mean rate by antiquity")
```

Genres from all rates are listed.

```
genres <- unlist(str_extract_all(edx$genres,"[a-zA-Z- ]{1,}"))
```

The rates without genres are searched

```
r3 <- edx %>% filter(str_detect(genres,"no genres listed"))
```

Once the movie is found (it was only Pull My Daisy), all rates from this movie are searched.

```
r4 <- nrow(edx %>% filter(str_detect(title,"Pull My Daisy")))
```

Information about Pull My Daisy is searched in the internet. There isn't a general accord about the genres of this movie, but some sources tag it as a Comedy. According to this, the data set is filled.

```
edx <- edx %>% mutate(genres=ifelse(title=="Pull My Daisy (1958)", "Comedy", genres))
```

Now that all possible genres are known, data is grouped by genres. As many movies belong to more than one genre, they are in more than one group. The number of movies per group and the mean of their mean rates is calculated. All this information is saved in a data frame called "genres".

```
genresList <- c("Action", "Adventure", "Animation", "Children", "Comedy", "Crime", "Documentary", "Drama
              "Film-Noir", "Horror", "IMAX", "Musical", "Mystery", "Romance", "Sci-Fi", "Thriller", "War

genresInfo <- sapply(genresList, function(g){
  movieInfo %>% filter(str_detect(genres, g)) %>% summarise(meanrate=sum(meanmovierate*number)/sum(numbe
})

# Transform the "genresInfo" matrix into a data frame
genresInfo <- data.frame(genres=genresList,meanrate=unlist(genresInfo[1,]), number=unlist(genresInfo[2,]
```

Distribution of genres along data set movies is displayed. Data is sorted according to group sizes.

```
figure11 <- genresInfo %>% mutate(genres=fct_reorder(genres, number)) %>% ggplot(aes(genres,number)) + g
    coord_flip() + xlab("Genres") + ylab("Number of movies per genre")
```

Groups mean rates are also sorted and plotted.

```
figure12 <- genresInfo %>% mutate(genres=fct_reorder(genres, meanrate)) %>% ggplot(aes(genres,meanrate))
    coord_flip() + scale_x_discrete(position = "top") + xlab("Genres") + ylab("Genre mean rate")
```

In order to perform further analysis in genres data, the next step would be binarizing this data (Code not run).

```
new_edx <- edx %>%
    mutate(Action=ifelse(str_detect(genres,"Action"),1,0),
           Adventure=ifelse(str_detect(genres,"Adventure"),1,0),
           Animation=ifelse(str_detect(genres,"Animation"),1,0),
           Children=ifelse(str_detect(genres,"Children"),1,0),
           Comedy=ifelse(str_detect(genres,"Comedy"),1,0),
           Crime=ifelse(str_detect(genres,"Crime"),1,0),
           Documentary=ifelse(str_detect(genres,"Documentary"),1,0),
           Drama=ifelse(str_detect(genres,"Drama"),1,0),
           Fantasy=ifelse(str_detect(genres,"Fantasy"),1,0),
           FilmNoir=ifelse(str_detect(genres,"Film-Noir"),1,0),
```

```
            Horror=ifelse(str_detect(genres,"Horror"),1,0),
            IMAX=ifelse(str_detect(genres,"IMAX"),1,0),
            Musical=ifelse(str_detect(genres,"Musical"),1,0),
            Mystery=ifelse(str_detect(genres,"Mystery"),1,0),
            Romance=ifelse(str_detect(genres,"Romance"),1,0),
            SciFi=ifelse(str_detect(genres,"Sci-Fi"),1,0),
            Thriller=ifelse(str_detect(genres,"Thriller"),1,0),
            War=ifelse(str_detect(genres,"War"),1,0),
            Western=ifelse(str_detect(genres,"Western"),1,0))
```

User Id

First of all movieInfo is joined to the edx data. This way, all rows contain the mean rate of the movie and the actual rate of each user. Then data is grouped by userId. Number of rates and mean rates of each user are saved. The difference between rates and expected rates is calculated and the mean of these differences per user is saved. Standard deviations of both mean rates and mean differences are saved in order to make further comparisons. Finally, repeated movieIds per groups are searched.

```
userInfo <- edx %>% left_join(movieInfo, by="movieId") %>% group_by(userId) %>%
    summarise(number = length(rating),
              meanuser = mean(rating),
              sd1 = sd(rating),
              meandiffuser = mean(rating-meanmovierate),
              sd2 = sd(rating-meanmovierate),
              repeats=sum(duplicated(movieId)))
```

Distribution of number of users with different number of rates per user is displayed in a logarithmic scale.

```
figure13 <- userInfo %>% ggplot(aes(number)) + geom_histogram(binwidth = 0.09) +
    scale_x_log10() + scale_y_log10() + xlab("Number of users") + ylab("Number of rates per user")
```

Distribution of mean rates along users is plotted.

```
figure14 <- userInfo[sample(nrow(userInfo), 1000),] %>% ggplot(aes(userId, meanuser)) + geom_point() +
  xlab("User Id") + ylab("User mean rate")
```

Distribution of mean differences along users is also plotted.

