MovieLens Recommender System Capstone Project - Report

Sanjana.B

Contents

1 Executive Summary

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

The version of movielens dataset used for this final assignment contains approximately 10 Milions of movies ratings, divided in 8.5 Milions for training and 1.5 Milion for validation or testing. It is a small subset of a much larger 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 built on this dataset are evaluated and chosen

based on the RMSE - Root Mean Squared Error that should be at least lower than **0.8649**.

For accomplishing this goal, the **Movie+User Model** is capable to reach a RMSE of **0.863**, that is really good.

2 Method and 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).

Users	Movies
<*int*>	<*int*>
1 69878	10677

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

- **userId** (class integer) Unique user identification number .
- **movieId** (class numeric) Unique movie identification number
- **rating** (class numeric) One user‘s rating of one movie. Ratings are given out of a 5(1,1.5,2…)
- **timestamp** (class integer) Timestamp for one particular rating by one particular user.
- **title** (class character) Title of each movie with the release year.
- **genres** (class character) list of genres of each movie separated by pipe(|).

First 6 Rows of edx dataset

	movieId	title	year	genres
1	31	Dangerous Minds	1995	Drama
2	1029	Dumbo	1941	Animation|Children|Drama|Musical
3	1061	Sleepers	1996	Thriller
4	1129	Escape from New York	1981	Action|Adventure|Sci-Fi|Thriller
5	1172	Cinema Paradiso (Nuovo cinema Paradiso)	1989	Drama
6	1263	Deer Hunter	1978	Drama|War

	userId	rating	timestamp
1	1	2.5	1260759144
2	1	3.0	1260759179
3	1	3.0	1260759182
4	1	2.0	1260759185
5	1	4.0	1260759205
6	1	2.0	1260759151

2.2 Dataset Pre-Processing and Feature Engineering

After some initial data exploration, we see that each movie has many different genres. It's necessary to extract and separate them for better consistency. We also notice that the title contains the year of the movie release which can be made into a separate column in the data frame to be used for further analysis. Finally, we can obtain the year and the month for each rating from the timestamp column.

The pre-processing phase is comprised of the following:
1. Format timestamp to a human readable date
2. Extract the month and the year from the date
3. Extract the release year from the title
4. Separate each genre from the list of genres of each movie. It increases the size of both datasets.

After preprocessing the data, edx dataset looks like this:

Processed edx datadaset

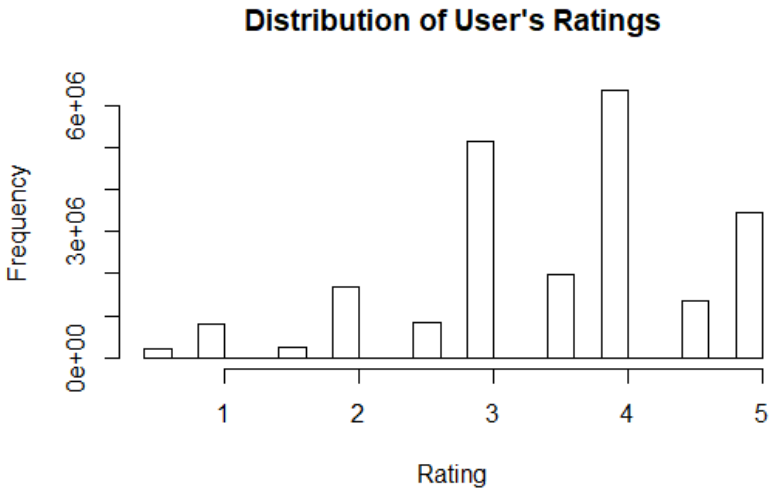
	userId	movieId	rating	title	release	year_rated	month_rated	genre
	<int>	*<dbl>*	*<dbl>*	*<chr>*	*<int>*	*<dbl>*	*<dbl>*	*<chr>*
1	1	122	5	Boomerang (1992)	1992	96	8	Comedy
2	1	122	5	Boomerang (1992)	1992	96	8	Romance
3	1	185	5	Net, The (1995)	1995	96	8	Action
4	1	185	5	Net, The (1995)	1995	96	8	Crime
5	1	185	5	Net, The (1992)	1995	96	8	Thriller

2.3 Rating frequency Distributions:

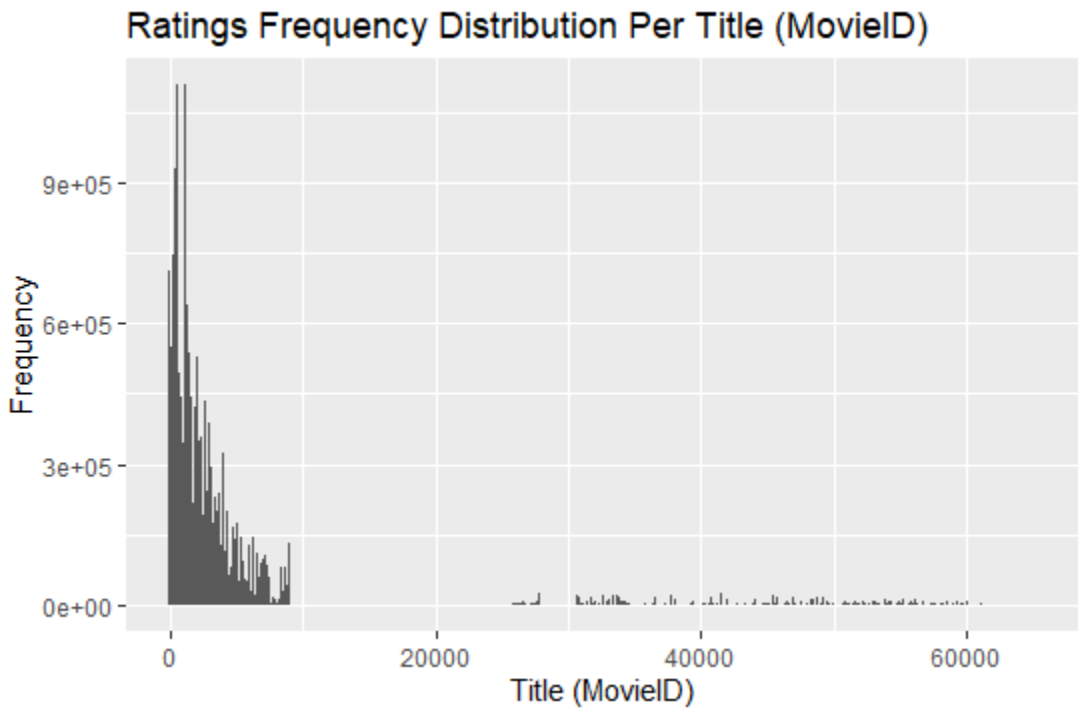
Overview of Rating Distribution

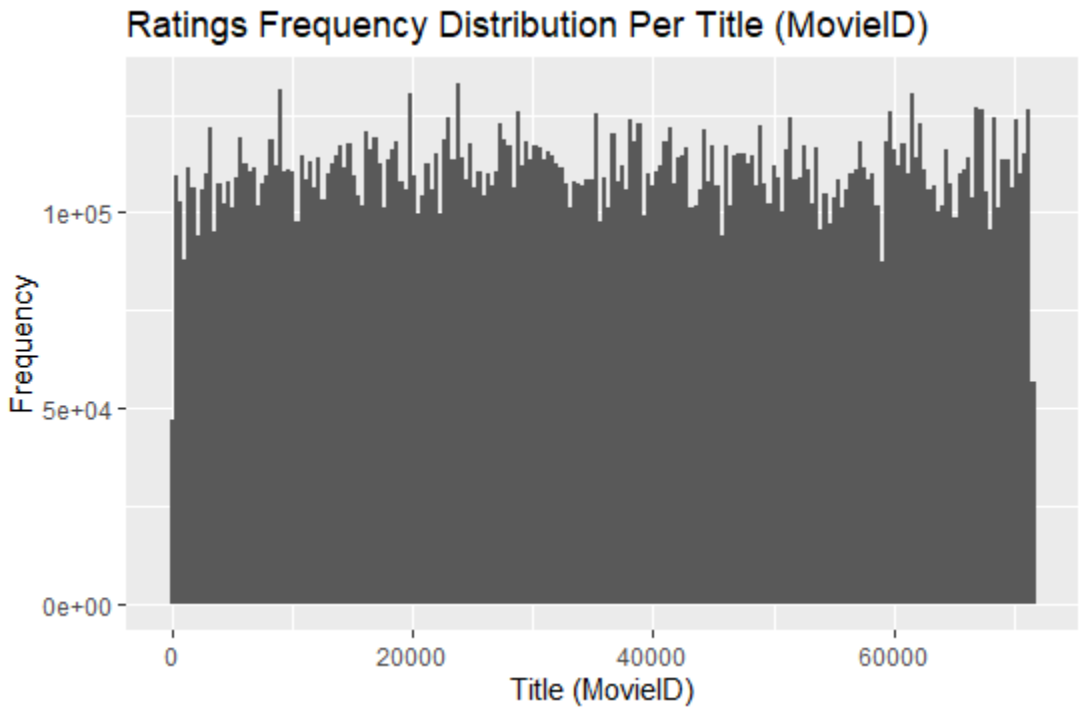
According to the given histogram, it can be seen that there are a small number of negative votes. It is noticed that half-Star votes are rarer than “Full-Star” votes.

