Data Science - Capstone - Project Movielens

JMMA

Febrary 2025

Contents

1. INTRODUCTION

The project is based on a dataset provided by EDX through https://grouplens.org/datasets/movielens/latest/ We will have to analyze the data using visualization knowledge to make summaries of what we observe.

The objective is to use the edx dataset provided to experiment with multiple parameters and/or use cross-validation to create a machine learning that gives us an RMSE < 0.86490.

Acccording to Mohit Uniyal (https://www.appliedaicourse.com/blog/root-mean-square-error-rmse/)

Root Mean Square Error (RMSE) is a crucial metric for evaluating the accuracy of machine learning models, especially in applications where large errors must be minimized. Its interpretability and sensitivity to significant deviations make it an essential tool for model comparison and improvement. While RMSE provides valuable insights, combining it with other metrics ensures a more comprehensive understanding of model performance. By effectively reducing RMSE, you can enhance predictive accuracy and build reliable models..

I. Loading libraries and the dataset

This loading of the dataset is provided by EDX through https://grouplens.org/datasets/movielens/latest/

```
############################################################
# 1.Create edx and final_holdout_test sets
############################################################

# 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(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(vtable)) install.packages("vtable", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(corrr)) install.packages("corrr", repos = "http://cran.us.r-project.org")
if(!require(ggcorrplot)) install.packages("ggcorrplot", repos = "http://cran.us.r-project.org")
if(!require(factoextra)) install.packages("factoextra", repos = "http://cran.us.r-project.org")
if(!require(recosystem)) install.packages("recosystem", repos = "http://cran.us.r-project.org")

# Libraries required to run the proyect
library(tidyverse)
library(caret)
library(kableExtra)
library(data.table)
library(vtable)
library(ggthemes)
library(corrr)
library(ggcorrplot)
library(factoextra)
library(recosystem)

options(digits = 5)
options(timeout = 120)
```

```
# Download and unzip dataset MovieLens from https://grouplens.org/datasets/movielens/latest/
dl <- "ml-10M100K.zip"
```

```r
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)
```

II. Initial data wrangling

This section of code is proporcion by EDX too.

```r
#Create a data frame ratings with columns (userId, movieId, rating, timestamp) to
#obtain ratings_file and convert (integer, integer, numeric, timestamp)
ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)

colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

#Create a data frame movies with columns (movieId, title, genres) to obtain
#movies_file and convert (movieId to integer)
movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)

colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

#Create data_set movielens with movies are in ratings by movieId
movielens <- left_join(ratings, movies, by = "movieId")
```

III. Creating test and train set (movielens)

The dataset is configure in two sets:

- **edx:** use for training the differents algorithms.
- **final_holdout_test:** use for evaluation RMSE to different algorithms and ensambled.

```r
#Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")

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

```r
temp <- movielens[test_index,]

#Make sure userId and movieId in final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

#Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

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

2. ANALYSIS

First of all is analysis the structure of data for know, what variables are relevants to realize training and evaluation RMSE.

I. Data structure

We will review EDX data frame what find out that columns and data have.

We will use function ktable(), offers a simple and flexible way to convert data frames and matrices into presentable tables, making it an essential tool for R users involved in data analysis, reporting, and documentation.

```r
############################################################
# 2. ANALYSIS
############################################################

############################################################
# I. Data structure
############################################################

# Create a complex HTML table using kableExtra for obtain data structure EDX
kable(head(edx, 5), caption = "Movielens") %>%
  kable_styling(bootstrap_options = c("striped", "hover"), html_font = "Palatino") %>%
  add_header_above(c(" " = 1,"Important data" = 3, " " = 3), color = "#B22222") %>%
  column_spec(2:4, bold = TRUE, color = "#27408B") %>%
  footnote(general = "This table lists the movies.")
```

Table 1: Movielens

	Important data					
	userId	movieId	rating	timestamp	title	genres
1	**1**	**122**	**5**	838985046	Boomerang (1992)	Comedy|Romance
2	**1**	**185**	**5**	838983525	Net, The (1995)	Action|Crime|Thriller
4	**1**	**292**	**5**	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|T
5	**1**	**316**	**5**	838983392	Stargate (1994)	Action|Adventure|Sci-Fi

6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama&#1

Note:
This table lists the movies.

The next step is obtain the types of data. We observe 6 types of variables:

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  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
```

- **userId:** type: Integer. This field is an identifier unique per user.
- **movieId:** type: Integer. This field is a identifier unique per movie.
- **rating:** type: Numeric. This field indicate a rating by one user. Rating have a point scale [0-5] to increments 0.5 points.
- **timestamp:** type: Integer. Indicate how time have passed since rating, expressed in seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.
- **title:** type: Character. This filed is name to film and date to release.
- **genres:** type: Character. This field is genres to film and use a pipe | to separated following genres: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, "no genre listed".

II. Summaries data

We will use summary function to obtain min, 1st quantil, median, mean, 3rd quantil, max to rating and rating_date movie.

Previously need to convert timestamp in the training set EDX and test set final_holdout_test. To achieve this wrangling data modified column timestamp and create new column rating_date and after eraser column timestamp and relocate rating_date after rating.

Other objective wrangling is extract year of the movie to the title and relocate movie_year after title. This new columns will use to elaborate after graphics summaries statistics.

```
############################################################
# II. Summaries structure
############################################################

#Convert timestamp column to datetime format in rating_date column
#in training set edx to calculate statistics
edx <- edx %>%
  mutate(timestamp = as_datetime(timestamp),
         rating_date = make_date(year(timestamp), month(timestamp))) %>%
select(-timestamp) %>%
  relocate(rating_date, .after = rating) %>%
  as.data.table()
```

```r
#Convert timestamp column to datetime format in rating_date column
#in test set final_holdout_test to calculate statistics
final_holdout_test <- final_holdout_test %>%
  mutate(timestamp = as_datetime(timestamp),
         rating_date = make_date(year(timestamp), month(timestamp))) %>%
  select(-timestamp) %>%
  relocate(rating_date, .after = rating) %>%
  as.data.table()

#Extract year to the title and create a new column name movie_year
#in training set edx
year_pattern <- "(?<=\\()\\d{4}(?=\\))"

edx <- edx %>%
  mutate(year_movie = as.Date(str_extract(title, year_pattern), format = "%Y")) %>%
  relocate(year_movie, .after = title) %>%
  as.data.table()

#Extract year to the title and create a new column name movie_year
#in test set final_holdout_test
final_holdout_test <- final_holdout_test %>%
  mutate(year_movie = as.Date(str_extract(title, year_pattern), format = "%Y")) %>%
  relocate(year_movie, .after = title) %>%
  as.data.table()
```

After wrangling data timestamp and year of the movie show in a table summary statistics.

```r
#Calculates the following summary statistics for the data frame
summary_edx <- summary(edx)

#The important statics is rating min, 1st Quantile, median, mean, 3rd Quantil, max
kable(summary_edx[,3:4], caption = "Statics rating") %>%
  kable_styling(bootstrap_options = c("striped", "hover"), html_font = "Palatino") %>%
  row_spec(3:4, bold = TRUE, color = "#00688B") %>%
  footnote(general = "This table lists rating and rating date min, 1st Quantile, median, mean, 3rd Quan
```

Table 2: Statics rating

rating	rating_date
Min. :0.50	Min. :1995-01-01
1st Qu.:3.00	1st Qu.:2000-01-01
Median :4.00	**Median :2002-10-01**
Mean :3.51	**Mean :2002-09-06**
3rd Qu.:4.00	3rd Qu.:2005-09-01
Max. :5.00	Max. :2009-01-01

Note:
This table lists rating and rating date min, 1st Quantile, median, mean, 3rd Quantiles, max.

Obtain of next summary statistics of training set EDX.

- Number of users.

- Number of movies.
- Number of ratings.
- Number of genres.
- Min year of rating movie.
- Max year of rating movie.
- EDX training data contains 10,667 movies with 9,000,055 ratings given by 69,878 users during the period 1995 to 2009.

```
############################################################
# III. Graphics statistics
############################################################

############################################################
# a. Number of ratings by user
############################################################

#Extract the summary_edx range years of movies
year_pattern_summary <- "\\d{4}"
min_rating_movie_year <- str_extract(summary_edx[1,4], year_pattern_summary)
max_rating_movie_year <- str_extract(summary_edx[6,4], year_pattern_summary)

#Extract main genres of movie
genres_pattern <- "([:alpha:]+[-]?[:alpha:]*[:space:]?[:alpha:]*[:space:]?[:alpha:]*)"
edx <- edx %>%
  mutate(real_genres = str_extract(genres, genres_pattern)) %>%
  relocate(real_genres, .after = genres) %>%
  as.data.table()

final_holdout_test <- final_holdout_test %>%
  mutate(real_genres = str_extract(genres, genres_pattern)) %>%
  relocate(real_genres, .after = genres) %>%
  as.data.table()

#Create summarize with metrics of movielens
edx %>%
  summarize('Number Users' = length(unique(userId)),
            'Number Movies' = length(unique(movieId)),
            'Number rating' = length(rating),
            'Number Genres' = length(unique(real_genres)),
            'Min year rating movie' = min_rating_movie_year,
            'Max year rating movie' = max_rating_movie_year) %>%
  kable(caption = "Metric Movielens") %>%
  kable_styling(bootstrap_options = c("striped", "hover"), html_font = "Palatino") %>%
  column_spec(1:6, bold = TRUE, color = "#27408B") %>%
  footnote(general = "This table lists metrics of movielens.")
```

Table 3: Metric Movielens

Number Users	Number Movies	Number rating	Number Genres	Min year rating movie	Max year rating movie
69878	**10677**	**9000055**	**20**	**1995**	**2009**

Note:
This table lists metrics of movielens.

III. Graphics Statistics

The most significant comparisons according to the columns that provide us with the training data to create graphs are:

- Number of ratings by user.
- Number of films with number of ratings.
- Number of film ratings per year of rating.
- Number of film ratings per rating values.
- Number of user ratings per average user ratings.
- Average ratings by genres of film.
- Average ratings by year of film.

a. Number of ratings by user

In this summary statistics that are 69.878 users, giving mean 129 movies ratings by user and 62 median. In the one hand, one user give only 10 ratings movies and in the other hand one user give an awesome rating of 6.616 movies; this cipher is a 1:51 ratio mean/this user.

```
#Create summarize with number of rating by user
ratings_by_user <- edx %>%
  group_by(userId) %>%
  summarize(n_ratings_by_user = n()) %>%
  ungroup()

#mean all number ratings of users
mean_ratings_by_users <- edx %>%
  group_by(userId) %>%
  summarize(n_ratings_by_user = n()) %>%
  pull(n_ratings_by_user) %>%
  mean() %>%
  signif(3)

#median all number ratings of users
median_ratings_by_users <- edx %>%
  group_by(userId) %>%
  summarize(n_ratings_by_user = n()) %>%
  pull(n_ratings_by_user) %>%
  median() %>%
  signif(3)

#Remove row userId because the unique row statistics important is number of ratings by user
ratings_by_user <- ratings_by_user %>%
  mutate(median_ratings_user = median(ratings_by_user$n_ratings_by_user)) %>%
  select(-userId, n_ratings_by_user, median_ratings_user) %>%
  as.data.table()

labs <- data.frame(n_ratings_by_user = 'Number of ratings by user',
                   median_ratings_user = 'Median ratings by user')

#Create a table of statistics training set Edx with mean, sd, min,
#percentil 25, percentil 75, max by number of ratings by user
sumtable(ratings_by_user, labels = labs)
```

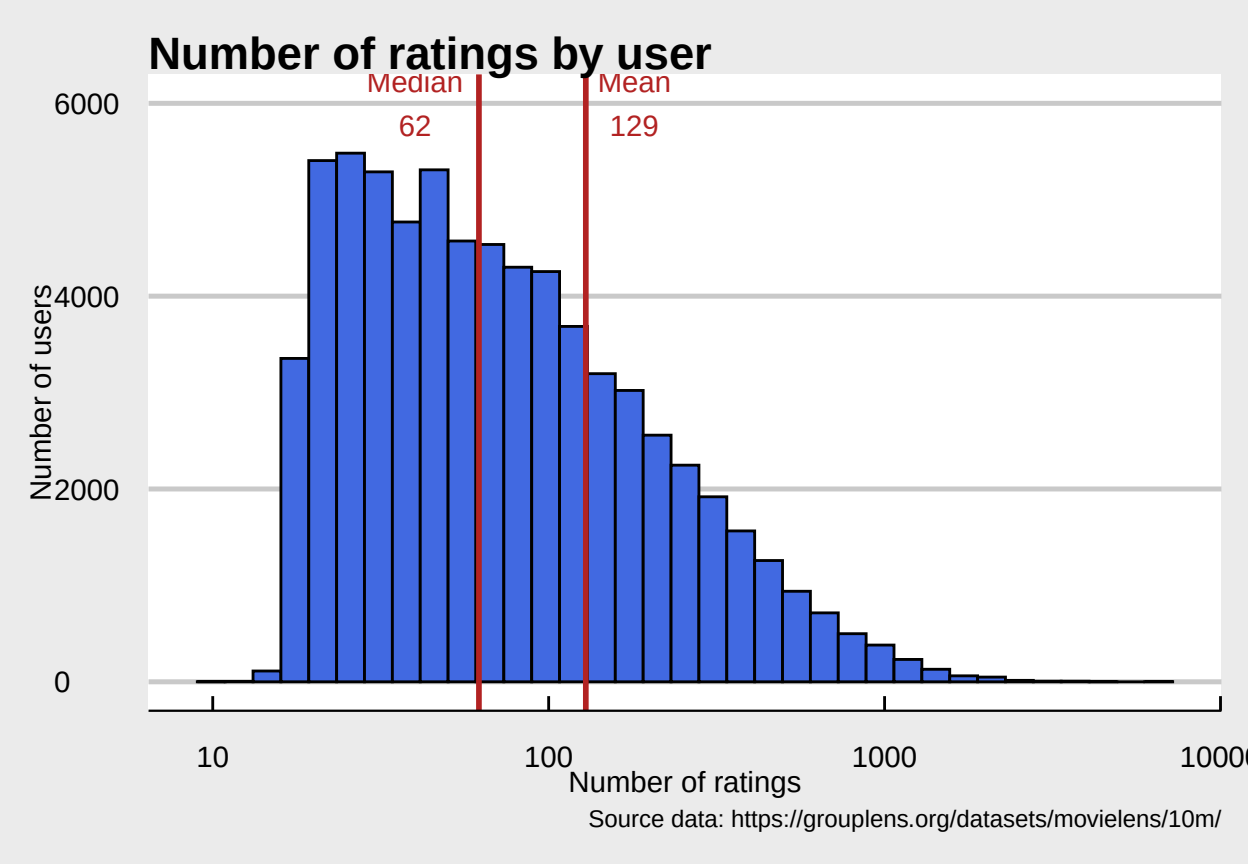
Table 4: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Number of ratings by user	69878	129	195	10	32	141	6616
Median ratings by user	69878	62	0	62	62	62	62