```
figure15 <- userInfo[sample(nrow(userInfo), 1000),] %>% ggplot(aes(userId, meandiffuser)) + geom_point()
  xlab("User Id") + ylab("User mean difference")
```

First approach

The validation set is also corrected. In this case, duplicated rates of the repeated movie from the same user would not me removed. However, there would be no problem. Both rates would be expected to be the same and so it would be. Anyway, there are no duplicated rates (see results section).

```
validation[validation$movieId==64997,] <- validation[validation$movieId==64997,] %>%
  mutate(movieId=34048, genres="Action|Adventure|Sci-Fi|Thriller")
validation <- validation %>% mutate(genres=ifelse(title=="Pull My Daisy (1958)", "Comedy", genres))
```

Mean rate of each movie and mean difference in each user's ratings is joined to the validation set

```
validation <- validation %>%
  left_join(select(movieInfo, movieId, meanmovierate), by="movieId") %>%
  left_join(select(userInfo, userId, meandiffuser), by="userId")
```

Ratings are predicted as the mean rate of the movie + the mean difference at rating from the user who is rating

```
y_hat <- validation$meanmovierate + validation$meandiffuser
```

Rates lower than the minimum (0.5) or higher than the maximum (5) are corrected

```
y_hat[y_hat < 0.5] <- 0.5
y_hat[y_hat > 5] <- 5
```

In addition, predicted rates are rounded in order to determine the accuracy. They are rounded to whole number when the timestamp is lower than the h_time and they are rounded to the closest ".5" when the timestamp is higher.

```
accurate_y <- ifelse(validation$timestamp < h_time,
                  round(validation$meanmovierate + validation$meandiffuser),
                  round((validation$meanmovierate + validation$meandiffuser)*2)/2)
```

Results

Dataset overview

Data sets contain 6 columns originally: moiveId, userId, timestamp, rating, title and genres. By default, the train set (edx) is 9000055 rows long and the test set (validation) is 999999 rows long. Althought after the corrections, the edx data set results 9000038 rows long and, after the modifications, the validation data set has 2 more columns: meanmovierate and meandiffuser.

```
str(edx)
```

```
## 'data.frame':    9000038 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
```

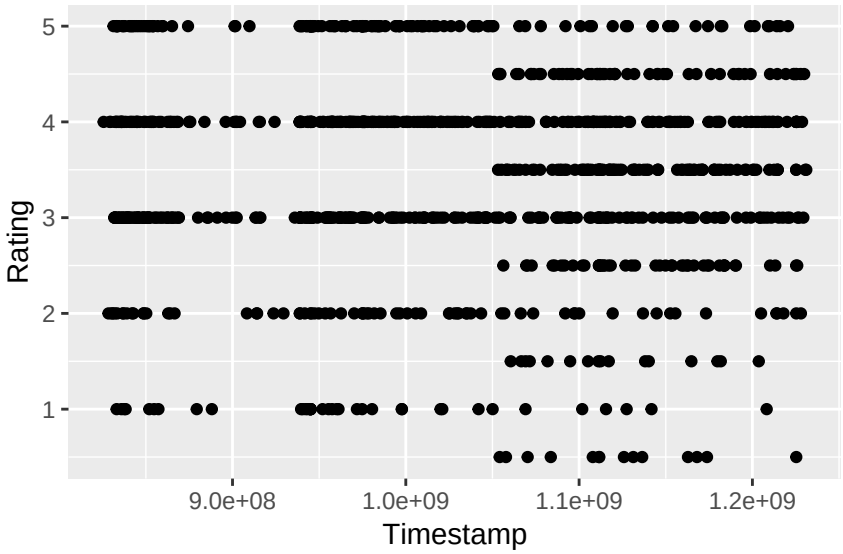
```
str(validation)
```

```
## 'data.frame':    999999 obs. of  8 variables:
##  $ userId       : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId      : num  231 480 586 151 858 ...
##  $ rating       : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp    : int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 113357
##  $ title        : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (19
##  $ genres       : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Ro
##  $ meanmovierate: num  2.94 3.66 3.06 3.53 4.42 ...
##  $ meandiffuser : num  1.679 1.679 1.679 -0.236 -0.236 ...
```

Ratings and timestamp

In figure 1 some insights are revealed. First, high scores seem to be more frequent than low scores. Second, the ".5" ratings are not present along all the timestamp. They appear for the first time in February 12, 2003. Third, ratings go from 0.5 to 5. There are no 0s.

```
figure1
```




```
as.POSIXct(h_time, origin="1970-01-01", tz="GMT")
```

```
## [1] "2003-02-12 17:31:34 GMT"
```

There are many repeated timestamps in rates from the same userId (data not shown). If we take in count that timestamps have accuracy of seconds, this must mean that multiples rates can be uploaded to the MovieLens web page at the same time.

Rates start in 1995, but actually there are only 2 rates in this year. These rates have the same timestamp and userId. They are quite away from the rest of rates, so they may be a trial from the MoiveLens admins. The real opening of the MovieLens web page may be around the timestamp 822873600, corresponding to the January 29, 1996.