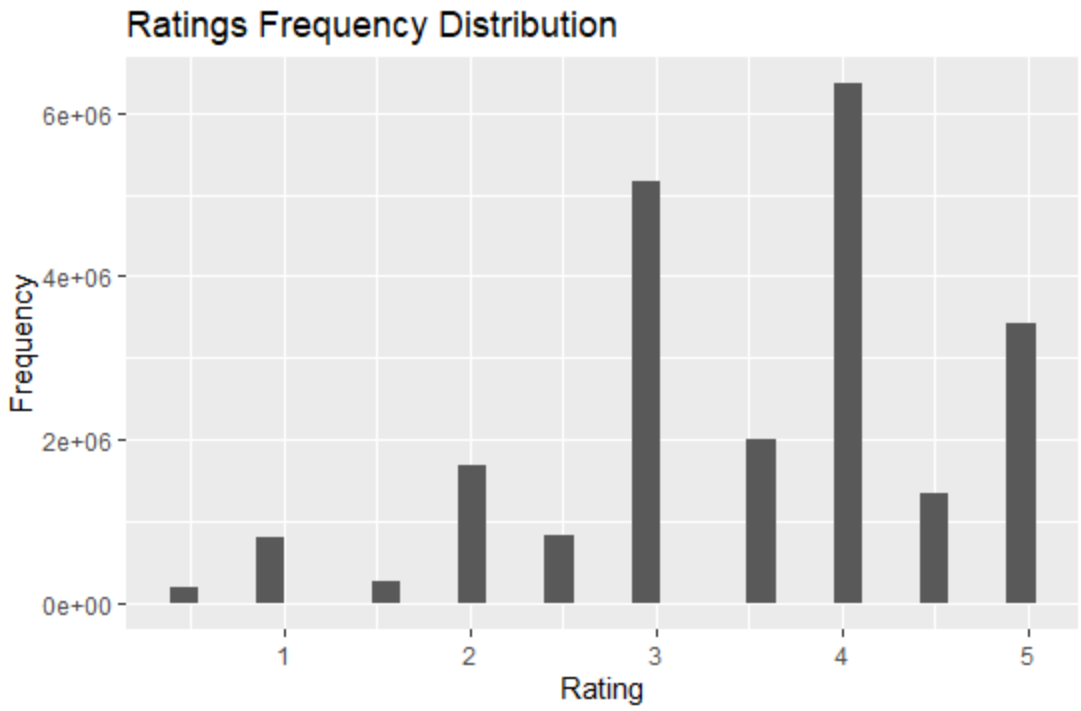
Distribution of User Ratings

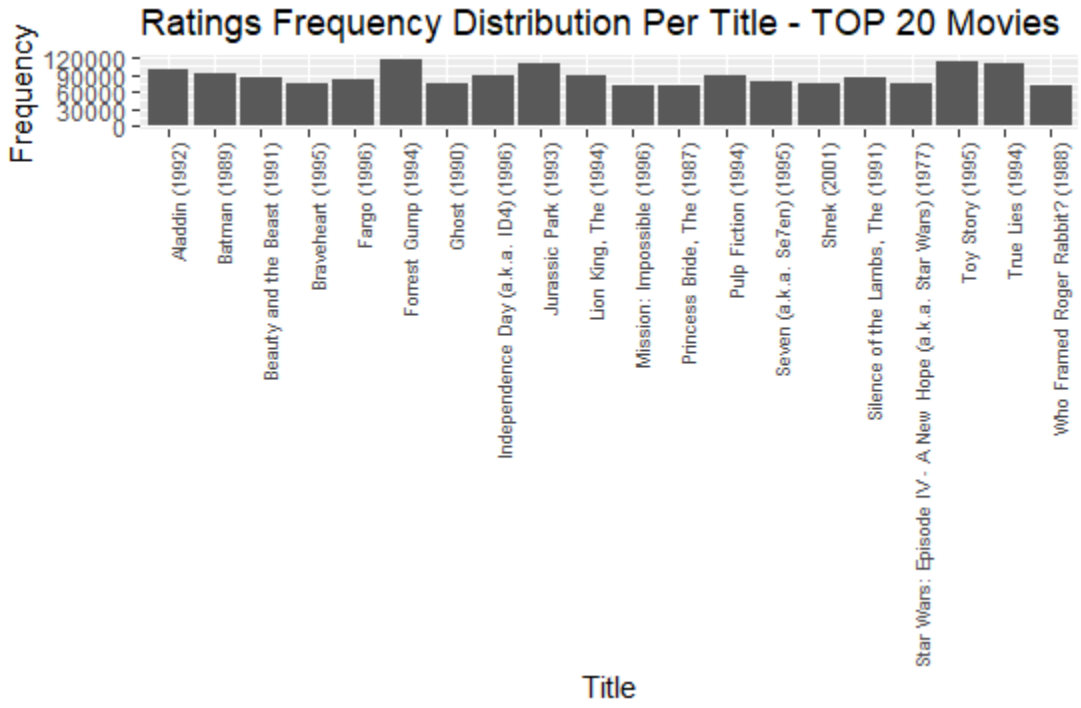


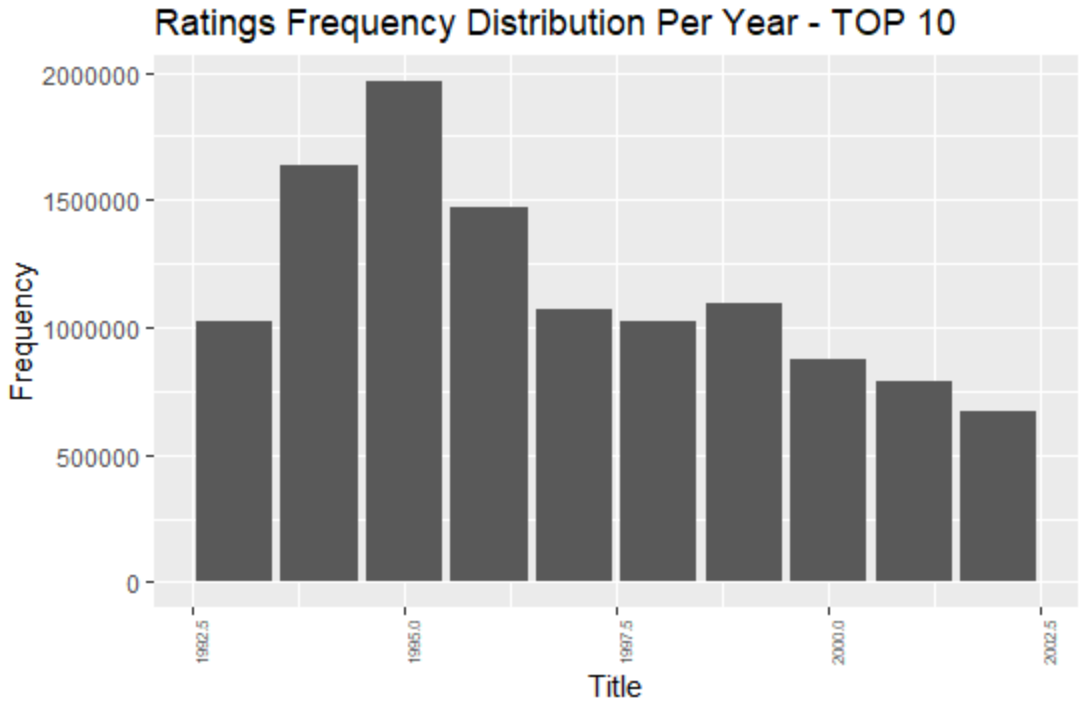
Rating frequency Distribution of User Ratings per Title