```
#Calculation of users with fewer ratings 30 with a total of 16271 out of 69878 users
length(which(ratings_by_user$n_ratings_by_user<=30))
```

```
## [1] 16271
```

```
#Show in a histogram graphic number user vs rating by user
ggplot(ratings_by_user, aes(x = n_ratings_by_user)) +
  geom_histogram(bins = 35, fill = "#4169E1", color = "black") +
  geom_vline(xintercept=mean_ratings_by_users, linewidth=1, color="#B22222") +
  geom_vline(xintercept=median_ratings_by_users, linewidth=1, color="#B22222") +
  annotate("text", x = 180, y = 6000, label = paste0("Mean\n",mean_ratings_by_users), color="#B22222", s
  annotate("text", x = 40, y = 6000, label = paste0("Median\n",median_ratings_by_users), color="#B22222
  scale_x_log10() +
  labs(title = "Number of ratings by user",
       x = "Number of ratings",
       y = "Number of users",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
  theme(axis.text.x=element_text(size=10, angle=0, face = "bold")) +
  theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))+
  theme_economist_white()
```



This histogram graphic shows a shift to the left with respect to the mean, which is 129 ratings per user and 62 of median. We have 16,271 users with less than 30 ratings out of a total of 69,878. This indicates that 23.78% of users are at a distance of 2.06 times the median and 4.3 times the mean. This means that we have quite a few users with little film experience, which could produce biases in the overall rating, either by giving very low or very high ratings.

b. Number of movies with number of ratings

In this summary statistics that are 10.677 movies, giving mean 843 ratings and 122 median. In the one hand, one movie give only 1 rating and in other hand one movie give an incredible quantity of 31.362 ratings; this cipher is a 1:37 ratio mean/this movie.

```
############################################################
# b. Number of movies with number of ratings
############################################################

#Create summarize with number of movies with number of ratings
ratings_by_movies <- edx %>%
  group_by(movieId) %>%
  summarize(n_ratings_by_movie = n()) %>%
  ungroup()

#mean all number ratings of movies
mean_ratings_by_movies <- edx %>%
  group_by(movieId) %>%
  summarize(n_ratings_by_movie = n()) %>%
```

Table 5: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Number of ratings by movie	10677	843	2238	1	30	565	31362
Median ratings by movie	10677	122	0	122	122	122	122

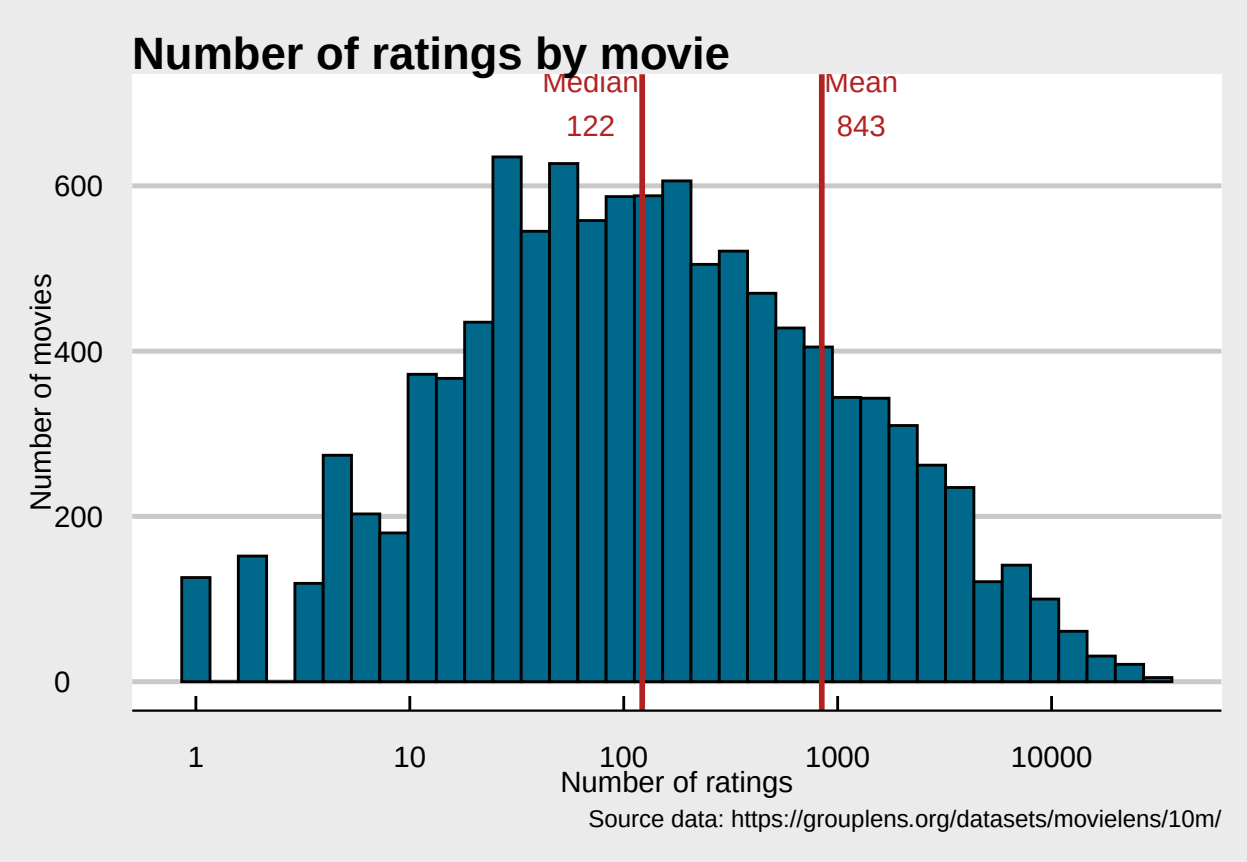
```
  pull(n_ratings_by_movie) %>%
  mean() %>%
  signif(3)

#median all number ratings of movies
median_ratings_by_movies <- edx %>%
  group_by(movieId) %>%
  summarize(n_ratings_by_movie = n()) %>%
  pull(n_ratings_by_movie) %>%
  median() %>%
  signif(3)

#Remove row movieId because the unique row statistics important is number of ratings by user
ratings_by_movies <- ratings_by_movies %>%
  mutate(median_movie_rating = median(ratings_by_movies$n_ratings_by_movie)) %>%
  select(-movieId, n_ratings_by_movie,median_movie_rating) %>%
  as.data.table()

labs <- data.frame(n_ratings_by_movie = 'Number of ratings by movie',
                   median_movie_rating = 'Median ratings by movie')

#Create a table of statistics training set Edx with mean, sd, min,
#percentil 25, percentil 75, max by number of ratings by movie
sumtable(ratings_by_movies, labels = labs)
```

```
#Calculation of movies with fewer ratings 30 with a total of 2688 out of 10677 users
length(which(ratings_by_movies$n_ratings_by_movie<=30))
```

```
## [1] 2688
```

```
#Show in a histogram graphic number movie vs rating by movie
ggplot(ratings_by_movies, aes(x = n_ratings_by_movie)) +
  geom_histogram(bins = 35, fill = "#00688B", color = "black") +
  geom_vline(xintercept=mean_ratings_by_movies, linewidth=1, color="#B22222") +
  geom_vline(xintercept=median_ratings_by_movies, linewidth=1, color="#B22222") +
  annotate("text", x = 1290, y = 700, label = paste0("Mean\n",mean_ratings_by_movies), color="#B22222",
  annotate("text", x = 70, y = 700, label = paste0("Median\n",median_ratings_by_movies), color="#B22222
  scale_x_log10() +
  labs(title = "Number of ratings by movie",
       x = "Number of ratings",
       y = "Number of movies",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
  theme(axis.text.x=element_text(size=10, angle=0, face = "bold")) +
```

```
  theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))+
  theme_economist_white()
```



This histogram shows that there are a few blockbuster films that have many ratings, producing a significant shift to the right of the mean with respect to the median. This means that the mean is 843 when the median is 122, the mean being 6.90 times higher than the most frequent number of ratings. On the other hand, we have 2,688 films (25.77%) with less than 30 ratings, which represents a low sample of ratings that can produce biases lower or higher than normal due to being little known.

c. Number of movies ratings of year-month of rating

This summary statistics we could observe, the interval of years of rating movies is [1995-2009]. In this period, the mean of rating annual is 600.004 ratings movies and the most rating year reach almost double of mean 1.144.349.

```
##########################################################
# c. Number of movies ratings of year-month of rating
##########################################################

#Create summarize with number of movies ratings of year rating
ratings_by_year_rating <- edx %>%
  mutate(year_rating = format(as.Date(rating_date, format="%d %m %Y"), "%Y")) %>%
  mutate(year_rating = as.Date(year_rating, format ="%Y")) %>%
  group_by(year_rating) %>%
  summarize(n_ratings_by_year_rating = n())
```

Table 6: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Number of ratings movies by year	15	600004	337041	2	469530	703317	1144349

```
labs <- data.frame(n_ratings_by_year_rating = 'Number of ratings movies by year')

