# HarvardX :PH125.9x Data Science: Capstone - Movielens Project: Predict movie ratings

*Roger Comas*

*February, 2020*

## Contents

# 1 Summary

This project is about the Movielens Project, from the *"HarvardX: PH125.9x Data Science: Capstone"*, the last course which of the *Data Science Professional Certificate* of HarvardX Universityn.

## 1.1 Aim of the project

The aim of the project is to create a machine learning algorithm to predict movie ratings. This system is used in movie platforms, like Netflix, as movie reccomendation system.

A dataset has been provided by the edx staff, with the actual information of different users and movie ratings, as well as other information, like the moment the user rated the movie or the genre of the movie.

## 1.2 Dataset

The information has been divided into 2 dataset. The training dataset is called "edx" and the other dataset, called "validation", is only used for the validation of the model created.

In this project, the model is going to be evaluated with the Root Mean Squared Error (RMSE), which follows the next expression:

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

The RMSE is one of the most used measures of the differences between values predicted by a model and the values observed. It is a measure of accuracy, used to compare forecasting errors of different models for a particular dataset, so a lower RMSE is better than a higher one.

The Movielens is automatically downloaded with the code provided by the edx staff:

```
################################
# Create edx set, validation 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")

# 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")
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")
```

As explained, the dataset is divided into a 2 subsets: the training set and the validation set with the following code:

```
# 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)` instead
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)
```

# 2 Data analysis

The first thing to do is to analyze the data received, to try to figure out which are the most important variables to predict the movie ratings.

## 2.1 Exploring the data

The information given by the edx dataset is the following:

```
head(edx) %>%
  print.data.frame()
```

```
##   userId movieId rating timestamp                          title
## 1      1     122      5 838985046               Boomerang (1992)
## 2      1     185      5 838983525                Net, The (1995)
## 4      1     292      5 838983421                Outbreak (1995)
## 5      1     316      5 838983392                Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                         genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

The summary of the information given by the dataset can be visualized with the following code:

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

## 2.2 Addition of a new column

As seen in last table, the year of the movie is included in the title. In the following code, we are going to extract the year from the title and create a new column with this information:

```r
# Create the pattern to extract the year
pattern <- "\\(([^()]+)\\)$"

# Create a column with the year of the movie in the edx dataset
edx <- edx %>% mutate(year = str_extract(title,pattern))

# Delete the parenthesis of the year column
edx$year <- substring(edx$year, 2, nchar(edx$year)-1)

# Create a column with the year of the movie in the validation dataset
validation <- validation %>% mutate(year = str_extract(title,pattern))

# Delete the parenthesis of the year column
validation$year <- substring(validation$year, 2, nchar(validation$year)-1)
```

Now we have the new column with the year of each movie:

```r
head(edx)
```

```
##   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)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                          genres year
## 1                Comedy|Romance 1992
## 2         Action|Crime|Thriller 1995
## 3  Action|Drama|Sci-Fi|Thriller 1995
## 4       Action|Adventure|Sci-Fi 1994
## 5 Action|Adventure|Drama|Sci-Fi 1994
## 6       Children|Comedy|Fantasy 1994
```

## 2.3 Number of movies and users

The total number of movies and users in the dataset to work is the following:

```r
edx %>%
  summarize(num_users = n_distinct(userId),
            num_movies = n_distinct(movieId))
```

```
##   num_users num_movies
## 1     69878      10677
```

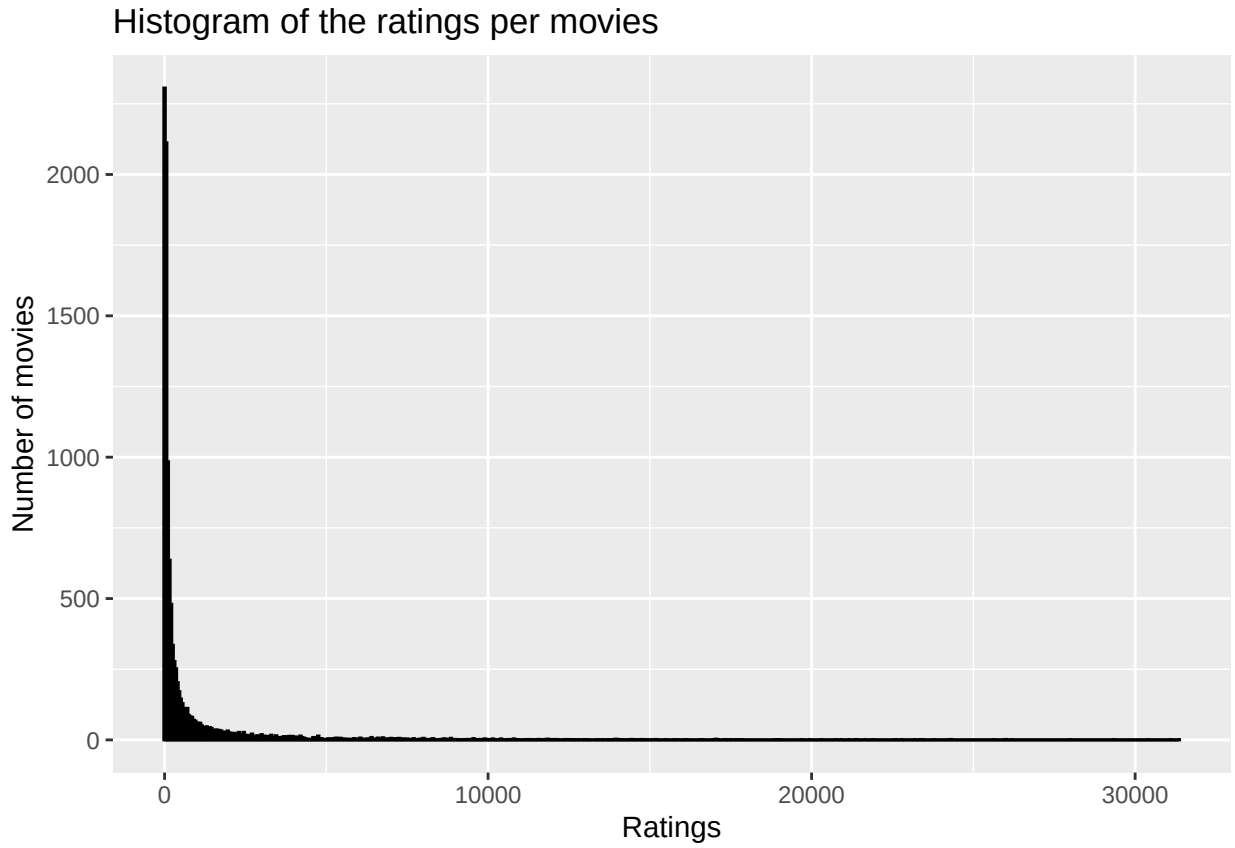
## 2.4 Analysing the data

Now that we have an idea of what we have in the dataset, such as the variables and how many information it has, it is time to analyse the dataset to understand the behaviour of the data and try to make the best prediction model.

### 2.4.1 Movie analysis

First, it is important to know if all the movies have approximately the same number of ratings. We can plot this as it follows:

```
# Plot of the number of ratings per movie
edx %>% group_by(movieId) %>%
  summarize(n_ratings = n()) %>%
  ggplot(aes(n_ratings)) +
  geom_histogram(binwidth = 50 , color = "black") +
  xlab("Ratings") +
  ylab("Number of movies") +
  ggtitle("Histogram of the ratings per movies")
```




As we can see, most of the movies have a low number of ratings and only a few of them have a large number of ratings.

The following table contains the 10 movies with a larger number of ratings:

```
edx %>%
  group_by(title) %>%
  summarize(count=n()) %>%
  top_n(10,count) %>%
  arrange(desc(count))
```

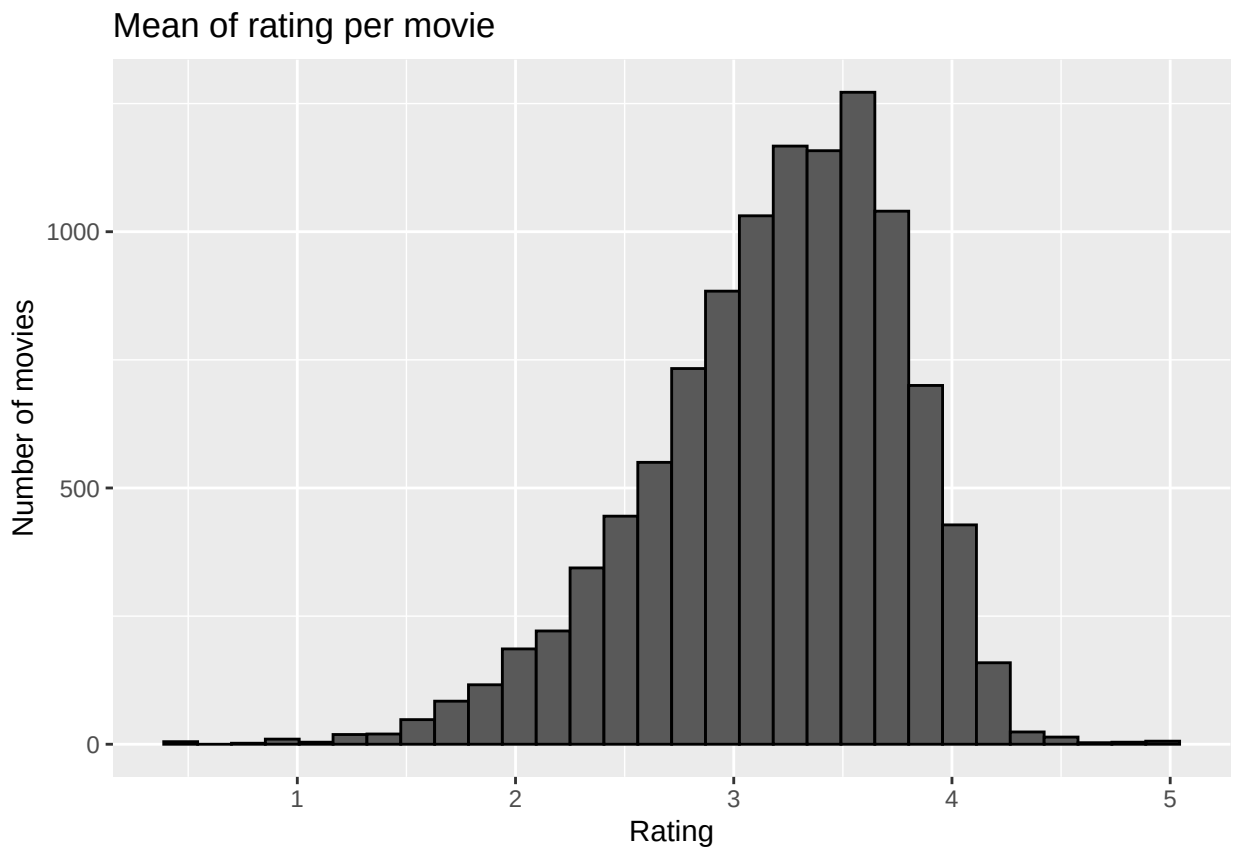
```
## # A tibble: 10 x 2
```

```
##    title                                                     count
##    <chr>                                                     <int>
##  1 Pulp Fiction (1994)                                       31362
##  2 Forrest Gump (1994)                                       31079
##  3 Silence of the Lambs, The (1991)                          30382
##  4 Jurassic Park (1993)                                      29360
##  5 Shawshank Redemption, The (1994)                          28015
##  6 Braveheart (1995)                                         26212
##  7 Fugitive, The (1993)                                      25998
##  8 Terminator 2: Judgment Day (1991)                         25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10 Apollo 13 (1995)                                          24284
```

From the last table, we can see that the most famous movies are also the movies with a higher number of ratings.

An important point is to know how the mean rating of the movies is distributted:

```
edx %>% group_by(movieId) %>%
  summarize(mean = mean(rating)) %>%
  ggplot(aes(mean)) +
  geom_histogram(bins = 30 , color = "black") +
  xlab("Rating") +
  ylab("Number of movies") +
  ggtitle("Mean of rating per movie")
```

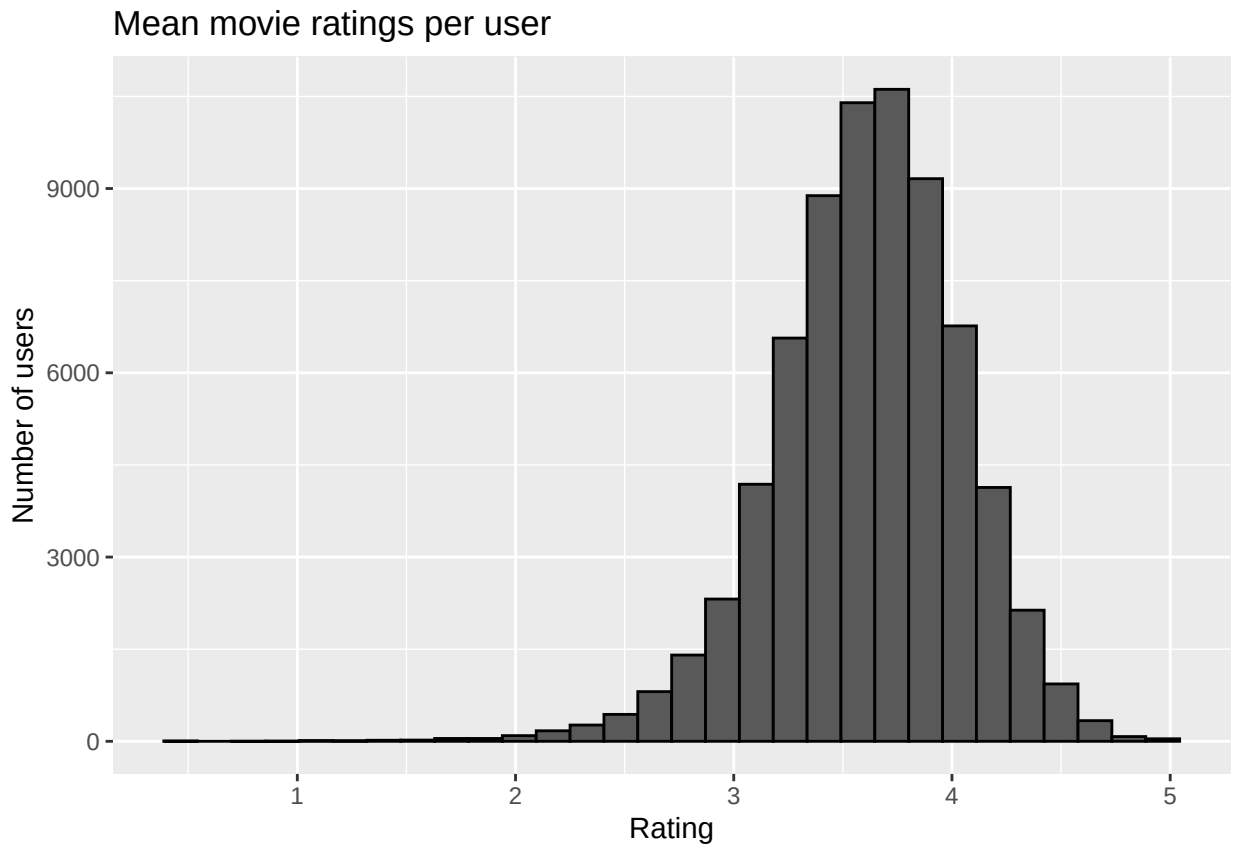


From the last plot, we can see that most of the ratings are between 3 and 4 stars. There are very few ratings between 0 - 1.5 and there are also few ratings between 4.5 - 5 stars.

### 2.4.2 User analysis

Another thing to analyse is the behavior of the users. To realise if there is a user effect, let's calculate the mean rating of every user and then use it in a histogram to see how the distribution is:

```
edx %>%
  group_by(userId) %>%
  summarize(mean = mean(rating)) %>%
  ggplot(aes(mean)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings per user")
```

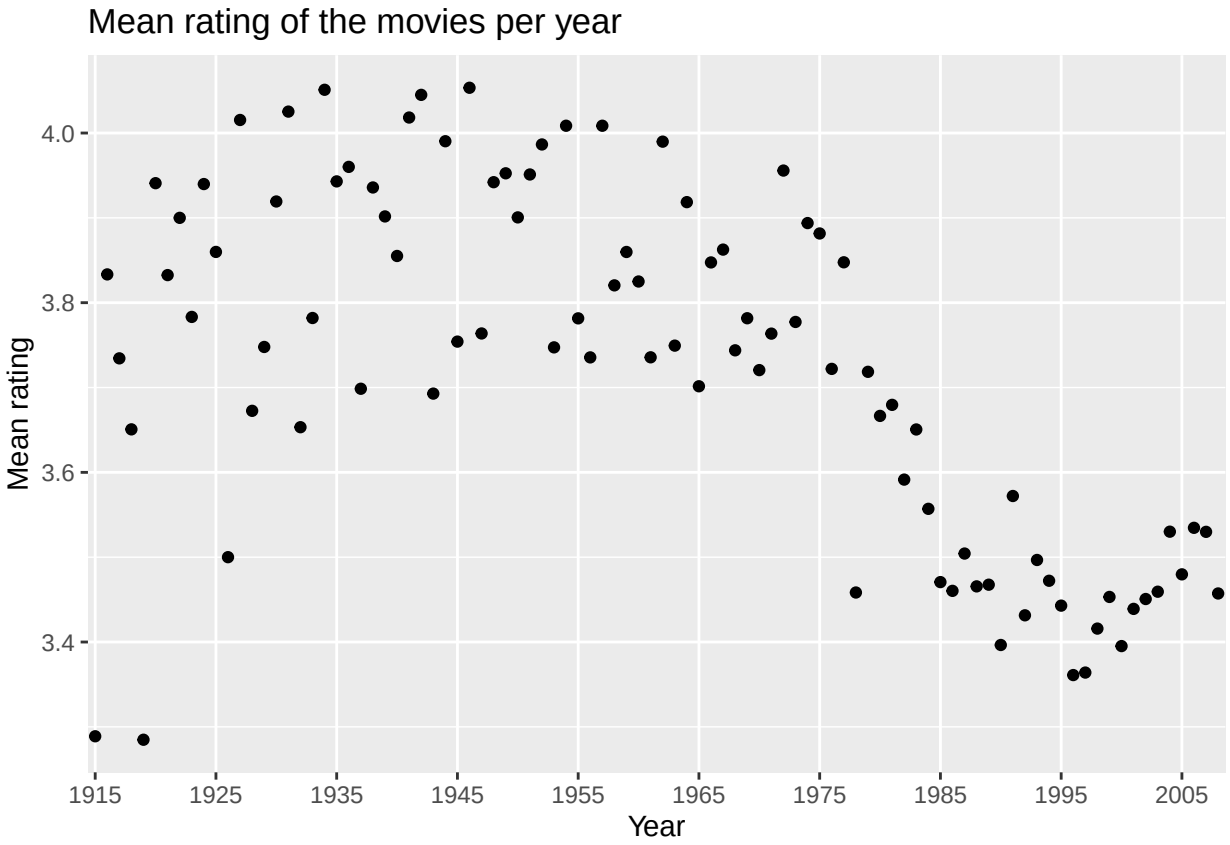



As we can see from the previous plot, there is a user effect in predicting the movie ratings. Not all the users rate in the same way and this is something we have to take care off when we are creating the prediction model.

### 2.4.3 Movie year analysis

Not all the people like the old or classic movies and not all the people like the newest movies, so the year of the movie is also another variable to keep in mind. The following plot shows the explained:

```
edx %>%
  group_by(year) %>%
  summarize(mean = mean(rating)) %>%
  ggplot(aes(year, mean, fill = year)) +
  geom_point()+
  ggtitle("Mean rating of the movies per year") +
  xlab("Year") +
  ylab("Mean rating") +
  scale_x_discrete(breaks = c(seq(1915,2008, 10))) +
  theme(legend.position= "none")
```




The plot confirms that the year of the movie is a variable to take into account in the prediciton model.

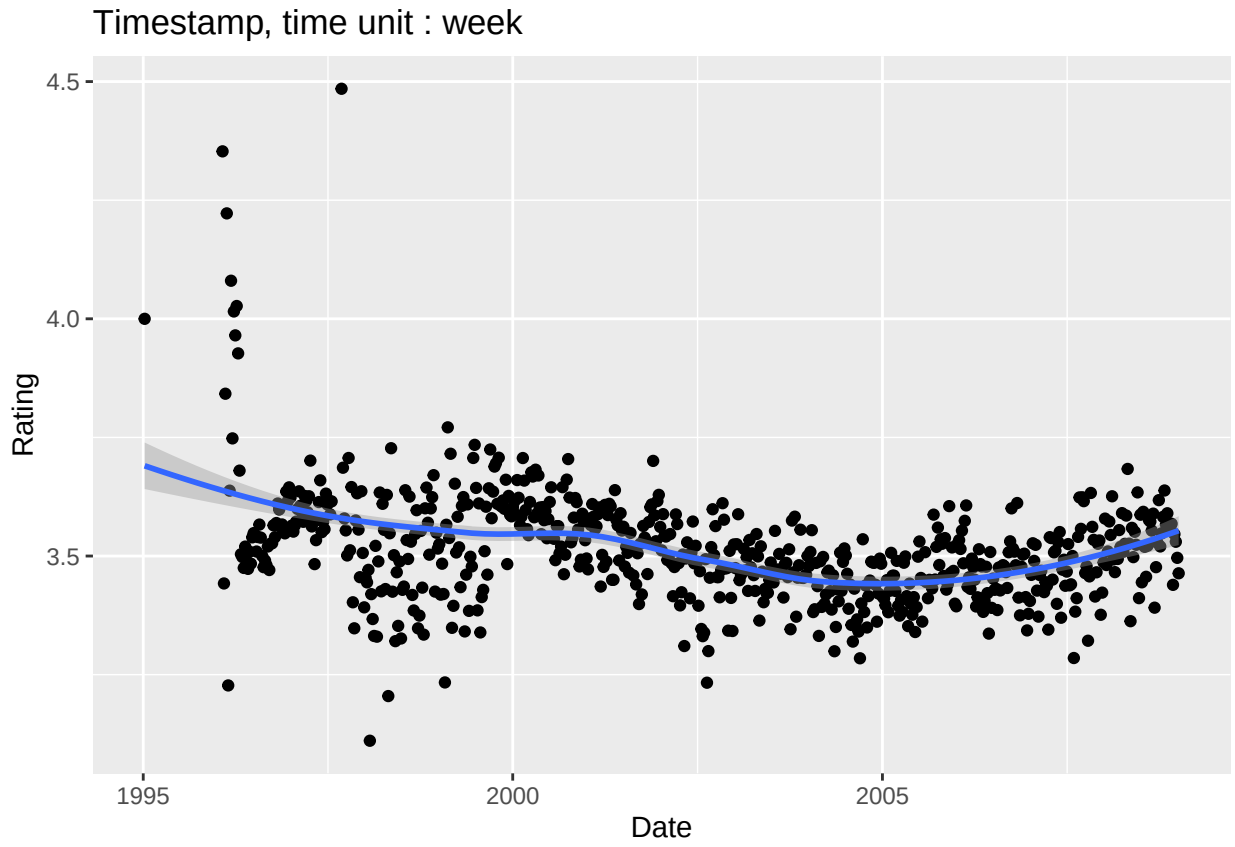
### 2.4.4 Timestamp analysis

One possible factor of a rating predict is the time when the movie is rated. This is stored in the column "timestamp" of the dataset. For example, in global economics crisis years, the ratings could not be the same as good economically years, or maybe in holidays people could rate in a different way than they normally.

In conclusion, another variable to analyze is the "timestamp", which is plotted with the following code:

```
library(lubridate)
edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
```

