HarvardX: PH125.9x Data Science - Capstone Project MovieLens - Rating Prediction

Léo Dange - https://github.com/ldange

November 29, 2020

Preamble

I had started the Data Science Program in September 2019 to improve my skills and my knowledge of data science in my daily job as Marketing Manager, but also to challenge myself. With no particular programming Skills or R Knowledge and education with a serious lack of fundamental mathematics and statistic, this process challenged me more than I thought. The Machine Learning chapter as been by far the hardest one, I am happy to be able to finish this journey and grateful to have learned so much through various aspects of Data Science.

I would like to thanks my family and my partner that always push me forward, providing trust and motivation all along.

Introduction

Related to the ***Harvardx Data Science Program - Capstone PH125.9x***, this project of **Rating Prediction** is a practical application of what I have learned in this program. It aim to build a Trained Machine Learning Algorithm able to predict user ratings from 0.5 to 5 5 (stars) using the **edx set** provided in the course PH125.9.

This Report will present you the ***Project Goal, the Data used, The Methods and Analysis of the results*** and at the end some ***Conclusion***.

The project will be delivered of the following files :

- The Report in .pdf
- The Report in .Rmd
- The Code and Script in .R

Project’s Goal

As explained in the Introduction, the Project’s Goal is to ***build a Trained Machine Learning Algorythm able to predict user ratings fromm 0.5 to 5 5 (stars) using the*** *edx set* ***provided in the course PH125.9.***

To evaluate our work we are going to use the Root Mean Square Error or RMSE :

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

The RMSE is frequently used to measure the difference between values predicted by a model and the observed values, in our case between the ***edx set (Train subset) and the Validation subset***. The best model will be defined as the one with the lowest RMSE and we aim to reach an **RMSE < 0.86490**

Data

Our two sets are based on the MovieLens 10M Dataset, which is composed of roughly: 10 million ratings, 100’000 tag applications applied to 10’000 movies by 72’000 users, and 18 genres/movie types.

We split this data set in two partitions with the code provided in the course, the edx set (train set) will hold 90% of the data, and the validation 10% left.

```
##########################################################
# Create edx set, validation set (final hold-out test set)
##########################################################

# Note: this process could take a couple of minutes

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

library(tidyverse)
library(caret)
library(data.table)
library(stringr)
library(lubridate)
library(gridExtra)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

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

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

# if using R 4.0 or later:
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")

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

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

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

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

The two sets are composed of 6 variables and we can see that the edx set is, as required, containing 90% of the data.

```
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 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 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)"
"Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller"
"Action|Drama|Sci-Fi|Thriller" "Action|Adventure|Sci-Fi" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

There is six features/variables/columns in both datasets :

- **userId**, ***integer*** containing the identification number of the user.
- **movieId**, ***numeric*** containing the identification number of the movie.
- **rating**, ***numeric*** containing the rating grade, given by a user for a movie
- **timestamp**, ***integer*** containing the timestamp for every rating given by a user.
- **title**, ***character*** containing the title of each movie and the year of release.
- **genres**, ***character*** containing a list of genre of each movie.

Exploratory Data Analysis

```
head(edx, n=3)
```

```
##    userId movieId rating timestamp            title
## 1:      1     122      5 838985046 Boomerang (1992)
## 2:      1     185      5 838983525  Net, The (1995)
## 3:      1     292      5 838983421  Outbreak (1995)
##                          genres
## 1:               Comedy|Romance
## 2:        Action|Crime|Thriller
## 3: Action|Drama|Sci-Fi|Thriller
```

To be able to conduct an Exploratory Data Analysis those data will need few transformations as we can quickly see by looking at the head, and the structure. ***The following modification will be done in a new data frame called edx_eda in order to keep the original one as requested*** :

- Convert the timestamp of rating into a readable date
- Extract year of rating from the readable date
- Extract year of release from the title

Those modifications will provide us the possibility to conduct a full EDA regarding the Rating, and Year of release which would not have been possible without them.

A Genres Analysis will be held after another modification that aims to extract genres from the title. this last modification will be performed on another dataset only use for the genre analysis, as it will significantly increase the number of rows.

