MovieLens Recommendation System - Code

Uday Adusumilli

16 Dec, 2021

Contents


```
# Install all needed libraries if it is not present

if(!require(tidyverse)) install.packages("tidyverse")
```

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

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

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

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

```
if(!require(kableExtra)) install.packages("kableExtra")
```

```
## Loading required package: kableExtra
```

```
## Warning: package 'kableExtra' was built under R version 4.1.1
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
if(!require(tidyr)) install.packages("tidyr")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(stringr)) install.packages("stringr")
if(!require(forcats)) install.packages("forcats")
if(!require(ggplot2)) install.packages("ggplot2")

if(!require(readr)) install.packages("readr")
if(!require(dplyr)) install.packages("dplyr")
if(!require(gridExtra)) install.packages("gridExtra")
```

```
## Loading required package: gridExtra
```

```
## Warning: package 'gridExtra' was built under R version 4.1.2
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
if(!require(dslabs)) install.packages("dslabs")
```

```
## Loading required package: dslabs
```

```
if(!require(data.table)) install.packages("data.table")
```

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

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

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

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

if(!require(ggrepel)) install.packages("ggrepel")

## Loading required package: ggrepel

## Warning: package 'ggrepel' was built under R version 4.1.2

if(!require(ggthemes)) install.packages("ggthemes")

## Loading required package: ggthemes

## Warning: package 'ggthemes' was built under R version 4.1.2

# Loading all needed libraries

library(dplyr)
library(tidyverse)
library(kableExtra)
library(tidyr)
library(stringr)
library(forcats)
library(ggplot2)

library(readr)
library(gridExtra)
library(dslabs)
library(data.table)
library(ggrepel)
library(ggthemes)

#################################################################
# Create edx set, validation set, and submission file
#################################################################

# 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")

## Loading required package: caret

## Warning: package 'caret' was built under R version 4.1.1

## Loading required package: lattice
```

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

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

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

The following block needs to be executed if the data isn't available locally. Execution of the following block requires an active internet connection.

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

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


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

Execution of the following block requires the data files to be preset in the ml-10M100K folder. The following block is an alternative to the above block.

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

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

With the execution of the above block, data loading is complete.

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

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

movies = subset(movies_old, select = c(movieId,title,genres))

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

# Validation set will be 10% of MovieLens data

set.seed(1, sample.kind = "Rounding") # if using R 3.5 or earlier, use `set.seed(1)` instead

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

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

# 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)

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

edx <- rbind(edx, removed)

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

library(kableExtra)
```

Top rows of the Edx file.

```
head(edx)

##   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
```

Users	Movies
69878	10677

1 Executive Summary

The purpose for this project is creating a recommender system using MovieLens dataset.

The version of movielens dataset used for this final assignment contains approximately 10 Milions of movies ratings, divided in 9 Milions for training and one Milion for validation. It is a small subset of a much larger (and famous) dataset with several millions of ratings. Into the training dataset there are approximately **70.000 users** and **11.000 different movies** divided in 20 genres such as Action, Adventure, Horror, Drama, Thriller and more.

After a initial data exploration, the recommender systems builted on this dataset are evaluated and choosen based on the RMSE - Root Mean Squared Error that should be at least lower than **0.87750**.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

```
# The RMSE function that will be used in this project is:
RMSE <- function(true_ratings = NULL, predicted_ratings = NULL) {
    sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

For accomplishing this goal, the **Regularized Movie+User+Genre Model** is capable to reach a RMSE of **0.8628**, that is really good.

2 Exploratory Data Analysis

2.1 Inital data Exploration

The 10 Millions dataset is divided into two dataset: `edx` for training purpose and `validation` for the validation phase.

The `edx` dataset contains approximately 9 Millions of rows with 70.000 different users and 11.000 movies with rating score between 0.5 and 5. There is no missing values (0 or NA).

edx dataset

```
edx %>% summarize(Users = n_distinct(userId),
                  Movies = n_distinct(movieId)) %>%
kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

Missing Values per Column

```
# edx <- edx %>% select(-X)
# validation <- validation %>% select(-X)
```

	x
userId	0
movieId	0
rating	0
timestamp	0
title	0
genres	0

```
sapply(edx, function(x) sum(is.na(x))) %>%
kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

The features/variables/columns in both datasets are six:

- **userId** `<integer>` that contains the unique identification number for each user.
- **movieId** `<numeric>` that contains the unique identification number for each movie.
- **rating** `<numeric>` that contains the rating of one movie by one user. Ratings are made on a 5-Star scale with half-star increments.
- **timestamp** `<integer>` that contains the timestamp for one specific rating provided by one user.
- **title** `<character>` that contains the title of each movie including the year of the release.
- **genres** `<character>` that contains a list of pipe-separated of genre of each movie.

	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|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

First 6 Rows of edx dataset

```
head(edx) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

2.2 Dataset Pre-Processing and Feature Engineering

After a initial data exploration, we notice that the `genres` are pipe-separated values. It's necessary to extract them for more consisten, robust and precise estimate. We also observe that the `title` contains the year where the movie war released and this it could be necessary to predic the movie rating. Finally, we can extract the year and the month for each rating.

The pre-processing phase is composed by this steps:

1. Convert `timestamp` to a human readable date format;
2. Extract the month and the year from the date;
3. Extract the release year for each movie from the title;
4. Separate each genre from the pipe-separated value. It increases the size of both datasets.

```
# Convert timestamp to a human readable date

edx$date <- as.POSIXct(edx$timestamp, origin="1970-01-01")
validation$date <- as.POSIXct(validation$timestamp, origin="1970-01-01")

# Extract the year and month of rate in both dataset

edx$yearOfRate <- format(edx$date,"%Y")
edx$monthOfRate <- format(edx$date,"%m")

validation$yearOfRate <- format(validation$date,"%Y")
validation$monthOfRate <- format(validation$date,"%m")

# Extract the year of release for each movie in both dataset
# edx dataset

edx <- edx %>%
   mutate(title = str_trim(title)) %>%
   extract(title,
           c("titleTemp", "release"),
```

```
          regex = "^(.*) \\(([0-9 \\-]*)\\)$",
          remove = F) %>%
   mutate(release = if_else(str_length(release) > 4,
                            as.integer(str_split(release, "-",
                                                 simplify = T)[1]),
                            as.integer(release))
   ) %>%
   mutate(title = if_else(is.na(titleTemp),
                          title,
                          titleTemp)
         ) %>%
  select(-titleTemp)

# validation dataset

validation <- validation %>%
   mutate(title = str_trim(title)) %>%
   extract(title,
           c("titleTemp", "release"),
           regex = "^(.*) \\(([0-9 \\-]*)\\)$",
           remove = F) %>%
   mutate(release = if_else(str_length(release) > 4,
                            as.integer(str_split(release, "-",
                                                 simplify = T)[1]),
                            as.integer(release))
   ) %>%
   mutate(title = if_else(is.na(titleTemp),
                          title,
                          titleTemp)
         ) %>%
  select(-titleTemp)

# Extract the genre in edx datasets

edx <- edx %>%
   mutate(genre = fct_explicit_na(genres,
                                  na_level = "(no genres listed)")
          ) %>%
   separate_rows(genre,
                 sep = "\\|")

# Extract the genre in validation datasets

validation <- validation %>%
   mutate(genre = fct_explicit_na(genres,
                                  na_level = "(no genres listed)")
          ) %>%
   separate_rows(genre,
                 sep = "\\|")

# remove unnecessary columns on edx and validation dataset

edx <- edx %>% select(userId, movieId, rating, title, genre, release, yearOfRate, monthOfRate)
```

userId	movieId	rating	title	genre	release	yearOfRate	monthOfRate
1	122	5	Boomerang	Comedy	1992	1996	8
1	122	5	Boomerang	Romance	1992	1996	8
1	185	5	Net, The	Action	1995	1996	8
1	185	5	Net, The	Crime	1995	1996	8
1	185	5	Net, The	Thriller	1995	1996	8
1	292	5	Outbreak	Action	1995	1996	8

```
validation <- validation %>% select(userId, movieId, rating, title, genre, release, yearOfRate, monthOfR
```

```
# Convert the columns into the desidered data type

edx$yearOfRate <- as.numeric(edx$yearOfRate)
edx$monthOfRate <- as.numeric(edx$monthOfRate)
edx$release <- as.numeric(edx$release)

validation$yearOfRate <- as.numeric(validation$yearOfRate)
validation$monthOfRate <- as.numeric(validation$monthOfRate)
validation$release <- as.numeric(validation$release)
```

After preprocessing the data, `edx` dataset looks like this:

Processed edx datadaset

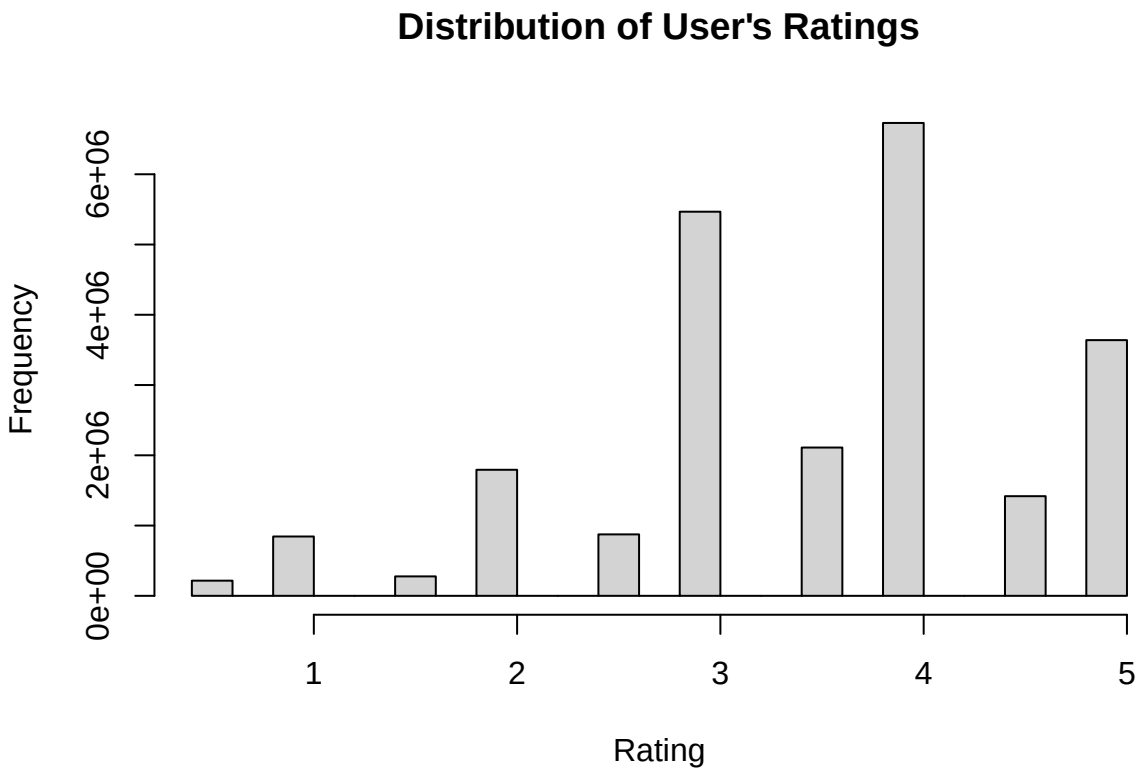
```
# Output the processed dataset

head(edx) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

2.3 Rating Distribution

Overview of Rating Distribution

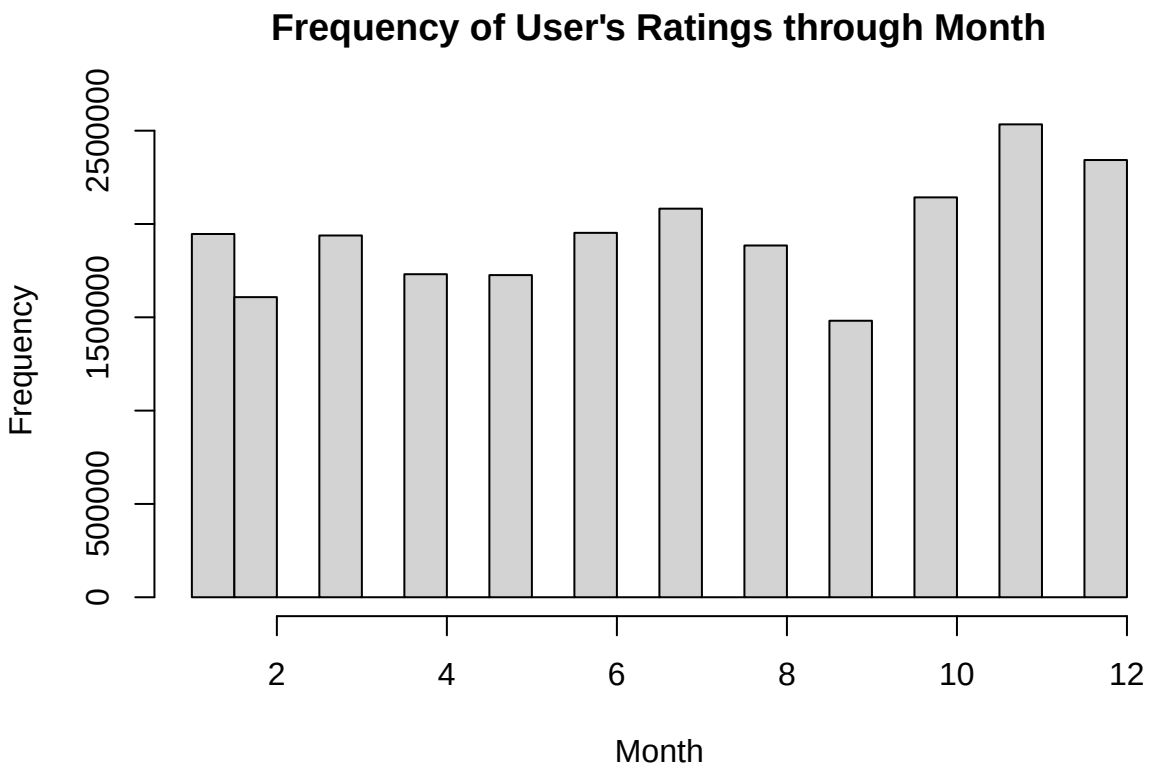
According to the histogram below, it shows that there are a small amount of negative votes (below 3). Maybe, the user tends to give a vote if he liked the movie. Half-Star votes are less common than "Full-Star" votes.

```
hist(edx$rating, main="Distribution of User's Ratings", xlab="Rating")
```

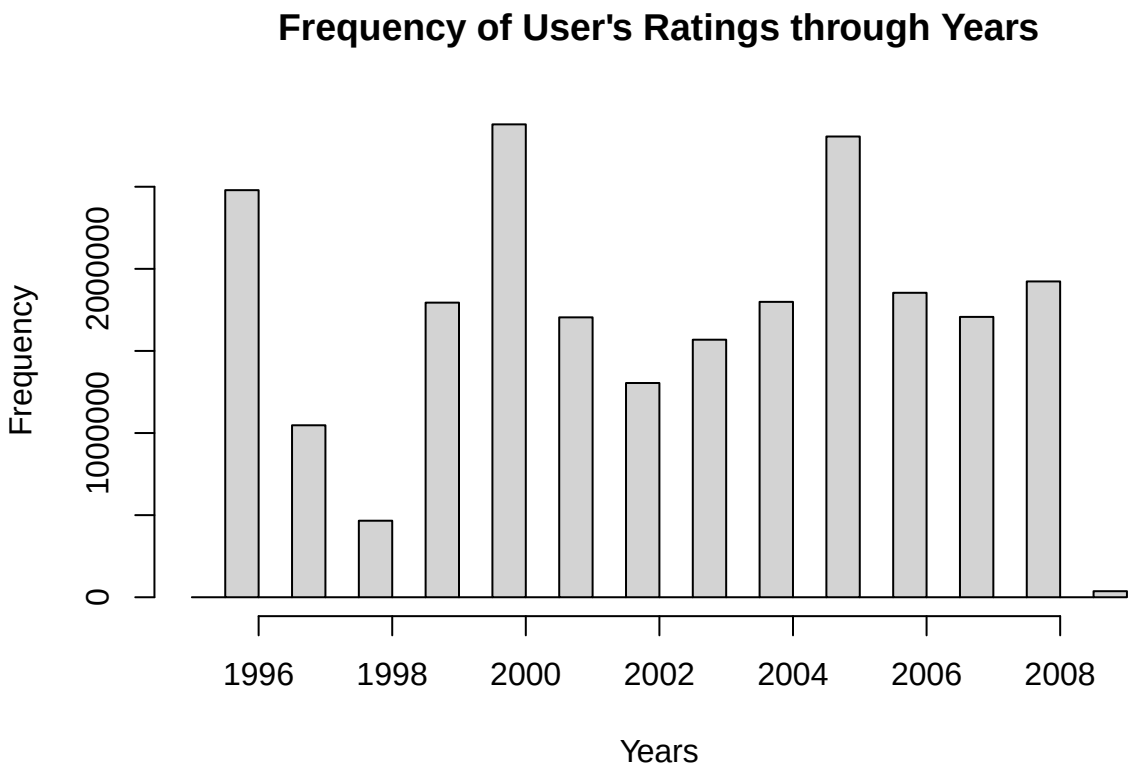


Overview of Rating Frequency through Months and Years