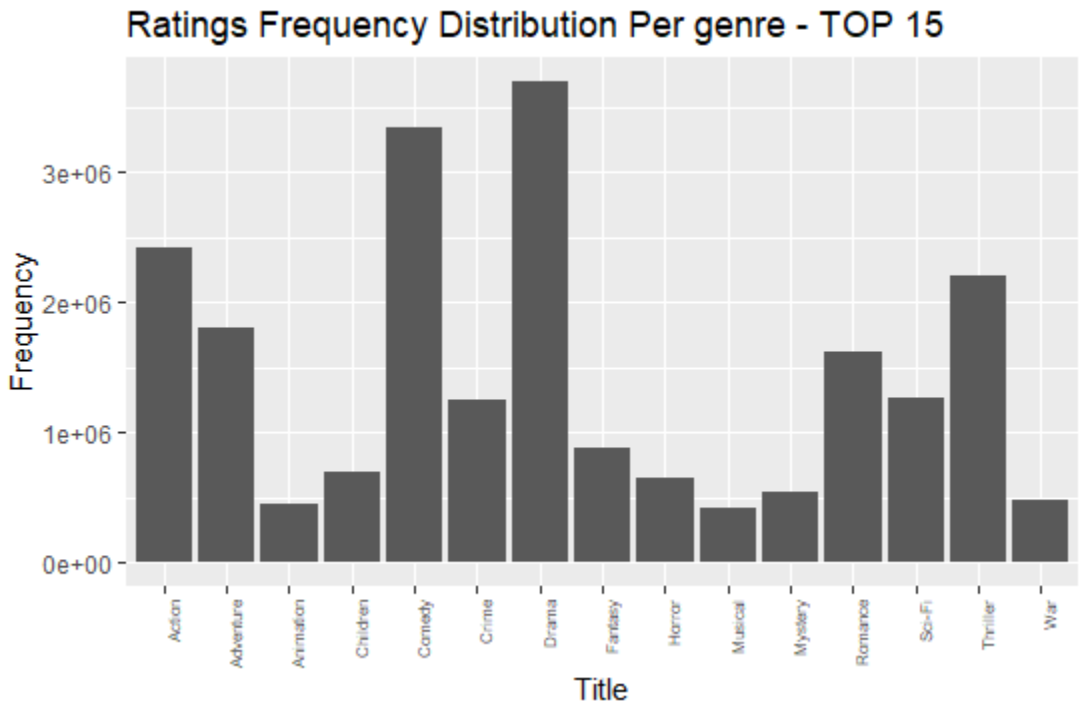



Ratings Frequency Distribution Per Title (MovieID)
Frequency
1e+05
5e+04
0e+00
0
20000
40000
60000
Title (MovieID)


Ratings Frequency Distribution
Frequency
6e+06
4e+06
2e+06
0e+00
1
2
3
4
5
Rating


Ratings Frequency Distribution Per Title - TOP 20 Movies
Frequency
120000
90000
60000
30000
0
Aladdin (1992)
Batman (1989)
Beauty and the Beast (1991)
Braveheart (1995)
Fargo (1996)
Forrest Gump (1994)
Ghost (1990)
Independence Day (a.k.a. ID4) (1996)
Jurassic Park (1993)
Lion King, The (1994)
Mission: Impossible (1996)
Princess Bride, The (1987)
Pulp Fiction (1994)
Seven (a.k.a. Se7en) (1995)
Shrek (2001)
Silence of the Lambs, The (1991)
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
Toy Story (1995)
True Lies (1994)
Who Framed Roger Rabbit? (1988)
Title


Ratings Frequency Distribution Per Year - TOP 10
Frequency
2000000
1500000
1000000
500000
0
1992.5
1995.0
1997.5
2000.0
2002.5
Title



Rating frequency distribution per title(top 10)

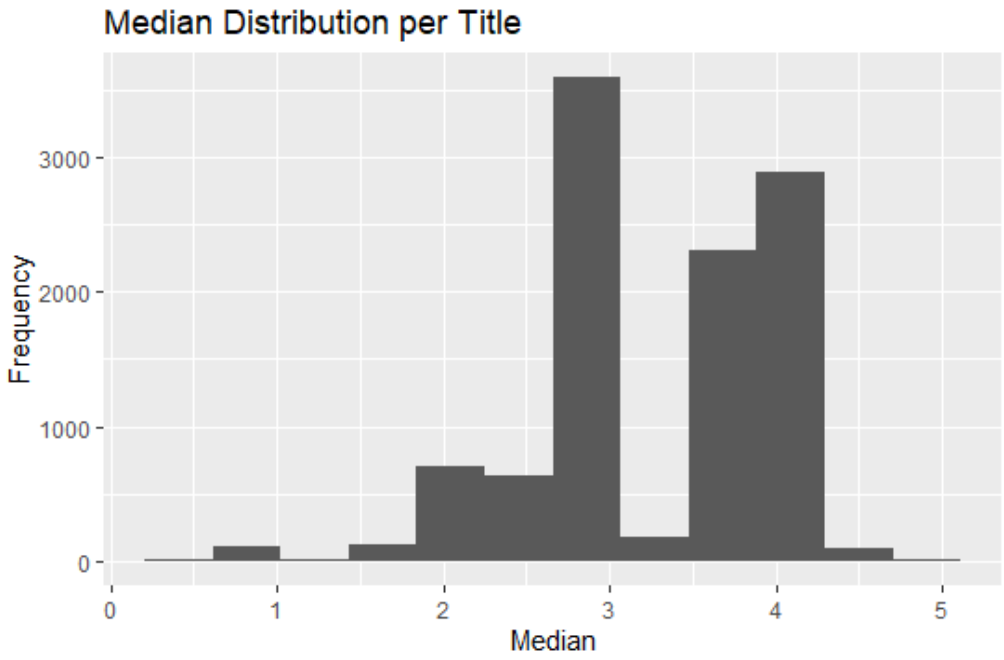
title	count
<chr>	*<int>*
1 Forrest Gump (1994)	117180
2 Toy Story (1995)	112450
3 Jurassic Park (1993)	111016
4 True Lies (1994)	107790
5 Aladdin (1992)	99880
6 Batman (1989)	91788
7 Pulp Fiction (1994)	89058
8 Lion King, The (1994)	89005
9 Independence Day (a.k.a. ID4) (1996)	88848
10 Silence of the Lambs, The (1991)	85665