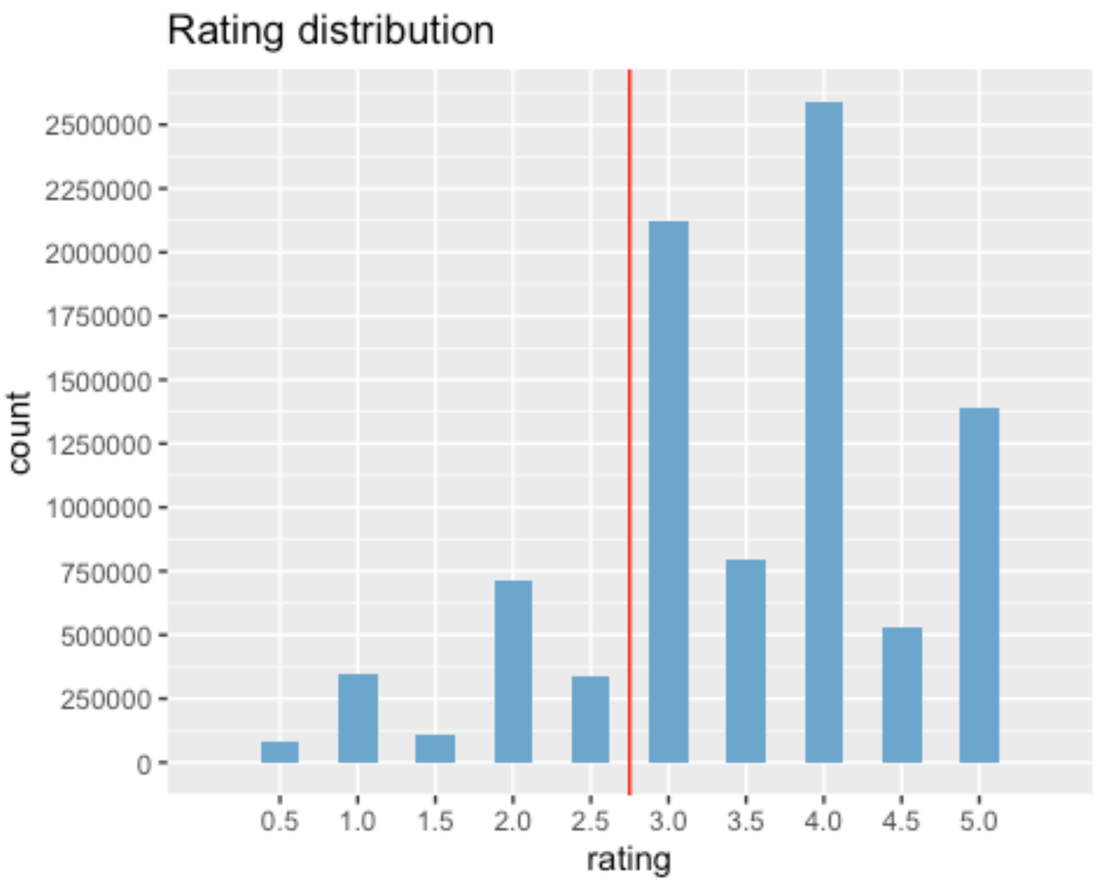
```r
# Extract Year of release and date of rating
edx_eda <- edx %>%
  mutate(year_of_release = as.numeric(str_sub(title, -5, -2))) %>%
  mutate(date_of_rating = as_datetime(timestamp))
```

Rating Analysis

Let's start our analysis with the Rating Analysis. This first analysis will help us understand what's the average rating, how are they given and how much time is a movie rated.

First, we will start by looking at the rating distribution. The following plot provides two major pieces of information.

```r
mean(edx_eda$rating)

## [1] 3.512465

edx_eda %>%
    ggplot(aes(rating)) +
    geom_histogram(binwidth = 0.25, fill = "skyblue3") +
    geom_vline(xintercept = 2.75, colour = "red") +
    scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
    scale_y_continuous(breaks = c(seq(0, 2750000, 250000))) +
    ggtitle("Rating distribution")
```



1. We can clearly see on the plot that users seem to rate movies more positively movies. Indeed, the formula below shows that more than 80% (0.8242329) of the given rating are between 3 and 5.

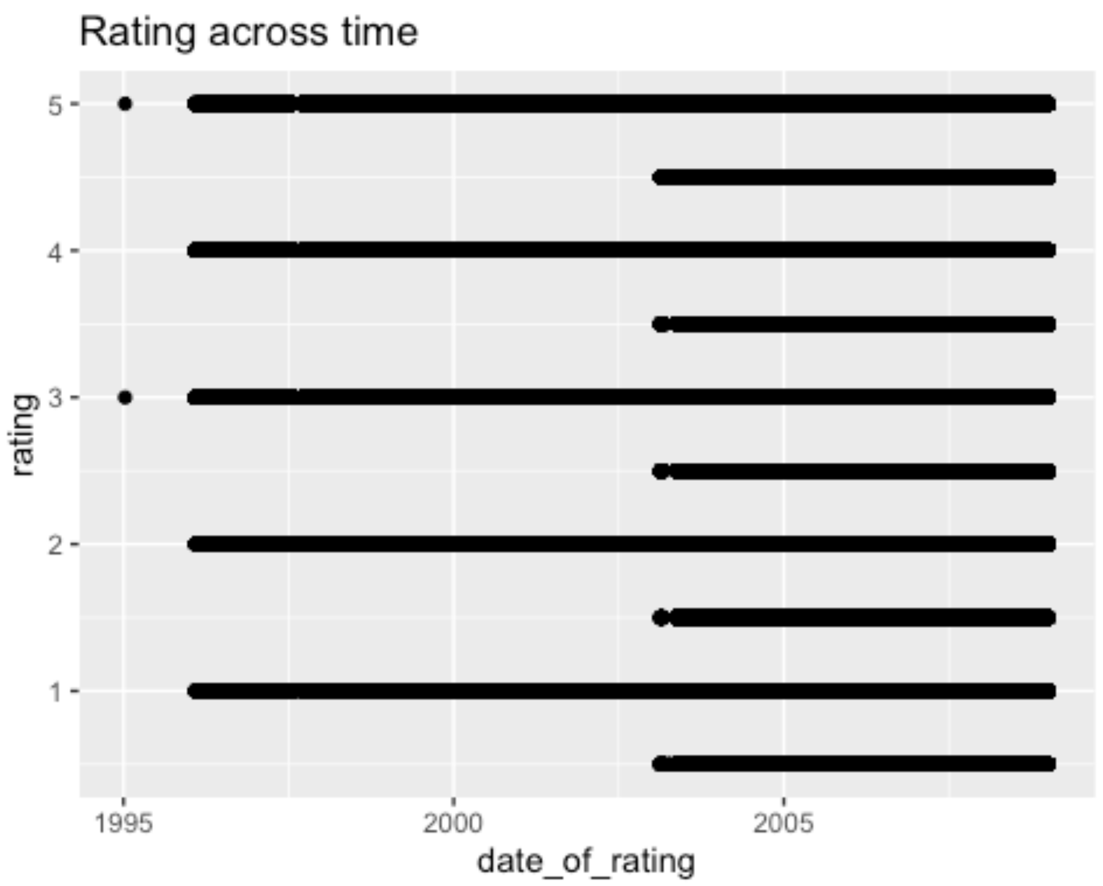
```
edx_eda %>% group_by(rating) %>% filter(rating >=3) %>% count() %>%sum() /
edx_eda %>% group_by(rating) %>% count() %>%sum()

## [1] 0.8242329
```

