MovieLens Project

HarvardX PH125.9x Data Science: Capstone

Loy Jong Sheng

10/17/2020

Contents

1 Introduction

According to statista.com, in 2019, it is estimated that 1.92 billions people around the world purchased goods and services online and e-retail sales have exceeded US$3.5 trillion. With numerous online goods and services, internet consumers are inundated with information overload and they have to spend lots of their time exploring or filtering the possibilities. Recommendation systems with machine learning algorithms play an important role to provide each individual user with some items they might be interested in, example such as movies, books, etc, and support them with better decision making.

2 Aim

This Capstone project aims to create a movie recommendation system that is capable of predicting movie ratings with a Residual Mean Squared Error (RMSE) lesser than 0.8649.

3 Approach

To achieve that, this project shall use the MovieLens dataset to develop and evaluate the Machine Learning (ML) algorithm. The dataset has 10000054 ratings, 69878 users, 10677 movies and 797 genres. Each user is represented only by an id. There is no user demographic information included in the dataset. All users rated at least 20 movies. Timestamp is represented in seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

In addition, the project shall perform the following key steps: 1)Download MovieLens dataset and data preparation, 2)Discover and visualize the MovieLens datasets to gain insights. 3)Explore and build a predictive model that can achieve a targeted RMSE value < 0.8649. 4)Present the final solution.

4 Download MovieLens dataset and data preparation

The MovieLens datasets is downloaded from the following grouplens link https://grouplens.org/datasets/movielens/10m/. The dataset is then split into two datasets namely edx, which is used to train the ML algorithm, and validation, which is used to evaluate the algorithms.

```
if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                                         repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table",
                                         repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2",
                                         repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate",
                                         repos = "http://cran.us.r-project.org")
if(!require(recosystem)) install.packages("recosystem",
                                         repos = "http://cran.us.r-project.org")
if(!require(Matrix)) install.packages("Matrix",
                                         repos = "http://cran.us.r-project.org")
if(!require(recommenderlab)) install.packages("recommenderlab",
                                         repos = "http://cran.us.r-project.org")
if(!require(tinytex)) install.packages("tinytex",
                                         repos = "http://cran.us.r-project.org")
library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(lubridate)
library(recosystem)
library(Matrix)
library(recommenderlab)
```

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

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

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

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

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

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

4.1 Create 90% edx dataset and 10% validation dataset from MovieLens dataset.

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

edx <- movielens[-test_index,]
temp <- movielens[test_index,]

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

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
#Remove Objects from a Specified Environment
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

5 Discover and visualize the MovieLens datasets to gain insights

In this section, the aim is to analyze and visualize the edx dataset to gain insights.

```r
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1:               Comedy|Romance
```

```
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```
edx %>%
  summarize(Num_Distinct_Movie = n_distinct(movieId),
            Num_Distinct_User = n_distinct(userId),
            Num_Distinct_Genres = n_distinct(genres),
            Tot_size = nrow(edx))
```

```
##   Num_Distinct_Movie Num_Distinct_User Num_Distinct_Genres Tot_size
## 1              10677             69878                 797  9000055
```

```
summary(edx)
```

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

From the table above, there are 9000055 ratings, 10677 movies, 69878 users and 797 genres in the edx dataset. It is observed that the "genres" contained more than one genre category for the movies. Each of the users and movies are given a unique identification number. The ratings given by users range from minimum of 0.5 to maximum of 5.

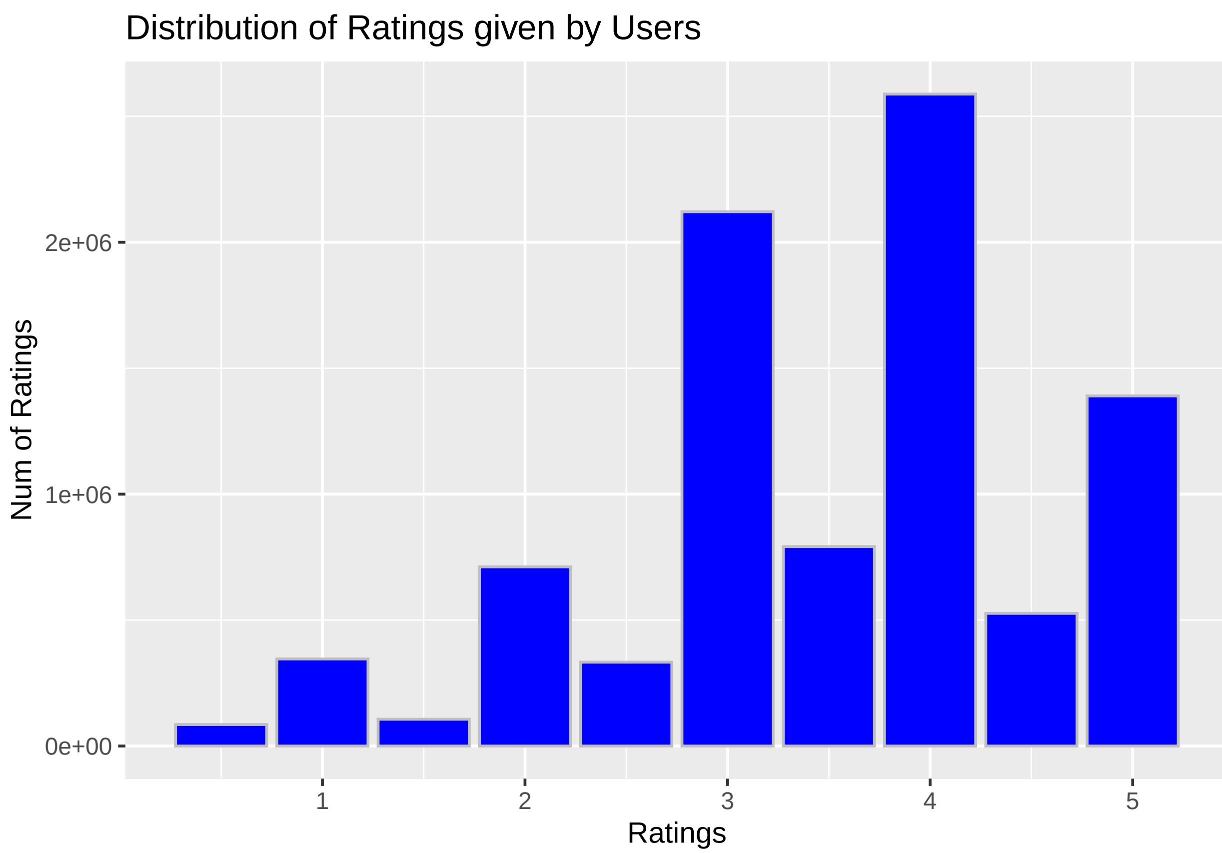
5.1 Exploring the ratings by users

```
#Top 5 ratings in order from most to least
Top_Given_Ratings <- edx %>% group_by(rating) %>%
  summarize(numRatings = n()) %>%
  arrange(desc(numRatings)) %>%
  top_n(5, numRatings)
Top_Given_Ratings
```

```
## # A tibble: 5 x 2
##   rating numRatings
##    <dbl>      <int>
## 1    4      2588430
```

```
## 2    3       2121240
## 3    5       1390114
## 4    3.5      791624
## 5    2        711422
```

```
Given_Ratings <- edx %>% group_by(rating) %>%  summarize(numRatings = n())
ggplot(data = Given_Ratings, mapping = aes(x = rating, y = numRatings)) +
  geom_col(fill = "blue", color = "grey") +
  labs(y = "Num of Ratings", x = "Ratings") +
  ggtitle("Distribution of Ratings given by Users")
```




```
summary(edx$rating)
```