sumtable(ratings_by_year_rating, labels = labs)
```

```
#Show in scatter a graphic number ratings of year of rating
ggplot(ratings_by_year_rating, aes(x = year_rating, y = n_ratings_by_year_rating)) +
  geom_point(color = "#4169E1") +
  geom_line(aes(year_rating, n_ratings_by_year_rating), col = "black") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  scale_y_continuous(breaks = seq(0, 1200000, 100000), labels = scales::label_number()) +
  labs(title = "Number ratings of year of rating",
       x = "Year of rating",
       y = "Number of ratings",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  geom_smooth(color = "black") +
  theme_economist()
```

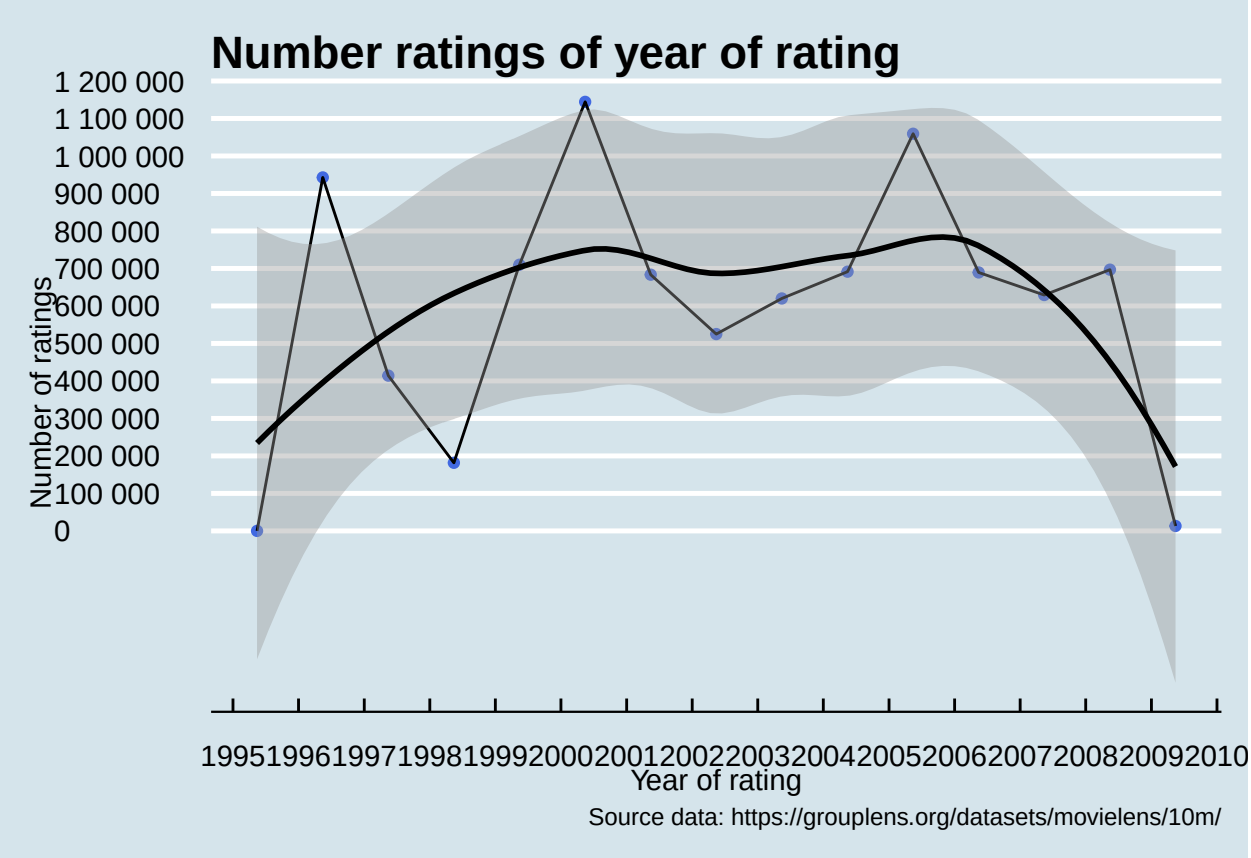


Table 7: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Number of ratings movies by month	157	57325	41199	2	38148	64464	264856

In this scatter plot, we have used an interval time [year] because with an only glance we will obtain which are the better years and worse years of ratings by user.

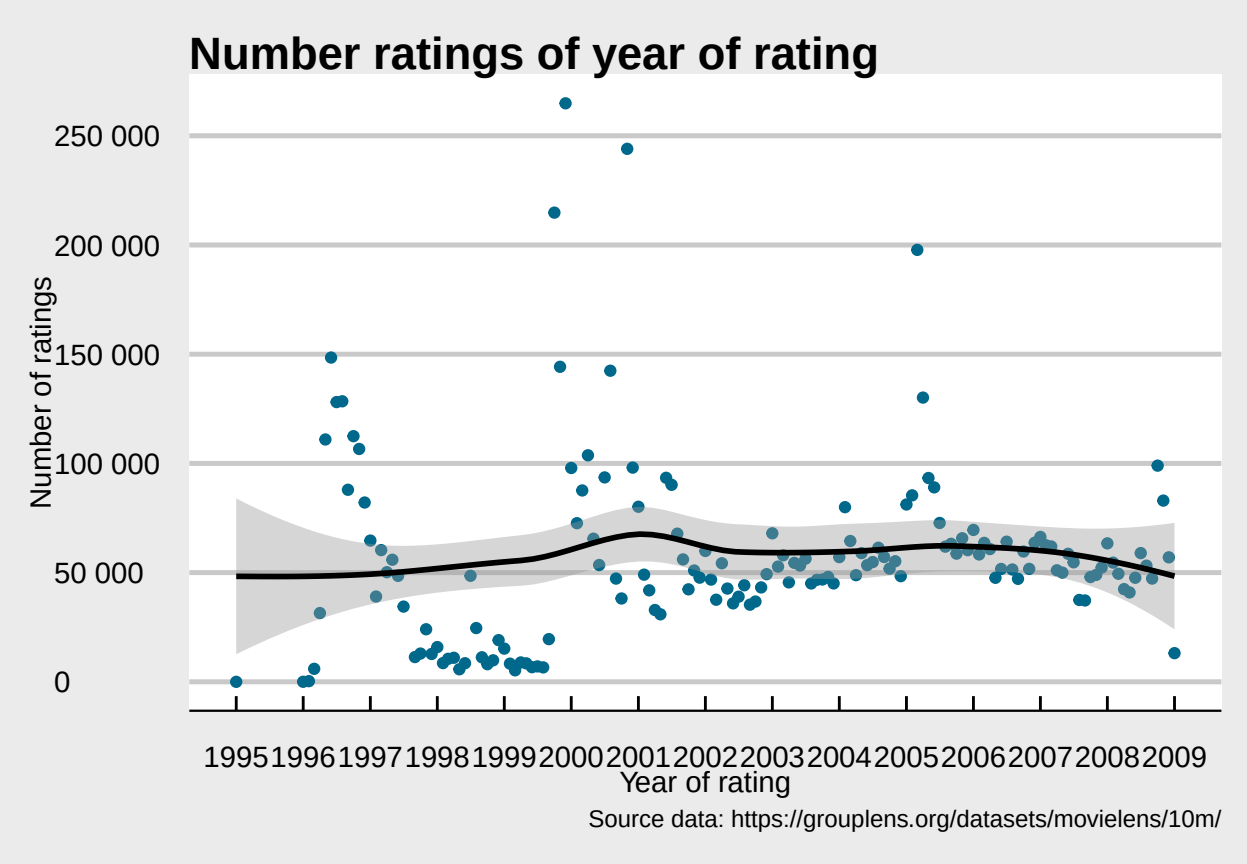
This strategy we will analyzer what years we gain more users in this case (1996, 2000, 2005) and what years lost users (1998). The extreme to interval year in 1995 we have only 2 ratings and 2009 we have only records in January with 13.123 ratings, so these years do not contribute anything to the analysis.

If we were a streaming company this analyzer we will permit find out what is the reason for which gain user in (1996,2000,2005) and lost in (1998). This analysis we will know how to improve our techniques of marketing.

This summary statistics we could observe, the interval of months of rating movies is [157 months, 1995-2009]. In this period, the mean of rating month is 57.235 ratings movies and the most rating moth reach more than quadruple of mean 264.856.

```
#Create summarize with number of movies ratings of month rating
ratings_by_month_rating <- edx %>%
  group_by(rating_date) %>%
  summarize(n_ratings_by_month_rating = n())

labs <- data.frame(n_ratings_by_month_rating = 'Number of ratings movies by month')
sumtable(ratings_by_month_rating, labels = labs)
```

```
#Show in a scatter graphic number ratings of month of rating
ggplot(ratings_by_month_rating, aes(x = rating_date, y = n_ratings_by_month_rating)) +
  geom_point(color = "#00688B") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  scale_y_continuous(breaks = seq(0, 300000, 50000), labels = scales::label_number()) +
  labs(title = "Number ratings of year of rating",
       x = "Year of rating",
       y = "Number of ratings",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  geom_smooth(color = "black") +
  theme_economist_white()
```



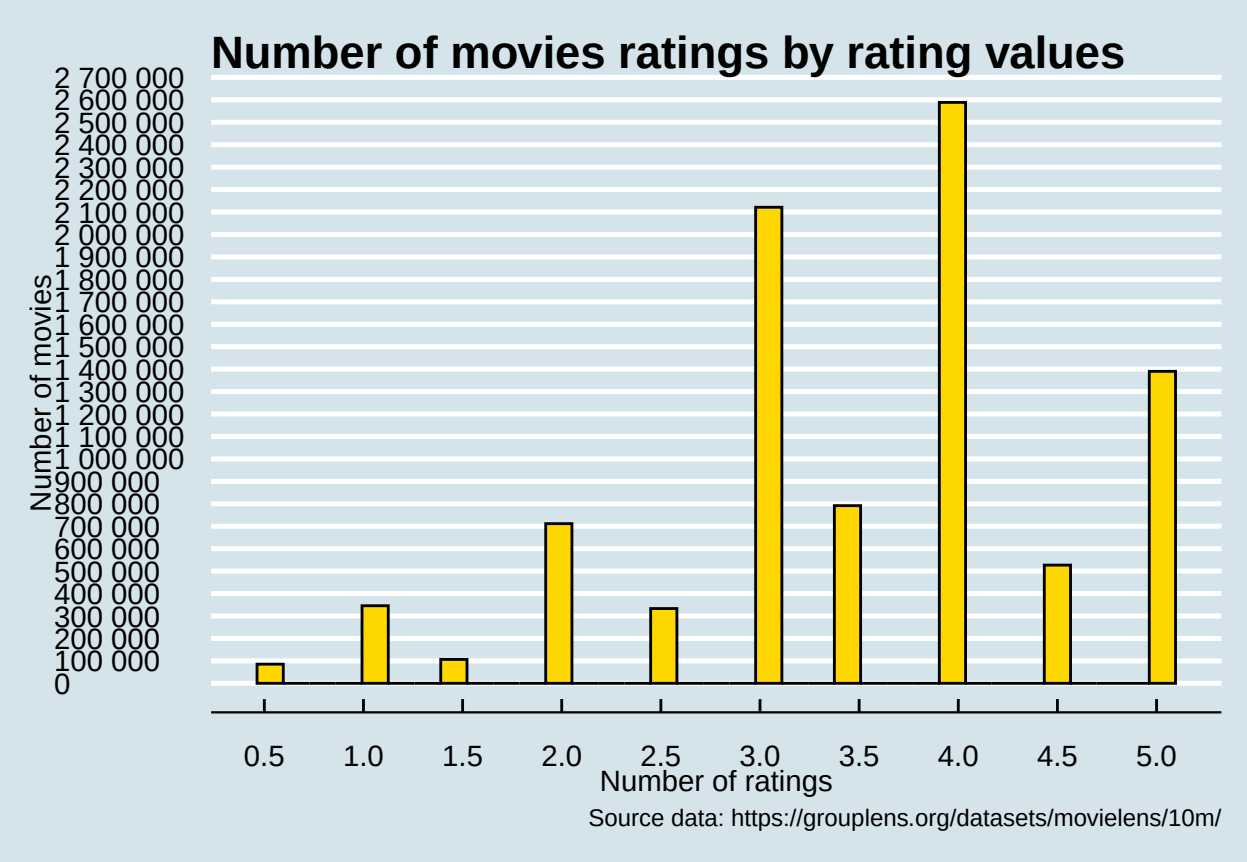
In this scatter plot, we have used an interval time [moth] but is more difficult to see because they are many points to detect what are the best and worst years of rating movie.

In the one hand, we could see the worst years are 1998 and 1999 but when we had used and interval year the worst year is 1998, in the other hand we could see the best years are 1996, 2000 and 2005, it is the same result to observe in the interval year.

d. Number of movies ratings by ratings values

```
############################################################
# d. Number of movies ratings by rating values
############################################################

#Show in a histogram graphic number of movies by rating values
ggplot(edx, aes(x = rating)) +
  geom_histogram(bins = 35, fill = "#FFD700", color = "black") +
  scale_x_continuous(breaks = seq(0,5,0.5)) +
  scale_y_continuous(breaks = seq(0,3000000,100000), labels = scales::label_number()) +
  labs(title = "Number of movies ratings by rating values",
       x = "Number of ratings",
       y = "Number of movies",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
  theme(axis.text.x=element_text(size=10, angle=0, face = "bold")) +
```

```
  theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))+
  theme_economist()
```



In this histogram graphic, we could verified the most frequently value movies ratings (median) is 4 and the mean is 3,51. These data are reveal that catalog movies are good because the major interval ratings values are [3-4], It is equivalent in scale [0-10 points] to 6-8 points.

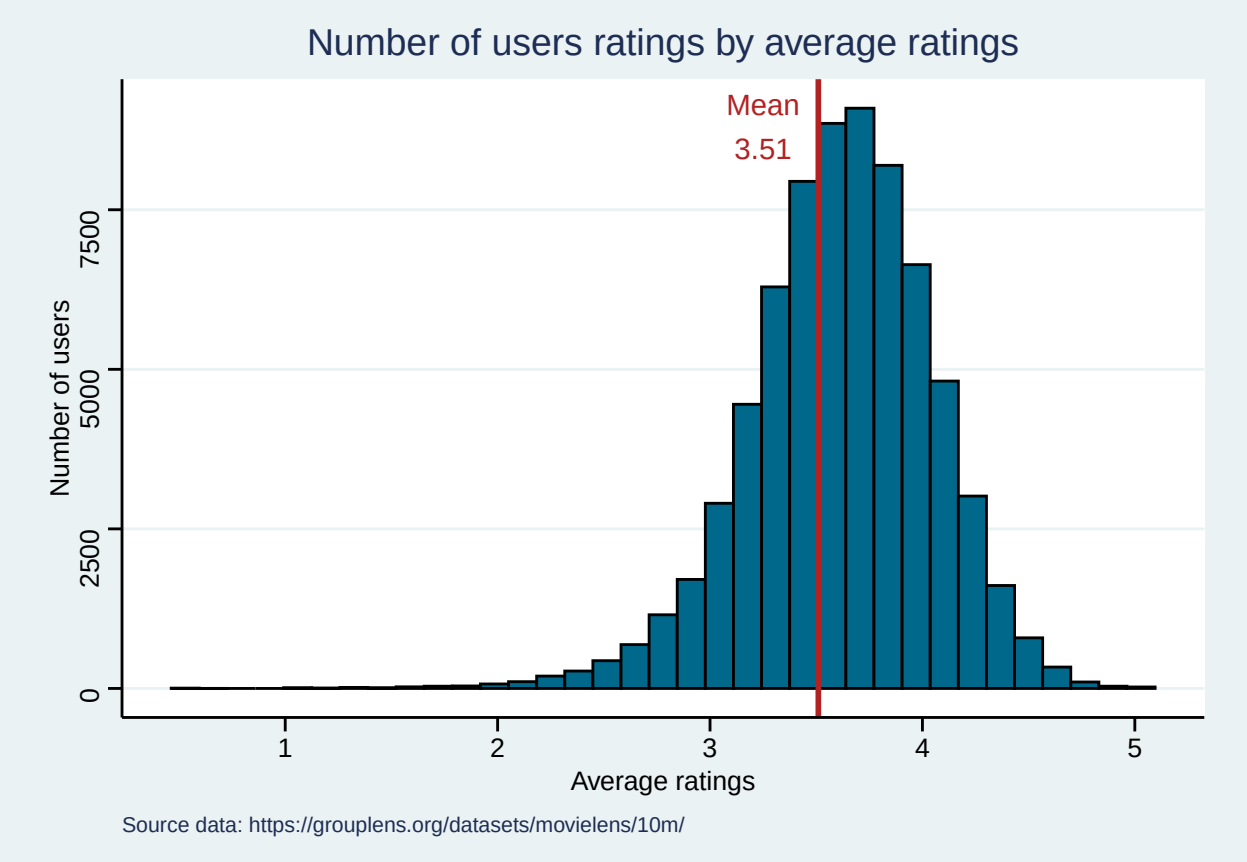
e. Number of users ratings by average ratings

```
##########################################################
# e. Number of users ratings by average ratings
##########################################################