2. It appeared that round ratings are more important than the half number ratings, which is confirmed by the formula below. Nearly 7'156'900 rates are round number, more than 74% of the total rating in the data set. **4, 3 & 5 (in this order) are by far the most given rates**. At the opposite, **0.5, 1.5 & 2.5 are the least given rates of all.**

```
edx_eda %>% group_by(rating) %>% filter(rating %in% c(1, 2, 3, 4, 5)) %>%
count() %>% sum()

## [1] 7156900
```

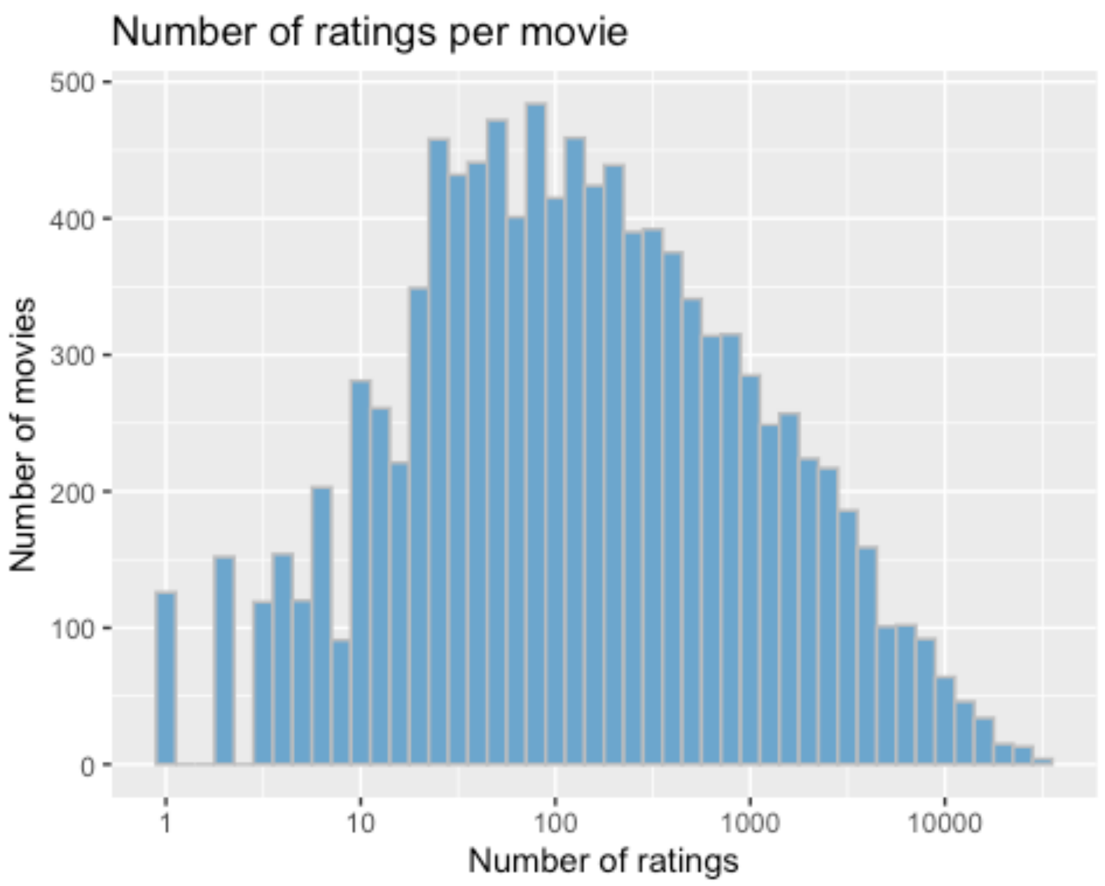
This second assumption might be partly influenced by another factor, to be sure we will look at the rating across time. A simple scatterplot shows that there is **no half rating before 2003**. This information might explain the important difference of rating between the whole number and the half number we highlight before, in our histogram.

```
edx_eda %>%
     ggplot(aes(date_of_rating,rating)) +
     geom_point(colors = "skyblue3") +
     ggtitle("Rating across time")

## Warning: Ignoring unknown parameters: colours
```



In the next plot we will observe the number of ratings per movie. This information is important for our prediction model.

```
edx %>%
    count(movieId) %>%
    ggplot(aes(n)) +
    geom_histogram(binwidth = 0.1, fill = "skyblue3", color = "grey") +
    xlab("Number of ratings") +
    scale_x_log10() +
    ylab("Number of movies") +
    ggtitle("Number of ratings per movie")
```



We observe that the distribution appears to be almost normal and most of the movies received between 40 and 120 ratings. At the extreme opposite, **around 125 movies have been rated only once** and **around 1000 movies have been rated 10'000 times**. This information is important for our prediction model as low rating numbers could result in an untrustworthy estimate.

To take this information into consideration, we will apply regularization and a penalty term to our models. It will help us to reduce the error by fitting a function appropriately on the training set and therefore avoid overfitting.