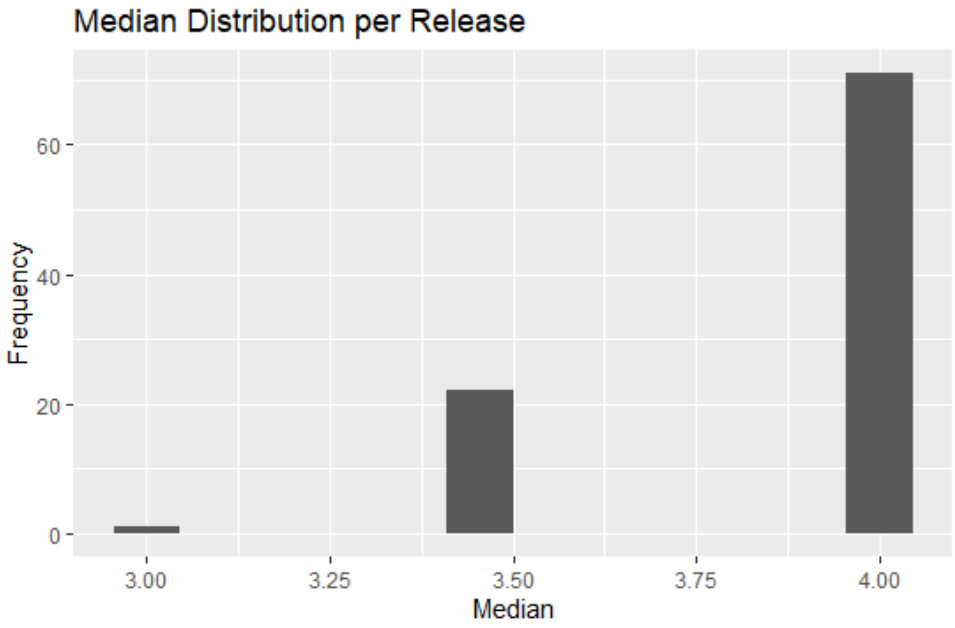
Rating frequency distribution per year

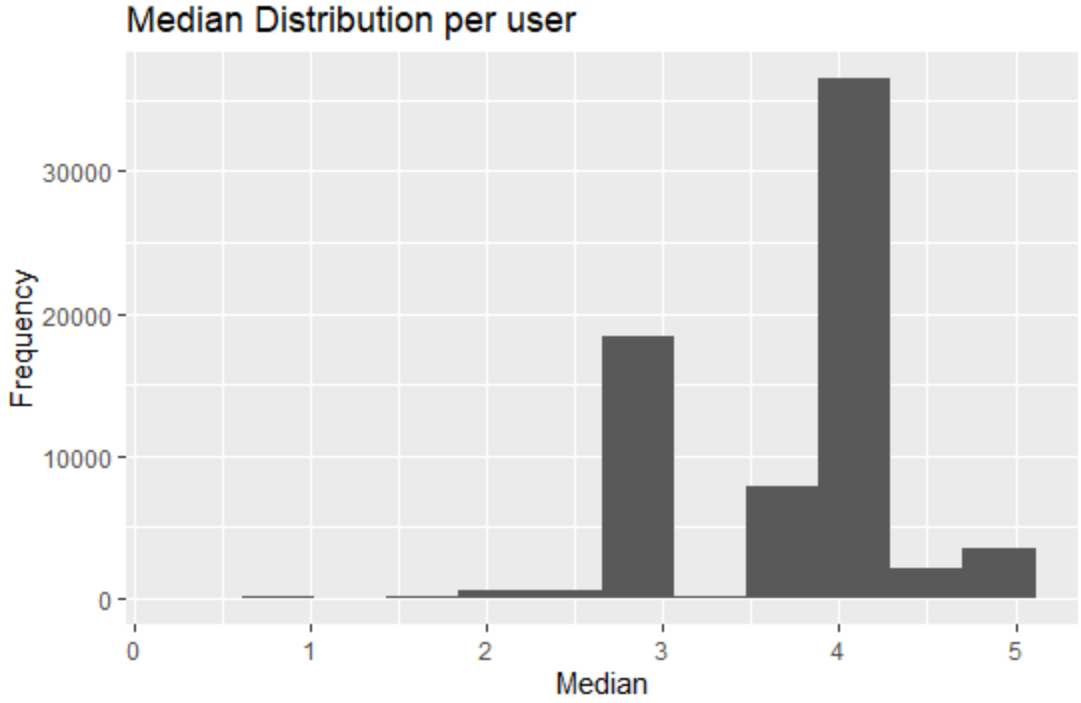
	release	count
	<*int*>	<*int*>
1	1995	1968372
2	1994	1635409
3	1996	1475053
4	1999	1094170
5	1997	1074284
6	1998	1025660
7	1993	1025377
8	2000	878117
9	2001	786606
10	2002	671877

Median rating Distributions

Median rating distribution histograms




Median Distribution per Release
Frequency
60
40
20
0
3.00
3.25
3.50
3.75
4.00
Median


Median Distribution per user
Frequency
30000
20000
10000
0
0
1
2
3
4
5
Median

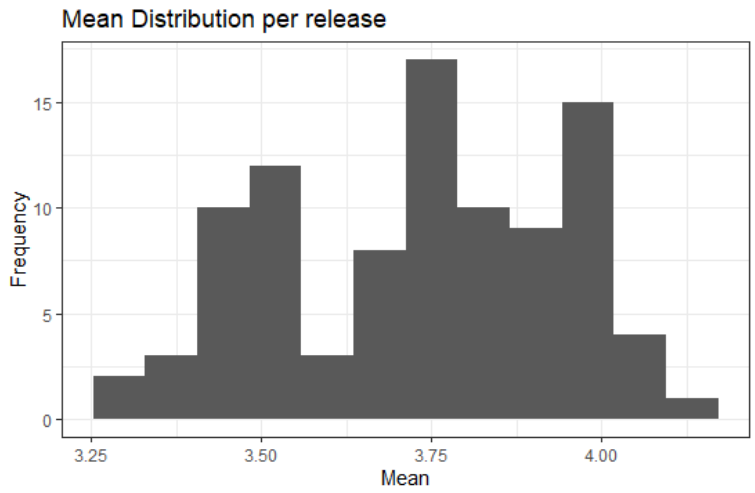
Median rating distribution tables

2.3.1 Median distribution per title(top 10)

	title	median
	<chr>	*<dbl>*
1	Blue Light, The (Das Blaue Licht) (1932)	5
2	Class, The (Entre les Murs) (2008)	5
3	Constantine's Sword (2007)	5
4	Fighting Elegy (Kenka erejii) (1966)	5
5	Godfather, The (1972)	5
6	Kids of Survival (1996)	5
7	More (1998)	5
8	Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	5
9	Shadows of Forgotten Ancestors (1964)	5
10	Shawshank Redemption, The (1994)	5

Median distribution per year(top 10)

	release	median
	<int>	*<dbl>*
1	1916	4
2	1918	4
3	1920	4
4	1921	4
5	1922	4
6	1923	4
7	1924	4
8	1925	4
9	1926	4
10	1927	4

