MovieLens Rate Prediction

Alan Bauza

6/16/2021

1.MOVIELENS INTRODUCTION

MovieLens is an online system that recommends movies to its users based on collaborative filter taking into consideration their preferences. GroupLens Research, a research lab that belongs to the University of Minnesota (United States) provides a free access to MovieLens' data set, that contains 10 million ratings, which will be used for this project to predict the movie ratings based on user preference. Before that a thorough data analysis will be performed on the data. Due to computational limitations and the impossibility to work with the whole data set, I will reduce the data set to be able to perform an analysis, from 9,000,055 items to 90,000 for the training set (renamed "edx_new", before "edx") and from 999,999 items to 10,000 for the validation set (renamed "validation_new", before "validation"). The end goal of the project is to achieve a system that predicts the rating with a lower value than 0.86490. The key steps taken on this project were creating the proper data set, creating new columns with the year and month of rating and year of release, analyzing the different columns and some of their interactions, calculating the RMSE and arriving to a conclusion.

2.MOVIELENS ANALYSIS

Installing and/or loading the necessary libraries

```
if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table",
                                          repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(lubridate)) install.packages("lubridate",
                                         repos = "http://cran.us.r-project.org")
```

```
## Loading required package: lubridate

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(tidyverse)

library(caret)

library(data.table)

library(lubridate)
```

Downloading the necessary data to a temporary file

```
dl <- tempfile()

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

Building the data set

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

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

colnames(movies) <- c("movieId", "title", "genres")

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

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

Partitioning the initial data with 90% of the info used for the training set, creating the training data set and creating a temporary data set that later will be used to create the validation set

```
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)

edx <- movielens[-test_index,]

temp <- movielens[test_index,]
```

Creating the validation set and making sure userId and movieId in it are also in the edx set

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

Adding rows removed from validation set back into edx set and removing all unnecessary objects

```
removed <- anti_join(temp, validation)
```

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

```
edx <- rbind(edx, removed)

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

Extracting fewer values due to computational limitations as explained in the introduction so my computer can process the data and creating the data sets that I will be working with

```
edx_new <- sample_n(edx, 90000)

validation_new <- sample_n(validation, 10000)
```

Checking how the data structure looks like

```
head(edx_new)
```

```
##    userId movieId rating  timestamp                            title
## 1:  50805    1095    4.0  952366935       Glengarry Glen Ross (1992)
## 2:  42011    6303    3.5 1121891132     Andromeda Strain, The (1971)
## 3:  45985     593    4.0  854490846 Silence of the Lambs, The (1991)
## 4:  19660    4210    4.0 1207988321                 Manhunter (1986)
## 5:  46073     186    4.0 1123689113               Nine Months (1995)
## 6:  13268     485    2.0  923662409          Last Action Hero (1993)
##                                 genres
## 1:                               Drama
## 2:                         Mystery|Sci-Fi
## 3:                Crime|Horror|Thriller
## 4: Action|Crime|Drama|Horror|Thriller
## 5:                         Comedy|Romance
## 6:    Action|Adventure|Comedy|Fantasy
```

We can see there are six columns: userId, that contains the Id that is used to identify each specific person that has rated a movie; movieId, that has the Id that represents each specific movie; timestamp, a numerical value of the specific moment in which the movie was rated (It will need to be transformed to a datetime item); title, that has both the name of the movie and the year it was released. Extracting each year would be of value for future analysis;and genres: movies are not necessarily classified into one genre but many, so each row value contains different amounts and types of genres.

Before analysis, I will be transforming different columns for a better examination.

Mutating timestamp into two new columns, one for the year of rating and another for the month of rating to establish comparisons later; and dropping the timestamp column as it is not needed anymore.

```
edx_new <- mutate(edx_new, rated_year = year(as_datetime(timestamp)))

edx_new <- mutate(edx_new, rated_month = month(as_datetime(timestamp)))

edx_new <- subset(edx_new, select = -timestamp)
```

Separating the release date from the title and placing that information in a new column named release

```
edx_new <- mutate(edx_new, release = as.numeric(substr(title, nchar(title)-4,
                                                       nchar(title)-1)))
```

Creating a new column named age_rated_movie that will show how many years passed since the movie was released and the year it was rated, to later analyze it

```
edx_new <- mutate(edx_new, age_rated_movie = rated_year - release)
```

Checking how everything looks now

```
head(edx_new)
```

```
##    userId movieId rating                             title
## 1:  50805    1095    4.0       Glengarry Glen Ross (1992)
## 2:  42011    6303    3.5     Andromeda Strain, The (1971)
## 3:  45985     593    4.0 Silence of the Lambs, The (1991)
## 4:  19660    4210    4.0                 Manhunter (1986)
## 5:  46073     186    4.0               Nine Months (1995)
## 6:  13268     485    2.0          Last Action Hero (1993)
##                                 genres rated_year rated_month release
## 1:                               Drama       2000           3    1992
## 2:                       Mystery|Sci-Fi       2005           7    1971
## 3:                Crime|Horror|Thriller       1997           1    1991
## 4: Action|Crime|Drama|Horror|Thriller       2008           4    1986
## 5:                      Comedy|Romance       2005           8    1995
## 6:    Action|Adventure|Comedy|Fantasy       1999           4    1993
##    age_rated_movie
## 1:               8
## 2:              34
## 3:               6
## 4:              22
## 5:              10
## 6:               6
```

Checking amount of users in the edx_new dataset

```
length(unique(edx_new$userId))
```

```
## [1] 36461
```

Checking amount of movies in the edx_new dataset

```
length(unique(edx_new$movieId))
```

```
## [1] 6633
```

Checking all the different rating values in an ordered manner and the amount in the edx_new dataset

```
sort(unique(edx_new$rating), decreasing = FALSE)
```

```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
length(unique(edx_new$rating))
```

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

We can see 10 ratings that go from 0.5 to 5.0, with 0.5 intervals. There is no 0.0 rating

Checking the amount of genres in the edx_new dataset

```
length(unique(edx_new$genres))
```

```
## [1] 668
```

We know after previously inspecting the genres column that each move belongs to more than one genre, so the real value is not actually 668.

Separating the genres in the genres column. Since this will alter the amount of rows and will create more than one review for each real review, I will be creating a new data set named edx_genres with this information not to affect the edx_new data set that will be used for training the model

```
edx_genres <- edx_new %>% as.data.frame() %>% separate_rows(genres, sep ="\\|")
```

Checking the amount of genres in the edx_genres dataset

```
length(unique(edx_genres$genres))
```

```
## [1] 19
```

Instead of 668 different genres like it was indicated by the genres column in the edx_new data set with mixed genres per row, we see that actually there are 19 genres.

Calculating the range of years of reviews

```
range(edx_new$rated_year)
```

```
## [1] 1996 2009
```

This data set only contains movies reviewed between 1996 and 2009.

Calculating the range of years of release

```
range(edx_new$release)
```

```
## [1] 1915 2008
```

The movies reviewed were released between 1915 and 2008.

Calculating the range of age of rates, the moment between they were released and the moment they were rated

```
range(edx_new$age_rated_movie)
```

```
## [1] -1 90
```

We can see there are -1 ratings which is impossible since it would mean that the movie was rated before it was released.