Rating Analysis Summary

To resume this first analysis, here is the key information we observed :

- The average rating is **3.512465**
- There is **more rating above 2.75** than under
- There is **more round number** than half
- Half rating have been **introduce in 2003** which could explain this difference
- Most of the movie received between **40 and 120 ratings**
- **125 movies** have only been **rated once**
- **~100 movies** have been rated more than **10'000 times**

Genres Analysis

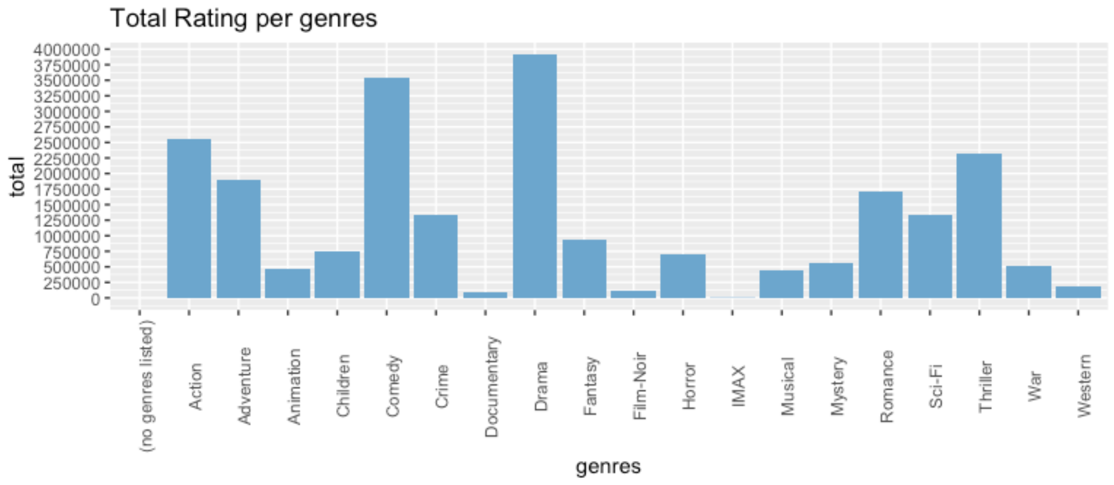
Our second analysis will concern the genres. This will help to understand the difference of rating comportment from a genre to another, the number of ratings for each genre and to establish if there is or not a correlation.

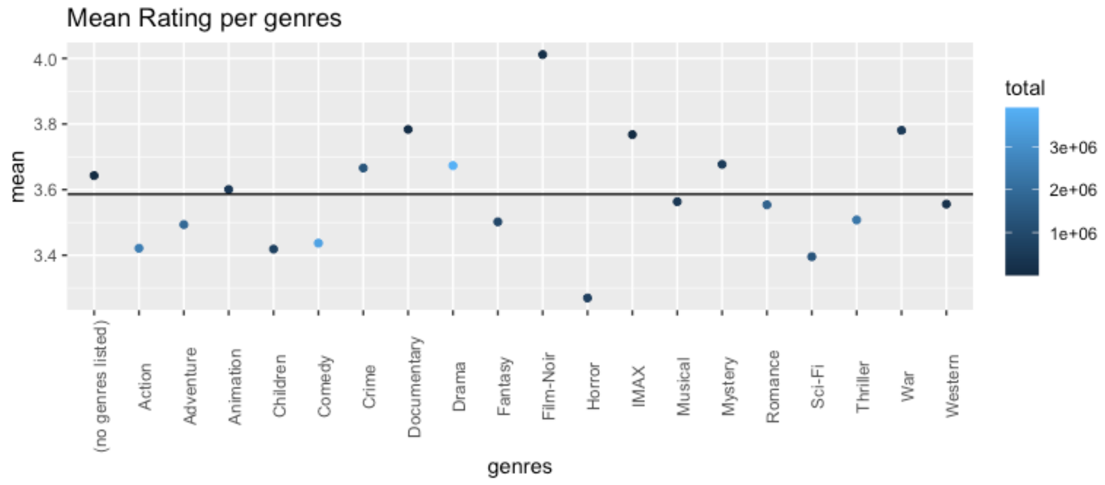
First of all, to be able to analyze each genre we will have to modify our Exploratory Data Analysis Base and separate the different genres that are pipe-separated.

```
# Split genres + add average rating per genres
edx_eda_genres <- edx_eda %>% separate_rows(genres, sep = '\\|')
edx_eda_genres <- edx_eda_genres %>% group_by(genres) %>%
mutate(average_rating = mean(rating))
```

In addition, I want to create an independent data frame that will contain the genre with the total rating and the average rating for each genre, in order to see if there is a correlation between those two variables.

```
# create Genres DFs
genres <- edx_eda_genres %>% group_by(genres) %>% count()
total_genres <- data.frame(genres = genres$genres, total =
as.numeric(genres$n))

mean_genre <- data.frame(genres = edx_eda_genres$genres, mean =
as.numeric(edx_eda_genres$average_rating))
mean_genre <- mean_genre[!duplicated(mean_genre$genres),]

movie_genres <- left_join(total_genres,mean_genre)

## Joining, by = "genres"

movie_genres <- movie_genres[order(-movie_genres$total),]

head(movie_genres, n=5)

##       genres   total     mean
## 9      Drama 3910127 3.673131
## 6     Comedy 3540930 3.436908
## 2     Action 2560545 3.421405
## 18  Thriller 2325899 3.507676
## 3  Adventure 1908892 3.493544

# Plot genres with the total rating and the mean rating
rating_plot <- movie_genres %>%
  ggplot(aes(genres, total, options(scipen=5))) +
  geom_col(fill = "skyblue3") +
  scale_y_continuous(breaks = c(seq(0, 50000000, 250000))) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Total Rating per genres")

mean_plot <- movie_genres %>%
  ggplot(aes(genres, mean)) +
  geom_point(aes(color = total)) +
  geom_hline(yintercept = 3.586) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Mean Rating per genres")

grid.arrange(rating_plot,mean_plot, ncol = 1)
```