```
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Timestamp, time unit : week") +
  xlab ("Date") +
  ylab("Rating")
```



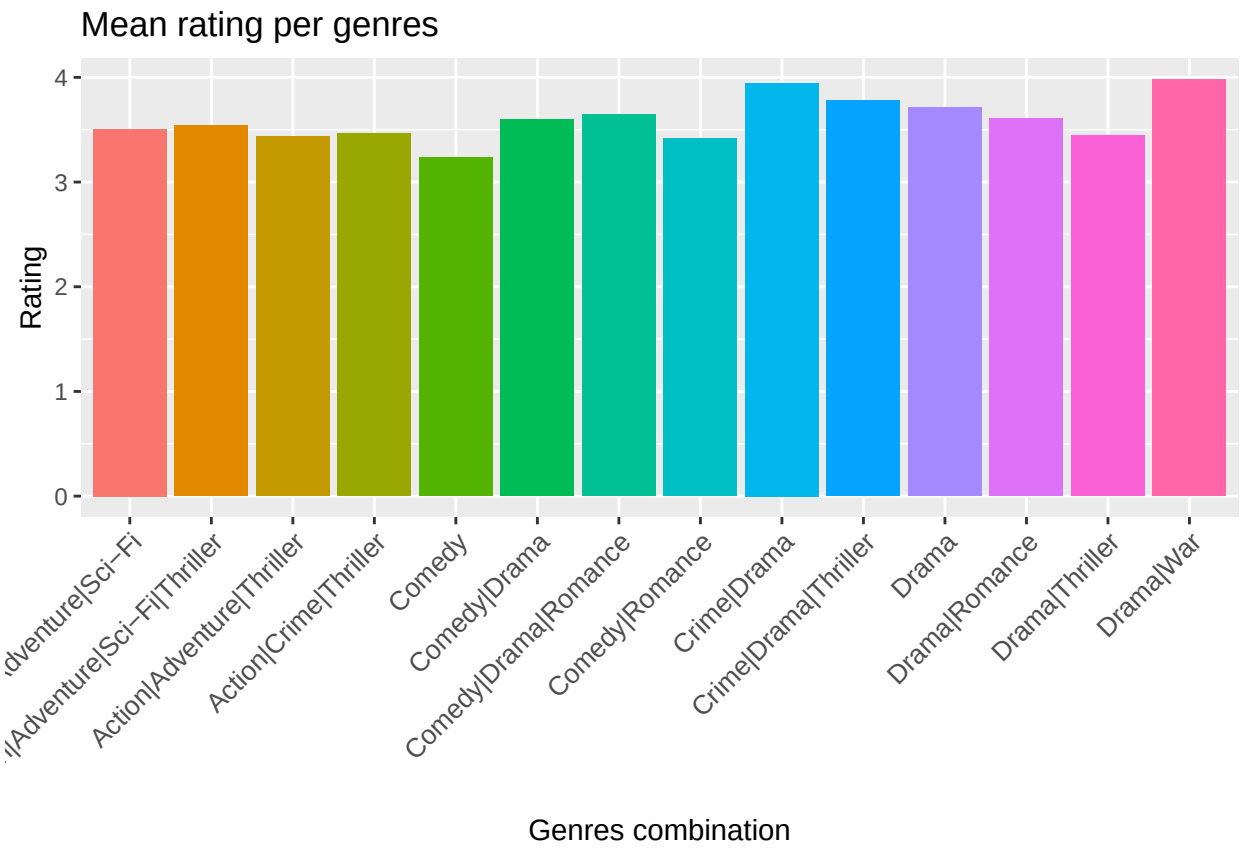

The conclusion from the previous plot is that the "timestamp" variable affect the rating of the movies.

### 2.4.5 Genre analysis

Finally, another variable to take into account is the genre of each movie. In the dataset, each movie can have various genres combined (for example "Comedy|Romance"). To see if it is an important variable, let's plot the mean rating of most important genres combinations (with more than 100,000 ratings):

```
edx %>%
  group_by(genres) %>%
  filter(n()>100000)%>%
  summarize( mean = mean(rating)) %>%
  arrange(mean) %>%
  ggplot(aes(genres, mean, fill = genres)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 10), legend.position = "none") +
  ggtitle("Mean rating per genres") +
```

```
  xlab("Genres combination") +
  ylab("Rating")
```




As seen in the last plot, each genre combination have significant different means, so it is a variable to take into account when modelling.

## 2.5 Create the prediction model

From the data analysis we have seen that there are some variables that have an effect, or bias, in the rating prediction. These variables are the following:

- Movie
- User
- Year of the movie
- Time of the rating
- Genres of the movie

So, in conclusion, to create a good prediction model, we should take into account all this variables.

### 2.5.1 Evaluation function

To evaluate the model we will use the RMSE, as explained in the **1.2. Dataset** point.

The first thing to do is to create the function to use in order to calculate the RMSE:

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

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

Once we have created this function, it is time to start modelling.

### 2.5.2 Simple model

We first create a simple model which follows the next equation:

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

Where $\mu$ is the mean rating of all the movies of the dataset, and $\epsilon_{u,i}$ is the error centered at 0. The average rating of the dataset is:

```
mu <- edx %>%
  summarize(mu = mean(rating))
mu
```

```
##         mu
## 1 3.512465
```

So this first model will take this number as if it was the rating of each movie. Now we are going to apply the RMSE function to know its value:

```
rmse_mu <- RMSE(validation$rating, mu[[1]])
rmse_mu
```

```
## [1] 1.061202
```

Let's save this result in a dataframe in which we will save all the models results to compare them:

```
rmse_results <- data.frame(model = "simple",
                           RMSE = rmse_mu)
rmse_results
```

```
##    model     RMSE
## 1 simple 1.061202
```

As we see it is not a good result, so we need to improve the model.

### 2.5.3 Model with biases

As we have seen in the data analysis, there are some variables in the dataset that affect to the rating of a movie. The aim of the following steps is to apply some biases effects of these different variables in order to find the best predicting model.

#### 2.5.3.1 Addition of the first bias

The models we are going to create are going to follow next equation:

$$Y_{u,i} = \mu + b_1 + \epsilon_{u,i}$$

Where $b_1$ is going to be the bias of one variable.

We will test all the variables as a bias and save it in the dataframe "rmse_results", so we can finally compare them and select the one that fits best(lower RMSE).

##### 2.5.3.1.1 Movie effect

```
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu[[1]]))

predicted_ratings <- validation %>%
  left_join(b_i) %>%
  mutate(pred = mu[[1]] + b_i)

rmse_movie <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie",
                                     RMSE = rmse_movie))
```

##### 2.5.3.1.2 User effect