Checking which rows are giving -1, ordering the data set by the age_rated_movie in ascending rate

```
edx_new[order(edx_new$age_rated_movie),]
```

```
##        userId movieId rating
##     1:   8010     987    3.0
##     2:  49073     981    3.0
##     3:  65117     987    1.0
##     4:  19985    2700    3.0
##     5:  58469      86    3.0
##    ---
## 89996:  13107    3310    4.0
## 89997:  11138    3310    4.5
## 89998:  51106    6987    4.0
## 89999:  67385    3309    2.5
## 90000:  32587    7065    3.5
##                                                                        title
##     1:                                                          Bliss (1997)
##     2:                                                 Dangerous Ground (1997)
##     3:                                                          Bliss (1997)
##     4:                          South Park: Bigger, Longer and Uncut (1999)
##     5:                                                    White Squall (1996)
##    ---
## 89996:                                                       Kid, The (1921)
## 89997:                                                       Kid, The (1921)
## 89998: Cabinet of Dr. Caligari, The (Das Cabinet des Dr. Caligari.) (1920)
## 89999:                                                    Dog's Life, A (1918)
## 90000:                                          Birth of a Nation, The (1915)
##                                    genres rated_year rated_month release
##     1:                      Drama|Romance       1996          10    1997
##     2:                              Drama       1996           9    1997
##     3:                      Drama|Romance       1996           9    1997
##     4:           Animation|Comedy|Musical       1999          11    1999
##     5:                     Adventure|Drama       1996          11    1996
##    ---
## 89996:                       Comedy|Drama       2007          10    1921
## 89997:                       Comedy|Drama       2008          10    1921
## 89998: Crime|Drama|Fantasy|Film-Noir|Horror       2008          11    1920
## 89999:                             Comedy       2007           7    1918
## 90000:                          Drama|War       2005           6    1915
##        age_rated_movie
##     1:              -1
##     2:              -1
##     3:              -1
##     4:               0
##     5:               0
##    ---
## 89996:              86
## 89997:              87
## 89998:              88
## 89999:              89
## 90000:              90
```

We can see it's only the first 3 elements that give a value of -1. It happened with 2 movies, Bliss and Dangerous Ground, that were both released in 1997 and rated in 1996 according to the data set. Further

investigation on the movies indicate that they were indeed released in 1997, so the timestamp must be wrong.

Replacing the -1 age_rated_movie value by 0 and the rated_year one by 1997, which makes more sense

```
edx_new$age_rated_movie[edx_new$age_rated_movie < 0] <- 0

edx_new$rated_year[21220] <- 1997

edx_new$rated_year[44930] <- 1997

edx_new$rated_year[84171] <- 1997
```

Checking that the changes did take place. The range should be now 0 to 90, and the values of the differences between the rated_year and release for those rows should be 0

```
range(edx_new$age_rated_movie)
```

```
## [1]  0 90
```

```
edx_new$rated_year[21220] - edx_new$release[21220]
```

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

```
edx_new$rated_year[44930] - edx_new$release[44930]
```

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

```
edx_new$rated_year[84171] - edx_new$release[84171]
```

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

All the proper changes took place.

Checking the average of the age of the rated movie when rated

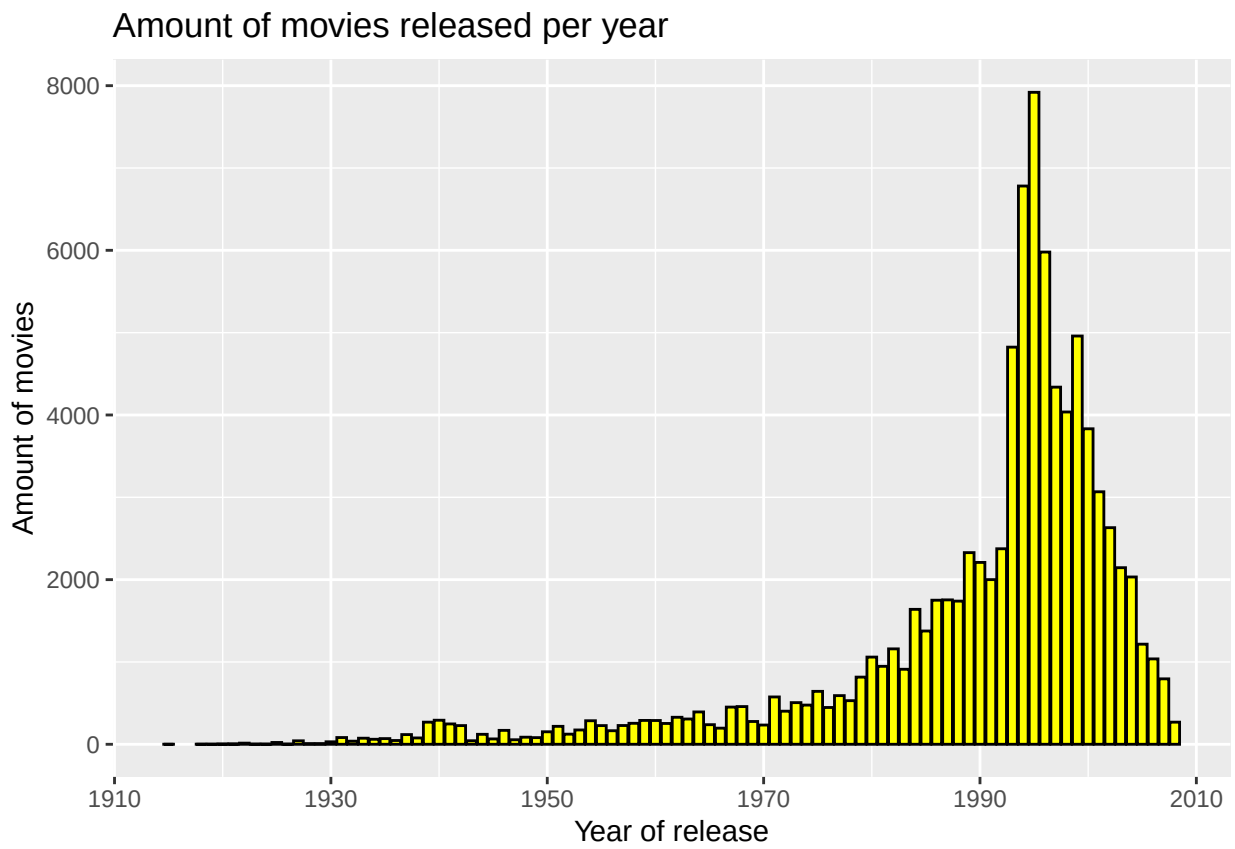
```
mean(edx_new$age_rated_movie)
```

```
## [1] 11.94684
```

In average movies were rated after 12 years of their release. Since the rating started in 1996 and there are movies as old as from 1915, this shows many movies had to be reviewed more recently. Let's check if there is a large amount of movies close to the 1996-2009 period (period of review).