#Create a summarize number of users ratings by average ratings
avg_ratings_by_user <- edx %>%
  group_by(userId) %>%
  summarize(average_rating = mean(rating)) %>%
  select(-userId) %>%
  as.data.table()

labs <- data.frame(average_rating = 'Number of users ratings by average ratings')
sumtable(avg_ratings_by_user, labels = labs)
```

Table 8: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Number of users ratings by average ratings	69878	3.6	0.43	0.5	3.4	3.9	5

```
#mean all the movies
mean_rating <- edx %>%
  pull(rating) %>%
  mean() %>%
  signif(3)

#Show in a histogram graphic number of users ratings by average ratings
ggplot(avg_ratings_by_user, aes(x = average_rating)) +
  geom_histogram(bins = 35, fill = "#00688B", color = "black") +
  geom_vline(xintercept=mean_rating, linewidth=1, color="#B22222") +
  annotate("text", x = 3.25, y = 8800, label = paste0("Mean\n",mean_rating), color="#B22222", size=4 , a
  labs(title = "Number of users ratings by average ratings",
       x = "Average ratings",
       y = "Number of users",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
  theme(axis.text.x=element_text(size=10, angle=0, face = "bold")) +
  theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))+
  theme_stata()
```



In this histogram graphic, we could see the majority rating by users are in the interval [3-4] equivalent to [6-8] points. This data tell us that users like our movies catalog.

f. Average ratings by genres of movie

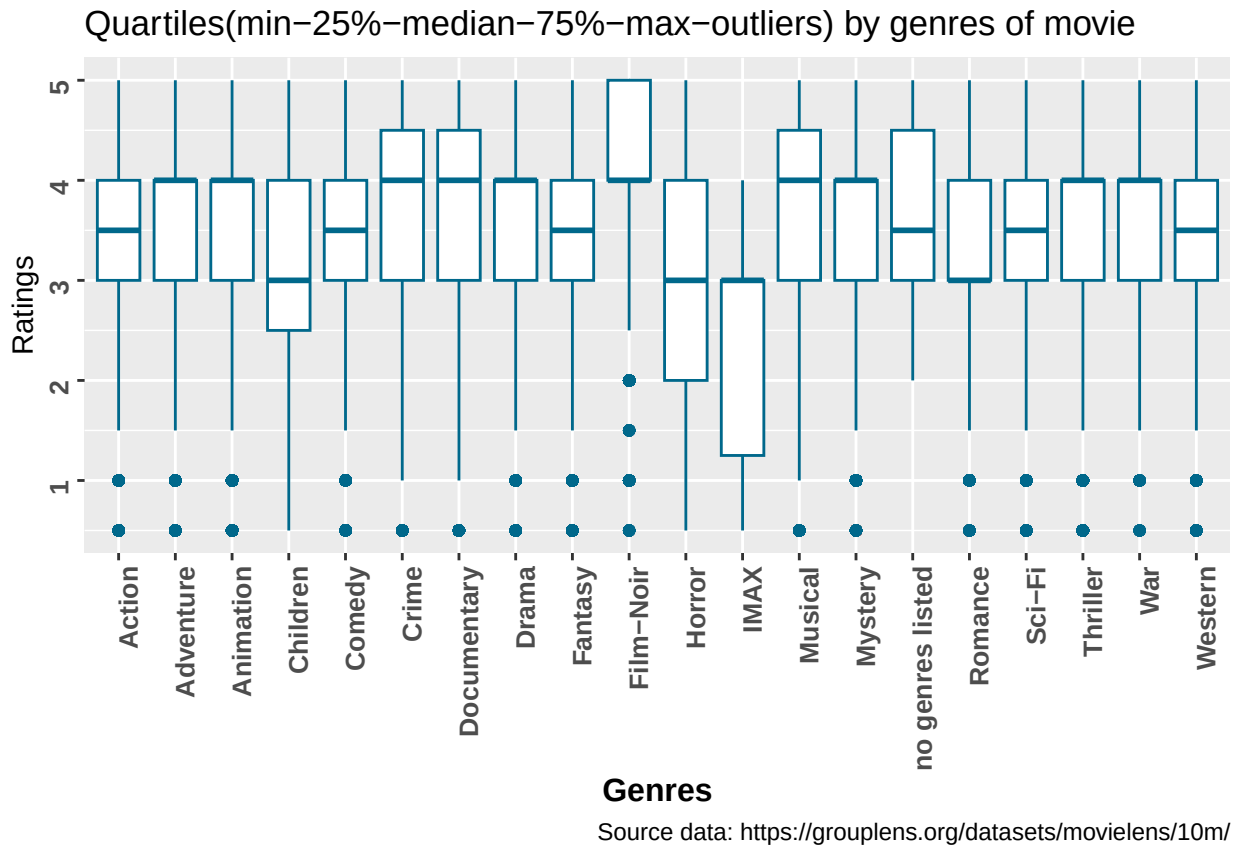
```
############################################################
# f. Average ratings by genres of movie
############################################################

#Create a summarize average ratings by genres of movie
avg_ratings_by_genres <- edx %>%
  group_by(real_genres) %>%
  select(real_genres,rating) %>%
  summarize(average_genres_rating = mean(rating)) %>%
  arrange(desc(average_genres_rating))


avg_ratings_by_genres %>%
  reframe('Genres' = real_genres,
          'Average genres' = average_genres_rating) %>%
  kable(caption = "Statics average rating by genres") %>%
  kable_styling(bootstrap_options = c("striped", "hover"), html_font = "Palatino") %>%
  column_spec(1, bold = TRUE, color = "#4169E1") %>%
  column_spec(2, bold = TRUE, color = "#00688B") %>%
  footnote(general = "Source data: https://grouplens.org/datasets/movielens/10m/")
```

Table 9: Statics average rating by genres

Genres	Average genres
Film-Noir	**4.1459**
Crime	**3.8712**
Documentary	**3.7873**
Mystery	**3.7008**
War	**3.6724**
Drama	**3.6665**
no genres listed	**3.6429**
Musical	**3.6420**
Adventure	**3.5646**
Animation	**3.5492**
Western	**3.5359**
Thriller	**3.5300**
Comedy	**3.4538**
Sci-Fi	**3.4336**
Action	**3.4214**
Fantasy	**3.3100**
Romance	**3.2804**
Children	**3.2476**
Horror	**3.0335**
IMAX	**2.3214**

Note:
Source data: https://grouplens.org/datasets/movielens/10m/

```
#show in a plot ratings by genres of movie
  ggplot(edx, aes(real_genres, rating)) +
  geom_boxplot(col = "#00688B") +
  labs(title = "Quartiles(min-25%-median-75%-max-outliers) by genres of movie",
       x = "Genres",
       y = "Ratings",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
  theme(axis.text.x=element_text(size=10, angle=90, hjust=1, face = "bold")) +
  theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))
```

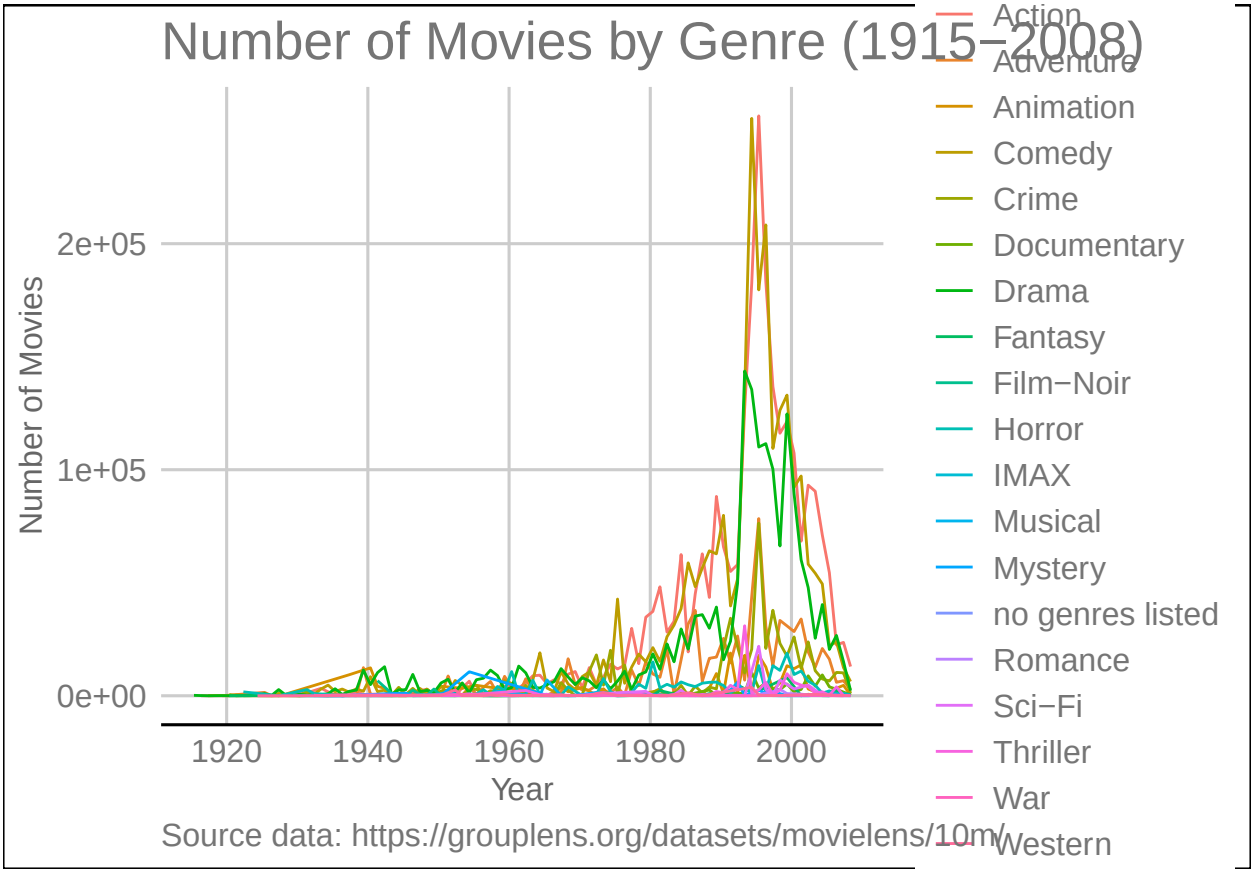


This statics say that top 3 genres ratings are Film-Noir, Crime and Documentary

g. Number of movies by genre year

```
############################################################
# g. Number of Movies by genre year
############################################################

#create a columns with genres that permit create a density plot
edx <- edx %>% mutate(
      'Action' = case_when(edx$real_genres == 'Action' ~ 1, .default = as.integer(0)),
      'Adventure' = case_when(edx$real_genres == 'Adventure' ~ 1, .default = as.integer(0)),
      'Animation' = case_when(edx$real_genres == 'Animation' ~ 1, .default = as.integer(0)),
      'Children' = case_when(edx$real_genres == 'Children' ~ 1, .default = as.integer(0)),
      'Comedy' = case_when(edx$real_genres == 'Comedy' ~ 1, .default = as.integer(0)),
      'Crime' = case_when(edx$real_genres == 'Crime' ~ 1, .default = as.integer(0)),
      'Documentary' = case_when(edx$real_genres == 'Documentary' ~ 1, .default = as.integer(0)),
      'Drama' = case_when(edx$real_genres == 'Drama' ~ 1, .default = as.integer(0)),
      'Fantasy' = case_when(edx$real_genres == 'Fantasy' ~ 1, .default = as.integer(0)),
      'Film-Noir' = case_when(edx$real_genres == 'Film-Noir' ~ 1, .default = as.integer(0)),
      'Horror' = case_when(edx$real_genres == 'Horror' ~ 1, .default = as.integer(0)),
      'IMAX' = case_when(edx$real_genres == 'IMAX' ~ 1, .default = as.integer(0)),
      'Musical' = case_when(edx$real_genres == 'Musical' ~ 1, .default = as.integer(0)),
      'Mystery' = case_when(edx$real_genres == 'Mystery' ~ 1, .default = as.integer(0)),
      'no genres listed' = case_when(edx$real_genres == 'no genres listed' ~ 1, .default = as.integer(0
```

```
      'Romance' = case_when(edx$real_genres == 'Romance' ~ 1, .default = as.integer(0)),
      'Sci-Fi' = case_when(edx$real_genres == 'Sci-Fi' ~ 1, .default = as.integer(0)),
      'Thriller' = case_when(edx$real_genres == 'Thriller' ~ 1, .default = as.integer(0)),
      'War' = case_when(edx$real_genres == 'War' ~ 1, .default = as.integer(0)),
      'Western' = case_when(edx$real_genres == 'Western' ~ 1, .default = as.integer(0))) %>%
  relocate('Action','Adventure','Animation','Children','Comedy','Crime','Documentary','Drama','Fantasy'
  as.data.table()

#Create a filter to view in a density plot with title, year movie and genre
filtered_title_year_genre <- edx %>%
  pivot_longer(c(Action, Adventure, Animation, Comedy, Crime, Documentary, Drama, Fantasy, `Film-Noir`,
  filter(value == "1") %>%
  select(title, year_movie, genre)