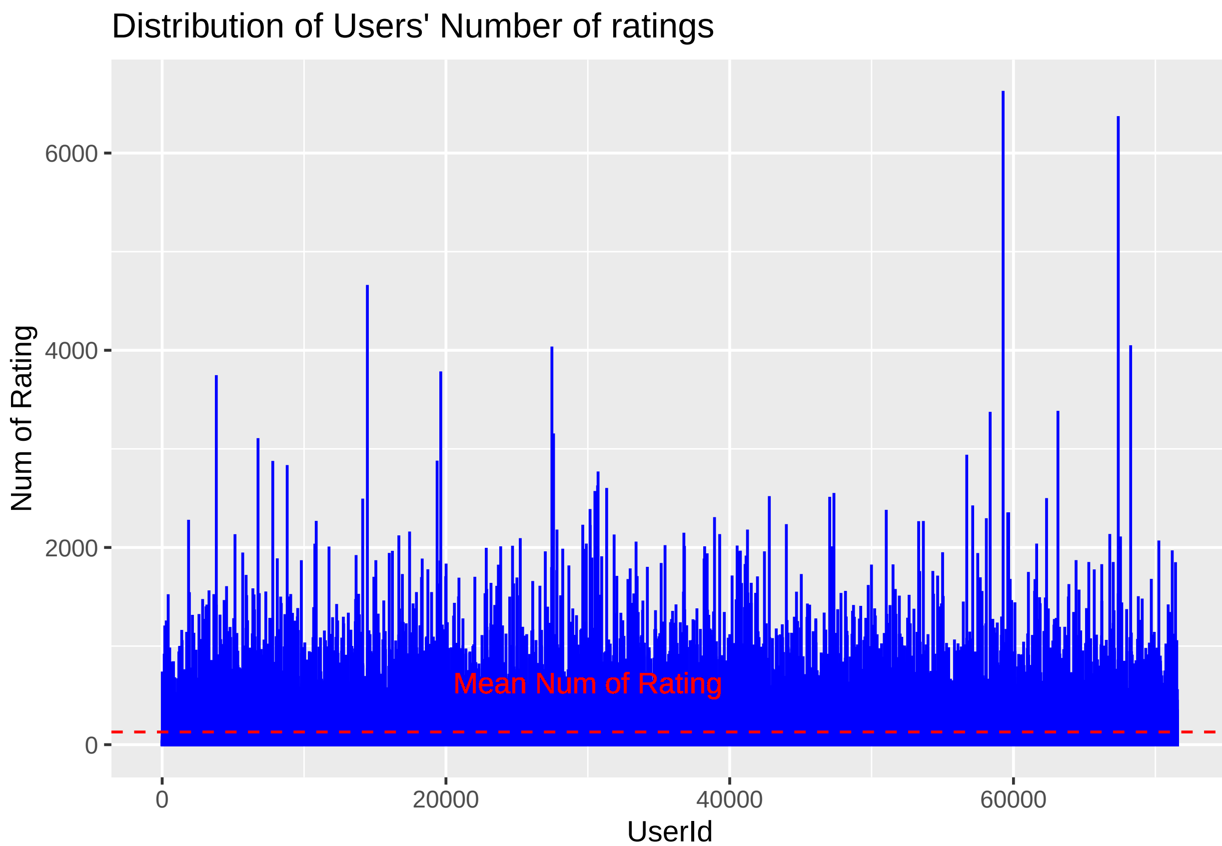
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

The distribution is observed to be left skewed. The median is 4 and mean is 3.512. There is no zero rating given. Most of the ratings given are either 3, 4 or 5. This showed that the users are more inclined to give higher ratings. It also showed that the users are lenient to the movies that they do not like.

```
NumRating_per_UserID <- edx %>% group_by(userId) %>%  summarize(numRating = n())

NumRating_per_UserID %>%
  ggplot(aes(x = userId, y = numRating)) +
```

```r
  geom_col(color = "blue") +
  geom_hline(yintercept= mean(NumRating_per_UserID$numRating),
             linetype="dashed", color = "red") +
  geom_text(aes(y=mean(NumRating_per_UserID$numRating)+500,
                label="Mean Num of Rating", x=30000.0),
                colour="red", angle=0) +
  labs(x = "UserId", y = "Num of Rating") +
  ggtitle("Distribution of Users' Number of ratings")
```



From above distribution, it is observed that there are users are very active in rating the movies while some are not.

```r
#Top 5 active users
Top_Active_Users <- edx %>% group_by(userId) %>%
  summarize(numRating = n()) %>%
  arrange(desc(numRating)) %>%
  top_n(5, numRating)
Top_Active_Users
```

```
## # A tibble: 5 x 2
##   userId numRating
##    <int>     <int>
## 1  59269      6616
## 2  67385      6360
## 3  14463      4648
```

```
## 4  68259      4036
## 5  27468      4023
```