Creating a histogram of the amount of movies released per year

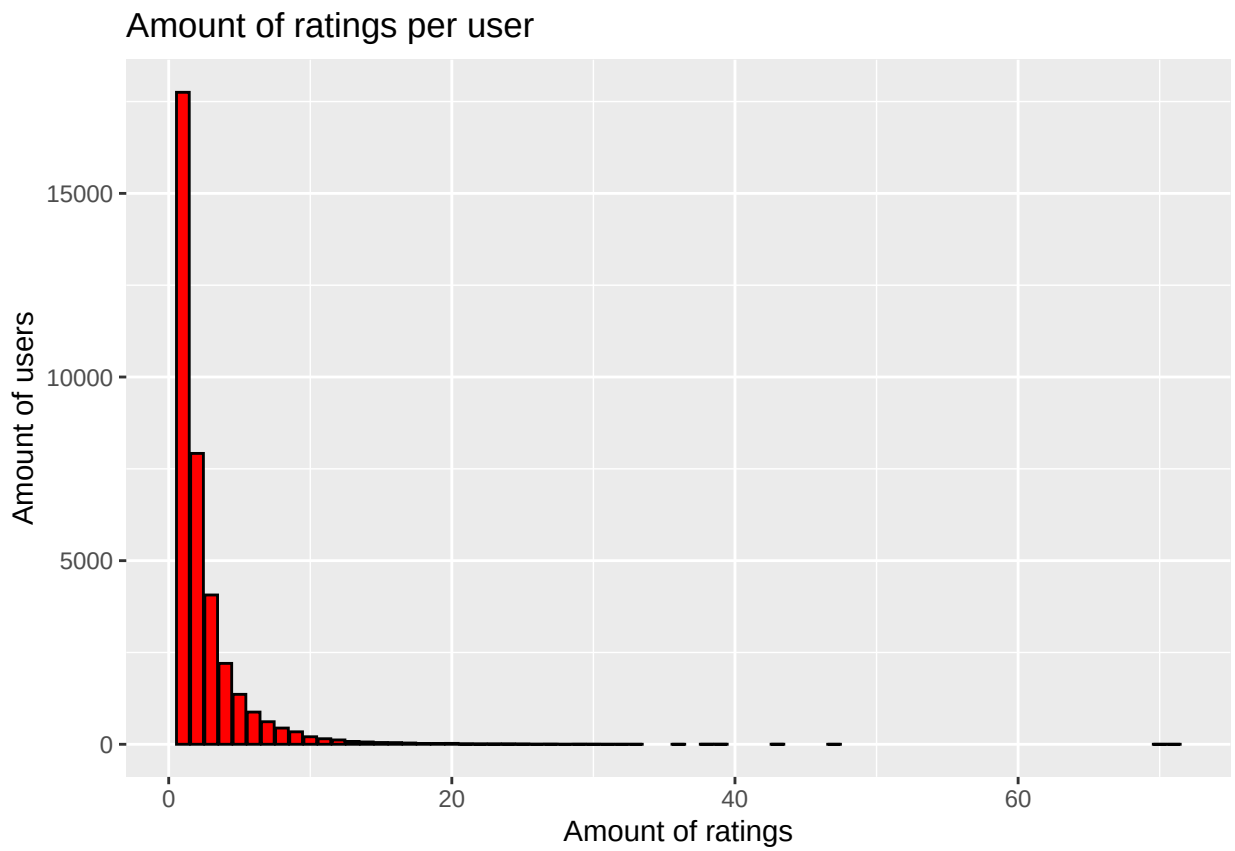
```
edx_new %>% group_by(release) %>% ggplot(aes(release)) +
  geom_bar(fill= "yellow", colour = "black") +
  labs(x = "Year of release", y = "Amount of movies",
       title = "Amount of movies released per year")
```



Like it was supposed previously, from the movies that have been reviewed there is a growing amount of movies released throughout the years with an acceleration in the 80's and a peak in the mid 90's.

Creating a histogram with the amount of ratings per user

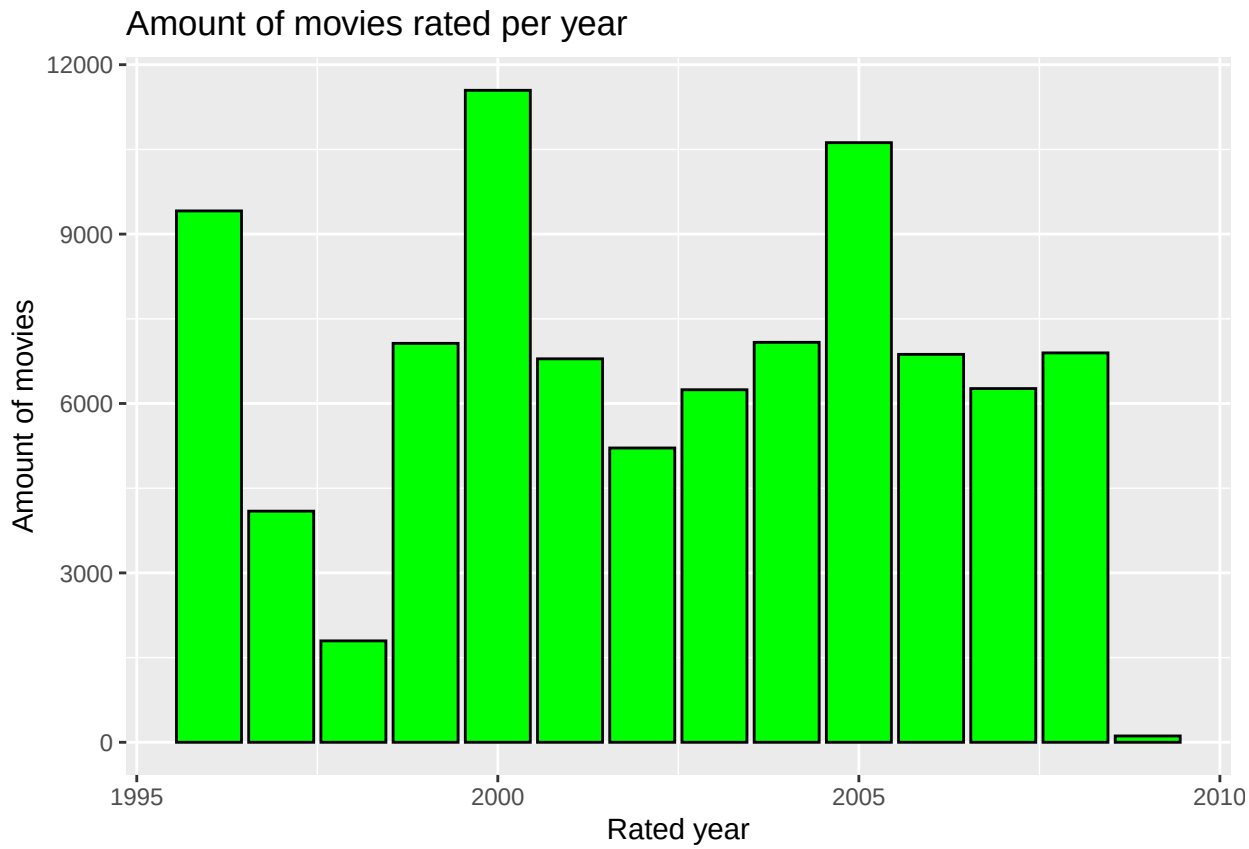
```
edx_new %>% group_by(userId) %>% dplyr::summarize(n = n()) %>% ggplot(aes(n)) +
  geom_bar(fill= "red", colour = "black") +
  labs(x ="Amount of ratings", y = "Amount of users",
       title = "Amount of ratings per user")
```



Most people have done few reviews, and the amount of more than 15 ratings per users are negligible.