```
hist(edx$monthOfRate, main="Frequency of User's Ratings through Month", xlab="Month")
```

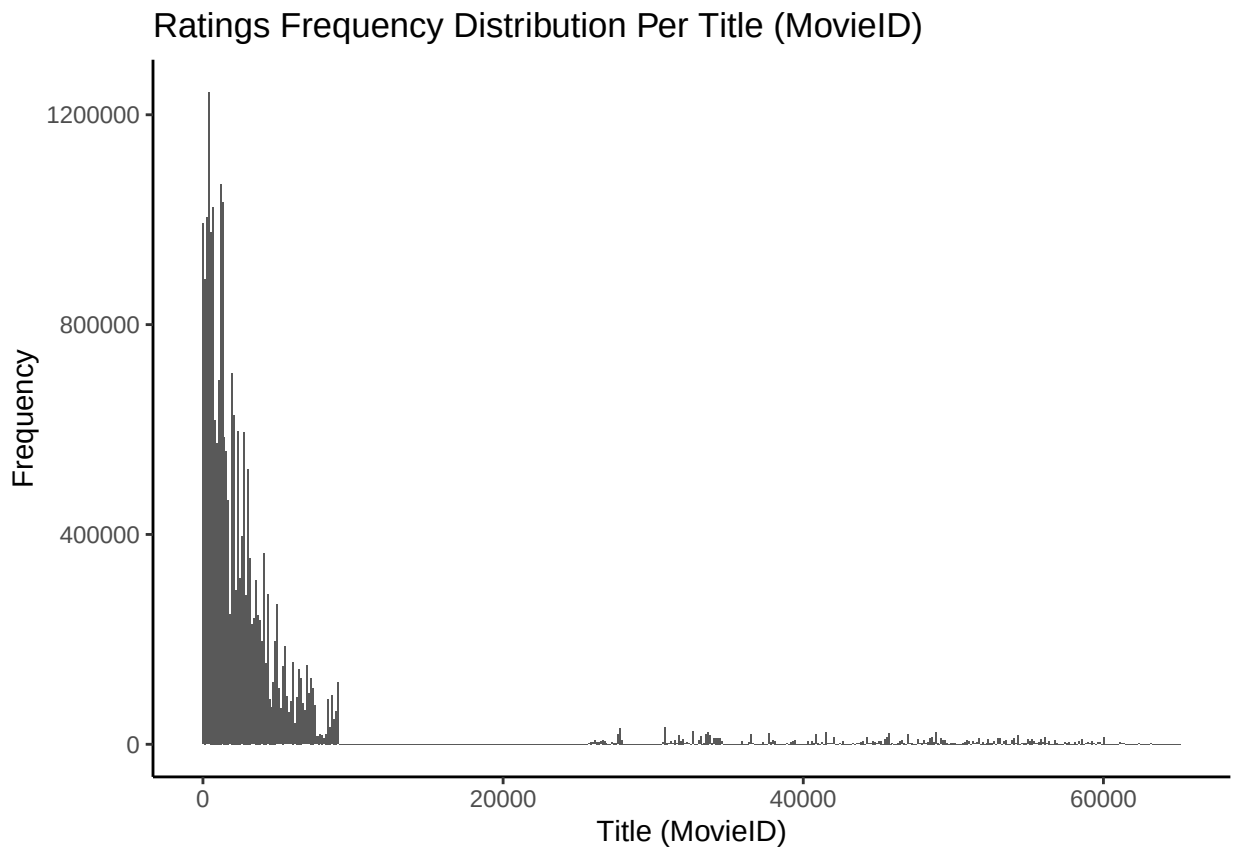



```
hist(edx$yearOfRate, main="Frequency of User's Ratings through Years", xlab="Years")
```



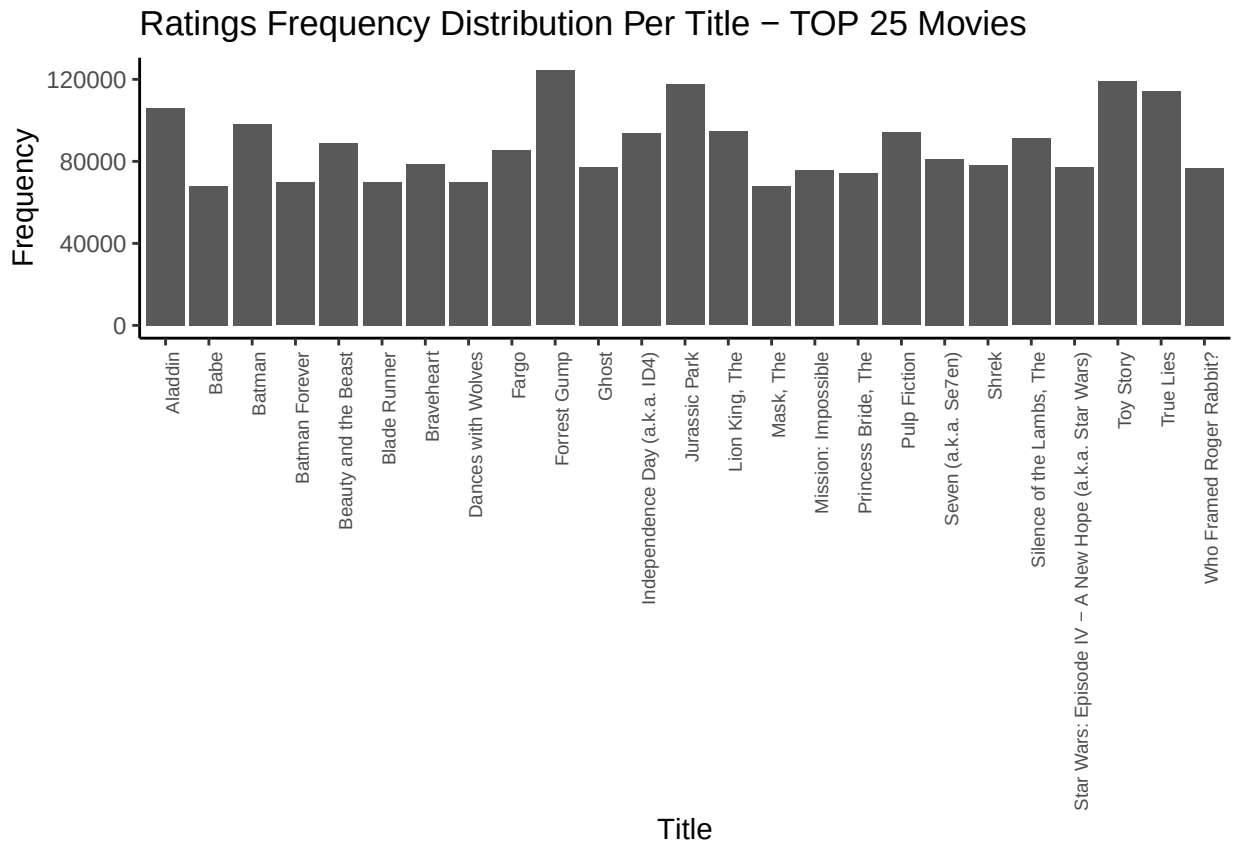
2.3.1 Numbers of Ratings per Movie

```
ggplot(edx, aes(movieId)) +
theme_classic()  +
geom_histogram(bins=500) +
labs(title = "Ratings Frequency Distribution Per Title (MovieID)",
     x = "Title (MovieID)",
     y = "Frequency")
```



2.3.2 Top Rated Movies