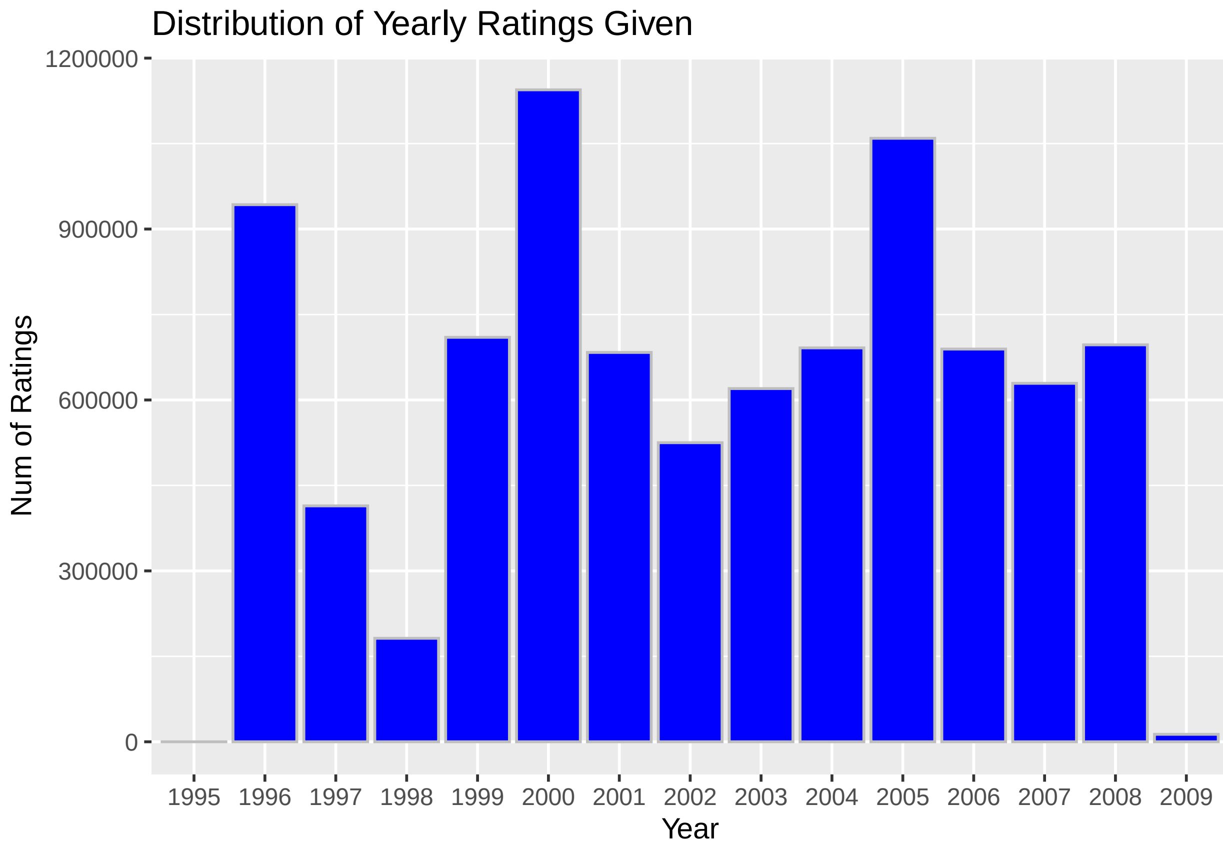
```
#Bottom 3 active users
Top_Active_Users <- edx %>% group_by(userId) %>%
  summarize(numRating = n()) %>%
  arrange(desc(numRating)) %>%
  top_n(-3, numRating)
Top_Active_Users
```

```
## # A tibble: 4 x 2
##   userId numRating
##    <int>     <int>
## 1  15719        13
## 2  50608        13
## 3  22170        12
## 4  62516        10
```

From the tables, it is observed that user ID 59269 rated 6616 times and user ID 62516 only rated 10 times. This showed that some users are active in rating the movies while some are not.

5.2 Exploring yearly ratings from 1995 to 2009

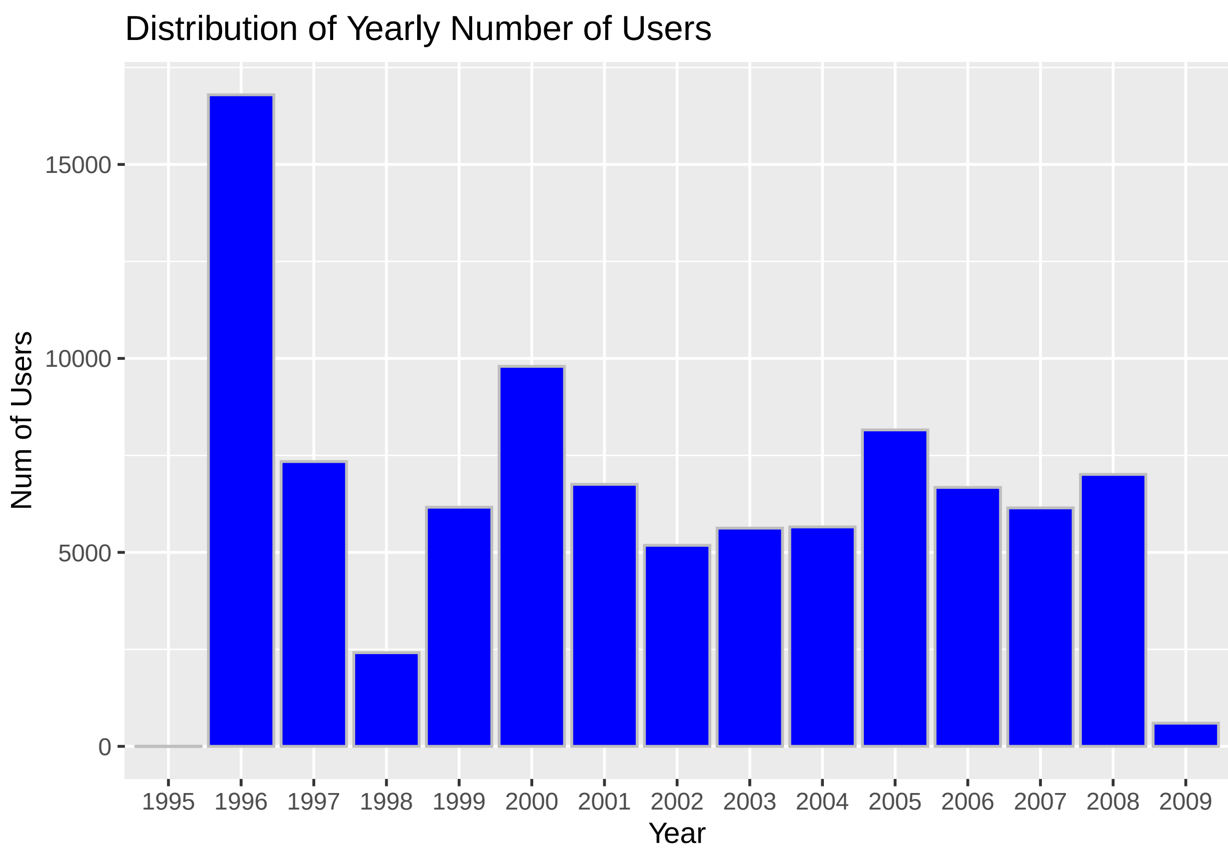
```
# Yearly Rating from 1995 to 2009
Yearly_Ratings <- edx %>%
  group_by(year = format(as_datetime(timestamp), format("%Y"))) %>%
  summarize(numRatings = n())
ggplot(data = Yearly_Ratings, mapping = aes(x = year, y = numRatings)) +
  geom_col(fill = "blue", color = "grey") +
  labs(x = "Year", y = "Num of Ratings") +
  ggtitle("Distribution of Yearly Ratings Given ")
```



From the distribution, it is observed that the number of users' participation in rating the movies are uneven year to year. The highest number of ratings occurred in 1996, 2000 and 2005 respectively.