Creating a histogram with the amount of movies rated per year

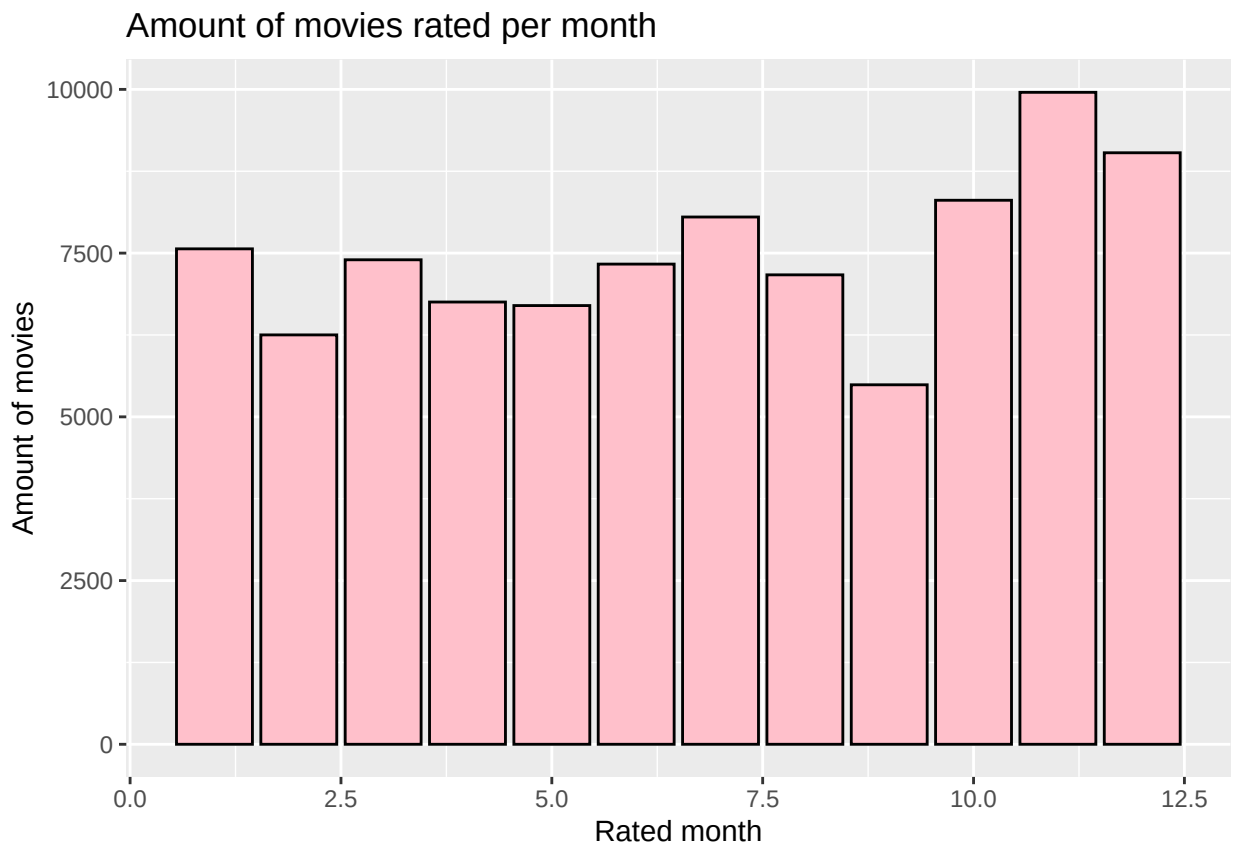
```
edx_new %>% group_by(rated_year) %>% ggplot(aes(rated_year)) +
  geom_bar(fill= "green", colour = "black") +
  labs(x = "Rated year", y = "Amount of movies",
       title = "Amount of movies rated per year")
```



We see fewer reviews in 1998 and 2009 (for 2009, probably because of the time when the data set was created, but it can't be confirmed) and 3 peaks in 1996, 2000, and 2005. From 1999 to 2009, except for the peaks and the drops in 1998 and 2009, the values are relatively steady around 6000 reviews per year.

Creating a histogram with the amount of movies rated per month

```
edx_new %>% group_by(rated_month) %>% ggplot(aes(rated_month)) +
  geom_bar(fill= "pink", colour = "black") +
  labs(x = "Rated month", y = "Amount of movies",
       title = "Amount of movies rated per month")
```