#show in a density plot number of movies by genre
ggplot(filtered_title_year_genre, aes(x = year_movie, color = genre)) +
  geom_line(stat = "count", aes(group = genre)) +
  labs(title = "Number of Movies by Genre (1915-2008)",
    x = "Year",
    y = "Number of Movies",
    caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
  theme(axis.text.x=element_text(size=10, angle=0, face = "bold")) +
  theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))+
  theme_gdocs() +
  scale_color_discrete(name = "Genre")
```



In this density graphic, we have used 20 total genres (action, adventure, animation, children, comedy, crime, documentary, drama, fantasy, film-noir, horror, imax, musical, mystery, no genres listed, romance, sci-fi, thriller, war, western).

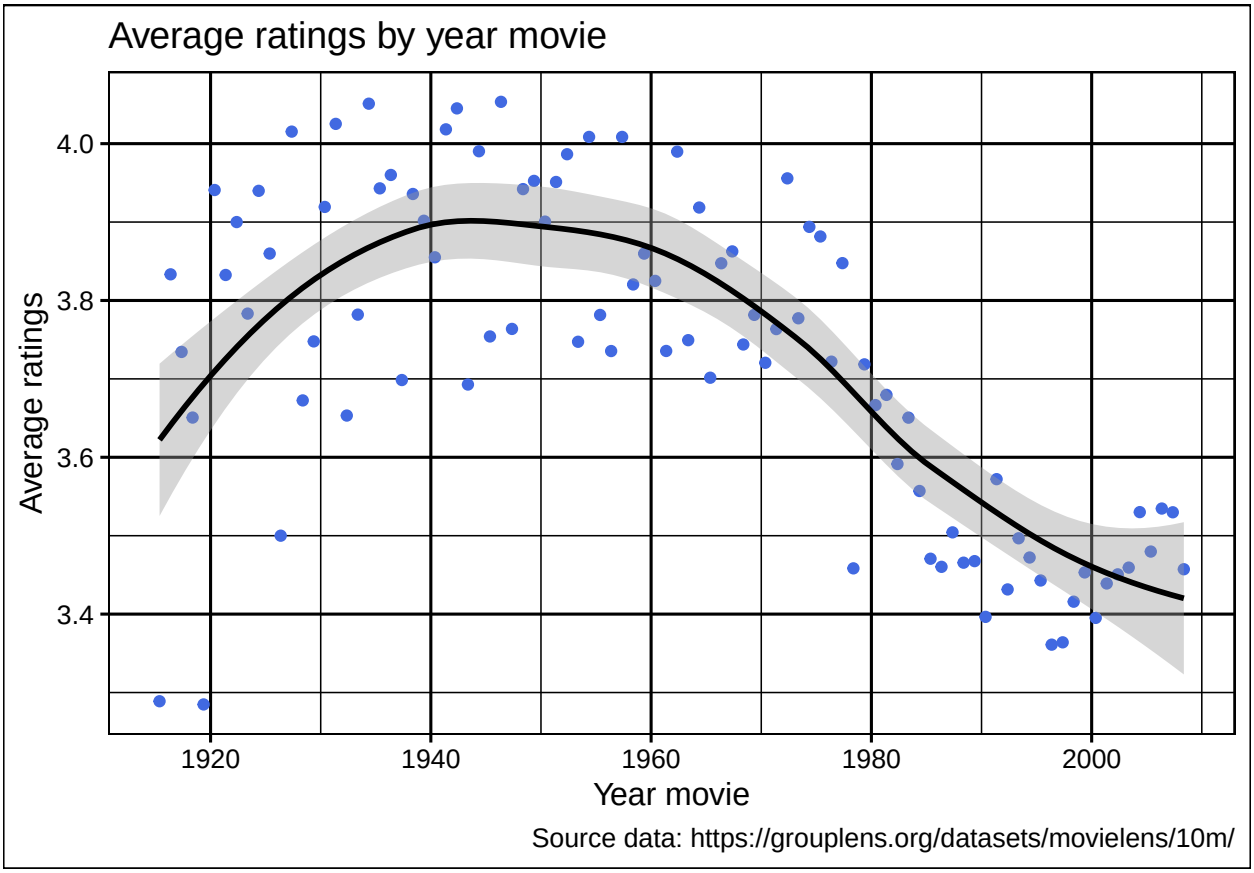
We could observe a long to 1915 -1980 all genres are viewed in the same way. At the beginning of 1980s have changed trend and began to rise action and comedy movies until 2000. At the middle 1990s also have changed trend to rise drama movies until 2000.

This pattern, we will permit design a strategy in our catalog of movies to decide what this type of genres offer our users.

h. Average ratings by year of movie

```
##########################################################
# h. Average ratings by year movie
##########################################################

#Create summarize average rating by year movie
avg_rating_by_year_movie <- edx %>%
  group_by(year_movie) %>%
  summarize (average_year_movie = mean(rating))

#Show in a scatter graphic average ratings by year movie
ggplot(avg_rating_by_year_movie, aes(x = year_movie, y = average_year_movie)) +
  geom_point(color = "#4169E1") +
  labs(title = "Average ratings by year movie",
```

```
      x = "Year movie",
      y = "Average ratings",
      caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  geom_smooth(color = "black") +
  theme_foundation()
```

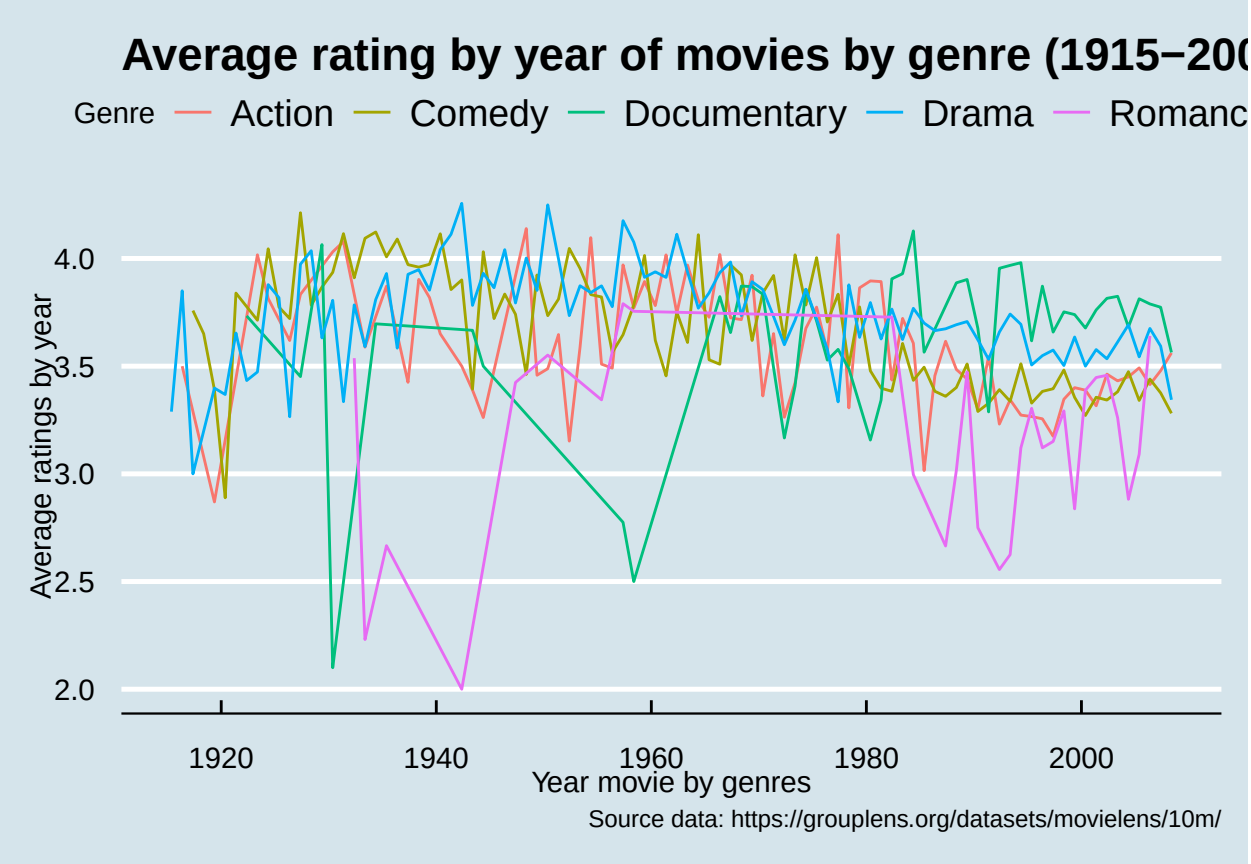


In this scatter plot, We could show that the movies with best values of rating are between 1940s to 1960s, which gives us the clue that our users are a senior people or retirees who have more time to watch movies.

i. Average rating by year of movies by genre

```
##########################################################
# i. Average rating by year of movies by genre
##########################################################

#Create summarize average rating by year movie with five most important genres
avg_rating_by_year_movie_genres <- edx %>%
  filter(real_genres == "Action" | real_genres == "Comedy" | real_genres == "Documentary" | real_genres
  group_by(year_movie,real_genres) %>%
  summarize (average_year_movie_genres = mean(rating))

#show in a density plot average rating by year of movies with five most important genres
ggplot(avg_rating_by_year_movie_genres, aes(x = year_movie, y = average_year_movie_genres, color = real
  geom_line(aes(x = year_movie, y = average_year_movie_genres )) +
```

```
labs(title = "Average rating by year of movies by genre (1915-2008)",
     x = "Year movie by genres",
     y = "Average ratings by year",
     caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
theme(axis.text.x=element_text(size=10, angle=0, face = "bold")) +
theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))+
theme_economist() +
scale_color_discrete(name = "Genre")
```



In this density graphic, We figure out a crisis documentary around the 1930s and the 1960s. The documentary gain most relevance from 1980s to 2000s.

Also we analyze a crisis of romance in the years of World War II, so is normal in war nobody see romances in this situation. Also romance genre have a continues crisis between 1980s to 2000s.

We will observe the most stable genres a long the time are drama and comedy.

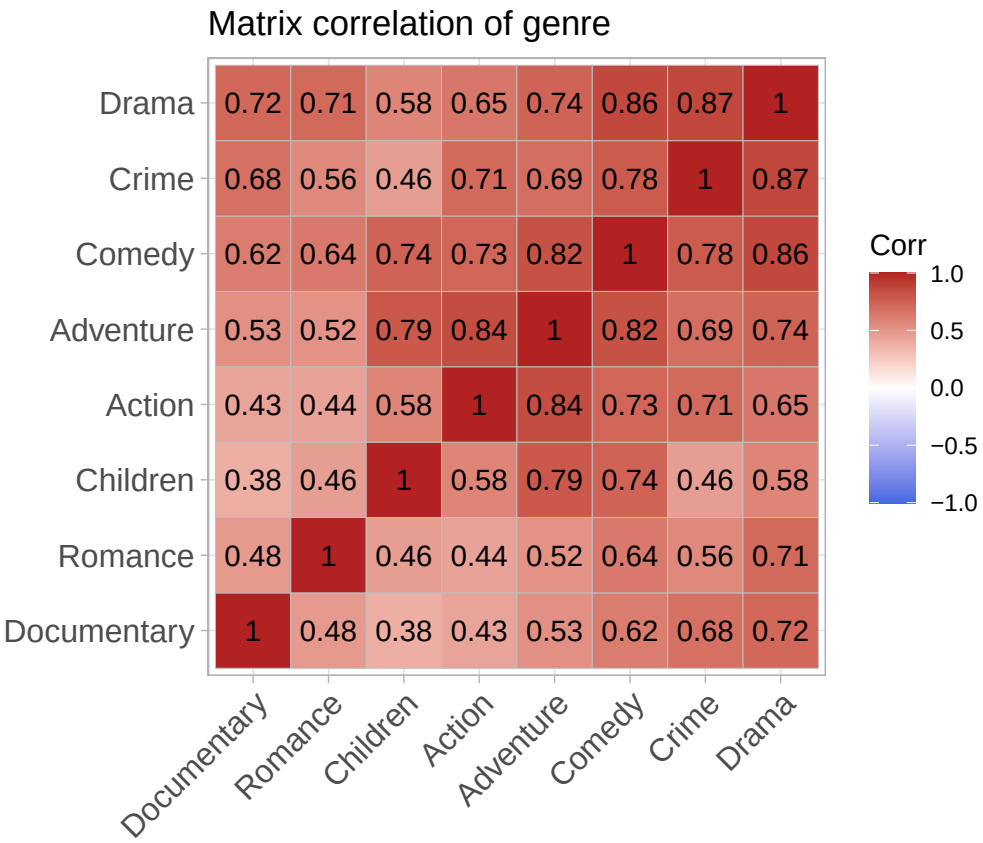
3. PCA (principal component analysis)

It is a statistical approach that can be used to analyze high-dimensional data and capture the most important information from it. This is done by transforming the original data into a lower-dimensional space and grouping the highly correlated variables.

```r
#create a data for realize a PCA study with 8 genres with the users above the median 62 rating movies
PCA <- edx %>%
  group_by(userId) %>%
  filter(n() >= 62) %>%
  select(Action,Adventure, Children, Comedy, Crime, Documentary, Drama, Romance) %>%
  summarize(Action = sum(Action),
            Adventure = sum(Adventure),
            Children = sum(Children),
            Comedy = sum(Comedy),
            Crime = sum(Crime),
            Documentary = sum(Documentary),
            Drama = sum(Drama),
            Romance = sum (Romance)) %>%
  select(Action,Adventure, Children, Comedy, Crime, Documentary, Drama, Romance)

#check is not null values in PCA data
colSums(is.na(PCA))
```

```
##      Action   Adventure    Children      Comedy       Crime Documentary
##           0           0           0           0           0           0
##       Drama     Romance
##           0           0
```

```r
#normalization using function scale
data_normalized <- scale(PCA)

#calculate correlation of matrix
corr_matrix_genres <- cor(data_normalized)

#Correlation plot
ggcorrplot(corr_matrix_genres,
           method = "square",
           hc.order = TRUE,
           type = "full",
           lab = TRUE,
           title = "Matrix correlation of genre",
           ggtheme = ggplot2::theme_light(),
           colors = c("#4169E1", "white", "#B22222"))
```



The result of the correlation matrix can be interpreted as follows:

1. The higher the value, the more positively correlated the two variables are.
2. The closer the value is to -1, the more negatively correlated they are.