```
edx %>%
   group_by(title) %>%
   summarise(count = n()) %>%
   arrange(desc(count)) %>%
   head(n=25) %>%
   ggplot(aes(title, count)) +
   theme_classic()  +
   geom_col() +
   theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 7)) +
   labs(title = "Ratings Frequency Distribution Per Title - TOP 25 Movies",
        x = "Title",
        y = "Frequency")
```

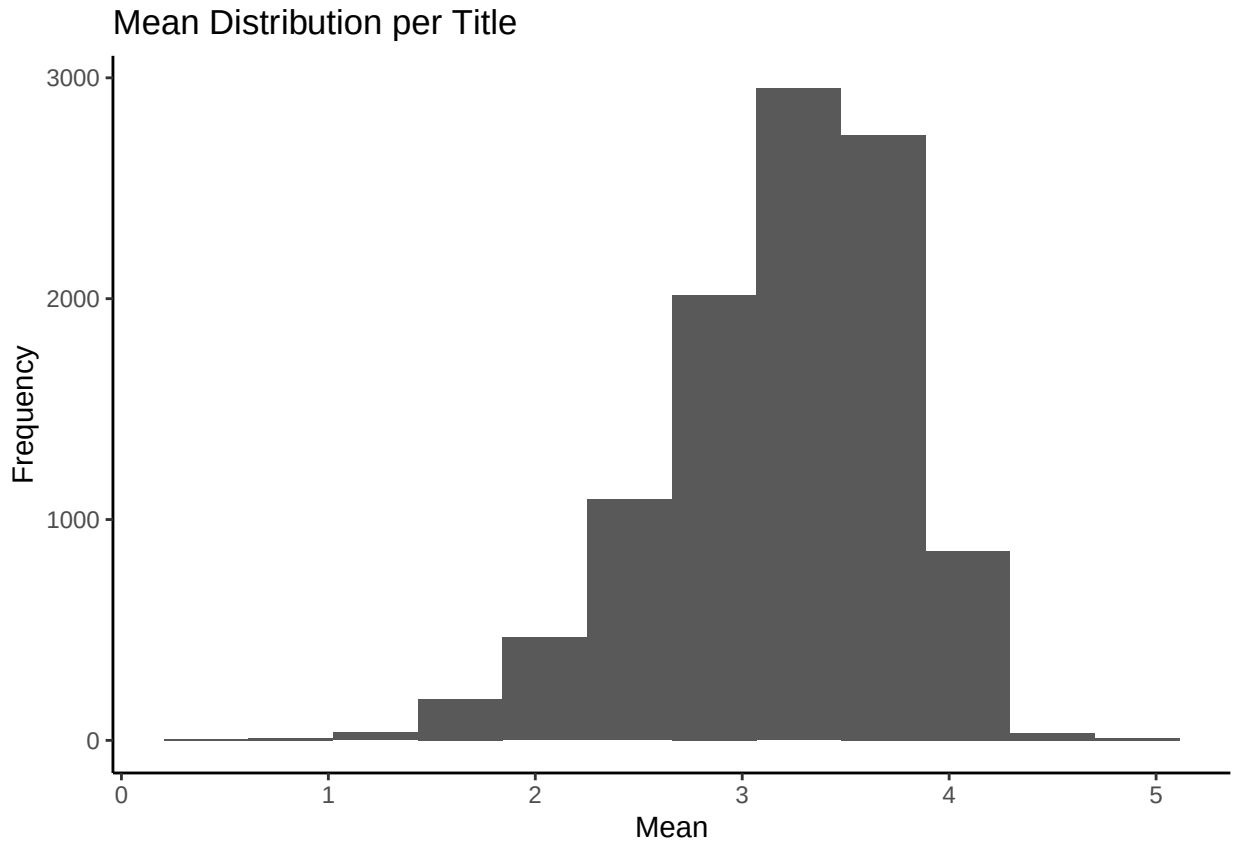



```
edx %>%
   group_by(title) %>%
   summarise(count = n()) %>%
   arrange(desc(count)) %>%
   head(n=25) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

2.3.3 Mean Distribution per Title (Movie ID)

```
edx %>%
   group_by(title) %>%
   summarise(mean = mean(rating)) %>%
   ggplot(aes(mean)) +
   theme_classic()  +
   geom_histogram(bins=12) +
   labs(title = "Mean Distribution per Title",
        x = "Mean",
        y = "Frequency")
```

title	count
Forrest Gump	124316
Toy Story	118950
Jurassic Park	117440
True Lies	114115
Aladdin	105865
Batman	98340
Lion King, The	94605
Pulp Fiction	94086
Independence Day (a.k.a. ID4)	93796
Silence of the Lambs, The	91146
Beauty and the Beast	89145
Fargo	85580
Seven (a.k.a. Se7en)	81244
Braveheart	78636
Shrek	78378
Ghost	77440
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars)	77016
Who Framed Roger Rabbit?	76825
Mission: Impossible	75968
Princess Bride, The	74045
Dances with Wolves	70101
Blade Runner	69785
Batman Forever	69656
Mask, The	68172
Babe	68124

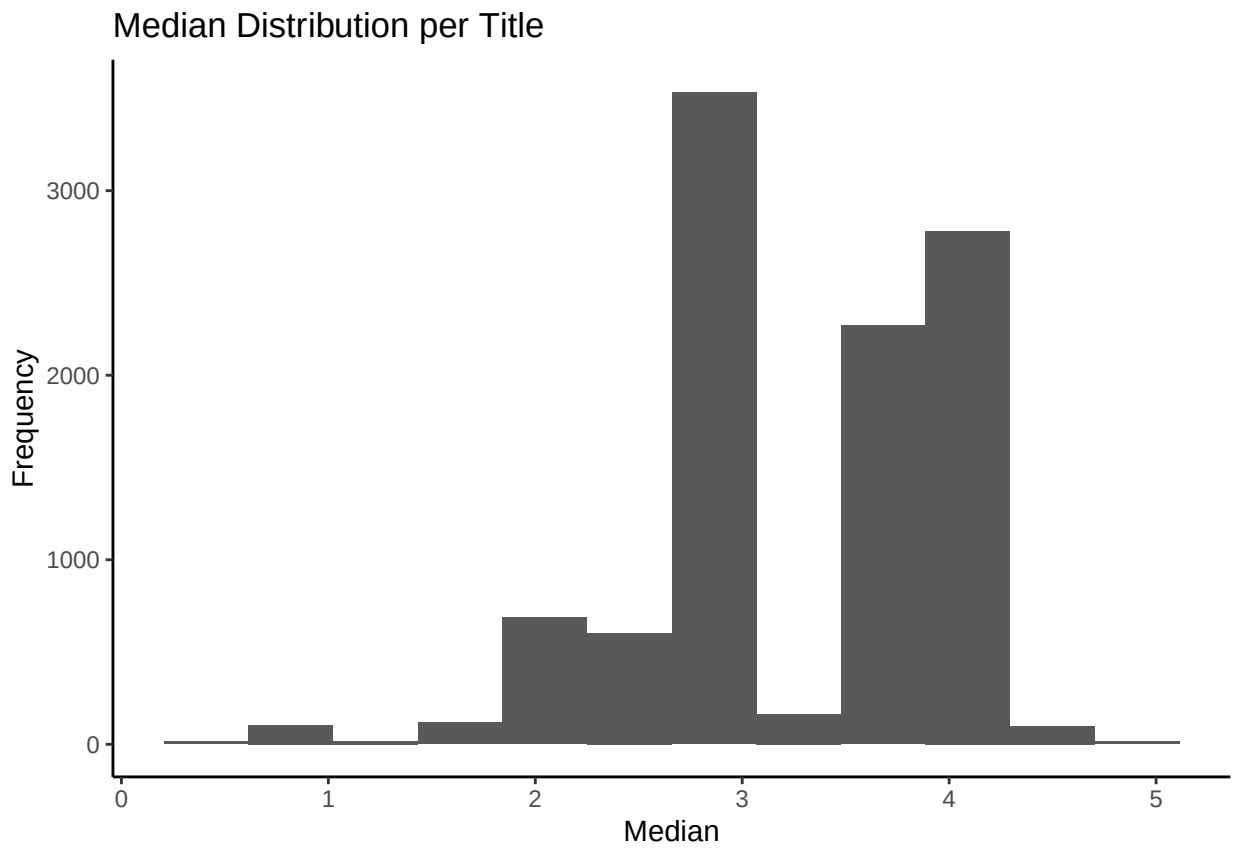


title	mean
Blue Light, The (Das Blaue Licht)	5.000000
Fighting Elegy (Kenka erejii)	5.000000
Hellhounds on My Trail	5.000000
Satan's Tango (SÃ¡tÃ¡ntangÃ³)	5.000000
Shadows of Forgotten Ancestors	5.000000
Sun Alley (Sonnenallee)	5.000000
Constantine's Sword	4.750000
Human Condition II, The (Ningen no joken II)	4.750000
Human Condition III, The (Ningen no joken III)	4.750000
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)	4.750000
Class, The (Entre les Murs)	4.666667
I'm Starting From Three (Ricomincio da Tre)	4.666667
Bad Blood (Mauvais sang)	4.500000
Demon Lover Diary	4.500000
End of Summer, The (Kohayagawa-ke no aki)	4.500000
Kansas City Confidential	4.500000
Ladrones	4.500000
Life of Oharu, The (Saikaku ichidai onna)	4.500000
Man Named Pearl, A	4.500000
Mickey	4.500000
Please Vote for Me	4.500000
Power of Nightmares: The Rise of the Politics of Fear, The	4.500000
Testament of Orpheus, The (Testament d'OrphÃ©e)	4.500000
Tokyo!	4.500000
Valerie and Her Week of Wonders (Valerie a tÃ½den divu)	4.500000

```
edx %>%
   group_by(title) %>%
   summarise(mean = mean(rating)) %>%
   arrange(desc(mean)) %>%
   head(n=25) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

2.3.4 Median Distribution per Title (Movie ID)