```
head(sort(unique(edx$timestamp)),10)
```

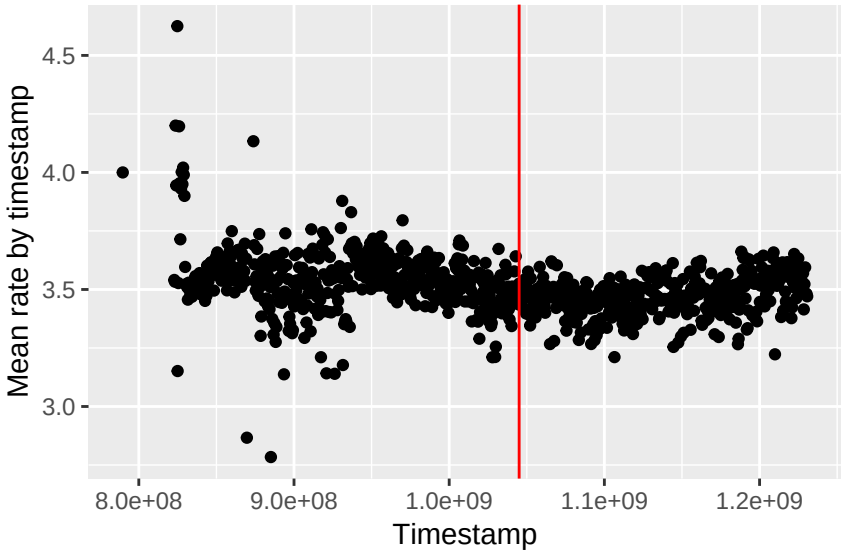
```
##  [1] 789652009 822873600 823185196 823185197 823185198 823185200 823185203
##  [8] 823185204 823185205 823185206
```

```
as.POSIXct(822873600, origin="1970-01-01", tz="GMT")
```

```
## [1] "1996-01-29 GMT"
```

Overall mean rating is stable along time, even when the .5 ratings appear. This mean is around 3.5.

```
figure2
```

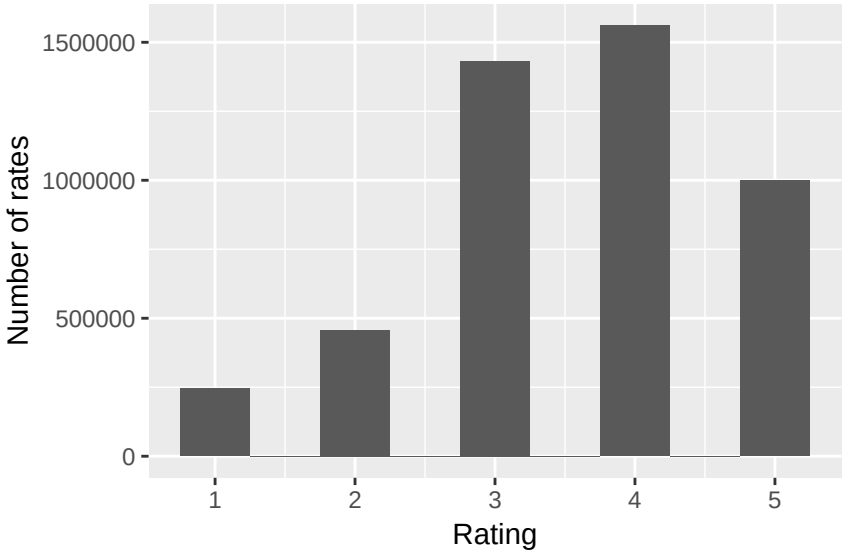



```
mean(edx$rating)
```

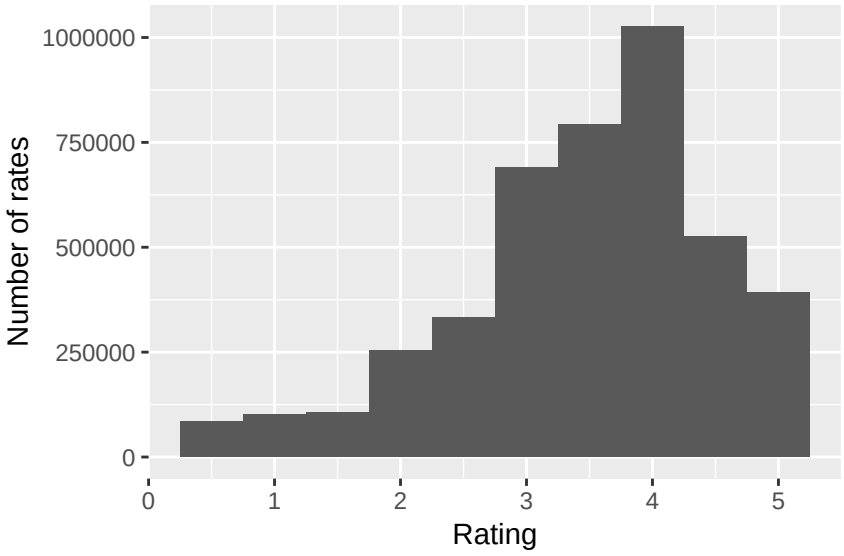
```
## [1] 3.512467
```

Rating distributions are equivalent with and without .5 rates, with 4 as the mode in both cases.

```
figure3
```




```
figure4
```



Movies data

One repeated movie is found: War of the Worlds

```
r1
```

```
## # A tibble: 2 x 5
##   movieId title               genres                  number meanmovierate
##     <dbl> <chr>               <chr>                    <int>         <dbl>
## 1   34048 War of the Worlds ~ Action|Adventure|Sci-F~   2460          3.18
## 2   64997 War of the Worlds ~ Action                     28          2.91
```

Some users voted for both repeated movies and in some cases with a different rate.