```
#summary importance of components
data.pca <- princomp(corr_matrix_genres)
summary(data.pca)
```

```
## Importance of components:
##                           Comp.1  Comp.2  Comp.3  Comp.4   Comp.5    Comp.6
## Standard deviation     0.30867 0.24115 0.17743 0.14598 0.063062 0.0411938
## Proportion of Variance 0.44837 0.27367 0.14815 0.10028 0.018714 0.0079855
## Cumulative Proportion  0.44837 0.72204 0.87019 0.97047 0.989183 0.9971688
##                           Comp.7     Comp.8
## Standard deviation     0.0245281 3.3553e-09
## Proportion of Variance 0.0028312 5.2979e-17
## Cumulative Proportion  1.0000000 1.0000e+00
```

We observe that eight principal components have been generated (Comp.1 to Comp.8), which also correspond to the number of variables in the data (genre). Each component explains a percentage of the total variance in the data set.

In the Cumulative Proportion section, the first principal component explains almost 44% of the total variance. This implies that almost half of the data in the set of 8 variables can be represented by the first principal

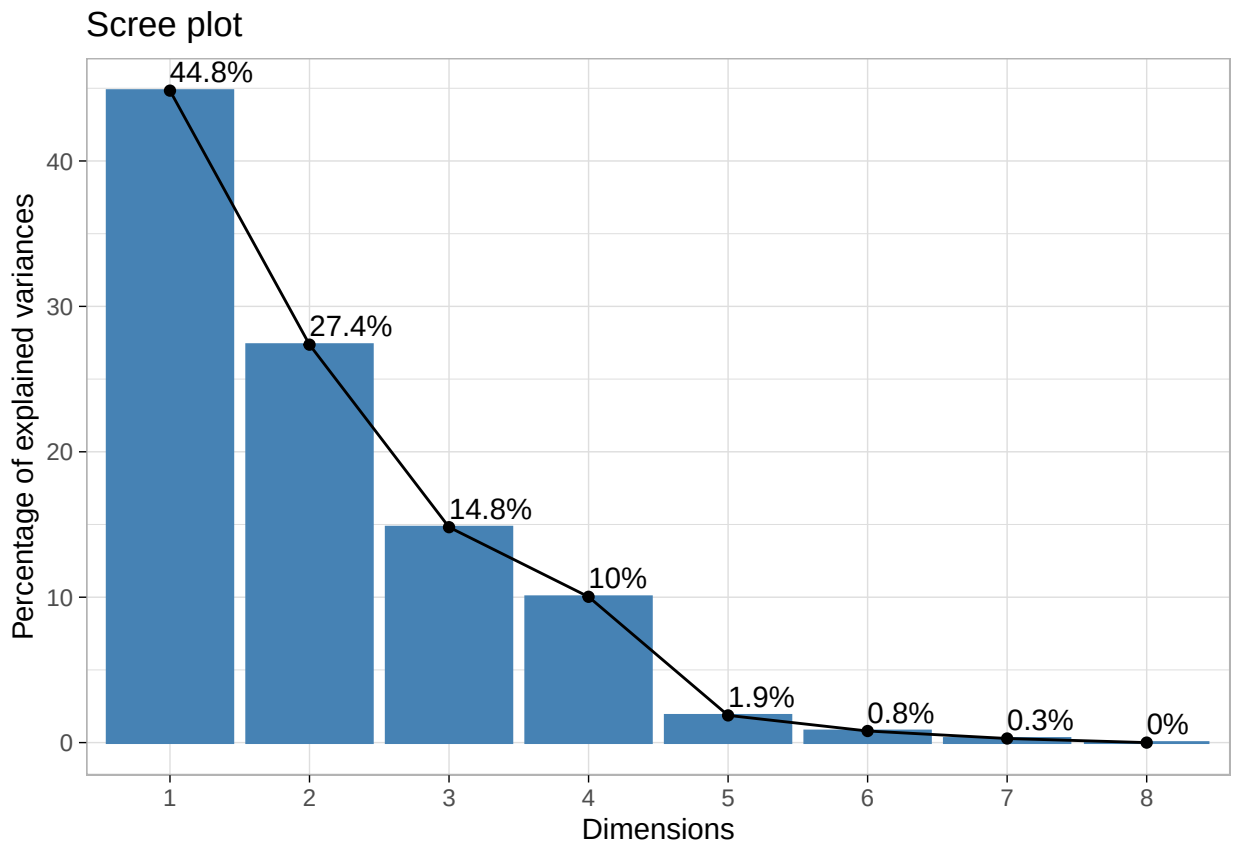
component alone. The second one explains 27.36% of the total variance. The cumulative proportion of Comp.1 and Comp.2 explains almost 72% of the total variance. This means that the first two principal components can accurately represent the data.

```
#Proportion acumulate Comp1. Comp2.
data.pca$loadings[, 1:2]
```

```
##                Comp.1   Comp.2
## Action        0.40454  0.42727
## Adventure     0.43228  0.24654
## Children      0.54179 -0.18565
## Comedy        0.15888  0.27029
## Crime        -0.15255  0.59559
## Documentary -0.45855  0.30896
## Drama        -0.16019  0.36376
## Romance      -0.26740 -0.25811
```

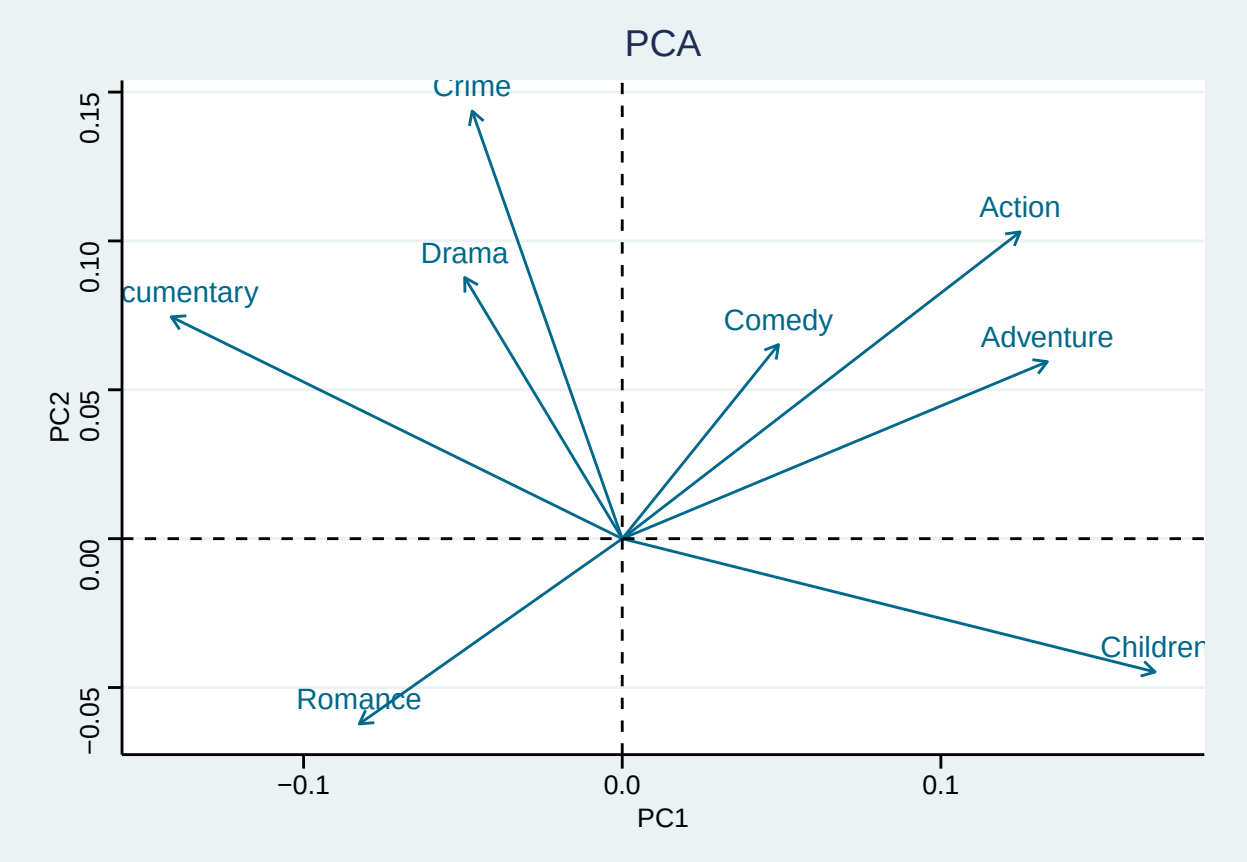
In the first place, that Comp1 could be a group of user (children) who watch children and adventure genre. In the second place, that Comp2 could be a group of user (adult) who watch crime and action genre.

```
#Scree plot
fviz_eig(data.pca,
         addlabels = TRUE,
         ggtheme = ggplot2::theme_light())
```



This plot shows the eigenvalues in a descending curve, from highest to lowest. The first three components can be considered the most significant, since they contain almost 87% of the total information in the data.

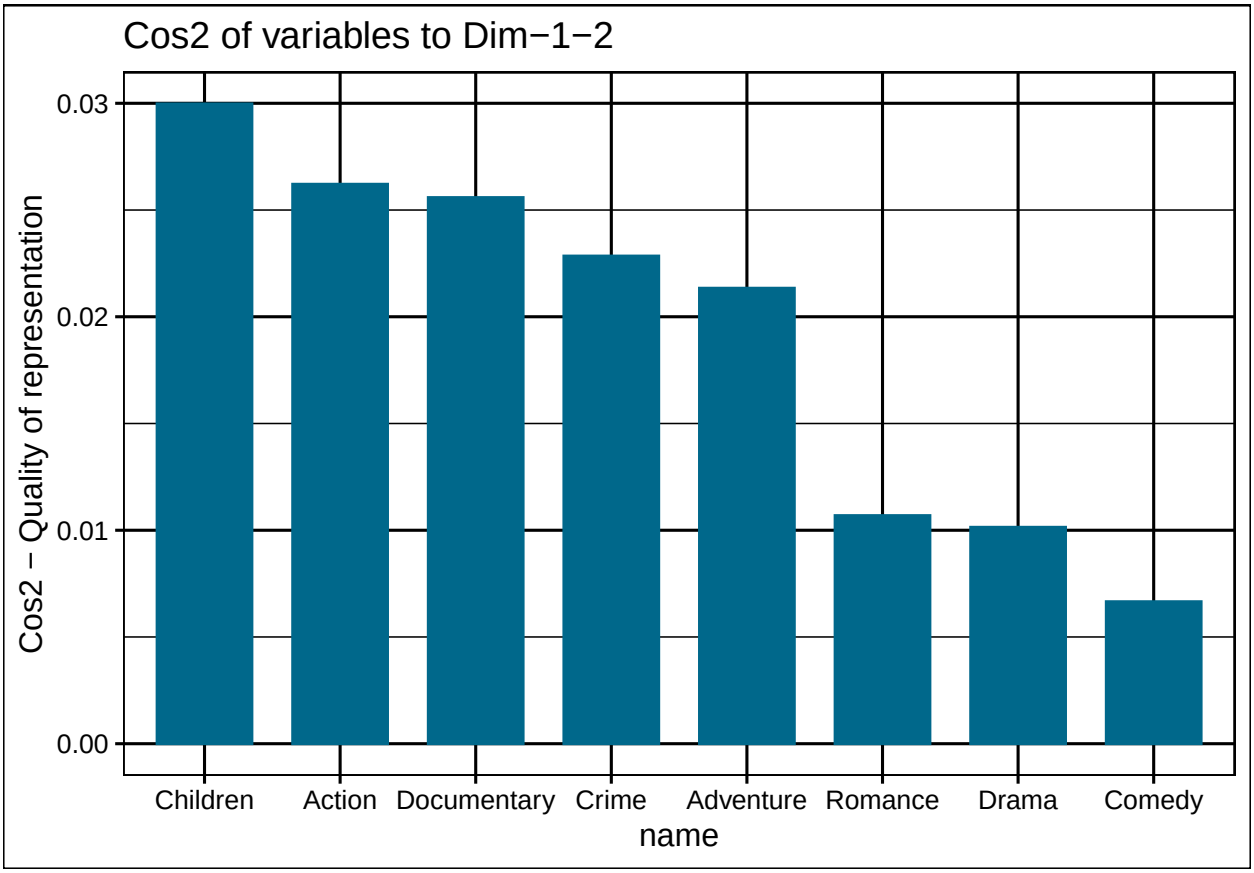
```
#Graphic pca var
fviz_pca_var(data.pca, col.var = "#00688B")+
  labs(title ="PCA", x = "PC1", y = "PC2") +
  theme_stata()
```



With the biplot, it is possible to visualize the similarities and dissimilarities of the samples, and it also shows the impact of each attribute on each of the principal components.

First, all the variables that are grouped together are positively correlated with each other, and this case, for example, action, adventure and action, which have a positive correlation with each other. This result is surprising because they have the highest values in the loading matrix with respect to the first principal component. So, the greater the distance between the variable and the origin, the better represented that variable is. Finally, the negatively correlated variables are shown on the opposite sides of the origin of the biplot.