```
edx %>%
   group_by(title) %>%
   summarise(median = median(rating)) %>%
   ggplot(aes(median)) +
   theme_classic()  +
   geom_histogram(bins=12) +
   labs(title = "Median Distribution per Title",
        x = "Median",
        y = "Frequency")
```




```
edx %>%
   group_by(title) %>%
   summarise(median = median(rating)) %>%
   arrange(desc(median)) %>%
   head(n=25) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

2.4 Genre Analysis

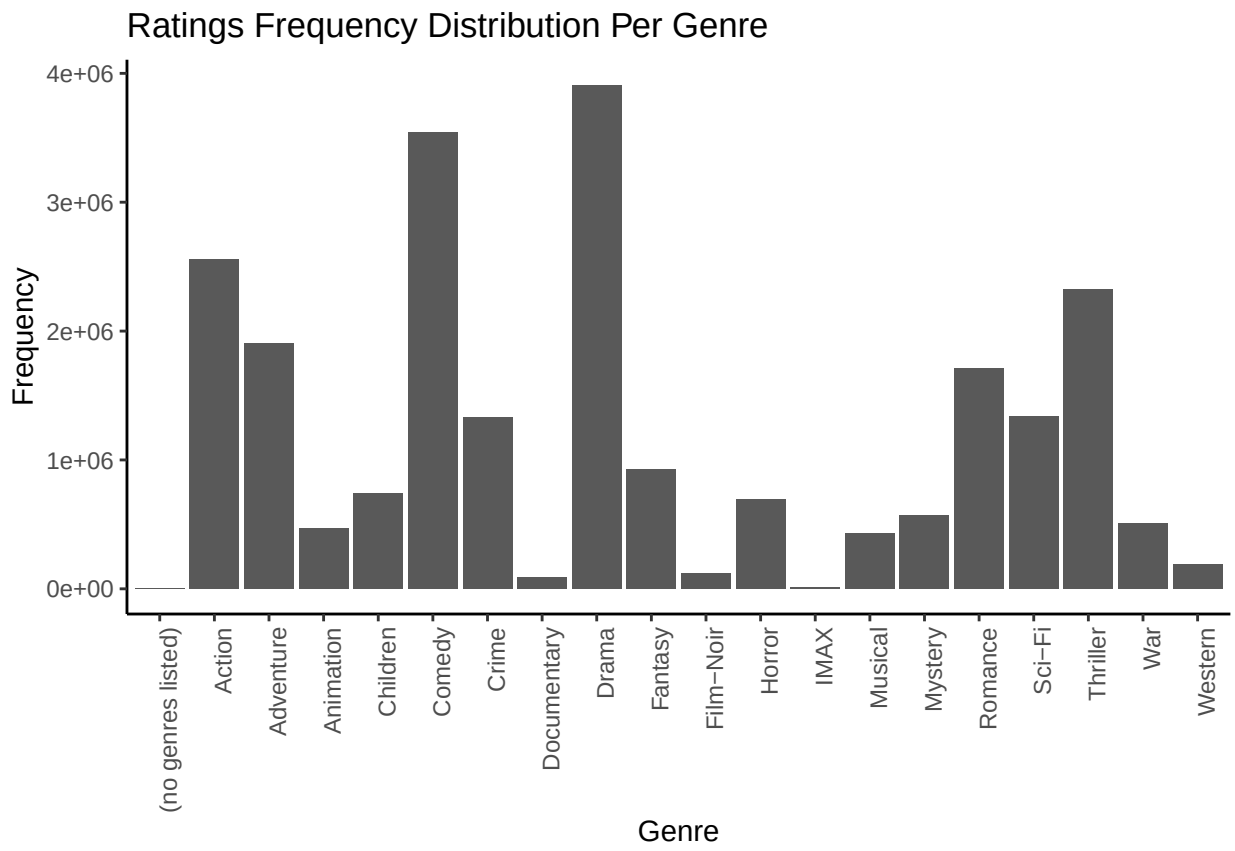
2.4.1 Rating Distribution per Genre

Overview of Rating distribution over Genre

```
edx %>%
   group_by(genre) %>%
   summarise(count = n()) %>%
   ggplot(aes(genre, count)) +
   theme_classic()  +
   geom_col() +
   theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
```

title	median
Blue Light, The (Das Blaue Licht)	5.00
Class, The (Entre les Murs)	5.00
Fighting Elegy (Kenka erejii)	5.00
Godfather, The	5.00
Hellhounds on My Trail	5.00
Kids of Survival	5.00
Satan's Tango (SÃ¡tÃ¡ntangÃ³)	5.00
Shadows of Forgotten Ancestors	5.00
Shawshank Redemption, The	5.00
Sun Alley (Sonnenallee)	5.00
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)	5.00
World of Apu, The (Apur Sansar)	5.00
Constantine's Sword	4.75
Human Condition II, The (Ningen no joken II)	4.75
Human Condition III, The (Ningen no joken III)	4.75
400 Blows, The (Les Quatre cents coups)	4.50
49 Up	4.50
Amelie (Fabuleux destin d'AmÃ©lie Poulain, Le)	4.50
American Beauty	4.50
Andrei Rublev (Andrey Rublyov)	4.50
Bad Blood (Mauvais sang)	4.50
Best of Youth, The (La Meglio gioventÃ¹)	4.50
Cabeza de Vaca	4.50
Casablanca	4.50
Celebration, The (Festen)	4.50

```
labs(title = "Ratings Frequency Distribution Per Genre",
     x = "Genre",
     y = "Frequency")
```

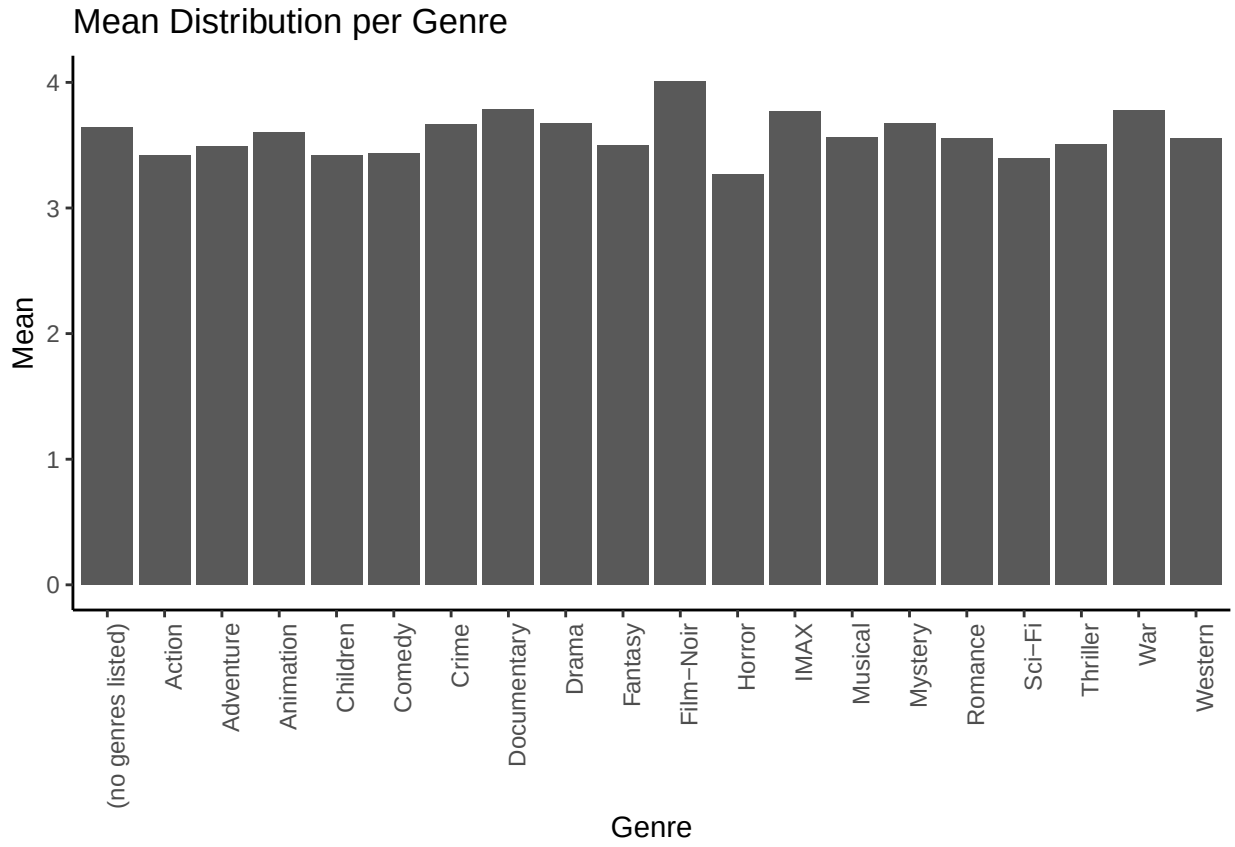



```
edx %>%
   group_by(genre) %>%
   summarise(count = n()) %>%
   arrange(desc(count)) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

2.4.2 Mean Distribution per Genre