```
r2
```

```
##    userId movieId rating                    title
## 1    1018   34048    3.0 War of the Worlds (2005)
## 2    1018   64997    3.0 War of the Worlds (2005)
## 3   12366   34048    2.5 War of the Worlds (2005)
## 4   12366   64997    3.0 War of the Worlds (2005)
## 5   29892   34048    4.5 War of the Worlds (2005)
## 6   29892   64997    4.0 War of the Worlds (2005)
## 7   30840   34048    1.5 War of the Worlds (2005)
## 8   30840   64997    1.5 War of the Worlds (2005)
## 9   40570   34048    0.5 War of the Worlds (2005)
## 10  40570   64997    1.0 War of the Worlds (2005)
## 11  45430   34048    2.0 War of the Worlds (2005)
## 12  45430   64997    2.5 War of the Worlds (2005)
## 13  49138   34048    3.0 War of the Worlds (2005)
## 14  49138   64997    2.5 War of the Worlds (2005)
## 15  52270   34048    3.0 War of the Worlds (2005)
## 16  52270   64997    3.0 War of the Worlds (2005)
## 17  52510   34048    3.0 War of the Worlds (2005)
## 18  52510   64997    3.0 War of the Worlds (2005)
## 19  53838   34048    2.5 War of the Worlds (2005)
```

```
## 20  53838   64997    2.5 War of the Worlds (2005)
## 21  59269   34048    3.0 War of the Worlds (2005)
## 22  59269   64997    2.5 War of the Worlds (2005)
## 23  60954   34048    3.5 War of the Worlds (2005)
## 24  60954   64997    3.5 War of the Worlds (2005)
## 25  61345   34048    3.5 War of the Worlds (2005)
## 26  61345   64997    3.5 War of the Worlds (2005)
## 27  61664   34048    2.5 War of the Worlds (2005)
## 28  61664   64997    3.0 War of the Worlds (2005)
## 29  62362   34048    2.0 War of the Worlds (2005)
## 30  62362   64997    2.5 War of the Worlds (2005)
## 31  64642   34048    4.0 War of the Worlds (2005)
## 32  64642   64997    3.0 War of the Worlds (2005)
## 33  67385   34048    4.0 War of the Worlds (2005)
## 34  67385   64997    4.0 War of the Worlds (2005)
```

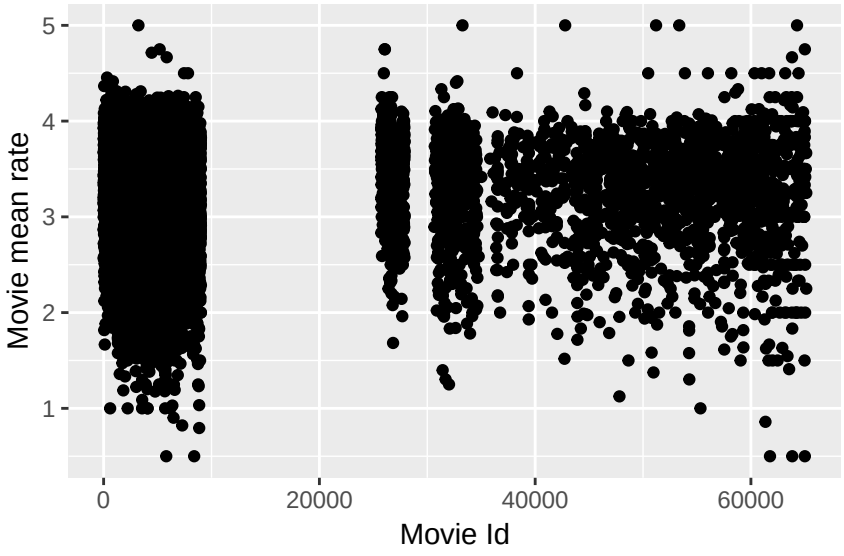
After correcting repeated data, there are 10676 different movies in the dataset

```
nrow(movieInfo)
```

```
## [1] 10676
```

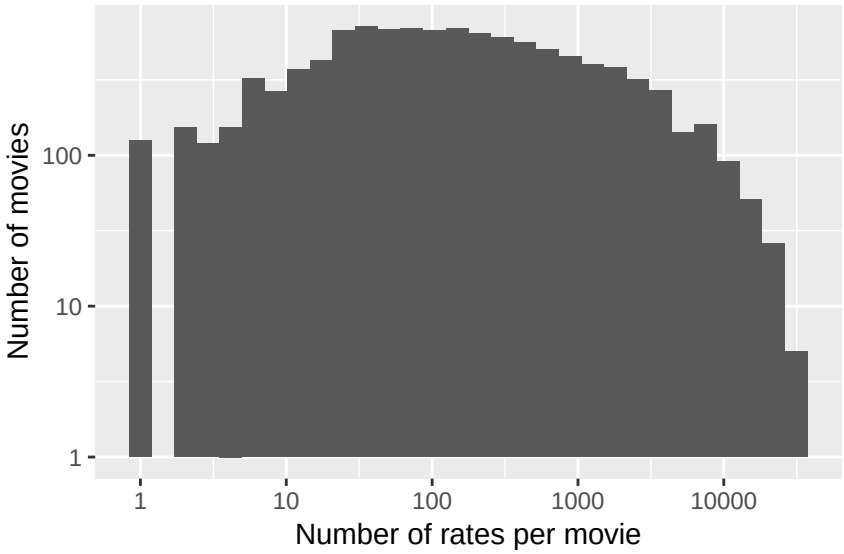
Mean rates per movie seems to be quite distributed. For this reason, they may be a good predictor.