5.3 Exploring yearly number of users from 1995 to 2009

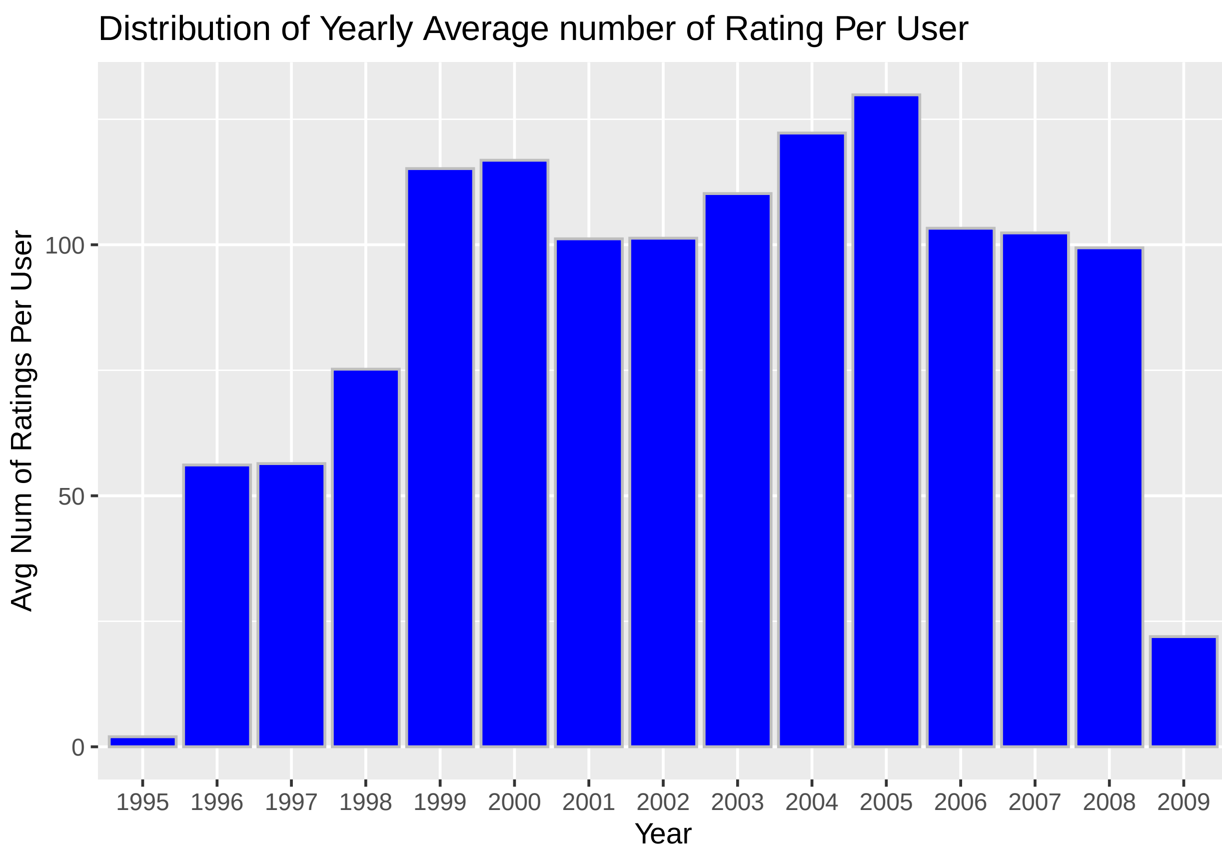
```
# Yearly Number of Users from 1995 to 2009
Yearly_NumUsers <- edx %>%
  group_by(year = format(as_datetime(timestamp), format("%Y"))) %>%
  summarize(numUsers = n_distinct(userId))
Yearly_NumUsers %>% ggplot(aes(x = year, y = numUsers)) +
  geom_col(fill = "blue", color = "grey") +
  labs(x = "Year", y = "Num of Users") +
  ggtitle("Distribution of Yearly Number of Users")
```



From the distribution, it is observed that year 1996, 2000 and 2005 have the highest number of users respectively, which explained the high number of ratings too.