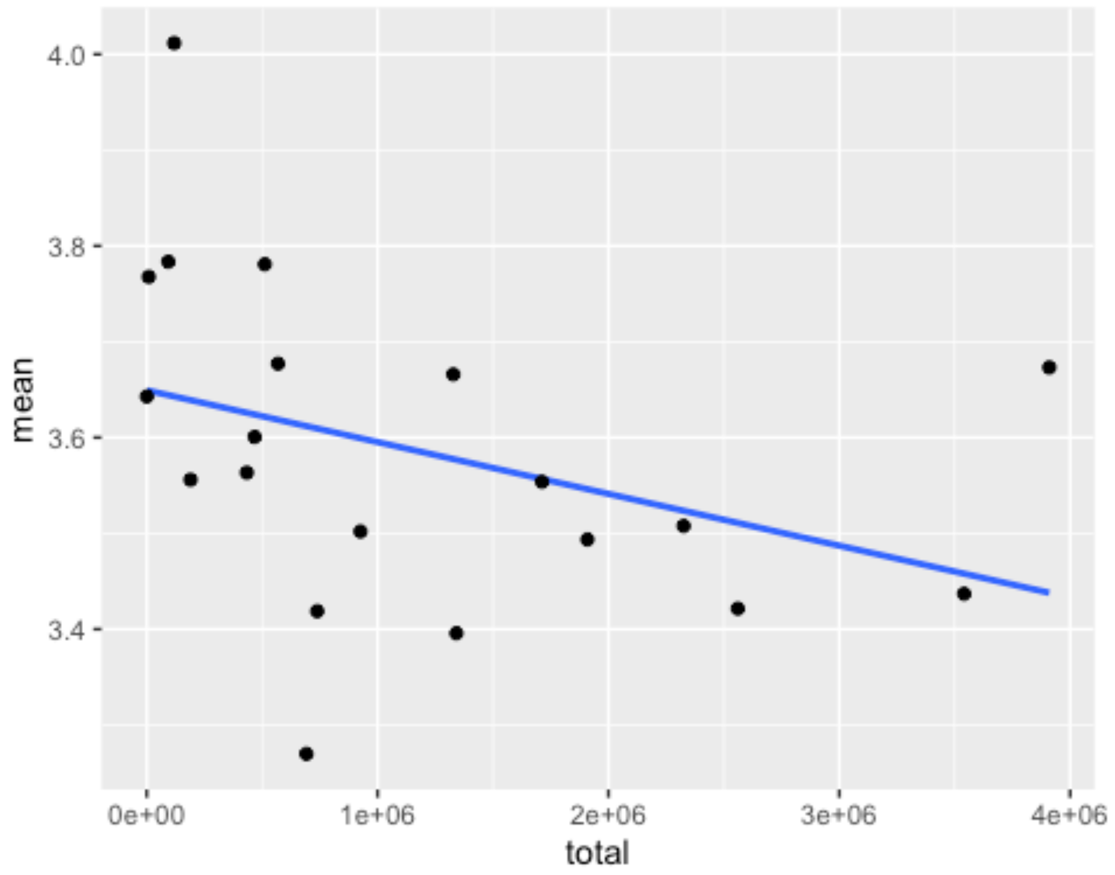


By plotting the Mean rating and the total rating per genre, it seems that there is no established correlation between those two variables. Despite having no correlation, some of the genres with a lower number of movies, such as Imax, Film-Noir, Documentary and War are way above the average rating of 3.586 and are clearly the best-rated category.

This assumption is confirmed below by a quick correlation test and the plot with the regression line. They show that there is a slightly negative correlation with a coefficient of -0.36 .

```
movie_genres %>% summarize(r = cor(total,mean))

##            r
## 1 -0.3668534
```

```r
movie_genres %>%
  ggplot(aes(total, mean)) + geom_smooth(method=lm, se=FALSE) +
  geom_point()

## `geom_smooth()` using formula 'y ~ x'
```



Genres Analysis Summary

To resume this second analysis, here is the key information we observed :

- **Drama** is the most used genre of all with a total rate of **3'910'127**
- **IMAX** is the least used genre of all with a total rate of **8'181**
- **Film-Noir** is the best-rated genre with an average rate of **4.011625**
- **Horror** is the worst-rated genre with an average rate of **3.269815**
- **There is no real correlation** between the average rating and the number of the rating given to a category. Indeed the correlation coefficient is **-0.36**

Users Analysis

Our last analysis will help us to understand the users and their comportment which will complete the first analysis regarding the rating. Before starting we will quickly set up a data frame to conduct the analysis.