```
b_u <- edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu[[1]]))

predicted_ratings <- validation %>%
  left_join(b_u) %>%
  mutate(pred = mu[[1]] + b_u)

rmse_user <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "user",
                                     RMSE = rmse_user))
```

##### 2.5.3.1.3 Year effect

```
b_y <- edx %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - mu[[1]]))

predicted_ratings <- validation %>%
  left_join(b_y) %>%
  mutate(pred = mu[[1]] + b_y)
```

```
rmse_year <- RMSE(validation$rating, predicted_ratings$pred)
rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "year",
                                     RMSE = rmse_year))
```

#### 2.5.3.1.4 Genre effects

```
b_g <- edx %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu[[1]]))

predicted_ratings <- validation %>%
  left_join(b_g) %>%
  mutate(pred = mu[[1]] + b_g)

rmse_genre <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "genre",
                                     RMSE = rmse_genre))
```

#### 2.5.3.1.5 Date effect

```
b_d <- edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(b_d = mean(rating - mu[[1]]))

predicted_ratings <- validation %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_d) %>%
  mutate(pred = mu[[1]] + b_d)

rmse_date <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "date",
                                     RMSE = rmse_date))
```

Now that we have created all the models with 1 bias, let's compare the results and select the best model:

```
rmse_results
```

```
##    model      RMSE
## 1 simple 1.0612018
## 2  movie 0.9439087
## 3   user 0.9783360
## 4   year 1.0500259
## 5  genre 1.0184056
## 6   date 1.0575018
```

As we see, the best model is the Movie effect.

#### 2.5.3.2 Addition of the second bias

It is time to add a second bias to the model and see how is its effect. The model will follow the next equation:

$$Y_{u,i} = \mu + b_i + b_2 + \epsilon_{u,i}$$

Where $b_i$ is the bias of the movie effect and $b_2$ is the second bias we are going to add.

##### 2.5.3.2.1 Movie + year effect

```
b_y <- edx %>%
  left_join(b_i) %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - b_i - mu[[1]]))

predicted_ratings <- validation %>%
  left_join(b_i) %>%
  left_join(b_y) %>%
  mutate(pred = mu[[1]] + b_i + b_y)

rmse_yu <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+year",
                                     RMSE = rmse_yu))
```

##### 2.5.3.2.2 Movie + genre effect

```
b_g <- edx %>%
  left_join(b_i) %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - b_i - mu[[1]]))

predicted_ratings <- validation %>%
  left_join(b_i) %>%
  left_join(b_g) %>%
  mutate(pred = mu [[1]] + b_g + b_i)

rmse_gu <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+genre",
                                     RMSE = rmse_gu))
```

##### 2.5.3.2.3 Movie + date effect

```
b_d <- edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_i) %>%
  group_by(date) %>%
  summarize(b_d = mean(rating - mu[[1]]))
```

```
predicted_ratings <- validation %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_i) %>%
  left_join(b_d) %>%
  mutate(pred = mu[[1]] + b_i + b_d)

rmse_du <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+date",
                                     RMSE = rmse_du))
```

All the models with the second bias added are created, it is time to compare the results and select the best one:

```
rmse_results
```

```
##         model      RMSE
## 1      simple 1.0612018
## 2       movie 0.9439087
## 3        user 0.9783360
## 4        year 1.0500259
## 5       genre 1.0184056
## 6        date 1.0575018
## 7  movie+year 0.9439087
## 8 movie+genre 0.9439087
## 9  movie+date 0.9415172
```

The best model is the movie + user effect.

**2.5.3.3 Addition of the third bias** It is time to add a third bias to the model and see how is its effect. The model will follow the next equation:

$$Y_{u,i} = \mu + b_i + b_u + b_3 + \epsilon_{u,i}$$

Where $b_i$ is the bias of the movie effect, $b_u$ is the bias of the user effect and $b_3$ is the third bias we are going to add.

**2.5.3.3.1 Movie + user + year effect**

```
b_u <- edx %>%
  left_join(b_i) %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - b_i - mu[[1]]))

b_y <- edx %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - b_i - b_u - mu[[1]]))
```

```
predicted_ratings <- validation %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_y) %>%
  mutate(pred = mu [[1]] + b_i + b_u + b_y)

rmse_iuy <- RMSE(validation$rating, predicted_ratings$pred)
rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+user+year",
                                     RMSE = rmse_iuy))
```

#### 2.5.3.3.2 Movie + user + genre effect

```
b_u <- edx %>%
  left_join(b_i) %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - b_i - mu[[1]]))

b_g <- edx %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - b_i - b_u - mu[[1]]))


predicted_ratings <- validation %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_g) %>%
  mutate(pred = mu [[1]] + b_i + b_u + b_g)

rmse_iug <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+user+genre",
                                     RMSE = rmse_iug))
```

#### 2.5.3.3.3 Movie + user + date effect

```
b_u <- edx %>%
  left_join(b_i) %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - b_i - mu[[1]]))

b_d <- edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  group_by(date) %>%
  summarize(b_d = mean(rating - b_i - b_u - mu[[1]]))
```

```
predicted_ratings <- validation %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_d) %>%
  mutate(pred = mu [[1]] + b_i + b_u + b_d)

rmse_iud <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+user+date",
                                     RMSE = rmse_iud))
```

All the models with the third bias are created. Let's see the results and select the best one:

```
rmse_results
```

```
##                model      RMSE
## 1             simple 1.0612018
## 2              movie 0.9439087
## 3               user 0.9783360
## 4               year 1.0500259
## 5              genre 1.0184056
## 6               date 1.0575018
## 7         movie+year 0.9439087
## 8        movie+genre 0.9439087
## 9         movie+date 0.9415172
## 10  movie+user+year 0.8650043
## 11 movie+user+genre 0.8649469
## 12  movie+user+date 0.8652511
```

The best model is the movie+user+genre biases effect.

#### 2.5.3.4 Addition of a fourth bias

Let's add a fourth bias to the model and see how is its effect. The model will follow the next equation:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_4 + \epsilon_{u,i}$$

Where $b_i$ is the bias of the movie effect, $b_u$ is the bias of the user effect, $b_g$ is the genre effect bias and $b_4$ is the fourth bias we are going to add.

##### 2.5.3.4.1 Movie + user + genre + year effect