```
#Scree plot
fviz_cos2(data.pca,
          addlabels = TRUE,
          choice = "var",
          fill = "#00688B",
          color = "#00688B",
          axes = 1:2) +
     theme_foundation()
```



The objective is to determine to what extent each variable is represented in a given component. This quality of representation is called Cos2, it corresponds to the squared cosine and is computed with the function fviz_cos2.

A low value means that the variable is not perfectly represented by that component. On the other hand, a high value means a good representation of the variable in that component.

In the following graphic, children, action, documentary, and crime are the four variables with the highest cos2, so they contribute the most to PC1 and PC2.

4. MODELING RESULTS

We will use root mean squared error (RMSE) as our loss function. We can interpret RMSE similar to standard deviation. N is the number of user-movie combinations, $\mathbf{y_{u,i}}$ is the rating for movie $\mathbf{i}$ by user , and $\mathbf{\hat{y}_{u,i}}$ is our prediction, then RMSE is defined as follows

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

I. A first model

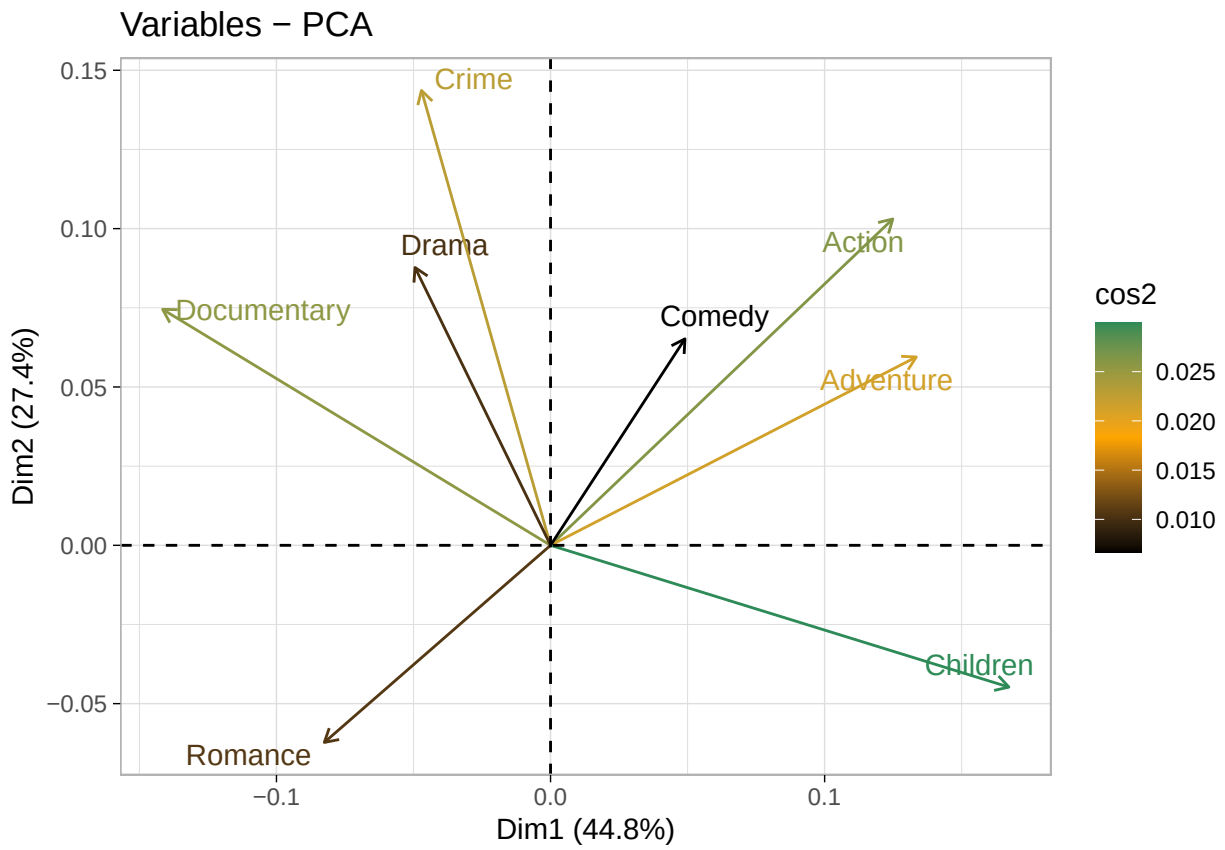
This model assumes the same rating for all movies and users with all the differences explained by random variation. The model is as follows:

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

Where, μ represents the true rating for all movies and users ϵ and represents independent errors sampled from the same distribution centered at zero.

We know that the estimate that minimizes the RMSE is the least squares estimate of μ and, in this case, is the average of all ratings. Suppose $\hat{\mu}$ represents the average of all ratings and we use $\hat{\mu}$ to predict all the unknown movie ratings. We obtain $\hat{\mu}$ and RMSE as shown below. That any number other than $\hat{\mu}$ would result into a higher RMSE.

```
#Graphic pca var
fviz_pca_var(data.pca, col.var = "cos2",
             gradient.cols = c("black", "#FFA500", "#2E8B57"),
             repel = TRUE,
      ggtheme = ggplot2::theme_light())
```



High cos2 attributes are colored sea green: children, action, documentary. Medium cos2 attributes are colored orange: adventure, crime. Finally, low cos2 attributes are colored black: comedy.

```
#############################################################
# 4. MODELING RESULTS
#############################################################

#############################################################
# I. A First model
#############################################################

#RMSE formula
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

```r
#Mean movies rating (mu) to training set (edx) 3.51
edx_mu_hat <- mean(edx$rating)

#Naive RMSE obtain use test set (final_holdout_test) rating and mean (mu) to training set (edx)
result_nai <- RMSE(final_holdout_test$rating, edx_mu_hat)
result_naive <- tibble(Model = "Naive RMSE", RMSE = result_nai)
  result_naive %>%
  knitr::kable()
```

Model	RMSE
Naive RMSE	1.0612

```r
#Any number other than ^mu_hat would result into a higher RMSE
predictions <- rep(2.5, nrow(final_holdout_test))
result_dis_mu <- RMSE(final_holdout_test$rating, predictions)

result_distinct_mu <- tibble(Model = "Distinct value mu_hat RMSE", RMSE = result_dis_mu)
  result_distinct_mu %>% kable()
```

Model	RMSE
Distinct value mu_hat RMSE	1.4664

II. Modeling movie effects

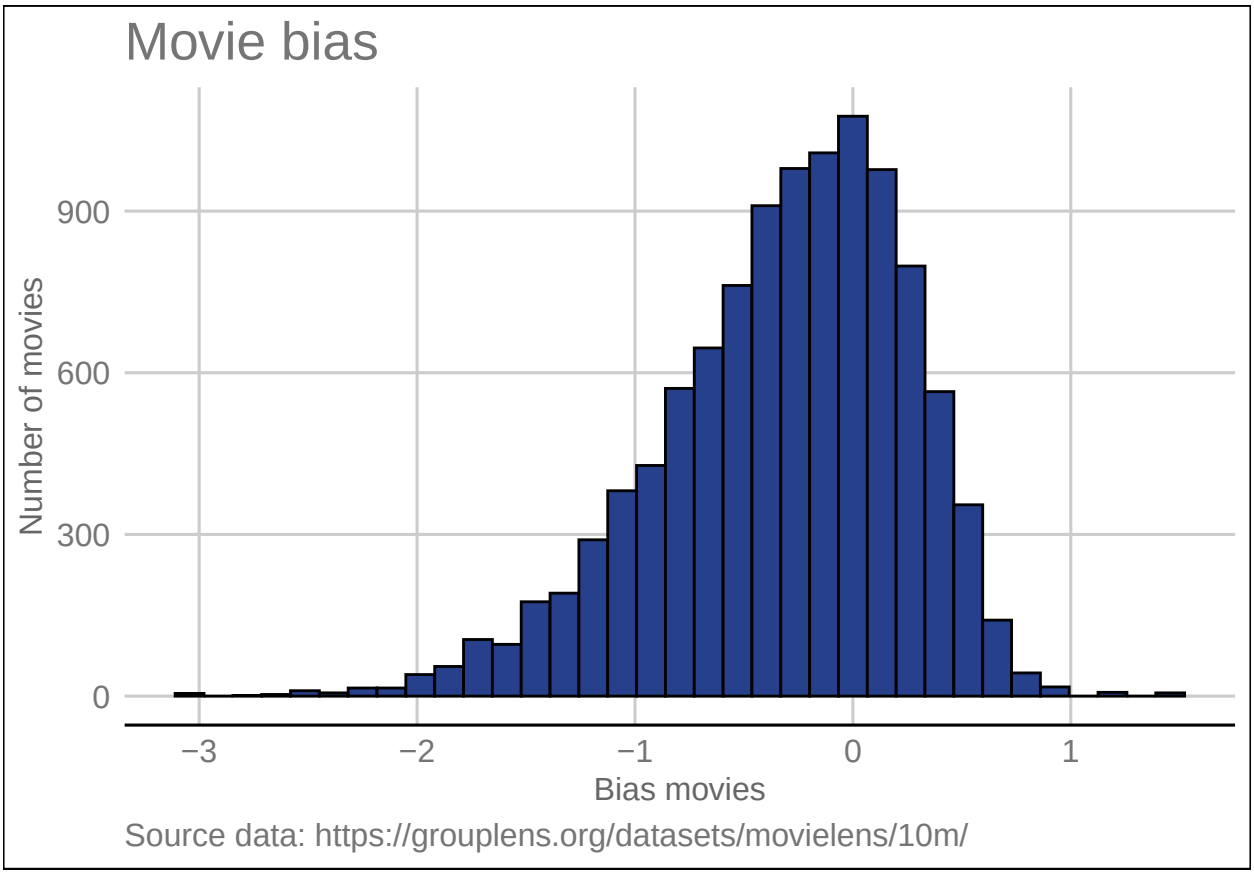
We know that some movies are just generally rated higher than others. Our first model can be improved by adding a term $\mathbf{b_i}$ that represents the average ranking for movie . The improved model can be described as follows:

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

We know that the least squares estimate $\hat{\mathbf{b}}_\mathbf{i}$ is just the average of $\mathbf{Y_{u,i}} - \hat{\mu}$ for each movie.

```r
############################################################
# II. Modeling movie effects
############################################################

#The least squares estimate ^b_i is just the average of Y(u,i) - ^mu for each movie
mu <- mean(edx$rating)
movie_bias <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

#Histogram graphic show 0 equivalent 3.51 mean rating movie, 1.5 is equal to 5 rating and -3 is equal t
ggplot(movie_bias, aes(x = b_i)) +
  geom_histogram(bins = 35, fill = "#27408B", color = "black") +
  labs(title = "Movie bias",
       x = "Bias movies",
       y = "Number of movies",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
```

```
  theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
  theme(axis.text.x=element_text(size=10, angle=0, face = "bold")) +
  theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))+
  theme_gdocs()
```




```
#create a predicted rating add mu + b_i (bias movie)
predicted_ratings <- mu + final_holdout_test %>%
  left_join(movie_bias, by='movieId') %>%
  pull(b_i)

#calculate RMSE with predicted_ratings and test set final_holdout_test
model_1 <- RMSE(predicted_ratings, final_holdout_test$rating)

model_1_rmse <- tibble(Model = "Movie Effect Model", RMSE = model_1)

model_1_rmse %>% kable()
```

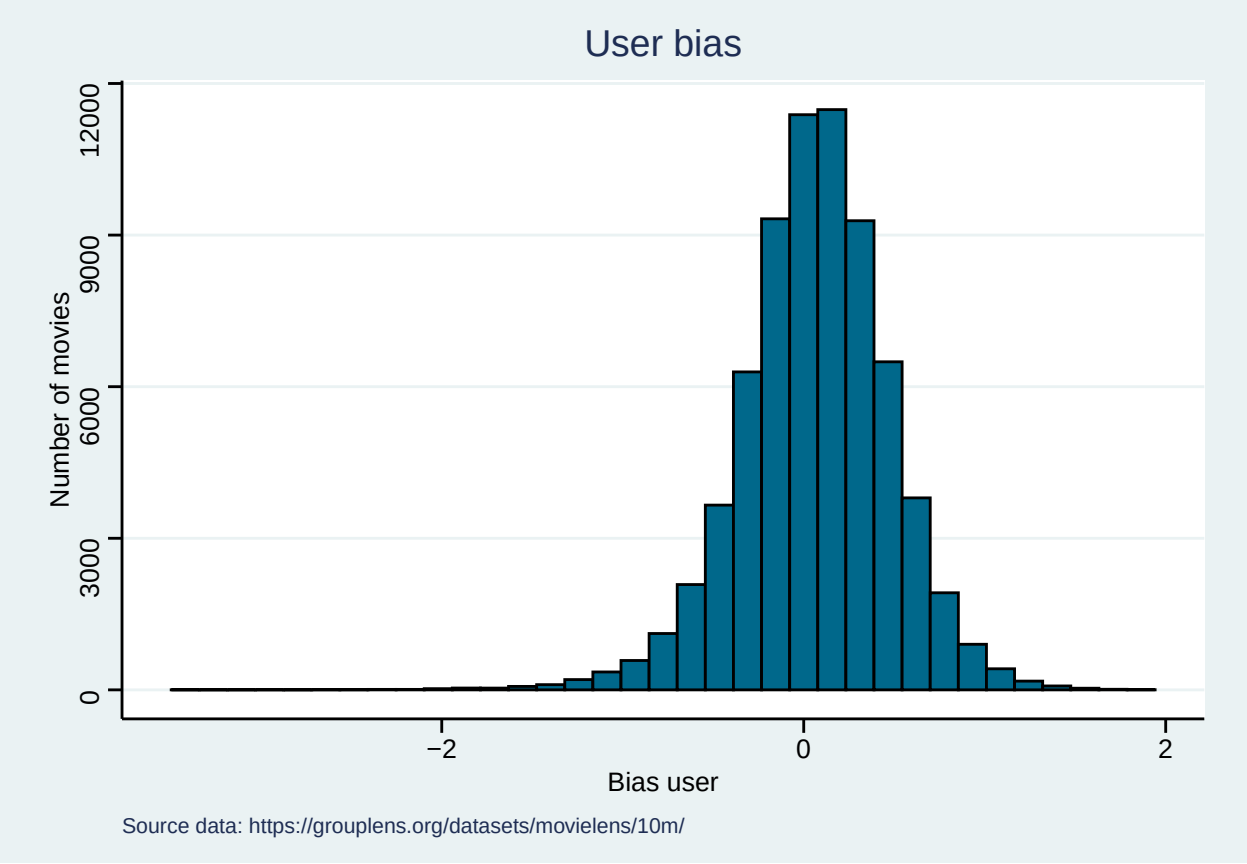
Model	RMSE
Movie Effect Model	0.94391

We could see in the histogram graphic, these estimates for movies vary substantially. We know that some movies are good whereas others are not.