```
figure5
```



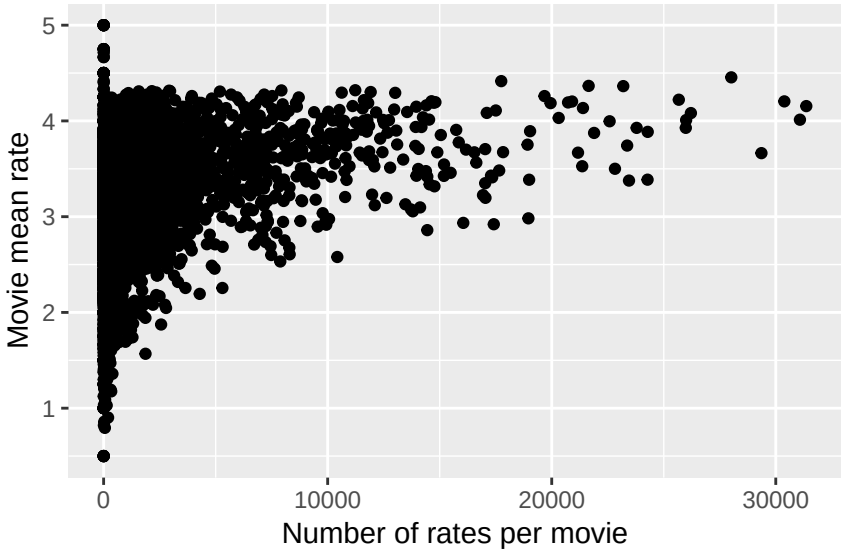
One of the weaknesses of this parameter could be the fact that there are movies with a low number of rates or even just 1. However they are not a majority.

```
figure6
```



The popularity of movies does not seem to have a clear effect in the mean rating, since the correlation between the mean rates and the number of rates per movie has a correlation of 0.2.

```
figure7
```

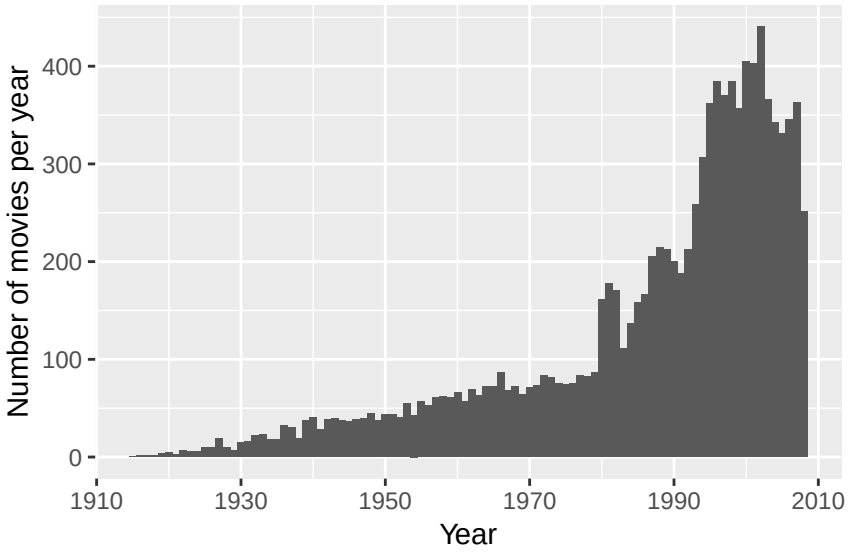



```
cor(movieInfo$number,movieInfo$meanmovierate)
```

```
## [1] 0.2114028
```

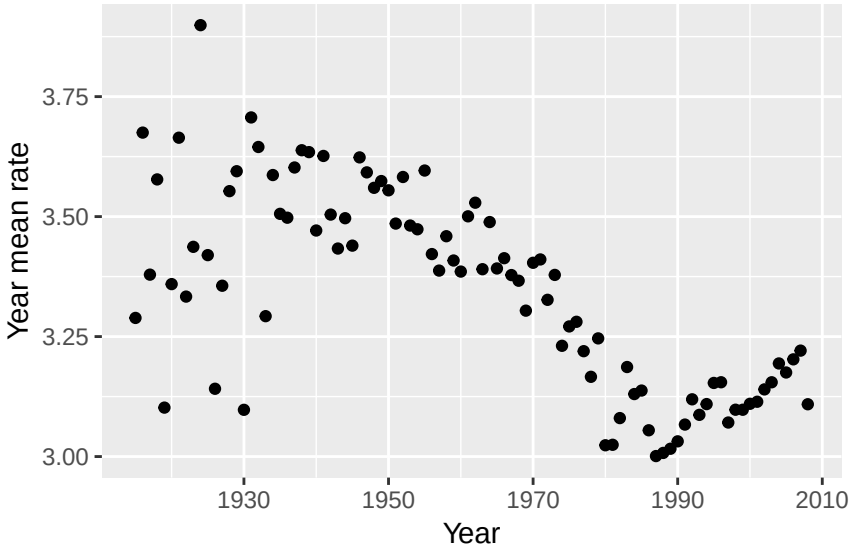
Movies are distributed by their release year between 1915 and 2008.