```
edx %>%
   group_by(genre) %>%
   summarise(mean = mean(rating)) %>%
   ggplot(aes(genre, mean)) +
   theme_classic()  +
   geom_col() +
   theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
   labs(title = "Mean Distribution per Genre",
        x = "Genre",
        y = "Mean")
```

genre	count
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183
Crime	1327715
Fantasy	925637
Children	737994
Horror	691485
Mystery	568332
War	511147
Animation	467168
Musical	433080
Western	189394
Film-Noir	118541
Documentary	93066
IMAX	8181
(no genres listed)	7



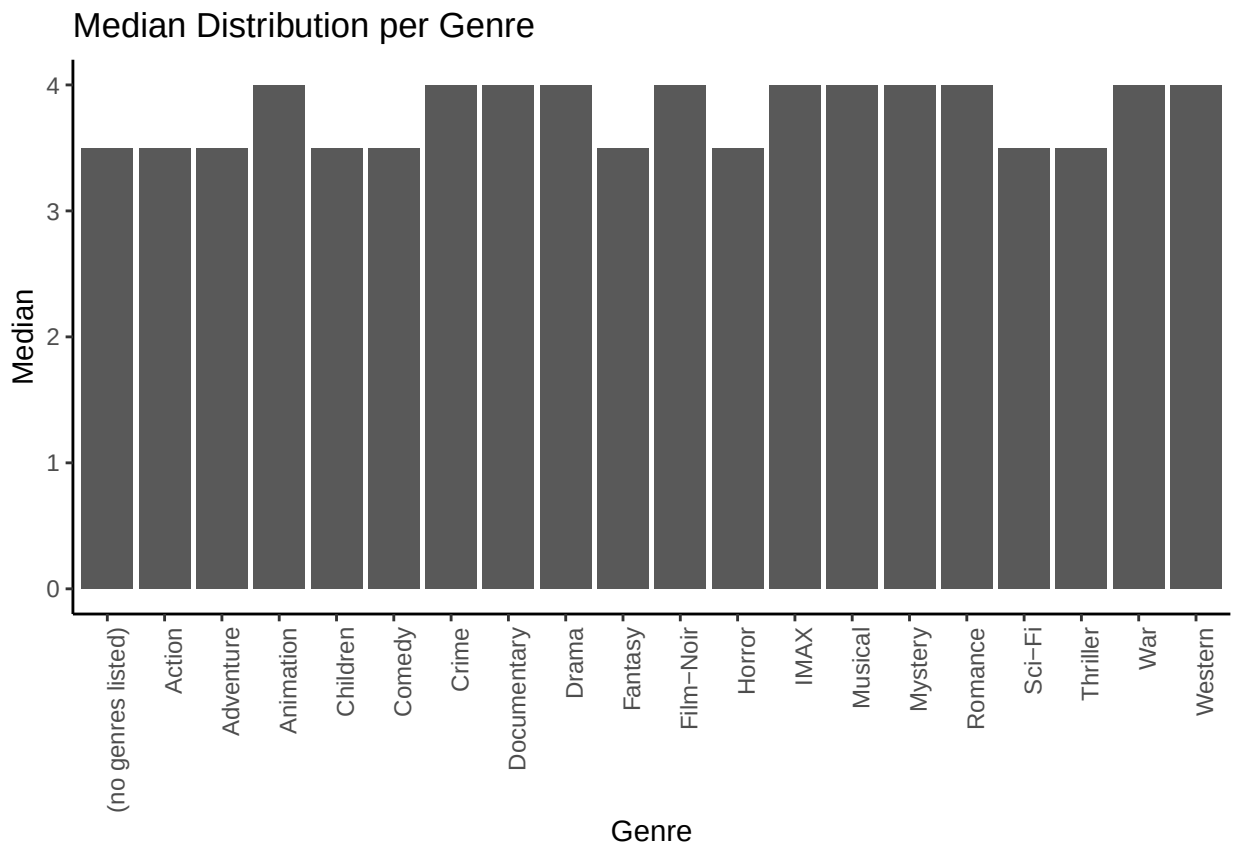

```
edx %>%
   group_by(genre) %>%
```

genre	mean
Film-Noir	4.011625
Documentary	3.783487
War	3.780813
IMAX	3.767693
Mystery	3.677001
Drama	3.673131
Crime	3.665925
(no genres listed)	3.642857
Animation	3.600644
Musical	3.563305
Western	3.555918
Romance	3.553813
Thriller	3.507676
Fantasy	3.501946
Adventure	3.493544
Comedy	3.436908
Action	3.421405
Children	3.418715
Sci-Fi	3.395743
Horror	3.269815

```
   summarise(mean = mean(rating)) %>%
   arrange(desc(mean)) %>%
   head(n=35) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

2.4.3 Median Distribution per Genre

```
edx %>%
   group_by(genre) %>%
   summarise(median = median(rating)) %>%
   ggplot(aes(genre, median)) +
   theme_classic()  +
   geom_col() +
   theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
   labs(title = "Median Distribution per Genre",
        x = "Genre",
        y = "Median")
```




```
edx %>%
   group_by(genre) %>%
   summarise(median = median(rating)) %>%
   arrange(desc(median)) %>%
   head(n=35) %>%
   kable() %>%
   kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                 position = "center",
                 font_size = 10,
                 full_width = FALSE)
```

3 Analysis - Model Building and Evaluation

3.1 Naive Baseline Model

The simplest model that someone can build, is a Naive Model that predict ALWAYS the mean. In this case, the mean is approximately 3.5.

```
paste("The mean is:", as.character(mean(edx$rating)))
```

```
## [1] "The mean is: 3.52701897954609"
```

genre	median
Animation	4.0
Crime	4.0
Documentary	4.0
Drama	4.0
Film-Noir	4.0
IMAX	4.0
Musical	4.0
Mystery	4.0
Romance	4.0
War	4.0
Western	4.0
(no genres listed)	3.5
Action	3.5
Adventure	3.5
Children	3.5
Comedy	3.5
Fantasy	3.5
Horror	3.5
Sci-Fi	3.5
Thriller	3.5

3.1.1 Naive Mean-Baseline Model

The formula used is:

$$Y_{u,i} = \hat{\mu} + \varepsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0.

```
# Calculate the average of all movies

mu_hat <- mean(edx$rating)

# Predict the RMSE on the validation set

rmse_mean_model_result <- RMSE(validation$rating, mu_hat)

# Creating a results dataframe that contains all RMSE results

results <- data.frame(model="Naive Mean-Baseline Model", RMSE=rmse_mean_model_result)
```

The RMSE on the `validation` dataset is **1.05**. It is very far for the target RMSE (below 0.87) and that indicates poor performance for the model.

3.2 Movie-Based Model, a Content-based Approach

The first Non-Naive Model takes into account the content. In this case the movies that are rated higher or lower resperct to each other.

The formula used is:

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

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i.

```
# Calculate the average of all movies

mu_hat <- mean(edx$rating)

# Calculate the average by movie

movie_avgs <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))

# Compute the predicted ratings on validation dataset

rmse_movie_model <- validation %>%
   left_join(movie_avgs, by='movieId') %>%
   mutate(pred = mu_hat + b_i) %>%
   pull(pred)

rmse_movie_model_result <- RMSE(validation$rating, rmse_movie_model)

# Adding the results to the results dataset

results <- results %>% add_row(model="Movie-Based Model", RMSE=rmse_movie_model_result)
```

The RMSE on the `validation` dataset is **0.94**. It better than the Naive Mean-Baseline Model, but it is also very far from the target RMSE (below 0.87) and that indicates poor performance for the model.

3.3 Movie + User Model, a User-based approach

The second Non-Naive Model consider that the users have different tastes and rate differently.

The formula used is:

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

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i. The b_u is a measure for the mildness of user u, i.e. the bias of user u.

```
# Calculate the average of all movies

mu_hat <- mean(edx$rating)

# Calculate the average by movie

movie_avgs <- edx %>%
   group_by(movieId) %>%
```

```
   summarize(b_i = mean(rating - mu_hat))

# Calculate the average by user

user_avgs <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu_hat - b_i))

# Compute the predicted ratings on validation dataset

rmse_movie_user_model <- validation %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   mutate(pred = mu_hat + b_i + b_u) %>%
   pull(pred)

rmse_movie_user_model_result <- RMSE(validation$rating, rmse_movie_user_model)

# Adding the results to the results dataset

results <- results %>% add_row(model="Movie+User Based Model", RMSE=rmse_movie_user_model_result)
```

The RMSE on the `validation` dataset is **0.8635** and this is very good. The Movie+User Based Model reaches the desidered performance but applying the regularization techniques, can improve the performance just a little.