5.4 Exploring yearly number of average number of ratings per User

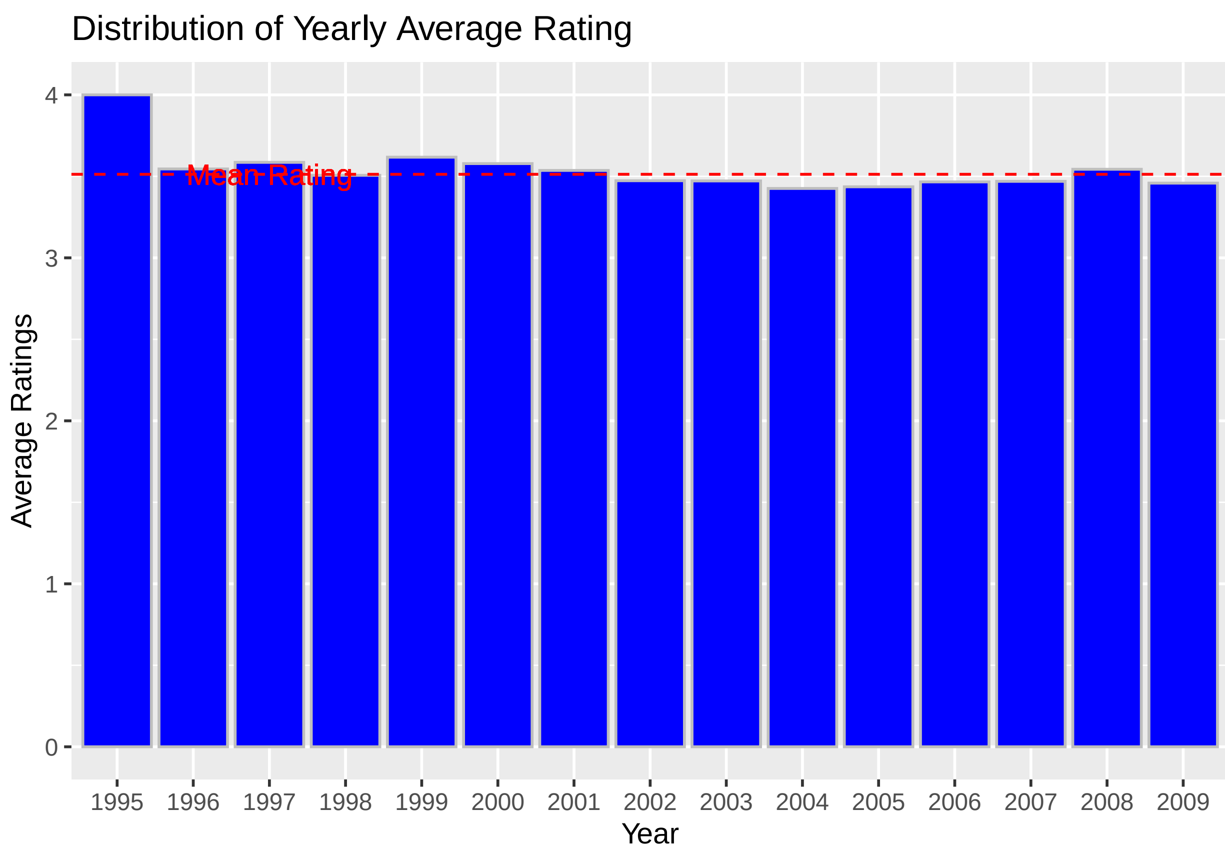
```
# Yearly Average number of Rating Per User from 1995 to 2009
Yearly_AvgNumRatingPerUser <- edx %>%
  group_by(year = format(as_datetime(timestamp), format("%Y"))) %>%
  summarize(AvgNumRatingPerUser = n()/n_distinct(userId))
Yearly_AvgNumRatingPerUser %>%
  ggplot(aes(x = year, y = AvgNumRatingPerUser)) +
  geom_col(fill = "blue", color = "grey") +
  labs(x = "Year", y = "Avg Num of Ratings Per User") +
  ggtitle("Distribution of Yearly Average number of Rating Per User")
```



From the distribution, it is observed that in year 1996, there are more users and less rating per user. However in 2000 and 2005, it is observed there are lesser users but higher rating per user.

5.5 Exploring yearly number of average number of ratings per movie

```
# Yearly Yearly Number of Average number of rating per Movie from 1995 to 2009
Yearly_AvgRatingPerUser <- edx %>%
  group_by(year = format(as_datetime(timestamp), format("%Y"))) %>%
  summarize(AvgRatingPerUser = sum(rating)/n())
Yearly_AvgRatingPerUser %>% ggplot(aes(x = year, y = AvgRatingPerUser)) +
  geom_col(fill = "blue", color = "grey") +
  geom_hline(yintercept= mean(edx$rating), linetype="dashed", color = "red") +
  geom_text(aes(y=mean(edx$rating), label="Mean Rating", x=3.0), colour="red", angle=0) +
  labs(x = "Year", y = "Average Ratings") +
  ggtitle("Distribution of Yearly Average Rating")
```



From the distribution, except for year 1995, the average ratings given by the users yearly from 1996 to 2009 do not vary much from the mean rating of 3.51.

5.6 Exploring the ratings by movies

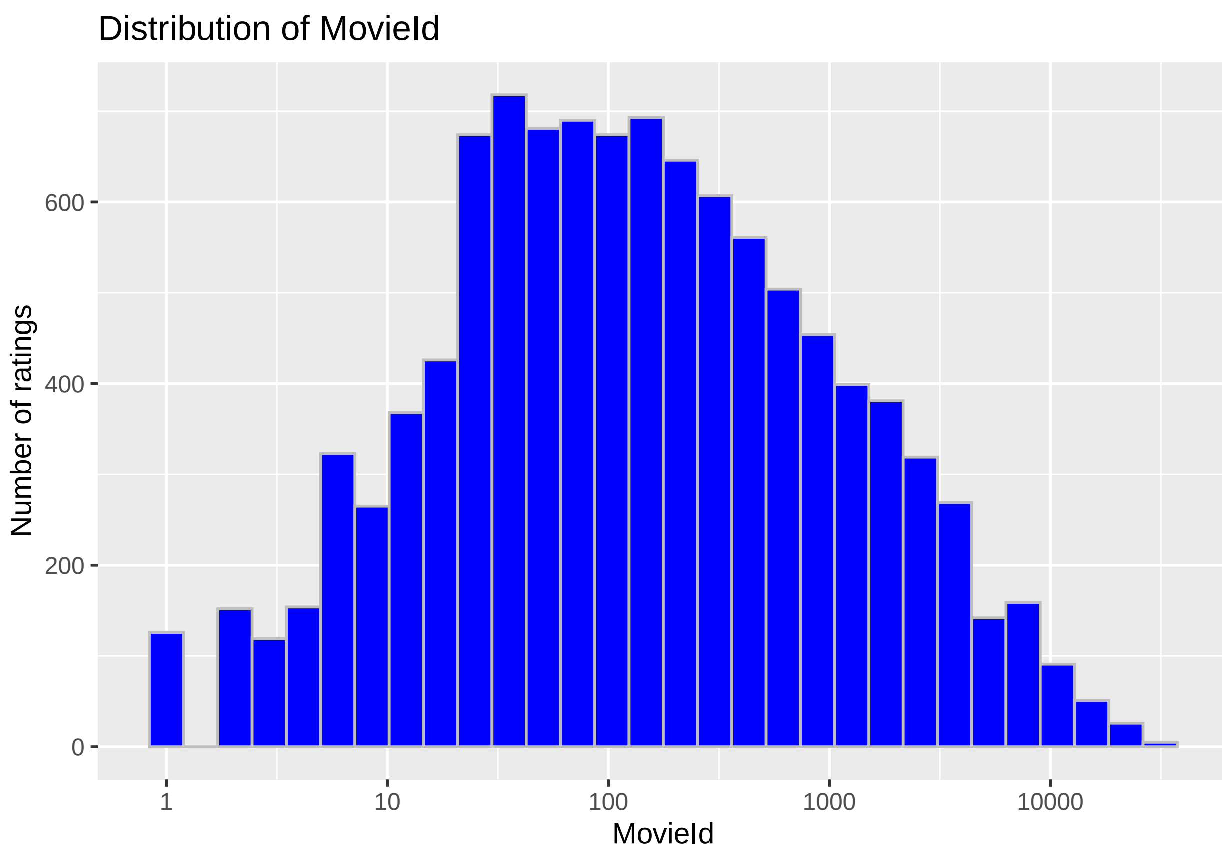
```
#Top years with most number of ratings
Top_Movies <- edx %>% group_by(movieId) %>%
  summarize(numRatings = n(), movieTitle = first(title)) %>%
  arrange(desc(numRatings)) %>%
  top_n(10, numRatings)
Top_Movies
```

```
## # A tibble: 10 x 3
##    movieId numRatings movieTitle
##      <dbl>      <int> <chr>
##  1     296      31362 Pulp Fiction (1994)
##  2     356      31079 Forrest Gump (1994)
##  3     593      30382 Silence of the Lambs, The (1991)
##  4     480      29360 Jurassic Park (1993)
##  5     318      28015 Shawshank Redemption, The (1994)
##  6     110      26212 Braveheart (1995)
##  7     457      25998 Fugitive, The (1993)
##  8     589      25984 Terminator 2: Judgment Day (1991)
##  9     260      25672 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (19~
## 10     150      24284 Apollo 13 (1995)
```