```
b_u <- edx %>%
  left_join(b_i) %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - b_i - mu[[1]]))

b_g <- edx %>%
```

```
  left_join(b_i) %>%
  left_join(b_u) %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - b_i - b_u - mu[[1]]))

b_y <- edx %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_g) %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - b_i - b_u - b_g - mu[[1]]))

predicted_ratings <- validation %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_g) %>%
  left_join(b_y) %>%
  mutate(pred = mu [[1]] + b_i + b_u + b_g + b_y)

rmse_iugy <- RMSE(validation$rating, predicted_ratings$pred)
rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+user+genre+year",
                                     RMSE = rmse_iugy))
```

#### 2.5.3.4.2 Movie + user + genre + date effect

```
b_u <- edx %>%
  left_join(b_i) %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - b_i - mu[[1]]))

b_g <- edx %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - b_i - b_u - mu[[1]]))

b_d <- edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_g) %>%
  group_by(date) %>%
  summarize(b_d = mean(rating - b_i - b_u - b_g - mu[[1]]))

predicted_ratings <- validation %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_g) %>%
  left_join(b_d) %>%
  mutate(pred = mu [[1]] + b_i + b_u + b_g + b_d)
```

```
rmse_iugd <- RMSE(validation$rating, predicted_ratings$pred)

rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+user+genre+date",
                                     RMSE = rmse_iugd))
```

The two models with the fourth bias added are created. Let's see the results to compare them and select the best one:

```
rmse_results
```

```
##                     model      RMSE
## 1                  simple 1.0612018
## 2                   movie 0.9439087
## 3                    user 0.9783360
## 4                    year 1.0500259
## 5                   genre 1.0184056
## 6                    date 1.0575018
## 7              movie+year 0.9439087
## 8             movie+genre 0.9439087
## 9              movie+date 0.9415172
## 10       movie+user+year 0.8650043
## 11      movie+user+genre 0.8649469
## 12       movie+user+date 0.8652511
## 13 movie+user+genre+year 0.8647606
## 14 movie+user+genre+date 0.8648392
```

As we see in the last data frame, the best model is the movie+user+genre+year effect.

**2.5.3.5 Addition of the fifth bias** This is the final step of adding biases to the model. We will add a fifth bias and see if the model improves. The model will follow the next equation:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_y + b_d + \epsilon_{u,i}$$

Where $b_i$ is the bias of the movie effect, $b_u$ is the bias of the user effect, $b_g$ is the genre effect bias, $b_y$ is the bias of the year of the movie and $b_d$ is the date of the timestamp (when was the movie rated) bias.

**2.5.3.5.1 Movie + user + genre + year + date effect**

```
b_u <- edx %>%
  left_join(b_i) %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - b_i - mu[[1]]))

b_g <- edx %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - b_i - b_u - mu[[1]]))
```

```
b_y <- edx %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_g) %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - b_i - b_u - b_g  - mu[[1]]))

b_d <- edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_g) %>%
  left_join(b_y) %>%
  group_by(date) %>%
  summarize(b_d = mean(rating - b_i - b_u - b_g - b_y - mu[[1]]))


predicted_ratings <- validation %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(b_i) %>%
  left_join(b_u) %>%
  left_join(b_g) %>%
  left_join(b_y) %>%
  left_join(b_d) %>%
  mutate(pred = mu [[1]] + b_i + b_u + b_g + b_y + b_d)

rmse_iugyd <- RMSE(validation$rating, predicted_ratings$pred)
rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "movie+user+genre+year+date",
                                     RMSE = rmse_iugyd))
```

Let's show the data frame with all the models and their RMSE:ç

```
rmse_results
```

```
##                          model      RMSE
## 1                       simple 1.0612018
## 2                        movie 0.9439087
## 3                         user 0.9783360
## 4                         year 1.0500259
## 5                        genre 1.0184056
## 6                         date 1.0575018
## 7                   movie+year 0.9439087
## 8                  movie+genre 0.9439087
## 9                   movie+date 0.9415172
## 10            movie+user+year 0.8650043
## 11           movie+user+genre 0.8649469
## 12            movie+user+date 0.8652511
## 13      movie+user+genre+year 0.8647606
## 14      movie+user+genre+date 0.8648392
## 15 movie+user+genre+year+date 0.8645752
```

To select the best model, let's execute the following code:

```
best_model <- rmse_results[which.min(rmse_results$RMSE),]
best_model
```

```
##                            model      RMSE
## 15 movie+user+genre+year+date 0.8645752
```

As seen, the model:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_d + b_y + \epsilon_{u,i}$$

has a good RMSE result, but it is possible to get a better one if we apply the regularization to the model.

### 2.5.4 Regularization of the best model

In this step we are going to take the best model created previously and use the regularization to see if the RMSE improves. We are using regularization to penalize the movies with a few number of ratings. The aim is to find the value of lambda, which is a tuning parameter, that minimizes the RMSE value:

```
lambdas <- seq(4, 6, 0.05)
rmses <- sapply(lambdas, function(l){

  b_u <- edx %>%
    left_join(b_i) %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu[[1]])/(n()+l))

  b_g <- edx %>%
    left_join(b_i) %>%
    left_join(b_u) %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_i - b_u - mu[[1]])/(n()+l))

  b_y <- edx %>%
    left_join(b_i) %>%
    left_join(b_u) %>%
    left_join(b_g) %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - b_i - b_u - b_g  - mu[[1]])/(n()+l))

  b_d <- edx %>%
    mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
    left_join(b_i) %>%
    left_join(b_u) %>%
    left_join(b_g) %>%
    left_join(b_y) %>%
    group_by(date) %>%
    summarize(b_d = sum(rating - b_i - b_u - b_g - b_y - mu[[1]])/(n()+l))