```
figure8
```



The means of mean rates of movies from around 1935 onwards follow a pattern. Means decrease from 1935 to the 80s and early 90s, where the worst means are found. Then means slightly increase.

```
figure9
```



Antiquity of movies in the rate time is calculated. There are a few rates which were done 1 or even 2 years before the release of the movie. This can be explained if rates are done after a preview. Anyway, 2 years before the release seems unusual, so these rates are observed. They are only 3 and all of them are from the same movie: Talk of Angels. This movie was released in 1998 but it was directed in 1996 (https://en.wikipedia.org/wiki/Talk_of_Angels), so this antiquity is possible.

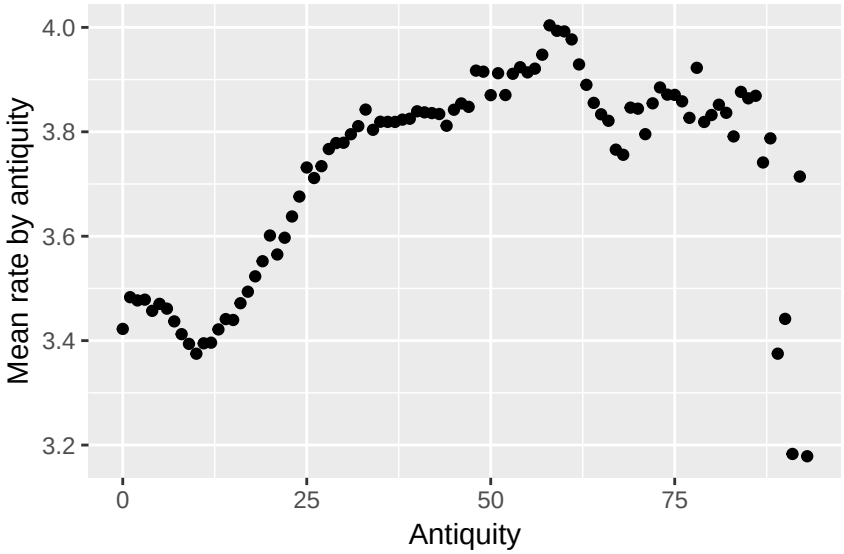
```
ant_edx %>% filter(antiquity == -2)
```

```
##   userId movieId rating timestamp                title genres rateyear
## 1   8010     887      3 844937869 Talk of Angels (1998)  Drama     1996
## 2  53545     887      1 842961411 Talk of Angels (1998)  Drama     1996
## 3  65117     887      2 843515388 Talk of Angels (1998)  Drama     1996
##   movieyear antiquity
```

```
## 1      1998        -2
## 2      1998        -2
## 3      1998        -2
```

When means are plotted by antiquity, it is possible to see the opposite pattern to figure 9, as expected. However this pattern is more stable. An interpretation of this pattern is that rates are better for old movies but not always for very old movies (which perfectly match my personal opinion).

```
figure10
```



There are 19 possible genres; Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, IMAX, Musical, Mystery, Romance, Sci-Fi, Thriller, War and Western. Additionally, there are 7 rates with no genres.

```
table(genres)
```

```
## genres
##           Action        Adventure        Animation         Children
##          2560528          1908903           467168           737994
##           Comedy            Crime      Documentary            Drama
##          3540930          1327715            93066          3910127
##          Fantasy        Film-Noir           Horror             IMAX
##           925637           118541           691485             8181
##          Musical          Mystery no genres listed          Romance
##           433080           568332                7          1712100
##           Sci-Fi         Thriller              War          Western
##          1341194          2325910           511147           189394
```

When the 7 rates without genres are displayed it is observed that all of them are from the same movie: Pull My Daisy. It is also verified that there are no more than 7 rates from this movie. The information is missing from the whole data set.

```
r3
```

```
##   userId movieId rating  timestamp                title             genres
## 1   7701    8606    5.0 1190806786 Pull My Daisy (1958) (no genres listed)
## 2  10680    8606    4.5 1171170472 Pull My Daisy (1958) (no genres listed)
## 3  29097    8606    2.0 1089648625 Pull My Daisy (1958) (no genres listed)
## 4  46142    8606    3.5 1226518191 Pull My Daisy (1958) (no genres listed)
```

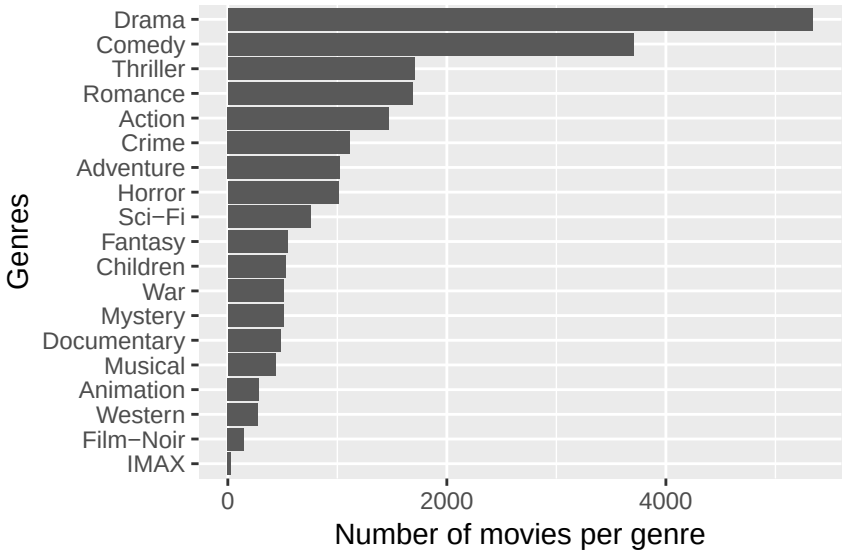
```
## 5  57696    8606    4.5 1230588636 Pull My Daisy (1958) (no genres listed)
## 6  64411    8606    3.5 1096732843 Pull My Daisy (1958) (no genres listed)
## 7  67385    8606    2.5 1188277325 Pull My Daisy (1958) (no genres listed)
```