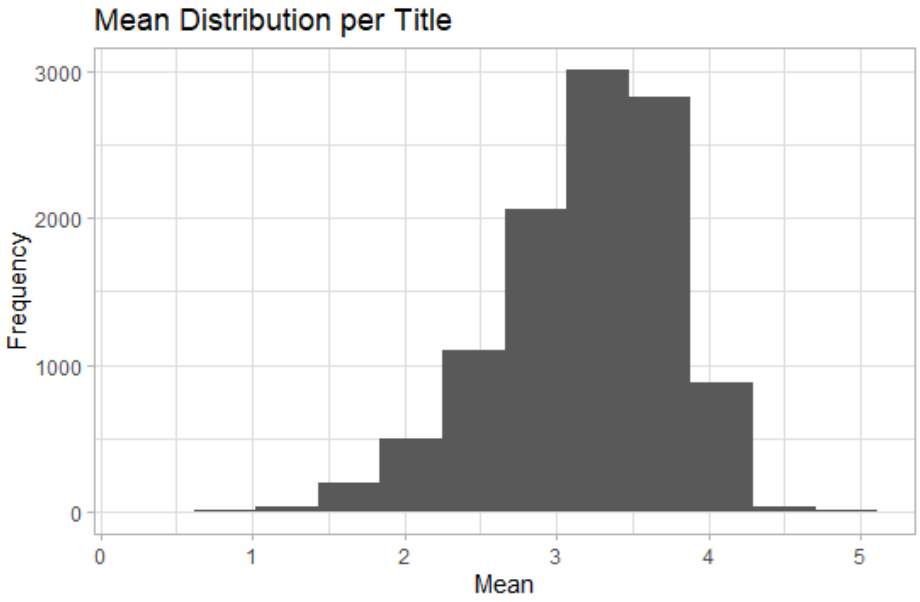


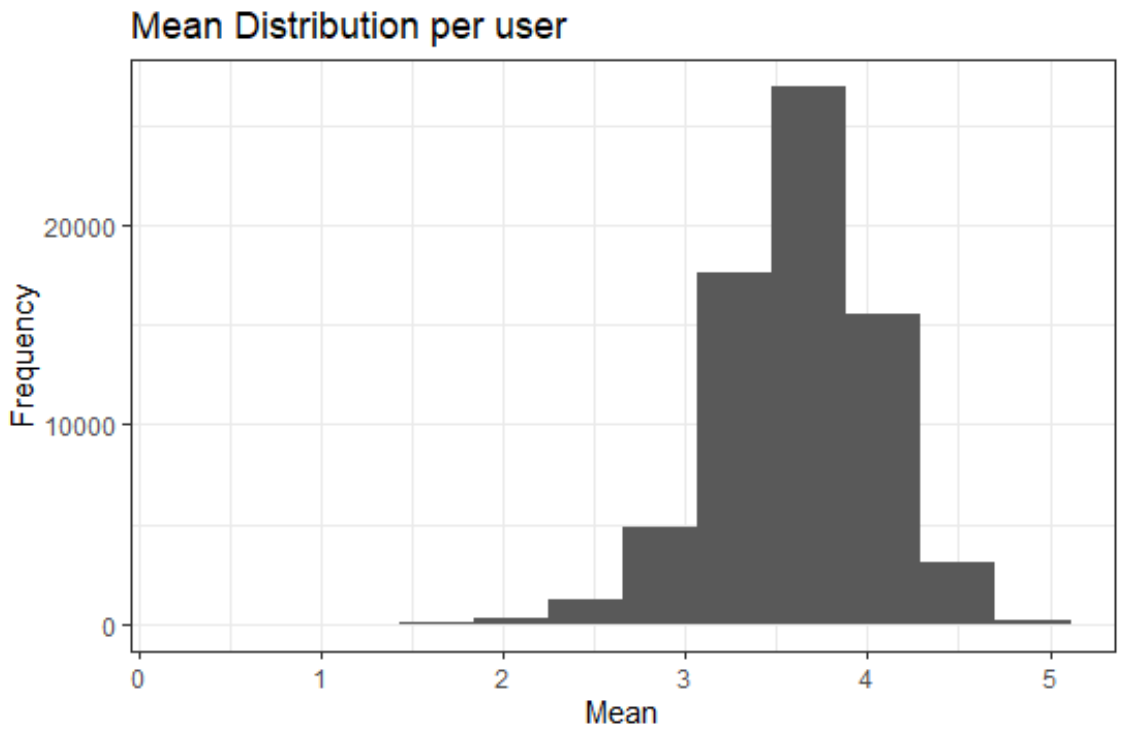
Median distribution per user(top 10)

	userId	median
	<*int*>	<*dbl*>
1	1	5
2	4	5
3	30	5
4	43	5
5	44	5
6	51	5
7	56	5
8	98	5
9	104	5
10	123	5

Mean Rating distributions

Mean Rating distribution histograms





Mean Rating Distribution Tables

Mean Rating Distribution Per Title (MovieID) - Top 10

	title	mean
	<chr>	*<dbl>*
1	Blue Light, The (Das Blaue Licht) (1932)	5
2	Class, The (Entre les Murs) (2008)	5
3	Constantine's Sword (2007)	5
4	Fighting Elegy (Kenka erejii) (1966)	5
5	Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	5
6	Shadows of Forgotten Ancestors (1964)	5
7	Sun Alley (Sonnenallee) (1999)	5
8	Sun Shines Bright, The (1953)	5
9	Human Condition II, The (Ningen no joken II) (1959)	4.75
10	Human Condition III, The (Ningen no joken III) (1961)	4.75

Mean Rating Distribution Per Release Year- Top 10

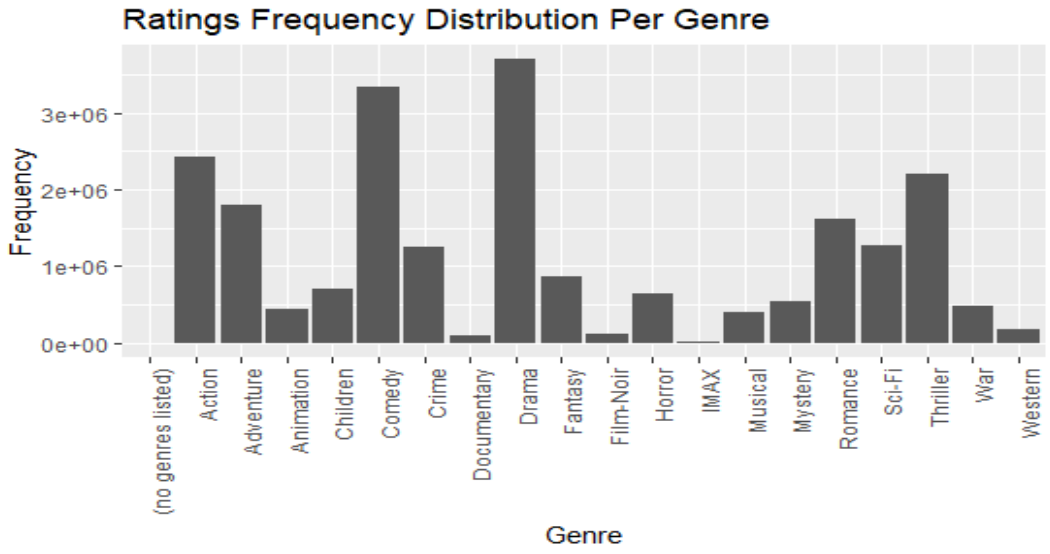
	release	mean
	<int>	*<dbl>*
1	1931	4.11
2	1934	4.08
3	1946	4.06
4	1944	4.05
5	1957	4.03
6	1927	4.01
7	1942	4.00
8	1952	4.00
9	1949	4.00
10	1962	3.99

Mean Rating Distribution Per User ID- Top 10

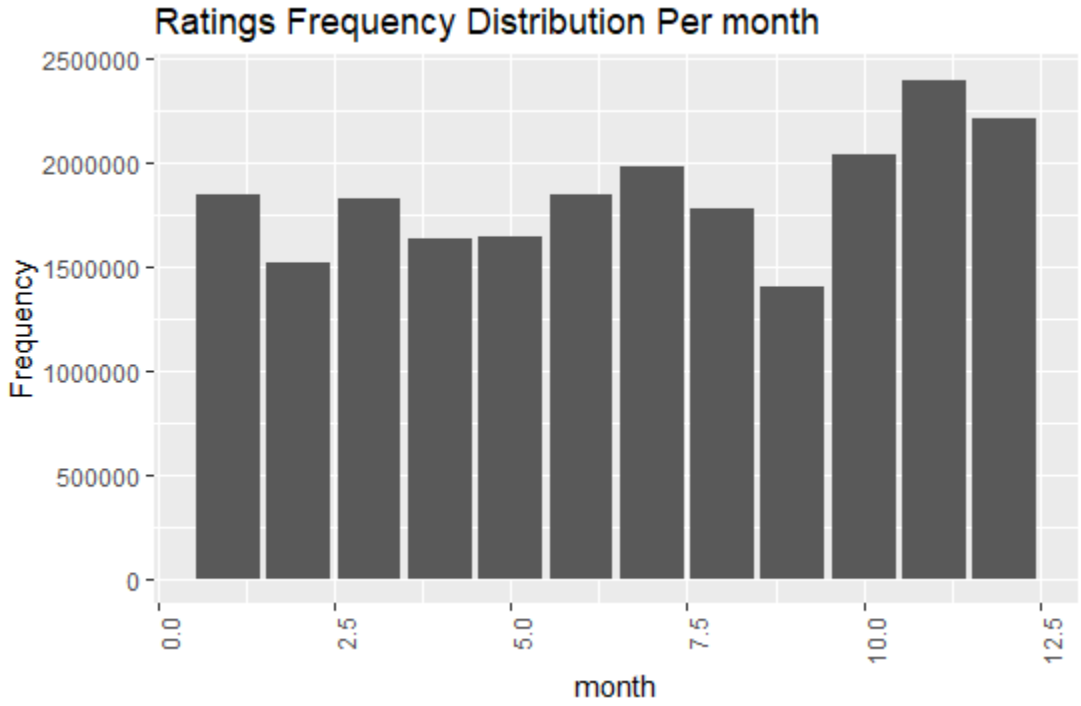
	userId	mean
	<int>	*<dbl>*
1	1	5
2	7984	5
3	11884	5
4	13027	5
5	13513	5
6	13524	5
7	15575	5
8	18965	5
9	22045	5
10	26308	5