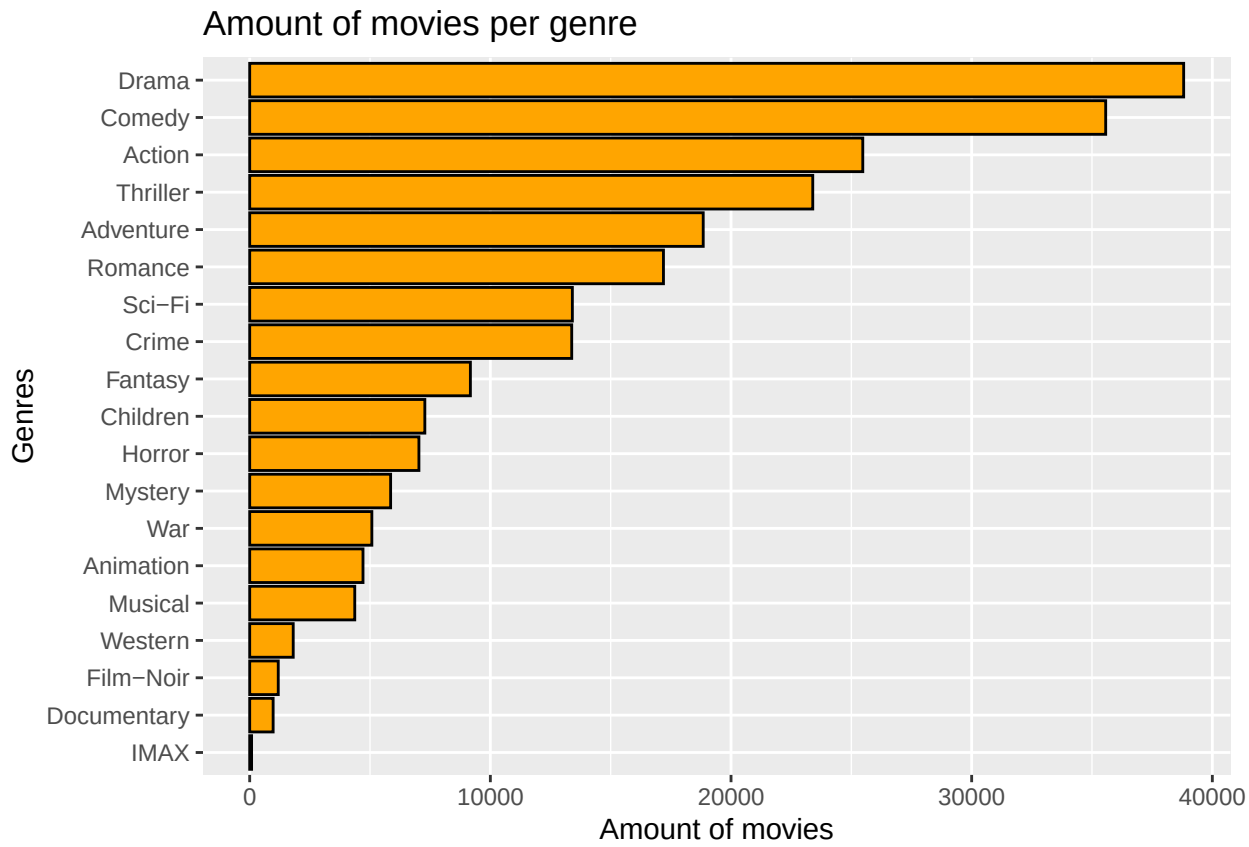


We can see a relatively steady amount of reviews from January to August (months 1 to 8), a drop in September (month 9), and an increase at the end of the year(months 10 to 12).

Creating a function to reorder any element, that will be applied to the column genres in edx_genres to plot the amount of movies per genre in a decreasing order

```
reorder_size <- function(x) {factor(x, levels = names(sort(table(x))))}

ggplot(edx_genres, aes(reorder_size(genres))) +
  geom_bar(fill= "orange", colour = "black") + coord_flip() +
  labs(y = "Amount of movies", x = "Genres", title = "Amount of movies per genre")
```



Drama, Comedy, Action, Thriller and Adventure are the most popular movies reviewed, and the least are IMAX, Documentary, Film-Noir and Western.

Calculating the average rating

```
mean(edx_new$rating)
```

```
## [1] 3.513594
```

The average rating is 3.51 (on a scale that goes up to 5, the higher the ranking, the better).

Creating a rating per movie dataframe with movies and their average mean to calculate how many movies have a perfect rating and how many have the worst one

```
rating_per_movie <- edx_new %>% group_by(title) %>%
  summarize(movie_rating_mean = mean(rating))
```

Checking all the movies with a perfect rating

```
length(rating_per_movie$title[rating_per_movie$movie_rating_mean == 5])
```

```
## [1] 130
```

There are 130 movies with perfect rating.

Checking all the movies with the worst rating

```
length(rating_per_movie$title[rating_per_movie$movie_rating_mean == 0.5])
```

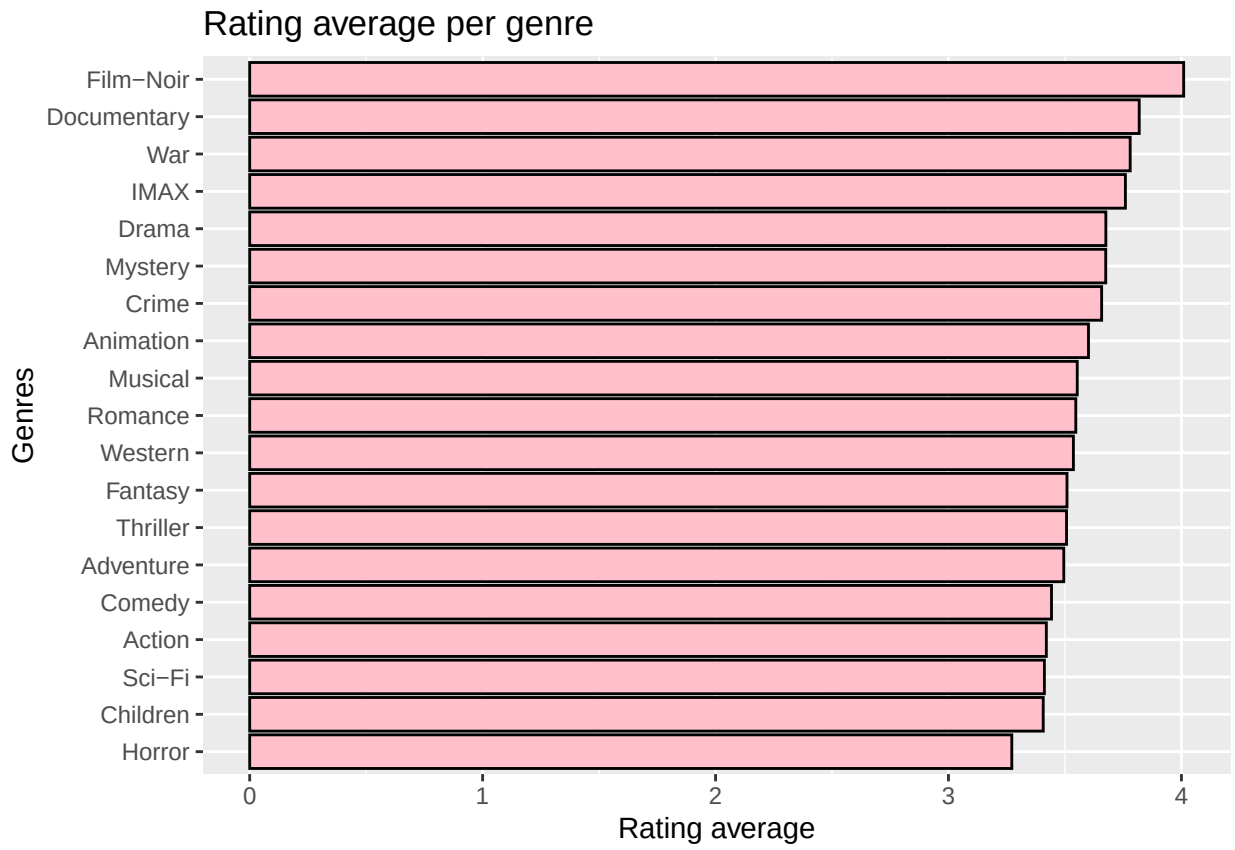
```
## [1] 46
```

There are 46 movies with the worst possible rating.