```
r4
```

```
## [1] 7
```

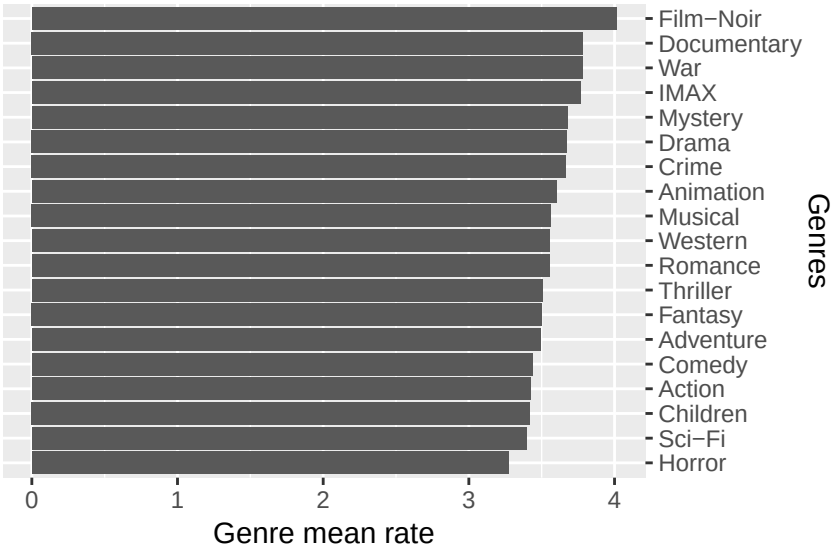
In figure 11 it is displayed the distribution of the data set movies along the different genres. Drama and Comedy are the most common genres. They cover around the half and a third of the movies respectively.

```
figure11
```



Also de mean rates per genres are displayed. They are similar.

```
figure12
```



Binarizing the genres data allowed to perform chi-squared tests and random forest models. They were some of the different approaches tried to improve accuracy in the old version of this exercise. However, they returned unsuccessfully results (data and code not shown). Chi-squared tests were performed by genre and user. In

all genres around a 10%-5% of users had a p-value lower than 0.05, but also many warnings about the low confidence of this results were returned. Random forest over the whole data set was impossible to be carried. Random forest by user was a giant simulation which resulted in overfitting: More than 50% accuracy in the training set and less than a 30% in the test set.

User Id

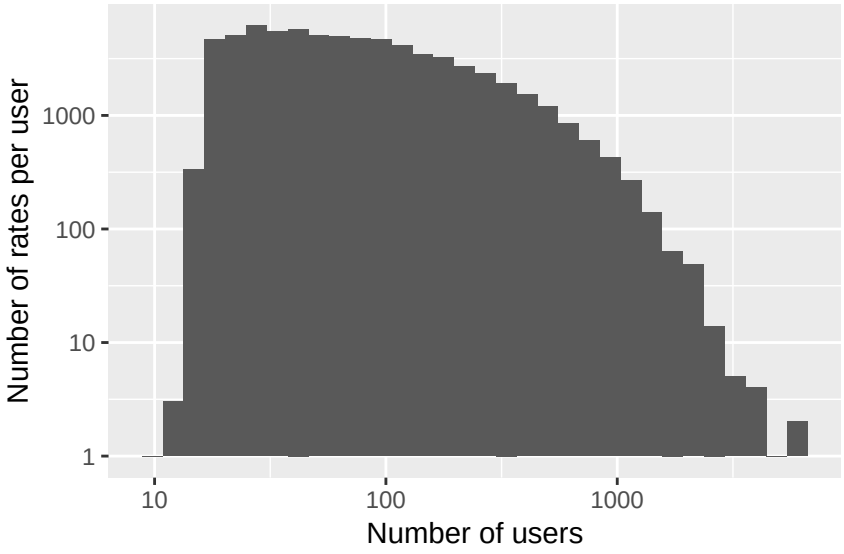
There are no repeated rates for a movieId from the same user.

```
unique(userInfo$repeats)
```

```
## [1] 0
```

The majority of users (95%) have 20 or more rates and there are no users with less than 10 rates. Since many users have a high number of rates, data grouping by userId may be a very powerful way to build models. Each user could have its own model.