Genre Analysis

2.4. Rating Distribution per Genre



Rating Distribution per Month



Rating Distribution Tables

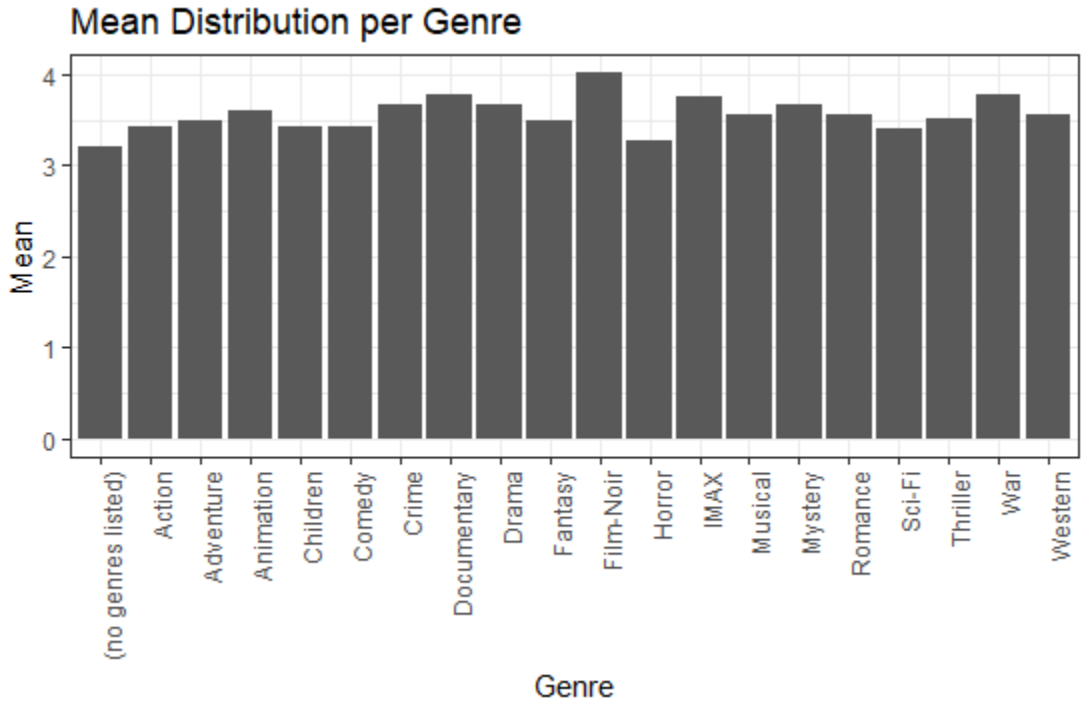
Ratings Frequency Distribution Per Genre

	genre	count
	<chr>	*<int>*
1	Drama	3693674
2	Comedy	3343505
3	Action	2418271
4	Thriller	2197031
5	Adventure	1802802
6	Romance	1616899
7	Sci-Fi	1267340
8	Crime	1254235
9	Fantasy	874268
10	Children	696914
11	Horror	652597
12	Mystery	537130
13	War	482621
14	Animation	440852
15	Musical	408685
16	Western	178863
17	Film-Noir	112168
18	Documentary	87957
19	IMAX	7739
20	(no genres listed)	5

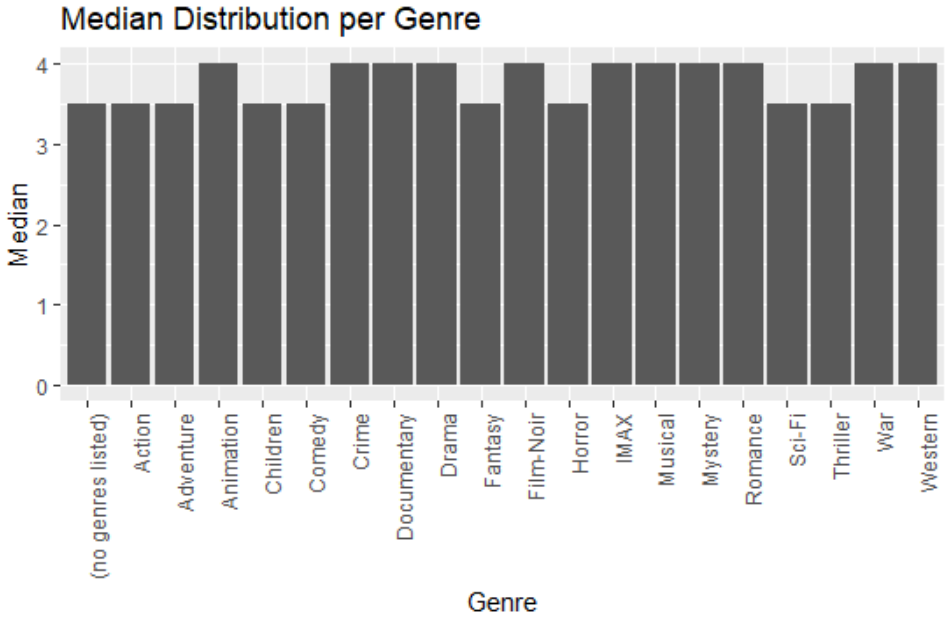
Ratings Frequency Distribution Per Month

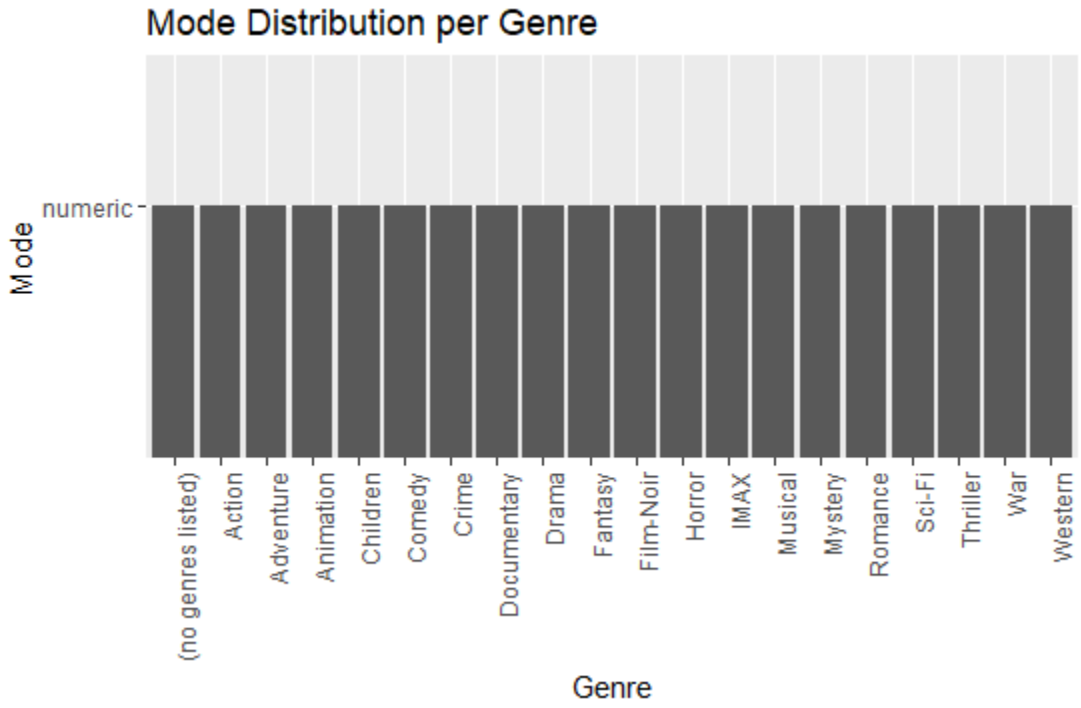
	month_rated	count
	<*dbl*>	<*int*>
1	11	2394359
2	12	2203980
3	10	2035272
4	7	1972738
5	1	1843616
6	6	1837634
7	3	1827162
8	8	1773138
9	5	1638183
10	4	1633203
11	2	1515358
12	9	1398913

2.4.2 Mean Distribution per Genre



2.4.3 Median Distribution per Genre





Median distribution per genre

genre	median
<*chr*>	<*dbl*>
1 Animation	4
2 Crime	4
3 Documentary	4
4 Drama	4
5 Film-Noir	4
6 IMAX	4
7 Musical	4
8 Mystery	4
9 Romance	4
10 War	4

Mean distribution per genre

genre	mean
<*chr*>	<*dbl*>
1 Film-Noir	4.01
2 Documentary	3.78
3 War	3.78
4 IMAX	3.76
5 Mystery	3.68
6 Drama	3.67
7 Crime	3.67
8 Animation	3.60
9 Musical	3.56
10 Western	3.56

3 Analysis - Model Building and Evaluation

3.1 Naive Baseline Model

The simplest model possible, is a Naive Model that predicts the mean for all cases. In this case, the mean is **3.512465**, approximately **3.5**.