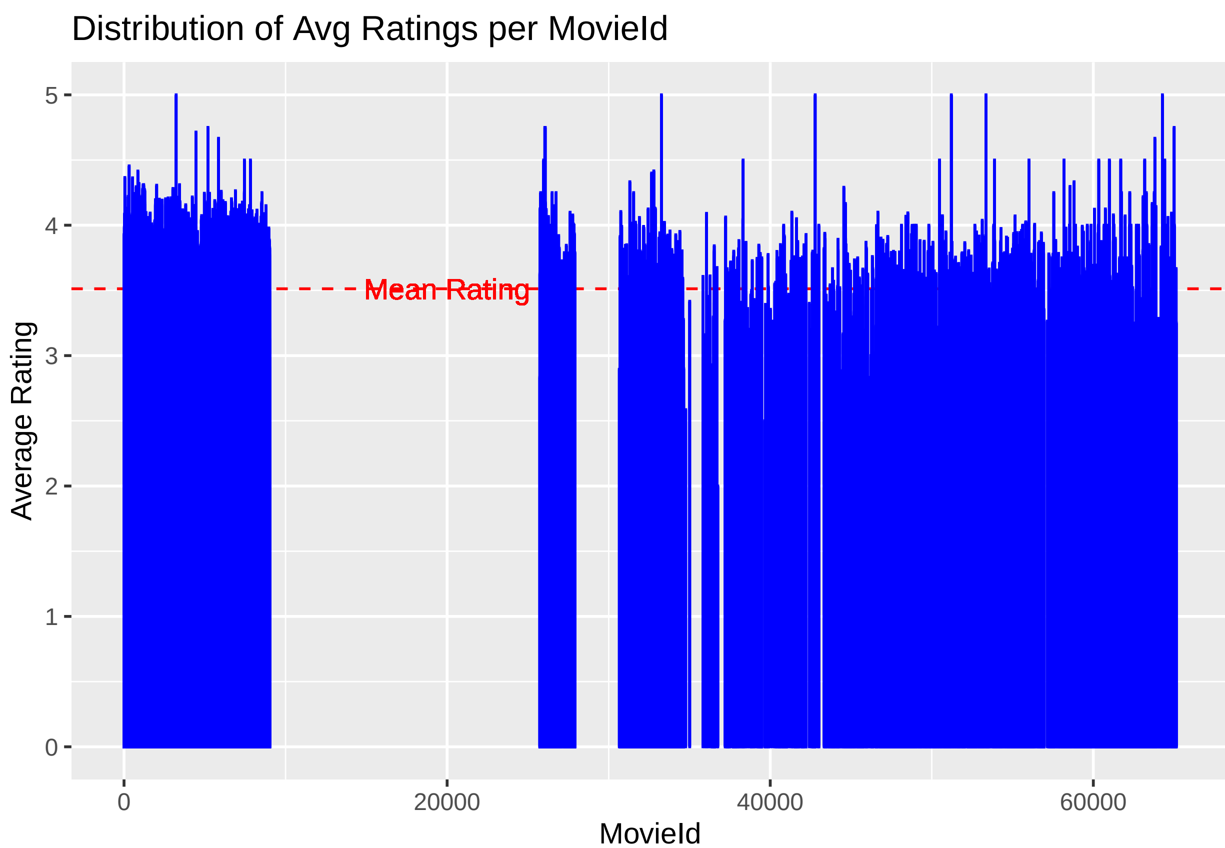
From the table, it is observed that the top three number of ratings were movie title "Pulp Fiction", "Forrest Gump" and "The Silence of the Lambs".

5.7 Exploring the number of ratings by movies

```
edx %>% group_by(movieId) %>% summarize(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(fill = "blue", color = "grey", bins = 30) +
  scale_x_log10() + ggtitle("Distribution of MovieId") +
  labs(x="MovieId" , y="Number of ratings")
```




```
AvgRating_per_MovieId <- edx %>% group_by(movieId) %>%
  summarize( AvgRatingsPerMovieId = sum(rating)/n())
AvgRating_per_MovieId %>% ggplot(aes(x = movieId, y = AvgRatingsPerMovieId)) +
  geom_hline(yintercept= mean(edx$rating), linetype="dashed", color = "red") +
  geom_text(aes(y=mean(edx$rating), label="Mean Rating", x=20000.0), colour="red") +
  geom_col(color = "blue") + labs(x = "MovieId", y = "Average Rating") +
  ggtitle("Distribution of Avg Ratings per MovieId")
```



From the distributions, some movies are infrequently rated while some are frequently rated. It is also observed that the popular movies are typically given higher ratings than those less popular.