Creating a histogram of the rating average per genre

```
rating_per_genre <- edx_genres %>% group_by(genres) %>%
  summarize(genre_rating_mean = mean(rating))

ggplot(rating_per_genre, aes(reorder(genres, genre_rating_mean), genre_rating_mean)) +
  geom_bar(fill= "pink", colour = "black", stat = "identity") + coord_flip() +
  labs(y = "Rating average", x = "Genres", title = "Rating average per genre")
```

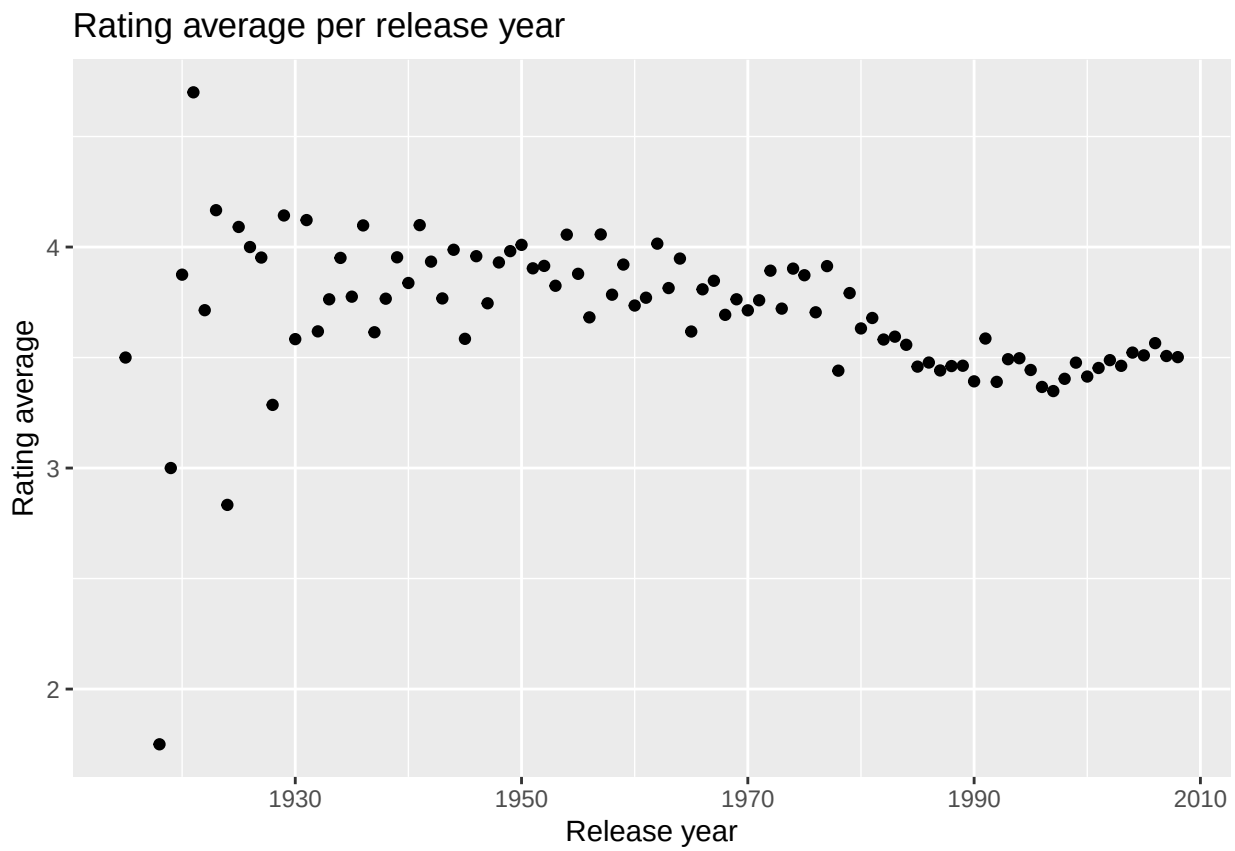


The three best rated are Film-Noir, Documentary and War, and the three worst, Horror, Children and Sci-Fi.

Creating a scatter plot with the rating average per release year

```
rating_per_release <- edx_new %>% group_by(release) %>%
  summarize(release_rating_mean = mean(rating))

ggplot(rating_per_release, aes(release, release_rating_mean)) +
  geom_point(stat = "identity") +
  labs(y = "Rating average", x = "Release year",
       title = "Rating average per release year")
```

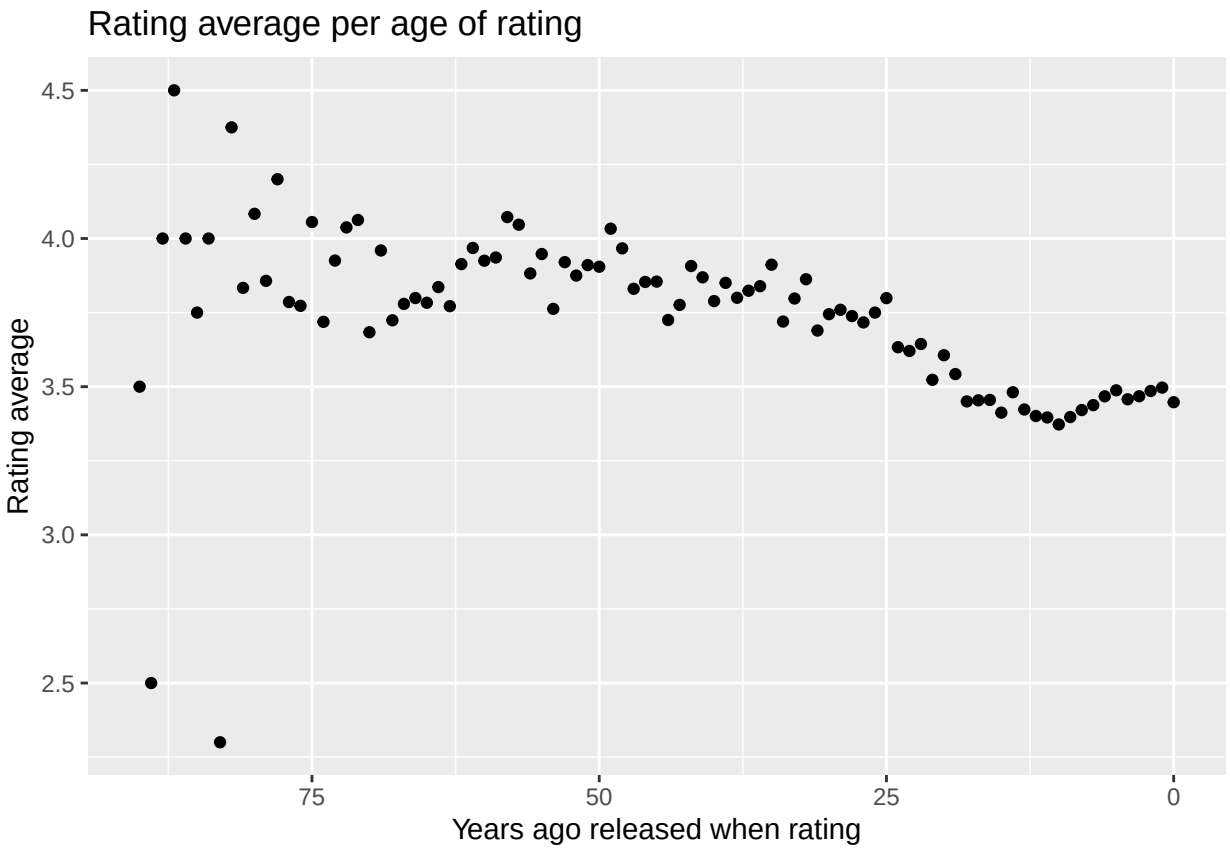


We can see that the rating averages vary less and less with time and they become more stable towards the average rating of 3.51.