3.1.1 Naive Mean-Baseline Model

The formula used is:

$Yu,i = \hat{}\mu + "u,i$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0.
Mu_hat is 3.527021 and the rmse on the validation set is **1.052698**, approximately **1.05**. It is larg er than 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 type of movie. In this case the movies that are rated higher or lower respect to each other.
The formula used is:
$Y_{u,i} = \hat{\mu} + b_i + \_{u,i}$
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 RMSE on the validation dataset is **0.9417822**. It better than the Naive Mean-Baseline Model but it is also much higher than 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 considers that each user has different preference of movies and rate differently according to their perspectives.
The formula used is:
$Y_{u,i} = \hat{\mu} + b_i + b_u + \_{u,i}$
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 RMSE on the validation dataset is 0.8639665 which is very good. The Movie+User Based Model
has obtained the required performance but by applying regularization, the model can be improved by some amount.

4 Results

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

	model	RMSE
1	Naive Mean-Baseline Model	1.0526979
2	Movie-Based Model	0.9417822
3	Movie+User Based Model	0.8639665

5 Conclusion

After training different models, it's very clear that movieId and userId are the main contributors for prediction. Without regularization, the model has achieved the desired performance, but by applying regularization and adding the genre predictor, performance can be improved and RMSE can be reduced.

6 Appendix

```
##Installing required packages
install.packages("forcats")
install.packages("kableExtra")

#Loading the libraries for the project
library(ggplot2)
library(kableExtra)
library(stringr)
library(tidyr)
library(tibble)
library(tidyverse)
library(dslabs)
library(dbplyr)
library(caret)
library(broom)

# Create proj set, valid set, and final file

m <- tempfile()

#Downloading data from web to m
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", m)

#Unzipping the file and storing it in movie_data
movie_data <- str_split_fixed(readLines(unzip(m, "ml-10M100K/movies.dat")), "\\::", 3)

#Assigning column names to movie_data
colnames(movie_data) <- c("movieId", "title", "genres")

#Converting movie_data to a data frame
movie_data <- as.data.frame(movie_data) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],title = as.character(title),genres = as.character(genres))
```

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

#Forming the movielens table by combining rating and movie details
movielens <- left_join(rating_table, movie_data, by = "movieId")

#Setting seed to 1
set.seed(1)

#Splitting movie data into training and test sets(proj and valid sets respectively)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.15, list = FALSE)
proj <- movielens[-test_index,]
test <- movielens[test_index,]
head(proj)
head(test)

# Validation set will be 15% of MovieLens data
# Make sure userId and movieId in valid set are also in proj set
valid <- test %>% semi_join(proj, by = "movieId") %>% semi_join(proj, by = "userId")

# Add rows removed from valid set back into proj set

removed <- anti_join(test, valid)
proj <- rbind(proj, removed)
rm(m, rating_table, movie_data, test_index, test, movielens, removed)

# Exploratory Data Analysis

## Inital data Exploration

#The 10 Millions dataset is divided into two dataset: "proj" for training purpose and "valid" for the validation phase.


#**proj dataset**
#Defining the root mean square error function
RMSE <- function(true_ratings = NULL, predicted_ratings = NULL) {
  +     sqrt(mean((true_ratings - predicted_ratings)^2))}
```

```
#Viewing fisrt 6 rows of proj and valid dataframes
head(proj)
head(valid)

#Finding the number of unique users and movies in the proj dataframe
proj %>% summarize(Users = n_distinct(userId), Movies = n_distinct(movieId))

#***********Dataset Pre-Processing and Feature Engineering**********************


# Convert timestamp to a human readable date


#Adding a date column to both frames with time zone as Greenwich Mean Time
valid$date <- as.POSIXct(valid$timestamp, origin='1970-01-01',tz="GMT")
proj$date <- as.POSIXct(proj$timestamp,origin='1970-01-01',tz="GMT")

#Viewing first 6 rows of proj and valid dataframes
head(valid)
head(proj)

#Adding month and year columns to both dataframes
valid$month_rated <- format(valid$date,"%m")
valid$year_rated <- format(valid$date,"%y")
proj$year_rated <- format(proj$date,"%y")
proj$month_rated <- format(proj$date,"%m")

#valid <- valid %>% mutate(title = str_squish(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)))

#Adding release date column to valid
valid <- valid %>%mutate(title = str_trim(title)) %>%extract(title, c("titleTemp", "release"),regex = "^(.*) \\(([0-9 \\-]*)\\)$",remove = F) %>%mutate(release = if_else(str_length(release) > 4,as.numeric(str_split(release, "-",simplify = T)[1]),as.numeric(release))) %>%mutate(title = if_else(is.na(titleTemp),title,titleTemp))
valid<-valid %>% select(-titleTemp)
```

```
#Viewing fisrt 6 rows of proj and valid dataframes
head(valid)
head(proj)

#Adding release date column to proj
proj <- proj %>% mutate(title = str_squish(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)))
proj<-proj %>% select(-titleTemp)
head(proj)

# Extract the genre in valid datasets
valid <- valid %>%mutate(genre = fct_explicit_na(genres,na_level = "(no genres listed)")) %>%separate_rows(genre, sep = "\\|")

# Extract the genre in proj datasets
proj <- proj %>% mutate(genre = fct_explicit_na(genres,na_level = "(no genres listed)")) %>% separate_rows(genre,sep = "\\|")

#Viewing fisrt 6 rows of proj and valid dataframes
head(proj)
head(valid)

#Removing the unnecessary columns in both data frames
valid <- valid %>% select(-genres,-timestamp,-date)
proj <- proj %>% select(-genres,-timestamp,-date)

# Convert the columns into the desired data type

valid$month_rated <- as.numeric(valid$month_rated)

valid$year_rated <- as.numeric(valid$year_rated)

proj$year_rated <- as.numeric(proj$year_rated)

proj$month_rated <- as.numeric(proj$month_rated)

#******************Processed proj datadaset*****************************
```

```
head(proj)

#******************Processed valid datadaset**************************

head(valid)

#***************END OF DATA PROCESSING*************************************

#********MOVIE AND RATING HISTOGRAMS*************************************


#movies released per year
hist(proj$release)

#rating distribution
hist(proj$rating, main="Distribution of User's Ratings", xlab="Rating")

#*********RATING FREQUENCY DISTRIBUTION HISTOGRAMS*****

### Numbers of Ratings per Movie

ggplot(proj, aes(movieId)) + theme_grey()  + geom_histogram(bins=700) + labs(title = "Ratings Frequency Distribution Per Title (MovieID)",x = "Title (MovieID)",y = "Frequency")


#Number of ratings per user id

ggplot(proj, aes(userId)) +theme_grey()  +geom_histogram(bins=200) +labs(title = "Ratings Frequency Distribution Per Title (MovieID)",x = "Title (MovieID)",y = "Frequency")

#*********************RATING FREQUENCY DISTRIBUTION PLOTS************
#Rating frequency distribution

ggplot(proj, aes(rating)) + theme_grey()  + geom_histogram() +labs(title = "Ratings Frequency Distribution ", x = "Rating", y = "Frequency")