3.4 Movie + User + Genre Model, the Genre Popularity

The formula used is:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_{u,g} + \epsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i. The b_u is a measure for the mildness of user u, i.e. the bias of user u. The $b_{u,g}$ is a measure for how much a user u likes the genre g.

```
# Calculate the average of all movies

mu_hat <- mean(edx$rating)

# Calculate the average by movie

movie_avgs <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))

# Calculate the average by user

user_avgs <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   group_by(userId) %>%
```

```
   summarize(b_u = mean(rating - mu_hat - b_i))

genre_pop <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   group_by(genre) %>%
   summarize(b_u_g = mean(rating - mu_hat - b_i - b_u))

# Compute the predicted ratings on validation dataset

rmse_movie_user_genre_model <- validation %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   left_join(genre_pop, by='genre') %>%
   mutate(pred = mu_hat + b_i + b_u + b_u_g) %>%
   pull(pred)

rmse_movie_user_genre_model_result <- RMSE(validation$rating, rmse_movie_user_genre_model)

# Adding the results to the results dataset

results <- results %>% add_row(model="Movie+User+Genre Based Model", RMSE=rmse_movie_user_genre_model_r
```

The RMSE on the `validation` dataset is **0.8634** and this is very good. The Movie+User+Genre Based Model reaches the desidered performance but adding the `genre` predictor, doesn't improve significantly the model's performance. Applying the regularization techniques, can improve the performance just a little.

3.5 Regularization

The regularization method allows us to add a penalty λ (lambda) to penalizes movies with large estimates from a small sample size. In order to optimize b_i, it necessary to use this equation:

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i)^2 + \lambda \sum_i b_i^2$$

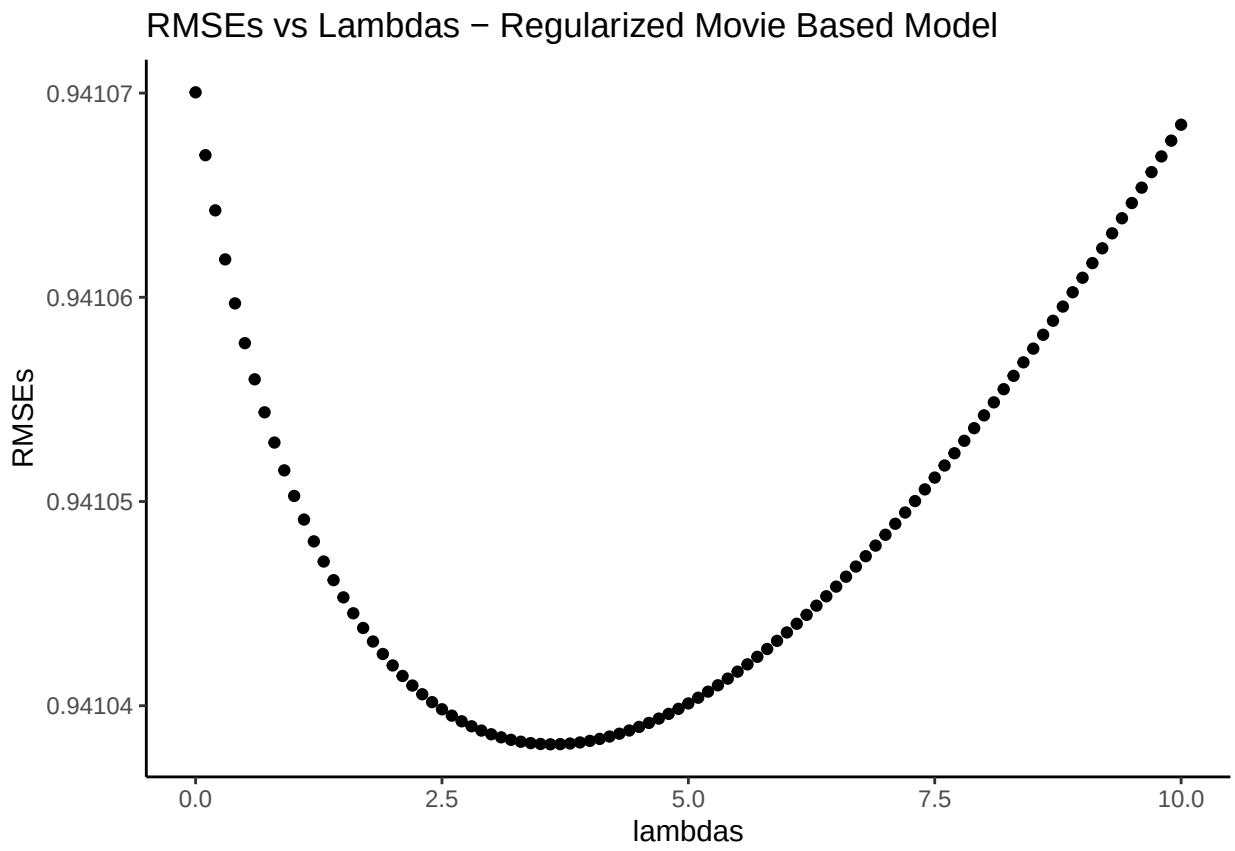
reduced to this equation:

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

3.5.1 Regularized Movie-Based Model

```
# Calculate the average of all movies

mu_hat <- mean(edx$rating)

# Define a table of lambdas

lambdas <- seq(0, 10, 0.1)

# Compute the predicted ratings on validation dataset using different values of lambda
```

```r
rmses <- sapply(lambdas, function(lambda) {

  # Calculate the average by user

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

   # Compute the predicted ratings on validation dataset

   predicted_ratings <- validation %>%
      left_join(b_i, by='movieId') %>%
      mutate(pred = mu_hat + b_i) %>%
      pull(pred)

   # Predict the RMSE on the validation set

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

# plot the result of lambdas

df <- data.frame(RMSE = rmses, lambdas = lambdas)

ggplot(df, aes(lambdas, rmses)) +
   theme_classic()  +
   geom_point() +
   labs(title = "RMSEs vs Lambdas - Regularized Movie Based Model",
        y = "RMSEs",
        x = "lambdas")
```



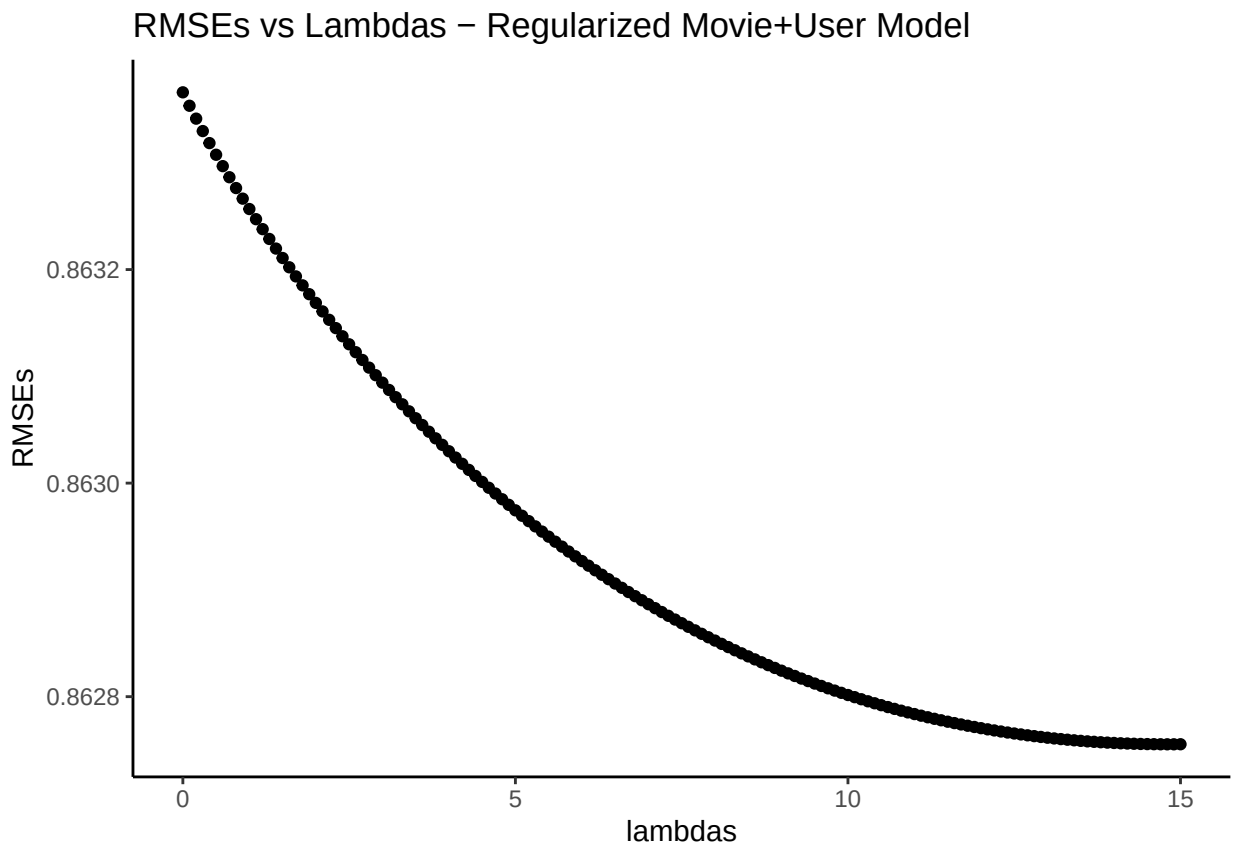

```
# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]

# Predict the RMSE on the validation set

rmse_regularized_movie_model <- min(rmses)

# Adding the results to the results dataset

results <- results %>% add_row(model="Regularized Movie-Based Model", RMSE=rmse_regularized_movie_model
```

The RMSE on the `validation` dataset is **0.8635** and this is very good. The Movie+User Based Model reaches the desidered performance but applying the regularization techniques, can improve the performance just a little.