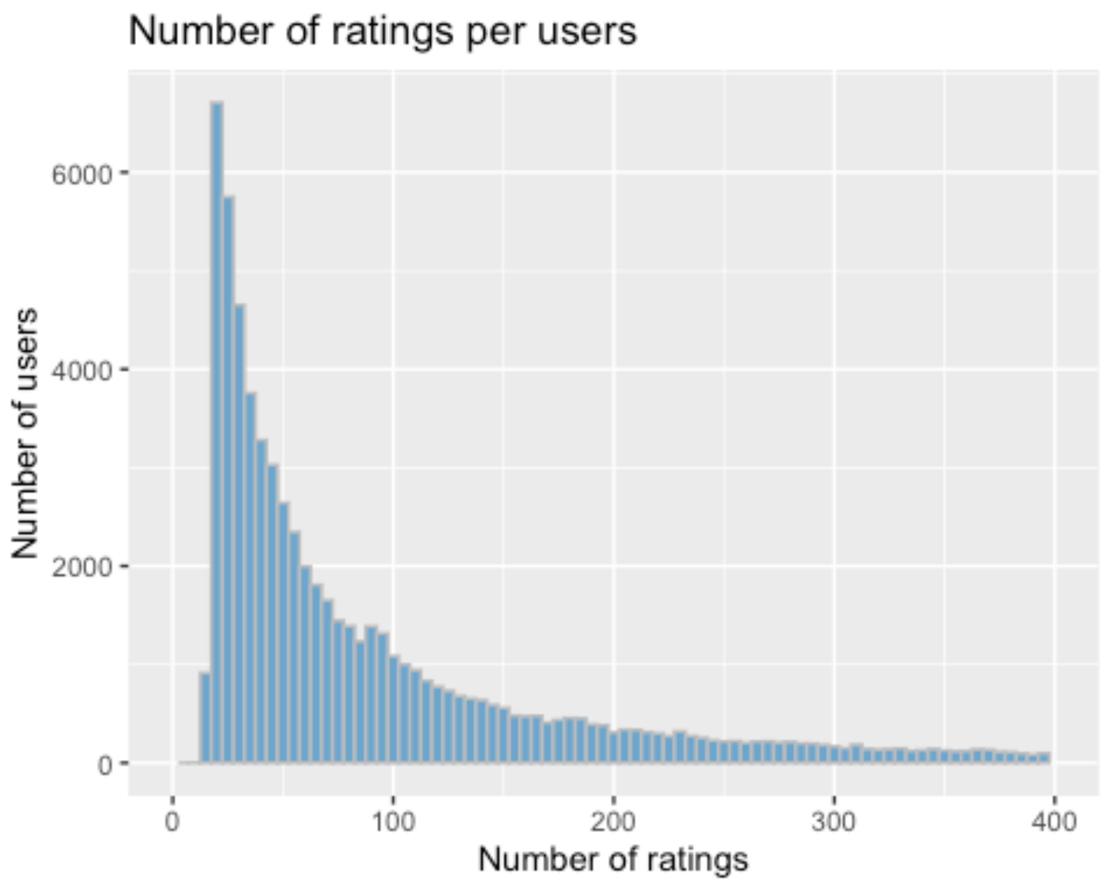
```
user_rating <- edx_eda %>% group_by(userId) %>% count()
user_rating <- data.frame(userId = user_rating$userId, total =
as.numeric(user_rating$n))

user_mean_rating <- edx_eda %>% group_by(userId) %>% summarize(Mean
=mean(rating))

## `summarise()` ungrouping output (override with `.groups` argument)

user_mean_rating <- data.frame(userId = user_mean_rating$userId, mean =
as.numeric(user_mean_rating$Mean))

user_rating <- left_join(user_rating, user_mean_rating)

## Joining, by = "userId"

head(user_rating, n = 7)

##   userId total     mean
## 1      1    19 5.000000
## 2      2    17 3.294118
## 3      3    31 3.935484
## 4      4    35 4.057143
## 5      5    74 3.918919
## 6      6    39 3.948718
## 7      7    96 3.864583
```

We want to know the number of users and the average number of ratings per user. We see that as stated before, there is in the whole data set around 10M rating, 90% are in the edx set. Divided by the number of users, it gives us the average number of ratings of **~129 ratings per user**. But this is just an average.

```
# Number of user
total_user <- user_rating %>% count()
total_user

##       n
## 1 69878

# Mean of rating per user
edx_eda %>% nrow() / total_user

##          n
## 1 128.7967
```

We want to plot the Number of ratings per user and compare it to the mean of rating per user.

```
# Plotting the number of rating per user
edx %>%
    count(userId) %>%
    ggplot(aes(n)) +
    geom_histogram(binwidth = 5, fill = "skyblue3", color = "grey") +
    scale_x_continuous() + xlim(0,400) +
    xlab("Number of ratings") +
    ylab("Number of users") +
    ggtitle("Number of ratings per users")
```