Creating a scatter plot with the rating average per age of rating

```
rating_per_age_rated <- edx_new %>% group_by(age_rated_movie) %>%
  summarize(age_rated_rating_mean = mean(rating))

ggplot(rating_per_age_rated, aes(age_rated_movie, age_rated_rating_mean)) +
  geom_point(stat = "identity") +
  labs(y = "Rating average", x = "Years ago released when rating",
       title = "Rating average per age of rating") +
  scale_x_reverse()
```

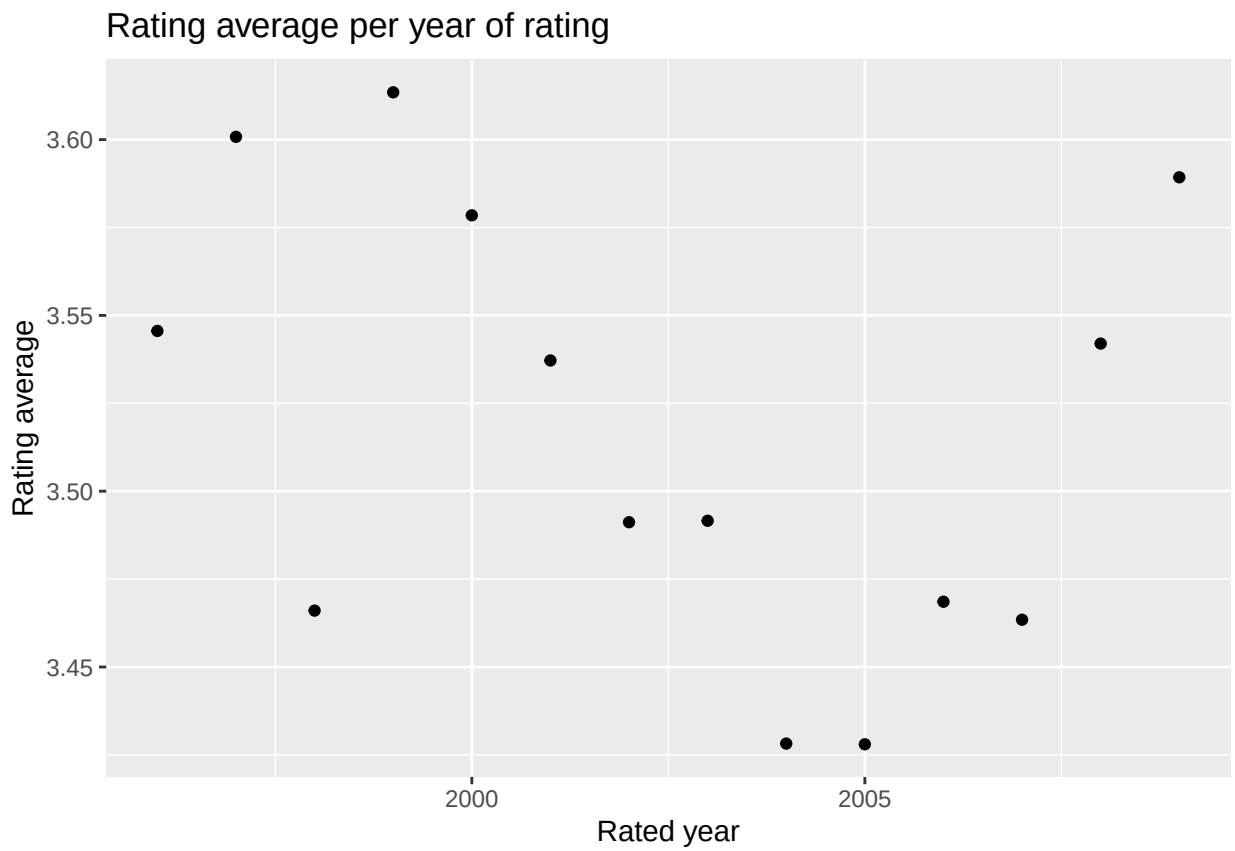


We see the same behavior as in the previous image, the older the age of ratings, the more it varies.

Creating a scatter plot with the rating average per year of rating

```
rating_per_year_rating <- edx_new %>% group_by(rated_year) %>%
  summarize(year_rated_rating_mean = mean(rating))

ggplot(rating_per_year_rating, aes(rated_year, year_rated_rating_mean)) +
  geom_point(stat = "identity") +
  labs(y = "Rating average", x = "Rated year",
       title = "Rating average per year of rating")
```

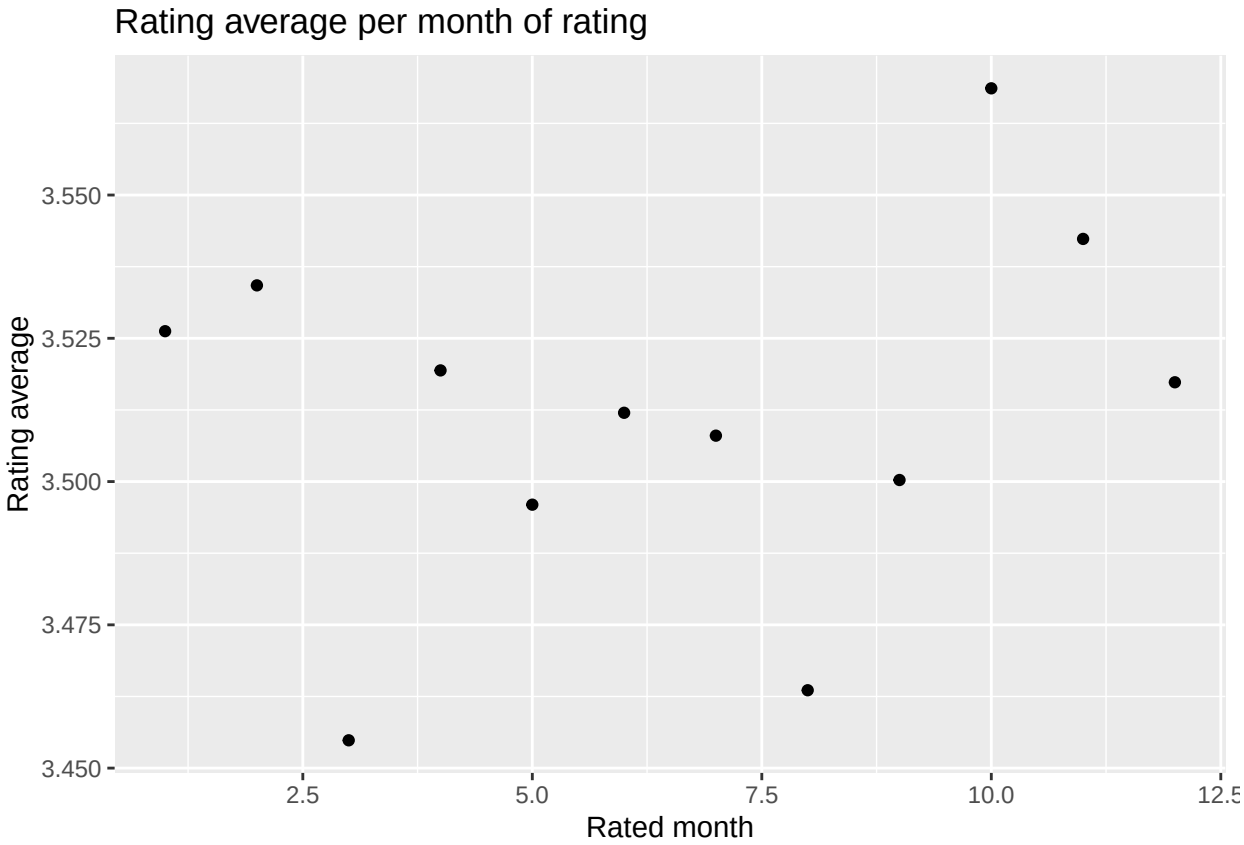


All the values are close to the average, 3.51. In general, they first tend to go upwards, then downwards, then upwards again, but these changes are actually minuscule.