5.8 Exploring the ratings by genres

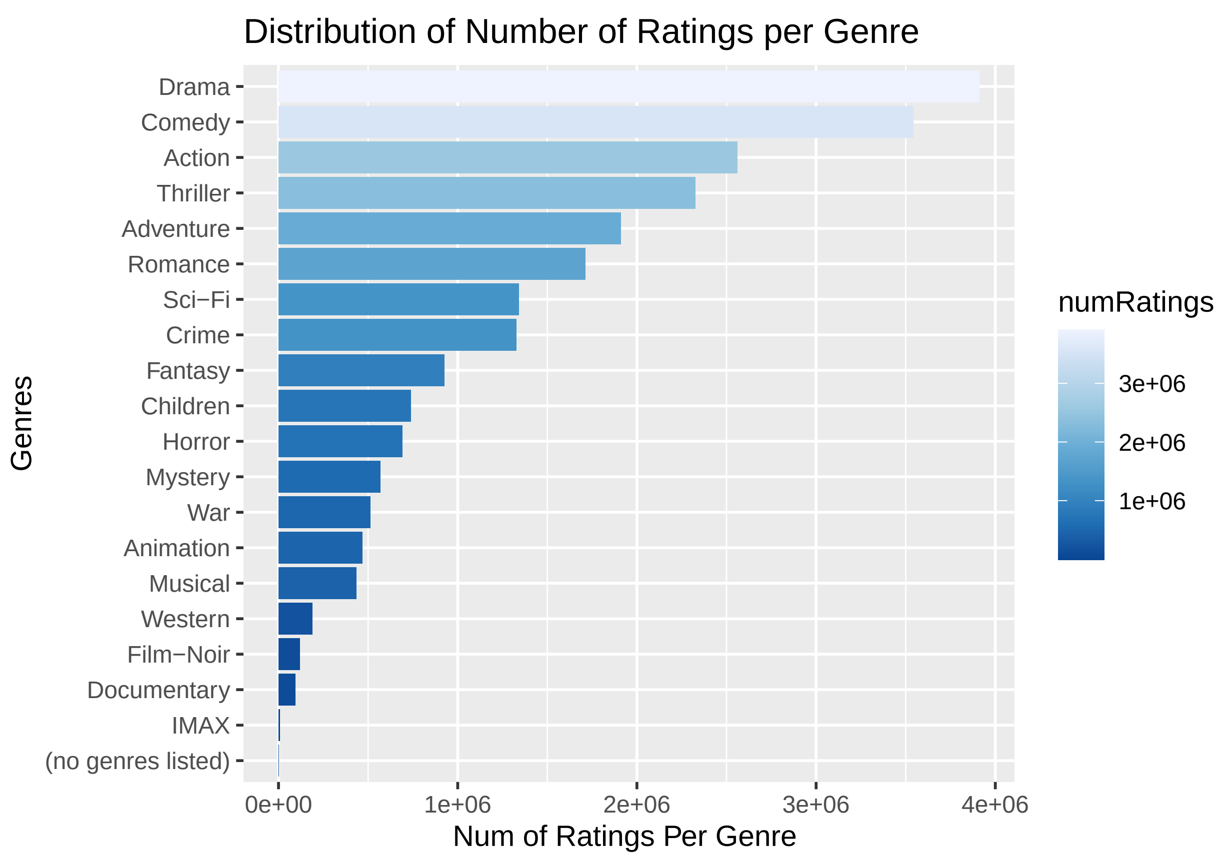
```
#Top genres
Top_genres_rating <- edx %>% group_by(genres) %>%
  summarize(numRatings = n()) %>%
  arrange(desc(numRatings)) %>%
  top_n(10, numRatings)
Top_genres_rating
```

```
## # A tibble: 10 x 2
##    genres                      numRatings
##    <chr>                            <int>
##  1 Drama                           733296
##  2 Comedy                          700889
##  3 Comedy|Romance                  365468
##  4 Comedy|Drama                    323637
##  5 Comedy|Drama|Romance            261425
##  6 Drama|Romance                   259355
##  7 Action|Adventure|Sci-Fi         219938
##  8 Action|Adventure|Thriller       149091
##  9 Drama|Thriller                  145373
## 10 Crime|Drama                     137387
```

```r
#Split the genres
Genres_split <- edx %>% separate_rows(genres, sep = "\\|")
Genres_split %>% summarize(Total_Distinct_Genre = n_distinct(genres))
```

```
## # A tibble: 1 x 1
##   Total_Distinct_Genre
##                  <int>
## 1                   20
```

```r
Distinct_genres_group <- Genres_split %>%
  group_by(genres) %>%
  summarize(numRatings = n())
Distinct_genres_group %>%
  ggplot(aes(reorder(genres, numRatings), numRatings, fill= numRatings)) +
  geom_bar(stat = "identity") + coord_flip() +
  scale_fill_distiller(palette = "Blues") + labs(y = "Num of Ratings Per Genre ", x = "Genres") +
  ggtitle("Distribution of Number of Ratings per Genre")
```




```r
NoGenreListed <- Genres_split %>%
  filter(genres == "(no genres listed)") %>%
  summarize(userid = userId, rating = rating, movieid = movieId,
            movieTitle = title, genres = genres)
NoGenreListed
```