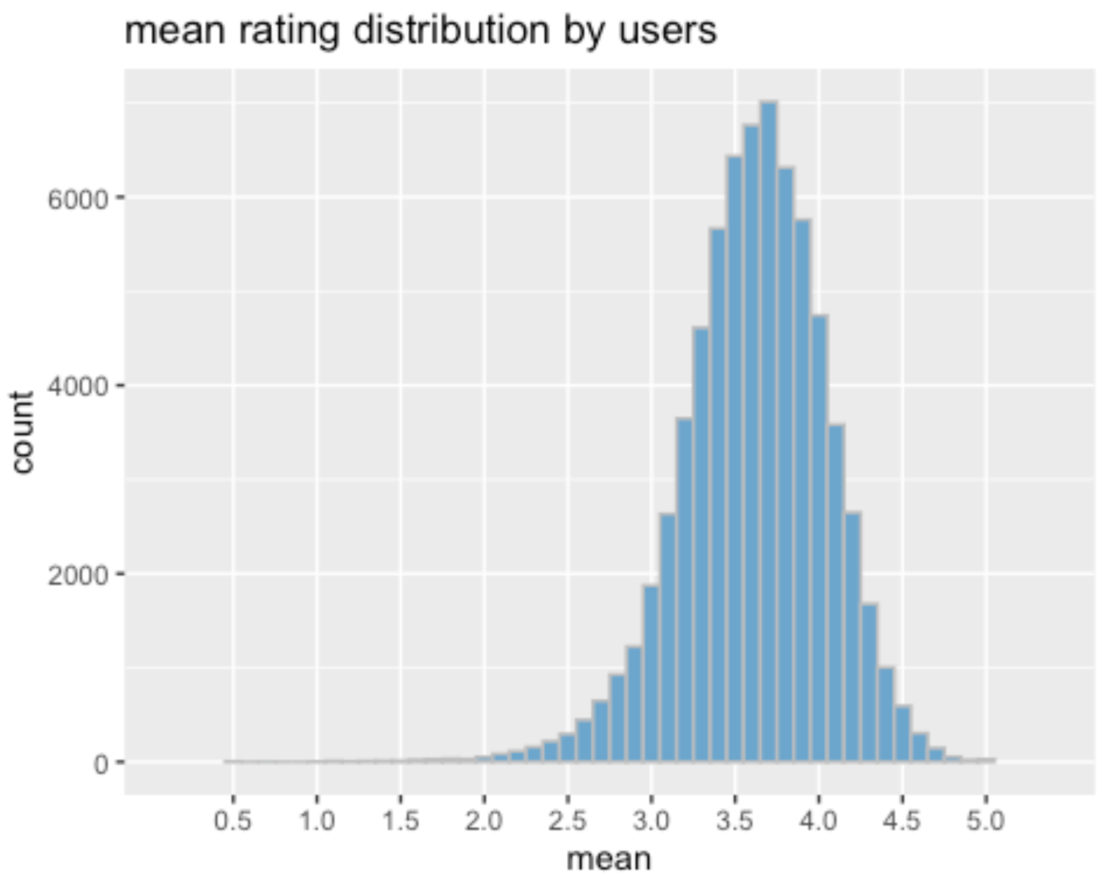
Despite having a high average of **~129 ratings per user** the following plot shows that the **majority of users (51.66%) gave between 15 and 65 ratings**, far from the average. Indeed, only 27.32% of users gave 129 ratings or more.

```
# Percent of user that gave more than 129 review
user_rating %>% filter(total >= 129) %>% count() / total_user

##           n
## 1 0.2732477
```

```r
# Percent of user that gave between 15 and 65 review
user_rating %>% filter(total %in% (15:65)) %>% count() / total_user

##           n
## 1 0.5166433
```

With the ***user_rating*** data collected in our dataframe we plot the mean rating distribution for the user.

```r
user_rating %>%
  ggplot(aes(mean)) +
  geom_histogram(binwidth = 0.1, fill = "skyblue3", color = "grey") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  ggtitle("mean rating distribution by users")

## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```



This last plot shows a normal distribution of mean rating per user, and we see that most users have an average rating ranging from 3.4 to 3.9. We have to take several pieces of information before taking any conclusion from this plot :

1. This plot shows all users, some of them might have only given a few ratings and other hundreds of rating.

2. We saw that rating strongly fluctuates from a genre to another, this plot concern all genre.

User Analysis Summary

To resume this last analysis, here is the key information we observed :

- The average number of rating per user is **~129**
- But the vast majority of users have between **15 and 65 rating**.
- All genre combined, the majority of rating **range from 3.4 to 3.9, with an average of 3.586**

Methods & Analysis

We will now try to find a model with the appropriate approach to correctly predict rating. As stated previously, The best model will be defined as the one with the lowest RMSE and we aim to reach an **RMSE < 0.86490** . We will evaluate the differents approach to try to reach our goal :

- A basic model based on **the average movie rating** of the data where ***mu = 3.512465***.
- An improvement of the basic model with the addition of an **independent error term regarding movie bias**
- An improvement of the second model with the addition of an **independent error term regarding user bias**
- A Regularized model of our last improvement, as independent error term may induce error our the RMSE.

Average movie rating model

In this first model, we start by building the easiest possible predicting system, while predicting all the unknown rating based on ***mu***, as defined before.

$$Y_{u,i} = \mu,$$

$Y_{u,i}$ is the predicted rating of : - u is the user - i is the movie - μ is the average rating of 3.512465

```
#Compute mu & predict unknown rating based on mu
mu <- mean(edx$rating)
average_movie_rating <- RMSE(validation$rating, mu)
average_movie_rating

## [1] 1.061202
```

This approach gives us a first RMSE of 1.061202 which is far from being accurate.

Average movie rating with movie bias model

In our second model, we will start to implement an independent error term to take into consideration different ratings for a movie as b_i is the movie bias term. As we saw earlier in our analysis movie are not rated the same (in number and average rate), this bias term takes this into consideration.

```
#compute the movie bias b_i
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

## `summarise()` ungrouping output (override with `.groups` argument)
```

$$Y_{u,i} = \mu + b_i$$

```r
#Predicting the rating with mu and the bias term.
predicted_ratings_1 <- validation %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
avg_movie_rating_w_movie_bias <- RMSE(predicted_ratings_1, validation$rating)
avg_movie_rating_w_movie_bias

## [1] 0.9439087
```

We improved the accuracy of our RMSE with this first bias from 1.061202 to 0.9439087.

Average movie rating with movie & user bias model

In this second iteration of the average movie rating model, we are adding a user bias term in addition to the movie bias term introduce before. As for the movie bias, the user bias aims to reduce the effect of extreme ratings given by some users.