Creating a scatter plot with the rating average per month of rating

```
rating_per_month_rating <- edx_new %>% group_by(rated_month) %>%
  summarize(month_rated_rating_mean = mean(rating))

ggplot(rating_per_month_rating, aes(rated_month, month_rated_rating_mean)) +
  geom_point(stat = "identity") + labs(y = "Rating average", x = "Rated month",
                                       title = "Rating average per month of rating")
```



Again the values are close to the average and they don't seem to follow a clear trend.

The model approach will be the following: An RMSE formula will be created to calculate how concentrated the data is around the line of best fit. After that, we will be checking the tuning value of the lambda for the final calculation. Lambdas from 0 to 6 will be chosen with an interval of 0.1. The formula to best calculate the RMSEs values is Yu,i= mu+bi+bu. Mu assumes that all the movies and users will have the same value (average of ratings), bi the average rating for any movie "i", and bu the user specific effect. The lowest value of lambda will be used to calculate both bi and bu. After getting the proper lambda for the model, the same formula will be applied to the validation set to calculate the RMSE. The ideal value should be less than 0.86490, which means the predicted average rating of all movies put together will be less than 0.8690 points away (from a rating possibility that goes from 0.5 to 5) from the actual average rating.

3. RESULTS

Creating the RMSE formula

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

Calculating the best lambda and plotting a graph to visualize it better

```
Lambdas <- seq(0, 6, 0.1)

RMSEs <- sapply(Lambdas, function(l){
  mu <- mean(edx_new$rating)
```

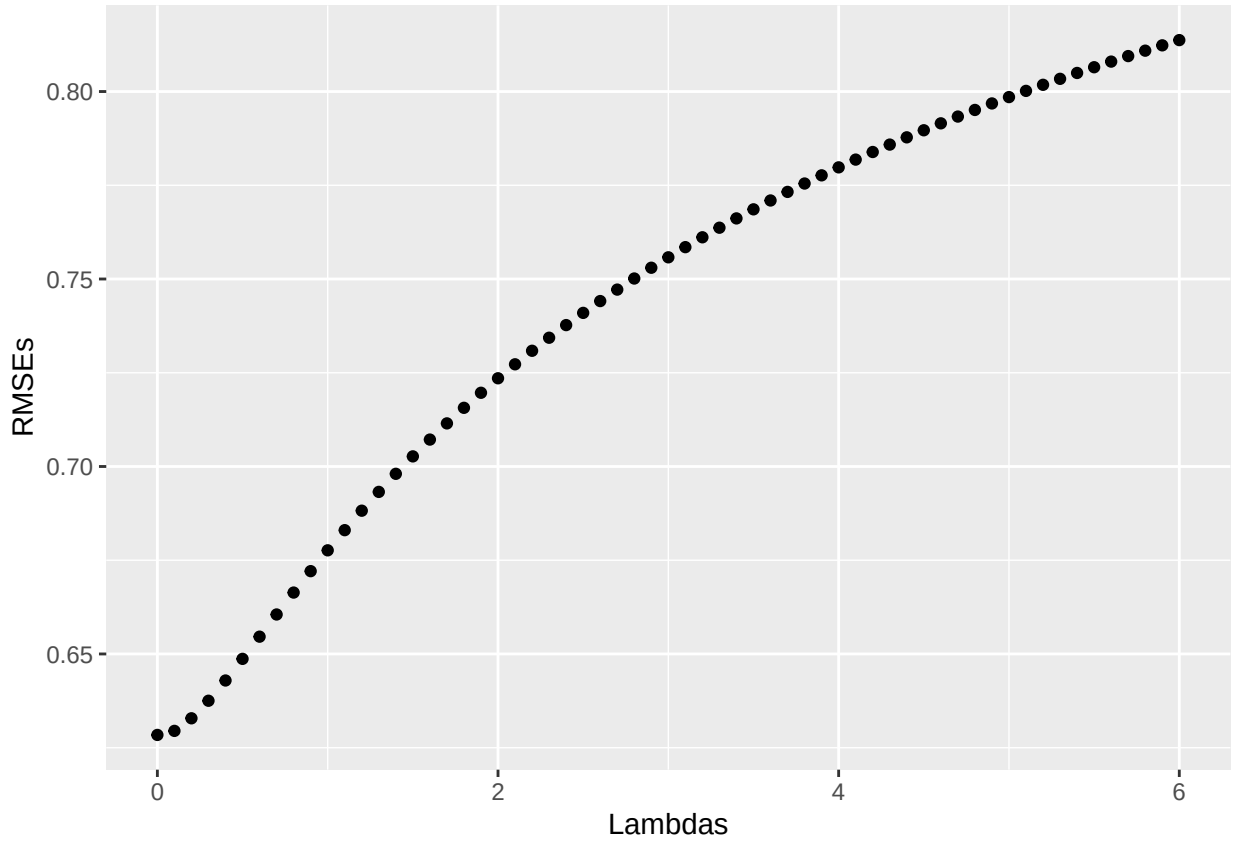
```
  bi <- edx_new %>% group_by(movieId) %>%
    dplyr::summarize(bi = sum(rating - mu)/(n() + l))

  bu <- edx_new %>% left_join(bi, by='movieId') %>%
    group_by(userId) %>% dplyr::summarize(bu = sum(rating - bi - mu)/(n() +l))

  predicted_ratings <- edx_new %>% left_join(bi, by = "movieId") %>%
    left_join(bu, by = "userId") %>% mutate(pred = mu + bi +  bu) %>% .$pred

  return(RMSE(predicted_ratings, edx_new$rating))
})

qplot(Lambdas, RMSEs)
```



Since the RMSEs increase as the lambdas do, the smaller the lambda, the better the result that will be obtained. We will use a lambda of 0.01.

Calculating the final RMSE

```
mu <- mean(validation$rating)

l <- 0.01

bi <- validation %>% group_by(movieId) %>%
  dplyr::summarize(bi = sum(rating - mu)/(n() + l))
```

```
bu <- validation %>% left_join(bi, by='movieId') %>% group_by(userId) %>%
    dplyr::summarize(bu = sum(rating - bi - mu)/(n() +l))

predicted_ratings <- validation %>% left_join(bi, by = "movieId") %>%
    left_join(bu, by = "userId") %>% mutate(pred = mu + bi +  bu) %>% .$pred

RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8251719
```

The result of the RMSE applied to the validation set is 0.8251719, which falls under the desired range of less than 0.86490, by 0.039721. The model applied here achieved the expected results.

4. CONCLUSION

As we have seen, the results gotten were under the value of 0.86490, which was the best possible result. The final goal was achieved, but do to computational restrictions I was forced to reduce the sample in order to be able to do any work at all.