#Ratings Frequency Distribution Per Title - TOP 20 Movies

proj %>% group_by(title) %>% summarise(count = n()) %>% arrange(desc(count)) %>%head(n=20) %>% ggplot(aes(title, count)) +theme_gray()  +geom_col() +theme(axis.text.x
```

= element_text(angle = 90, hjust = 1, size = 7)) +labs(title = "Ratings Frequency Distribution Per Title - TOP 20 Movies",x = "Title",y = "Frequency")

#Ratings Frequency Distribution Per Year - TOP 10

proj %>% group_by(release) %>% summarise(count = n()) %>%arrange(desc(count)) %>%head(n=10) %>%ggplot(aes(release, count)) +theme_gray() +geom_col() +theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 6)) +labs(title = "Ratings Frequency Distribution Per Year - TOP 10",x = "Title", y = "Frequency")

#Ratings Frequency Distribution Per genre - TOP 15

proj %>%group_by(genre) %>%summarise(count = n()) %>%arrange(desc(count)) %>% head(n=15) %>%ggplot(aes(genre, count)) +theme_gray() +geom_col() +theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 6)) +labs(title = "Ratings Frequency Distribution Per genre - TOP 15",x = "Title",y = "Frequency")
head(proj)

#**********RATING FREQUENCY DISTRIBUTION TABLES**********
#Rating frequency distribution per title

proj %>% group_by(title) %>% summarise(count = n()) %>% arrange(desc(count)) %>% head(n=25)

#Rating frequency distribution per release year

proj %>% group_by(release) %>% summarise(count = n()) %>% arrange(desc(count)) %>% head(n=25)

#**********MEDIAN DISTRIBUTION HISTOGRAMS**********
Median Distribution per Title (Movie) histogram

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

```
### Median distribution per release (year) histogram

proj %>%
      group_by(release) %>%
      summarise(median = median(rating)) %>%
      ggplot(aes(median)) +
      theme_gray()  +
      geom_histogram(bins=12) +
      labs(title = "Median Distribution per Release",x = "Median",y = "Frequency")

### Median distribution per user histogram

proj %>%
      group_by(userId) %>%
      summarise(median = median(rating)) %>%
      ggplot(aes(median)) +
      theme_gray()  +
      geom_histogram(bins=12) +
      labs(title = "Median Distribution per user",x = "Median",y = "Frequency")



#************************MEDIAN DISTRIBUTION TABLES*******************

### Median distribution per title(movie) table

proj %>%
   group_by(title) %>%
   summarise(median = median(rating)) %>%
  arrange(desc(median)) %>%
  head(n=25)

### Median distribution per release year table

proj %>%
      group_by(release) %>%
      summarise(median = median(rating)) %>%
      arrange(desc(median)) %>%
      head(n=25)
```

```
### Median distribution per user table

proj %>%
      group_by(userId) %>%
      summarise(median = median(rating)) %>%
      arrange(desc(median)) %>%
      head(n=25)

#*************************MEAN DISTRIBUTION HISTOGRAMS*****************

###Mean distribution per title histogram

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

###Mean distribution per release histogram

proj %>%
      group_by(release) %>%
      summarise(mean = mean(rating)) %>%
     ggplot(aes(mean)) +
     theme_bw()  +
     geom_histogram(bins=12) +
    labs(title = "Mean Distribution per release",x = "Mean", y = "Frequency")

###Mean distribution per user histogram

proj %>%
      group_by(userId) %>%
      summarise(mean = mean(rating)) %>%
      ggplot(aes(mean)) +
      theme_bw()  +
      geom_histogram(bins=12) +
      labs(title = "Mean Distribution per user",x = "Mean",y = "Frequency")
```

```
#************************MEAN DISTRIBUTION TABLES**********************

##MEAN DISTRIBUTION PER TITLE TABLE

proj %>%
      group_by(title) %>%
      summarise(mean = mean(rating)) %>%
      arrange(desc(mean)) %>%
      head(n=10)

##MEAN DISTRIBUTION PER RELEASE YEAR TABLE

proj %>%
      group_by(release) %>%
      summarise(mean = mean(rating)) %>%
      arrange(desc(mean)) %>%
      head(n=10)

##MEAN DISTRIBUTION PER USER TABLE
proj %>%
     group_by(userId) %>%
     summarise(mean = mean(rating)) %>%
     arrange(desc(mean)) %>%
     head(n=10)



##*****************************Rating distributions*****************************

### Rating Distribution per Genre

#**Overview of Rating distribution over Genre**

proj %>%
      group_by(genre) %>%
      summarise(count = n()) %>%
      ggplot(aes(genre, count)) +
      theme_gray()  +
      geom_col() +
```

```
      theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
      labs(title = "Ratings Frequency Distribution Per Genre",x = "Genre",y = "Frequency")

#**Overview of Rating distribution over months**

proj %>%
      group_by(month_rated) %>%
      summarise(count = n()) %>%
      ggplot(aes(month_rated, count)) +
      theme_gray()  +
      geom_col() +
      theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
      labs(title = "Ratings Frequency Distribution Per month",x = "month",y = "Frequency")

#**************************RATING FREQUENCY TABLES*******************

#rating per genre

proj %>%
     group_by(genre) %>%
     summarise(count = n()) %>%
      arrange(desc(count))

#rating per month

proj %>%
      group_by(month_rated) %>%
      summarise(count = n()) %>%
      arrange(desc(count))

#**********************Rating distribution plots**********************************
### Mean Distribution per Genre

proj %>%
  group_by(genre) %>%
  summarise(mean = mean(rating)) %>%
  ggplot(aes(genre, mean)) +
  theme_bw()  +
  geom_col() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
```

```
  labs(title = "Mean Distribution per Genre",x = "Genre",y = "Mean")

### Median Distribution per Genre

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

### Mode Distribution per Genre
proj %>%
       group_by(genre) %>%
       summarise(mode = mode(rating)) %>%
       ggplot(aes(genre, mode)) +
       theme_gray()  +
      geom_col() +
      theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
      labs(title = "Mode Distribution per Genre",x = "Genre",y = "Mode")

#**********Distribution tables**********

####median distribution per genre

proj %>%
       group_by(genre) %>%
       summarise(median = median(rating)) %>%
       arrange(desc(median)) %>%
       head(n=10)

###mean distribution per genre

proj %>%
       group_by(genre) %>%
       summarise(mean = mean(rating)) %>%
       arrange(desc(mean)) %>%
       head(n=10)
```

```
#********** Analysis - Model Building and Evaluation**********

## Naive Baseline Model
mean(edx$rating)

### Naive Mean-Baseline Model

# Calculate the average of all movies
mu_hat <- mean(proj$rating)
mu_hat

# Predict the RMSE on the valid set
rmse_mean_result <- RMSE(valid$rating, mu_hat)
rmse_mean_result

# Creating a results dataframe that contains all RMSE results
rmse_results <- data.frame(model="Naive Mean-Baseline Model", RMSE=rmse_mean_result)
rmse_results

## Movie-Based Model, a Content-based Approach
# Calculate the average per each movie

movie_rmse <- proj %>%
      group_by(movieId) %>%
      summarize(b_i = mean(rating - mu_hat))
movie_rmse

# Predict the ratings for valid dataset

rmse_valid <- valid %>%
      left_join(movie_rmse, by='movieId') %>%
      mutate(pred = mu_hat + b_i) %>%
      pull(pred)
rmse_valid

rmse_valid_result <- RMSE(valid$rating, rmse_valid)
rmse_valid_result

# Adding the results to the results dataset
```

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

#The RMSE on the ```validation``` dataset is **0.9417822**. It is slightly better than the Naive Mean-Baseline Model, but it is also far from the required RMSE (below 0.87) leading to poor performance for the model.

Movie + User Model, a User-based approach

```
# Calculate the average by movie
movie_avgs <- proj %>%
      group_by(movieId) %>%
      summarize(b_i = mean(rating - mu_hat))


# Calculate the average for every user
user_avgs <- proj %>%
  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_plus_user_model <- valid %>%
      left_join(movie_avgs, by='movieId') %>%
      left_join(user_avgs, by='userId') %>%
      mutate(pred = mu_hat + b_i + b_u) %>%
      pull(pred)

rmse_movie_plus_user_model_result <- RMSE(valid$rating, rmse_movie_plus_user_model)

# Adding the results to the results dataset
rmse_results <- rmse_results %>% add_row(model="Movie+User Based Model", RMSE=rmse_movie_plus_user_model_result)
rmse_results

#The movie plus user based model has achieved the
#required resultant RMSE of 0.863 which is less than 0.8649
#The model can be improved by using regularisation technique.
```