```r
# Compute the bias term, b_u
b_u <- edx %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

$$Y_{u,i} = \mu + b_i + b_u$$

```r
#Predicting the rating with mu and the movie and user bias term.
predicted_ratings_2 <- validation %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

avg_movie_rating_w_user_bias <- RMSE(predicted_ratings_2, validation$rating)
avg_movie_rating_w_user_bias

## [1] 0.8653488
```

Our new RMSE of **0.8653488** is closed to our goal, but the based model use for our RMSE cannot permit us to reach our goal.

Regularized user and movie effect model

In our two previous models (and as see in our analysis), some movies have only a few rating, and for some of the extreme one which affects b_i. Same for the user, some user gave only a few rating with really high/low rating and have a strong effect on b_u. Those effects introduce a larger estimate on both ways (positive or negative), large errors can therefore increase the RMSE and reduce his precision.

On this new model, we will regularize instead of computing standard error and confident interval as we did before. The regularization will permit us to penalize large estimate from small sample size on b_i & b_u. This method will also allow us to reduce the effect of overfitting. We want to find the value of lambda that could minimize our RMSE.

First, we define our lambdas from a sequence as follow

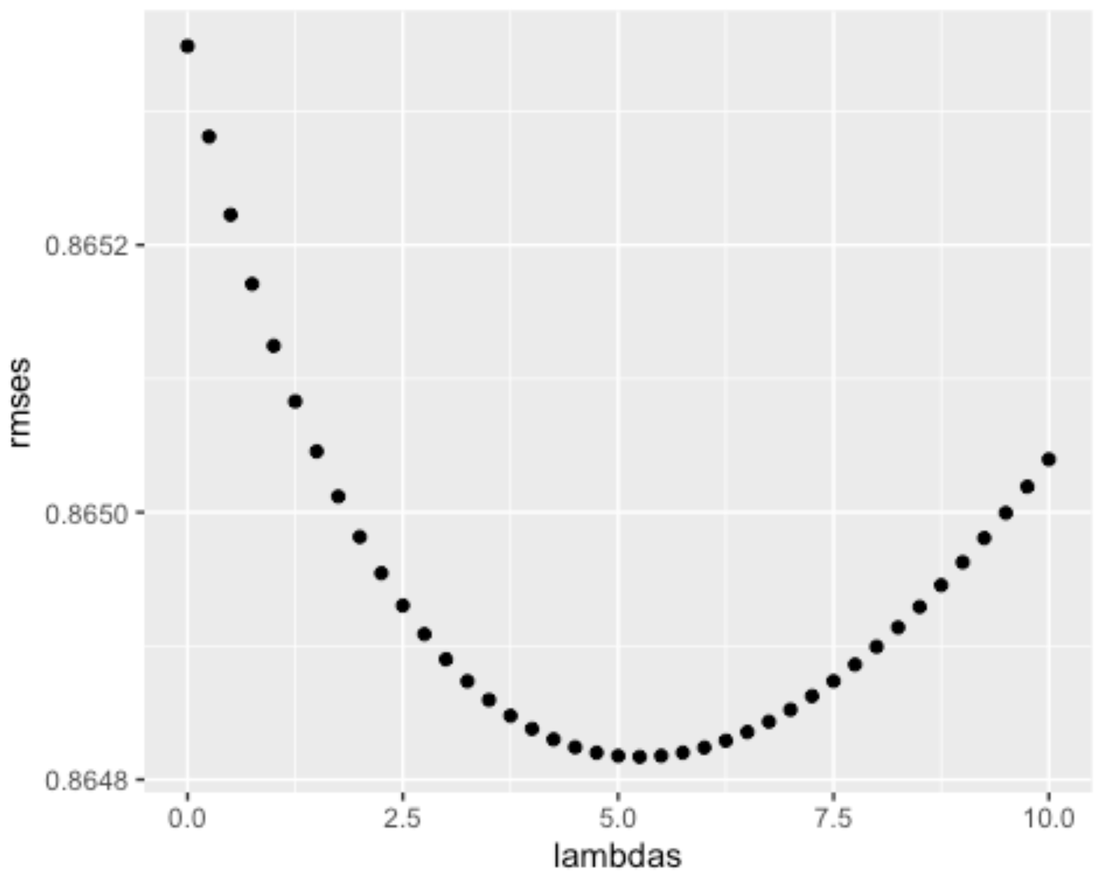
```
lambdas <- seq(from=0, to=10, by=0.25)
```

Then, using a sapply function (based on our lambdas), we redefined μ b_i & b_u before predicting our third ratings and returning the RMSE.

```
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
   b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings_3 <- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings_3, validation$rating))
})
```

We then, plot the lambdas vs our RMSES and the that the minimizing lambda is indeed 5.25

```
qplot(lambdas, rmses)
```



```
lambdas[which.min(rmses)]

## [1] 5.25
```

Using the minimum lambda, the models looks like this :

```
lambdas_min <- lambdas[which.min(rmses)]

b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambdas_min))

## `summarise()` ungrouping output (override with `.groups` argument)

b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambdas_min))

## `summarise()` ungrouping output (override with `.groups` argument)

predicted_ratings <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

And give us our targeted RMSE of **0.864817**

```
regularized_model <- RMSE(predicted_ratings, validation$rating)
```

Results

By improving step by step our RMSE with the introduction of our first bias term, up to the regularization of those terms, we improved the accuracy of our RMSE from ***1.061202*** to ***0.864817***.

Conclusion

After improving our model we have finally build a functional machine learning algorithm, able to predict movie rating through our database. We might have been able to improve more our models using more data from the MovieLens database such as the year of release of the movie, or with the addition of more data on the user such as his age or his location (as the taste might variate from countries or culture). But our model work on its own for now.