We had estimated the $\hat{\mu}$, the average of all ratings, as 3.5. So a $\hat{\mathbf{b}}\mathbf{i}$ of 1.5 implies $\hat{\mathbf{y}}_{\mathbf{u},\mathbf{i}} = \hat{\mu} + \hat{\mathbf{b}}_{\mathbf{i}} = \mathbf{3.5} + \mathbf{1.5} = \mathbf{5}$ which is a perfect five-star rating. Our prediction improves once we use $\hat{\mathbf{y}}_{\mathbf{u},\mathbf{i}} = \hat{\mu} + \hat{\mathbf{b}}_{\mathbf{i}}$

III. Modeling user effects

```
############################################################
# III. Modeling user effects
############################################################

#We will compute an approximation by computing ^mu  and ^b_i and estimating ^b_u as the average of  ^y(
user_bias <- edx %>%
  left_join(movie_bias, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

#Histogram graphic bias user
ggplot(user_bias, aes(x = b_u)) +
  geom_histogram(bins = 35, fill = "#00688B", color = "black") +
  labs(title = "User bias",
       x = "Bias user",
       y = "Number of movies",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  theme(axis.title.x=element_text(size=12, angle=0, face = "bold"))+
  theme(axis.text.x=element_text(size=10, angle=0, face = "bold")) +
  theme(axis.text.y=element_text(size=10, angle=90, face = "bold"))+
  theme_stata()
```



```r
#create a predicted rating add mu + b_i (bias movie) + b_u (bias user)
predicted_ratings <- final_holdout_test %>%
  left_join(movie_bias, by='movieId') %>%
  left_join(user_bias, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

#calculate RMSE with predicted_ratings and test set final_holdout_test
model_2 <- RMSE(predicted_ratings, final_holdout_test$rating)

model_2_rmse <- tibble(Model = "Movie + User Effect Model", RMSE = model_2)

model_2_rmse %>% kable()
```

Model	RMSE
Movie + User Effect Model	0.86535

We compute an approximation by computing $\hat{\mu}$ and $\mathbf{\hat{b}_i}$ and estimating $\mathbf{\hat{b}_u}$ as the average of $\mathbf{\hat{y}_{u,i}} - \hat{\mu} - \mathbf{\hat{b}_i}$

The substantial variability across users is quiet evident from the above plot. This implies that a further improvement to our model may be:

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

IV. Regularized movie + user effect model

Regularization permits us to penalize large estimates that are formed using small sample sizes.

In order to account for this effect, regularization will be introduced into the model. The formula of the regularization introduced is:

$\frac{1}{\lambda + n_i} \sum_{u=1}^{ni} (Y_{u,i} - \hat{\mu})$

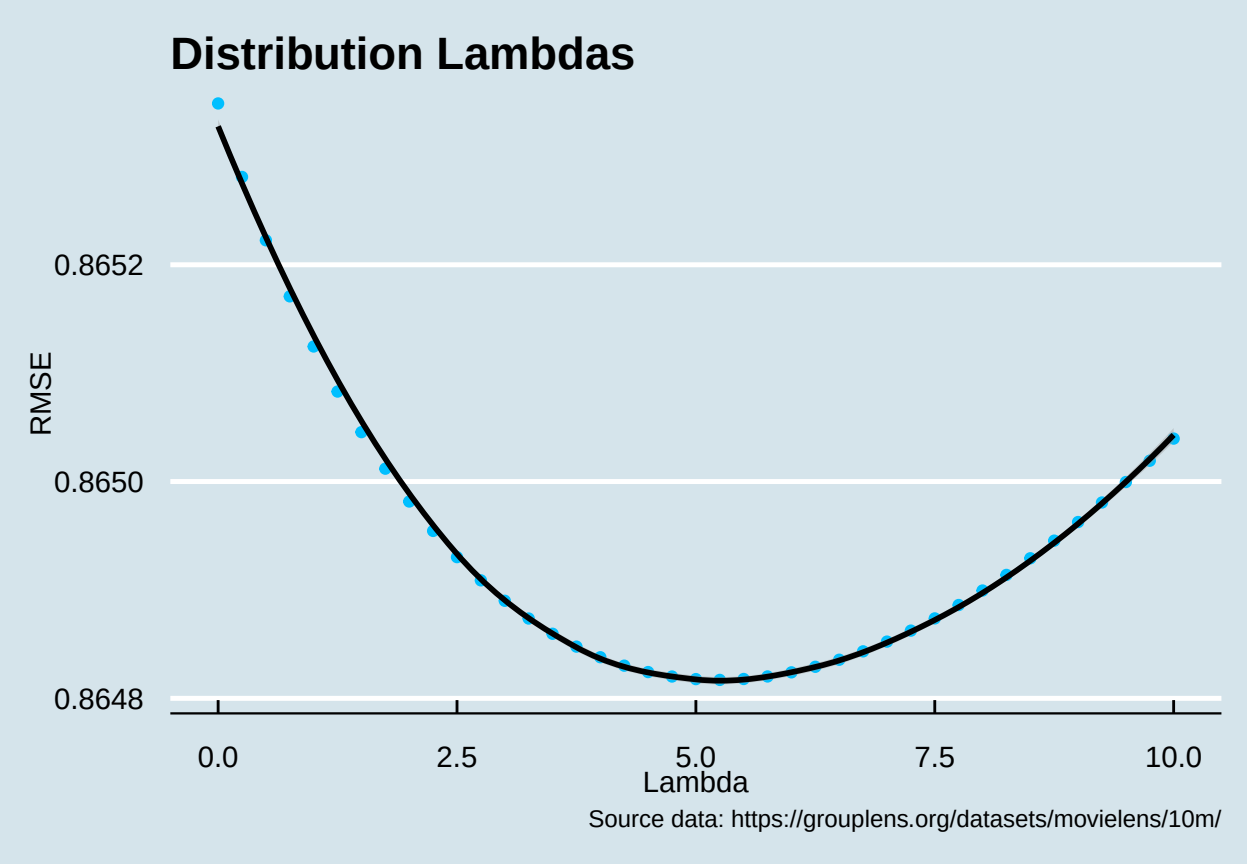
The idea is to add a tuning parameter λ to further reduce the RMSE. The idea is to penalize outliers from the movie bias and user bias sets which shall optimise the recommendation system.

```r
############################################################
# IV. Regularized movie + user effect model
############################################################

#The idea is to add a tuning parameter lamba to further reduce the RMSE
#The idea is to penalize outliers from the Movie Bias and User Bias sets which shall optimize the recom
lambdas_regularize <- seq(0, 10, 0.25)

rmse_regularize <- sapply(lambdas_regularize, function(l){

  edx_mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - edx_mu)/(n()+l))
```

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

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

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

#Discover which lamba will be best at reducing the RMSE
lambda <- lambdas_regularize[which.min(rmse_regularize)]
lambda
```

```
## [1] 5.25
```

```
ggplot(mapping = aes(x = lambdas_regularize, y = rmse_regularize)) +
  geom_point(color = "#00BFFF") +
  labs(title = "Distribution Lambdas",
       x = "Lambda",
       y = "RMSE",
       caption = "Source data: https://grouplens.org/datasets/movielens/10m/") +
  geom_smooth(color = "black") +
  theme_economist()
```




```
#We will use the best lamba to reduce our RMSE regularized Movie + User Effect Model
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

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

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

#calculate RMSE with predicted_ratings and test set final_holdout_test
model_3 <- RMSE(predicted_ratings, final_holdout_test$rating)

model_3_rmse <- tibble(Model = "Regularized Movie + User Effect Model", RMSE = model_3)

model_3_rmse %>% kable()
```

Model	RMSE
Regularized Movie + User Effect Model	0.86482

V. Regularized movie + user effect model + genres effect model

Regularization permits us to penalize large estimates that are formed using small sample sizes.

The idea is to add a tuning parameter λ to further reduce the RMSE. The idea is to penalize outliers from the movie bias,user and genres bias sets which shall optimize the recommendation system.

```
#########################################################################
# V. Regularized movie + user effect model + genres effecft model
#########################################################################

#The idea is to add a tuning parameter lamba to further reduce the RMSE
#The idea is to penalize outliers from the Movie, User and genres Bias sets which shall optimize the re

#We will use the best lamba to reduce our RMSE regularized Movie + User Effect Model + Genres Effect Mo

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

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

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

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

#calculate RMSE with predicted_ratings and test set final_holdout_test

model_4 <- RMSE(predicted_ratings, final_holdout_test$rating)

model_4_rmse <- tibble(Model = "Regularized Movie + User Effect Model + Genres Effect Model", RMSE = mo

model_4_rmse %>% kable()
```

Model	RMSE
Regularized Movie + User Effect Model + Genres Effect Model	0.86471

5. RECO(system using matrix factorization)

reco is an R package which implements many algorithms for sparse matrix factorizations. Focus is on applications for recommender systems.

```
##########################################################
# 5. RECO
##########################################################

niter <- 25
# Train the reco model with the fixed niter on the entire edx set
reco_train <- data_memory(edx$userId, edx$movieId, edx$rating)
reco_model <- Reco()

reco_model$train(reco_train, opts = list(dim = 10, niter = niter,  nthread = 8, costp_l2 = 0.1, costq_l1
```

```
## iter      tr_rmse          obj
##    0       1.0379   1.6585e+07
##    1       0.8872   1.3680e+07
##    2       0.8777   1.3400e+07
##    3       0.8725   1.3275e+07
##    4       0.8652   1.3175e+07
##    5       0.8553   1.3066e+07
##    6       0.8470   1.2963e+07
##    7       0.8417   1.2902e+07
##    8       0.8381   1.2857e+07
##    9       0.8355   1.2824e+07
##   10       0.8336   1.2805e+07
##   11       0.8320   1.2788e+07
##   12       0.8307   1.2775e+07
##   13       0.8295   1.2762e+07
##   14       0.8285   1.2753e+07
##   15       0.8277   1.2747e+07
##   16       0.8269   1.2739e+07
##   17       0.8262   1.2731e+07
##   18       0.8256   1.2728e+07
##   19       0.8250   1.2722e+07
##   20       0.8245   1.2717e+07
##   21       0.8241   1.2716e+07
##   22       0.8237   1.2711e+07
##   23       0.8232   1.2706e+07
##   24       0.8229   1.2704e+07
```

```
# Prepare final hold-out test data
reco_test <- data_memory(final_holdout_test$userId, final_holdout_test$movieId)

# Predict ratings for the final hold-out test set
predicted_holdout_ratings <- reco_model$predict(reco_test)

# Add predictions to final hold-out test set
final_holdout_test$predicted_rating <- predicted_holdout_ratings

# Calculate RMSE for the final hold-out test set
```

```r
model_reco <-  RMSE(final_holdout_test$predicted_rating, final_holdout_test$rating)
model_reco_rmse <- tibble(Model = "Reco matrix factorization", RMSE = model_reco)

model_reco_rmse %>% kable()
```

Model	RMSE
Reco matrix factorization	0.83534

6. RESULTS

```r
############################################################
# 6. RESULTS
############################################################

#Summarize the models RMSE

results_rmse <- tibble(Model = c("Naive RMSE", "Movie Effect Model", "Movie + User Effect Model", "Regul
                                 "Regularized Movie + User Effect Model + Genres Effect Model", "Reco ma
                       RMSE = c(result_nai, model_1, model_2, model_3, model_4, model_reco))

results_rmse %>% kable()
```

Model	RMSE
Naive RMSE	1.06120
Movie Effect Model	0.94391
Movie + User Effect Model	0.86535
Regularized Movie + User Effect Model	0.86482
Regularized Movie + User Effect Model + Genres Effect Model	0.86471
Reco matrix factorization	0.83534

7. CONCLUSION

We could see from the models used that adding more variables increases the accuracy of the result.

1. Naive rmse: We use one variable: mu (median rating), which is the average of the movie ratings, so the result is naive. Result is: 1.06120.
2. Movie Effect Model: We use two variables: mu (median rating) and b_i (movie bias). To the average of the movies we add the deviations of the average ratings of the movies. The result is: 0.94391 .
3. Movie + User Effect Model: We use three variables: mu (average rating), b_i (movie bias) and b_u (user bias). To the average of the movies we add the deviations of the average ratings of the movies and add the deviation of the average rating of the movies of the users. The result is: 0.86535.
4. Regularized Movie + User Effect Model: We use four variables: mu (median rating), b_i (movie bias), b_u (user bias), λ (lambda). To the movie mean we add the deviations from the movie rating means, add the deviation of the average rating of the movies of the users and use lambda with the idea of penalizing the outliers from the movie bias and user bias sets, which will optimize the recommendation system. The result is: 0.86482.

5. Regularized Movie + User Effect Model + Genres Effect Model: We use five variables: mu (median rating), b_i (movie bias), b_u (user bias), b_g(genres bias), λ (lambda). To the movie mean we add the deviations from the movie rating means, add the deviation of the average rating of the movies of the users, add the deviation the movies of the genres and use lambda with the idea of penalizing the outliers from the movie bias, user and genres bias sets, which will optimize the recommendation system. The result is: 0.86471.

6. Reco (matrix refactorization): using this recommendation system, we obtain the best of all results. **The result is: 0.83572 therefore objective completed, RMSE $<$ 0.86490.**

8. REFERENCES

[1] Yixuan Qiu, David Cortes, Chih-Jen Lin, Yu-Chin Juan, Wei-Sheng Chin, Yong Zhuang, Bo-Wen Yuan, Meng-Yuan Yang. Reco (system using a matrix factorization):

https://cran.rproject.org/web/packages/recosystem/recosystem.pdf