```
figure13
```




```
quantile(userInfo$number,probs=c(.05))
```

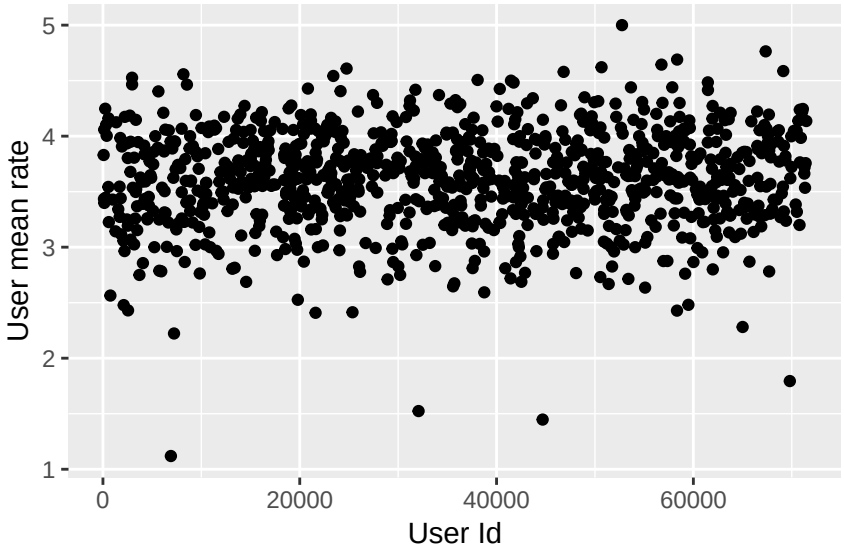
```
## 5%
## 20
```

```
min(userInfo$number)
```

```
## [1] 10
```

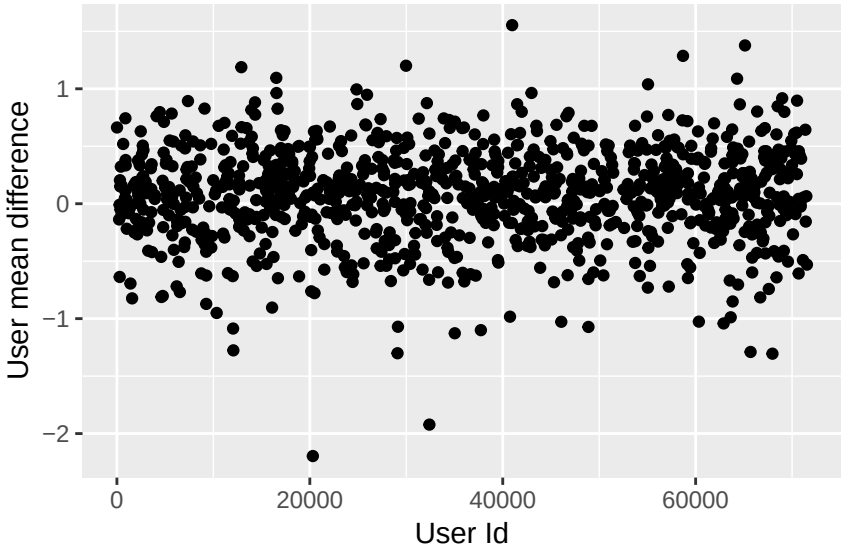
Mean rates per user may be a reasonably good parameter to predict user rates.

```
figure14
```



However, the mean of difference in actual rates about expected rates (the mean rate of each movie) also may be a good parameter.

```
figure15
```



When both parameters, which are means, are compared by the standard deviation of the data they come from, the second parameter shows a lower mean standard deviation. Actually it is around a 10% lower. According to this result, the mean of differences is a more trustable parameter.

```
mean(userInfo$sd1)
```

```
## [1] 0.9598492
```

```
mean(userInfo$sd2)
```

```
## [1] 0.8649452
```

```
mean(userInfo$sd2)/mean(userInfo$sd1)
```

```
## [1] 0.9011261
```

First approach

In the validation set there are about 300 rates from War of the Worlds, the repeated movie, but never two rates from the same user.

```
nrow(validation %>% filter(title == "War of the Worlds (2005)"))
```

```
## [1] 301
```

```
nrow(validation %>% filter(title == "War of the Worlds (2005)") %>% filter(userId %in% userId[duplicate
```

```
## [1] 0
```

There are no rates from Pull My Daisy, the movie without genres.

```
nrow(validation %>% filter(title=="Pull My Daisy (1958)"))
```

```
## [1] 0
```

After performing the simulation, the result is a RMSE of 0.86516. This RMSE is better than 0.87750. There is no need to improve the algorithm.

```
sqrt(mean((y_hat-validation$rating)^2))
```

```
## [1] 0.8651614
```

This rates, when rounded, result in a 35,6 % of accuracy.

```
mean(accurate_y == validation$rating)
```

```
## [1] 0.3558364
```

Conclusions

The mean rate of each movie and the mean rating difference of each user are parameters obtained from the train set. A prediction model can be built with these predictors and this model is good enough to reach the objective of this project (RMSE < 0.8775).

In order to improve this result, other parameters such as genres or antiquity may be useful. The development of individual models for each user may be an interesting strategy, since there are 10 or more rates per user and, in most cases, more than 20.

MovieLens data set has minor flaws: One movie is repeated and one movie has missing genres. These problems should be solved before further modeling.