3.5.2 Regularized Movie+User Model

```
# Calculate the average of all movies

mu_hat <- mean(edx$rating)

# Define a table of lambdas

lambdas <- seq(0, 15, 0.1)
```

```r
# Compute the predicted ratings on validation dataset using different values of lambda

rmses <- sapply(lambdas, function(lambda) {

   # Calculate the average by user

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

   # Calculate the average by user

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

   # Compute the predicted ratings on validation dataset

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

   # Predict the RMSE on the validation set

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

# plot the result of lambdas

df <- data.frame(RMSE = rmses, lambdas = lambdas)

ggplot(df, aes(lambdas, rmses)) +
   theme_classic()  +
   geom_point() +
   labs(title = "RMSEs vs Lambdas - Regularized Movie+User Model",
        y = "RMSEs",
        x = "lambdas")
```



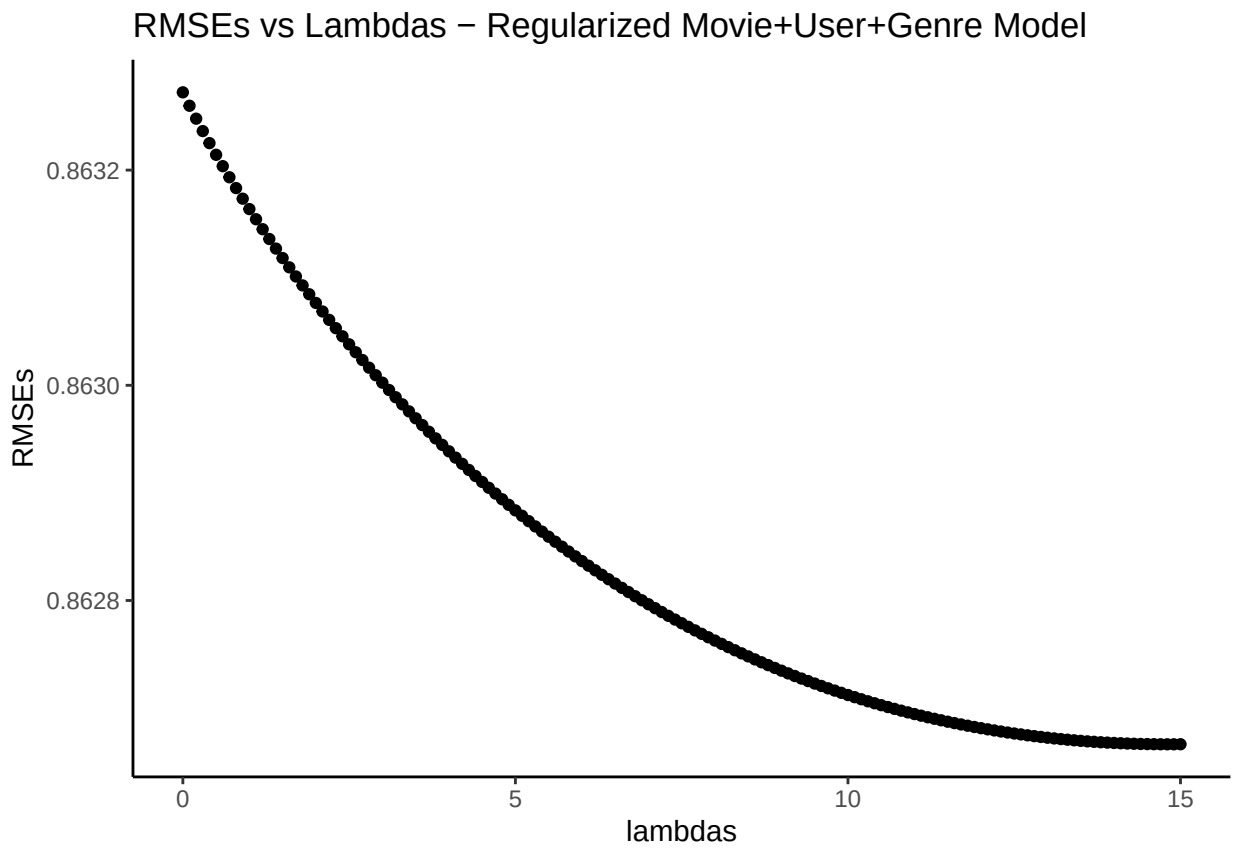

```
# Get the lambda value that minimize the RMSE

min_lambda <- lambdas[which.min(rmses)]

# Predict the RMSE on the validation set

rmse_regularized_movie_user_model <- min(rmses)

# Adding the results to the results dataset

results <- results %>% add_row(model="Regularized Movie+User Based Model", RMSE=rmse_regularized_movie_
```

The RMSE on the `validation` dataset is **0.8629**. The Regularized Movie+User Based Model improves just a little the result of the Non-Regularized Model.

3.5.3 Regularized Movie+User+Genre Model

```
# Calculate the average of all movies

mu_hat <- mean(edx$rating)

# Define a table of lambdas

lambdas <- seq(0, 15, 0.1)
```

```r
# Compute the predicted ratings on validation dataset using different values of lambda

rmses <- sapply(lambdas, function(lambda) {

   # Calculate the average by user

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

   # Calculate the average by user

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

    b_u_g <- edx %>%
      left_join(b_i, by='movieId') %>%
      left_join(b_u, by='userId') %>%
      group_by(genre) %>%
      summarize(b_u_g = sum(rating - b_i - mu_hat - b_u) / (n() + lambda))

   # Compute the predicted ratings on validation dataset

   predicted_ratings <- validation %>%
      left_join(b_i, by='movieId') %>%
      left_join(b_u, by='userId') %>%
      left_join(b_u_g, by='genre') %>%
      mutate(pred = mu_hat + b_i + b_u + b_u_g) %>%
      pull(pred)

   # Predict the RMSE on the validation set

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

# plot the result of lambdas

df <- data.frame(RMSE = rmses, lambdas = lambdas)

ggplot(df, aes(lambdas, rmses)) +
   theme_classic()  +
   geom_point() +
   labs(title = "RMSEs vs Lambdas - Regularized Movie+User+Genre Model",
        y = "RMSEs",
        x = "lambdas")
```




```
# Get the lambda value that minimize the RMSE

min_lambda <- lambdas[which.min(rmses)]

# Predict the RMSE on the validation set

rmse_regularized_movie_user_genre_model <- min(rmses)

# Adding the results to the results dataset

results <- results %>% add_row(model="Regularized Movie+User+Genre Based Model", RMSE=rmse_regularized_
```

The RMSE on the `validation` dataset is **0.8628** and this is the best result of the builted models. The Regularized Movie+User+Genre Based Model improves just a little the result of the Non-Regularized Model. As the Non-Regularized Model, the `genre` predictor doesn't improve significantly the model's performance.

4 Results

This is the summary results for all the model builted, trained on `edx` dataset and validated on the `validation` dataset.

```
# Shows the results

results %>%
```

model	RMSE
Naive Mean-Baseline Model	1.0525579
Movie-Based Model	0.9410700
Movie+User Based Model	0.8633660
Movie+User+Genre Based Model	0.8632723
Regularized Movie-Based Model	0.9410381
Regularized Movie+User Based Model	0.8627554
Regularized Movie+User+Genre Based Model	0.8626664

```
kable() %>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
          position = "center",
          font_size = 10,
          full_width = FALSE)
```

5 Conclusion

After training different models, it's very clear that `movieId` and `userId` contribute more than the `genre` predictor. Without regularization, the model can archieves and overtakes the desidered peformance, but the best is the enemy of the good and applying regularization and adding the `genre` predictor, it make possible to reach a RSME of **0.8628** that is the best result for the trained models.