  predicted_ratings <- validation %>%
    mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
    left_join(b_i) %>%
    left_join(b_u) %>%
```

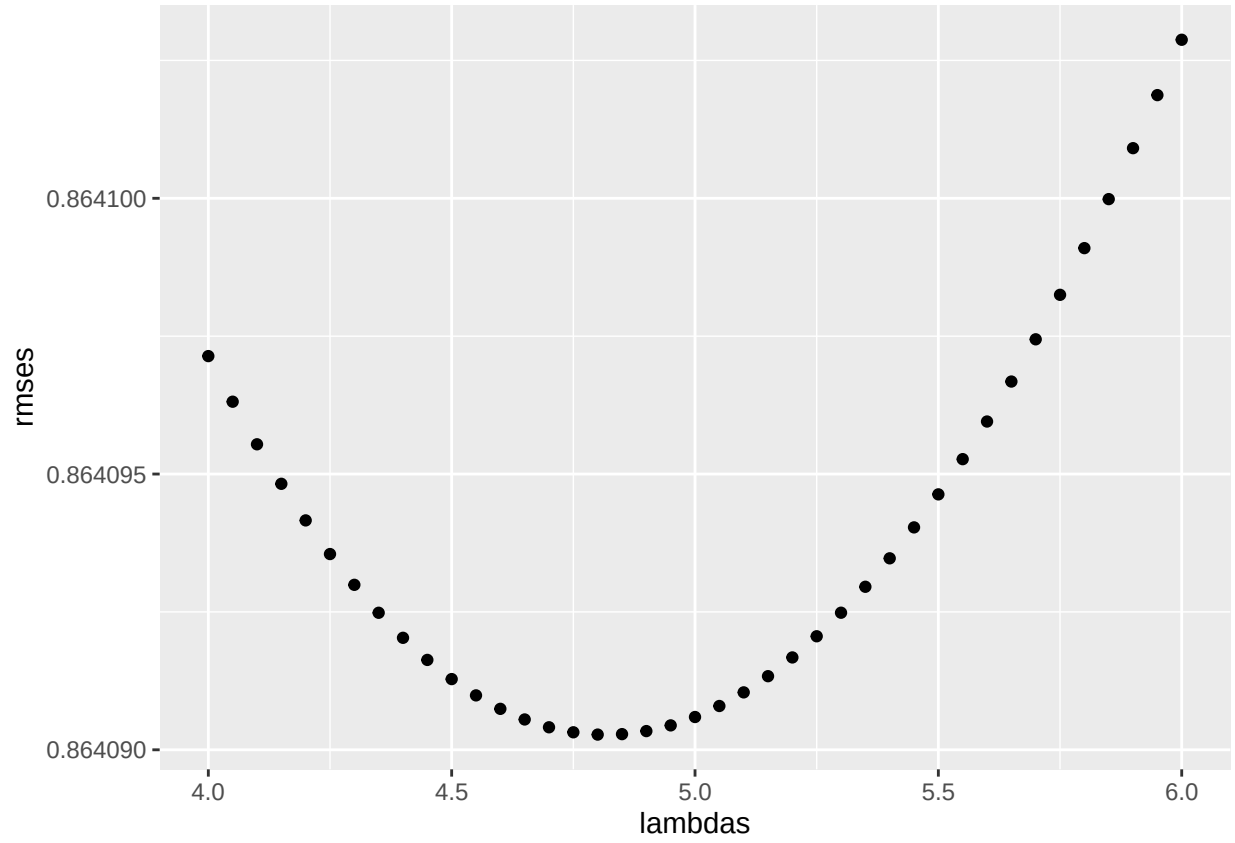
```
    left_join(b_g) %>%
    left_join(b_y) %>%
    left_join(b_d) %>%
    mutate(pred = mu [[1]] + b_i + b_u + b_g + b_y + b_d)

  predicted_ratings$pred <- ifelse(predicted_ratings$pred > 5, 5, predicted_ratings$pred)
  predicted_ratings$pred <- ifelse(predicted_ratings$pred < 0, 0 , predicted_ratings$pred)

  return(RMSE(validation$rating, predicted_ratings$pred))
})
```

Let's plot the values of lambda $\lambda$ to see its behaviour:

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




Finally, the optimal lambda and its RMSE calculated is:

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

```
## [1] 4.8
```

```
rmse_regularized <- min(rmses)
rmse_regularized
```

```
## [1] 0.8640903
```

```
rmse_results <- bind_rows(rmse_results,
                          data.frame(model= "regularized",
                                     RMSE = rmse_regularized))
```

# 3 Results

The final results of the different models, with their RMSE calculated and ordered by the best RMSE are the following ones:

```
rmse_results %>% arrange(RMSE)
```

```
##                          model      RMSE
## 1                  regularized 0.8640903
## 2  movie+user+genre+year+date 0.8645752
## 3       movie+user+genre+year 0.8647606
## 4       movie+user+genre+date 0.8648392
## 5            movie+user+genre 0.8649469
## 6             movie+user+year 0.8650043
## 7             movie+user+date 0.8652511
## 8                  movie+date 0.9415172
## 9                       movie 0.9439087
## 10                 movie+year 0.9439087
## 11                movie+genre 0.9439087
## 12                       user 0.9783360
## 13                      genre 1.0184056
## 14                       year 1.0500259
## 15                       date 1.0575018
## 16                     simple 1.0612018
```

So the final model is:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_y + b_d + \epsilon_{u,i}$$

In which we have to regularize the biases effect using the $\lambda$ that minimizes the RMSE calculation: 4.8

# 4 Conclusions

In this Movielens project we have created a model that predicts the movie rating with an RMSE of 0.8640903. This model takes into account five variables given by the dataset: the movie, the user, the genre of the movie, the year of the movie and the date in which is the movie rated. Each of these variables gives a bias effect on the mean rating of all the dataset, in order to predict a movie rating from a specific user.

A movie rating prediction model is used in some online platforms, such as Netflix, to give some personalized movie reccomendations to their users.

Despite the result of this report (RMSE = 0.8640903) is bettter than the result with the maximum score of the exercise (RMSE = 0.86490), I could have considered other systems of machine learning (Random Forests, for example) to have more predicting models and compare them to try to predict better RMSE results, but due to hardware restrictions, it hasn't been possible.

I also wanted to separate genres combinations of each movie into unique genres and see if the bias of the unique genres could give better results, but it was impossible because when I've tried to separate the string of characters of the genres R crashed.