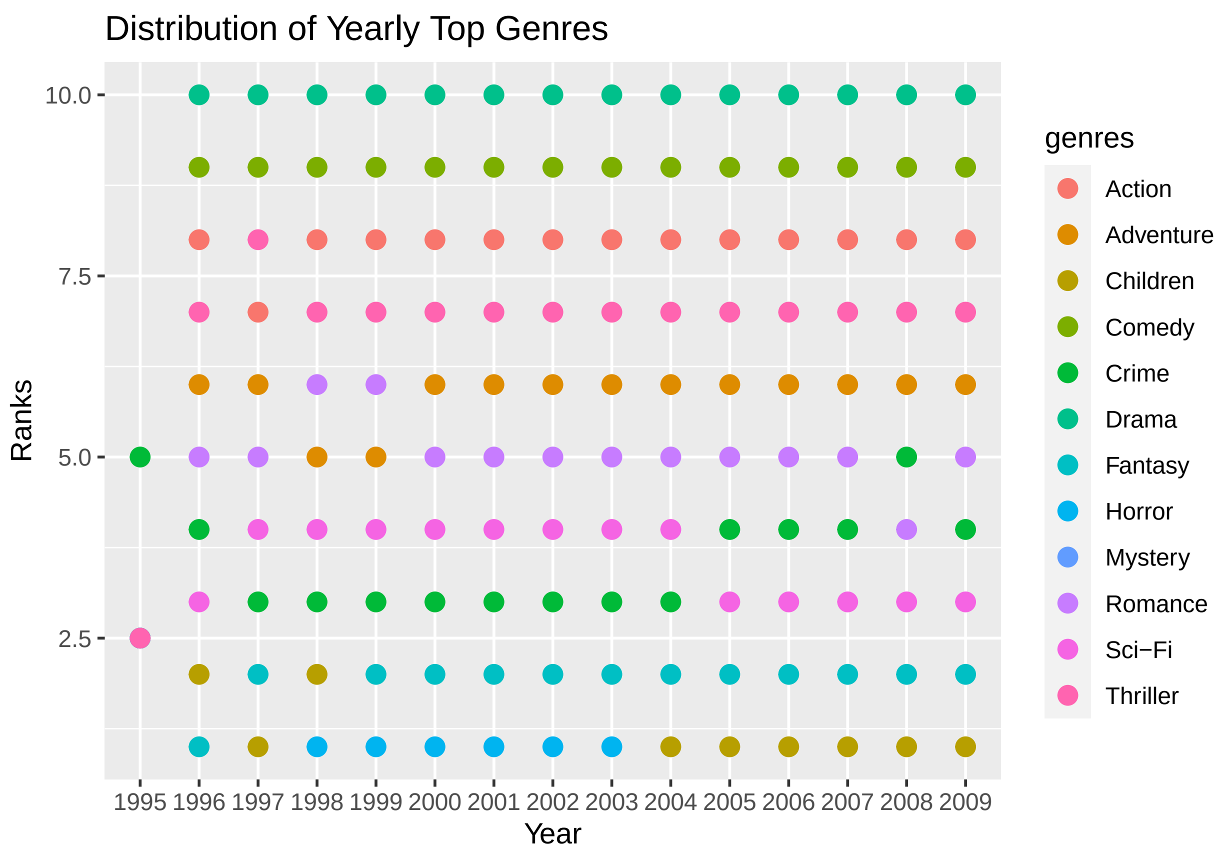
```
## # A tibble: 7 x 5
```

```
##   userid rating movieid movieTitle           genres
##    <int>  <dbl>   <dbl> <chr>                <chr>
## 1   7701    5      8606 Pull My Daisy (1958) (no genres listed)
## 2  10680    4.5    8606 Pull My Daisy (1958) (no genres listed)
## 3  29097    2      8606 Pull My Daisy (1958) (no genres listed)
## 4  46142    3.5    8606 Pull My Daisy (1958) (no genres listed)
## 5  57696    4.5    8606 Pull My Daisy (1958) (no genres listed)
## 6  64411    3.5    8606 Pull My Daisy (1958) (no genres listed)
## 7  67385    2.5    8606 Pull My Daisy (1958) (no genres listed)
```

From the distribution, it is observed that there are nineteen distinct genres and one with "no genres listed" category. A movie title "Pull My Daisy" falls into "no genre list" with seven users' ratings.

5.9 Exploring top 10 genres for each year

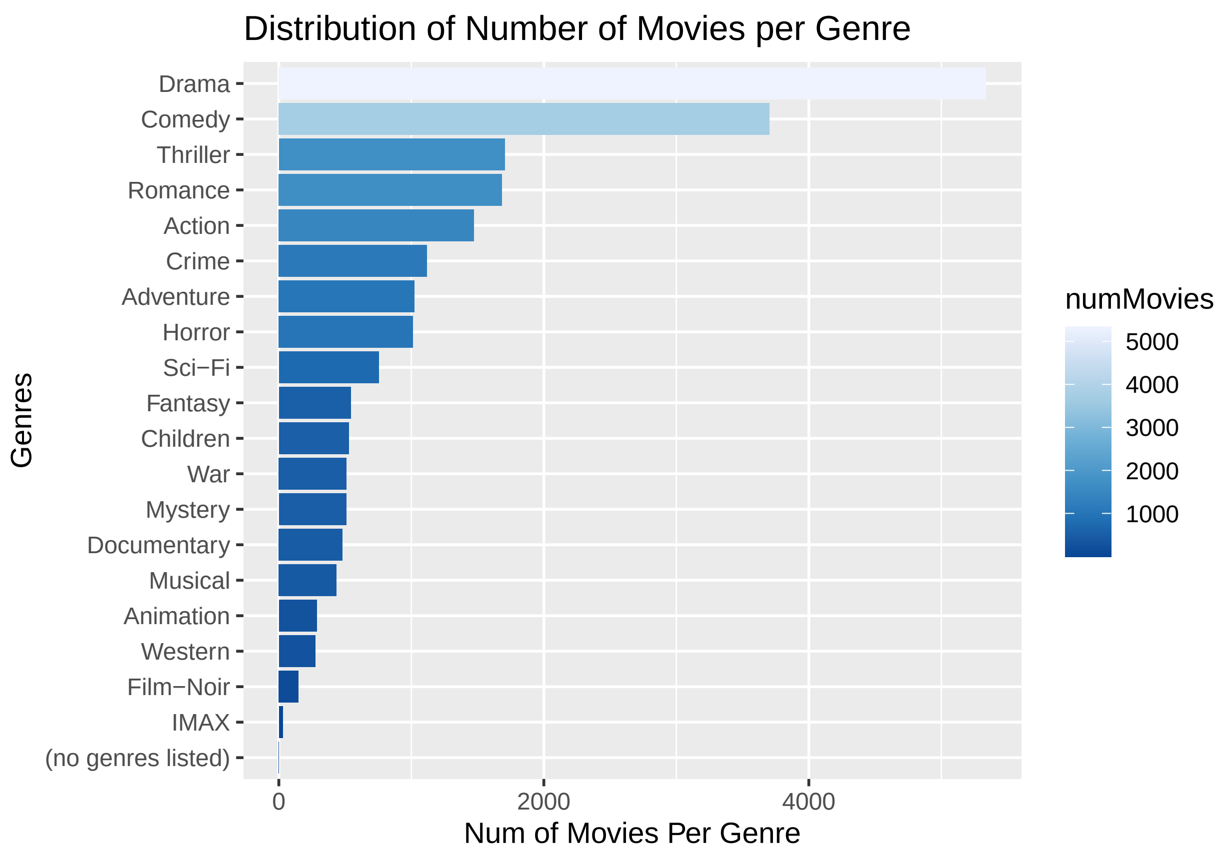
```
YearGenresGrp <- Genres_split %>%
  group_by(genres, year = format(as_datetime(timestamp), format("%Y"))) %>%
  summarize(numRatings = n())
Top10GenreYearly <- YearGenresGrp %>% group_by(year) %>%
  arrange(desc(numRatings)) %>%
  top_n(10, numRatings) %>%
  mutate(ranks = rank(numRatings))
Top10GenreYearly %>% ggplot(aes(x = year, y = ranks, color = genres)) +
  geom_point(size=3) +
  xlab("Year") + ylab("Ranks") +
  ggtitle("Distribution of Yearly Top Genres")
```



Comparing the top 10 genre rankings with highest number of ratings from 1996 to 2009, the average top 3 genres are Drama, Comedy and Actions. It is observed that the top 5 genres do not change significantly compared to the bottom five genres.

5.10 Exploring number of movies per genre

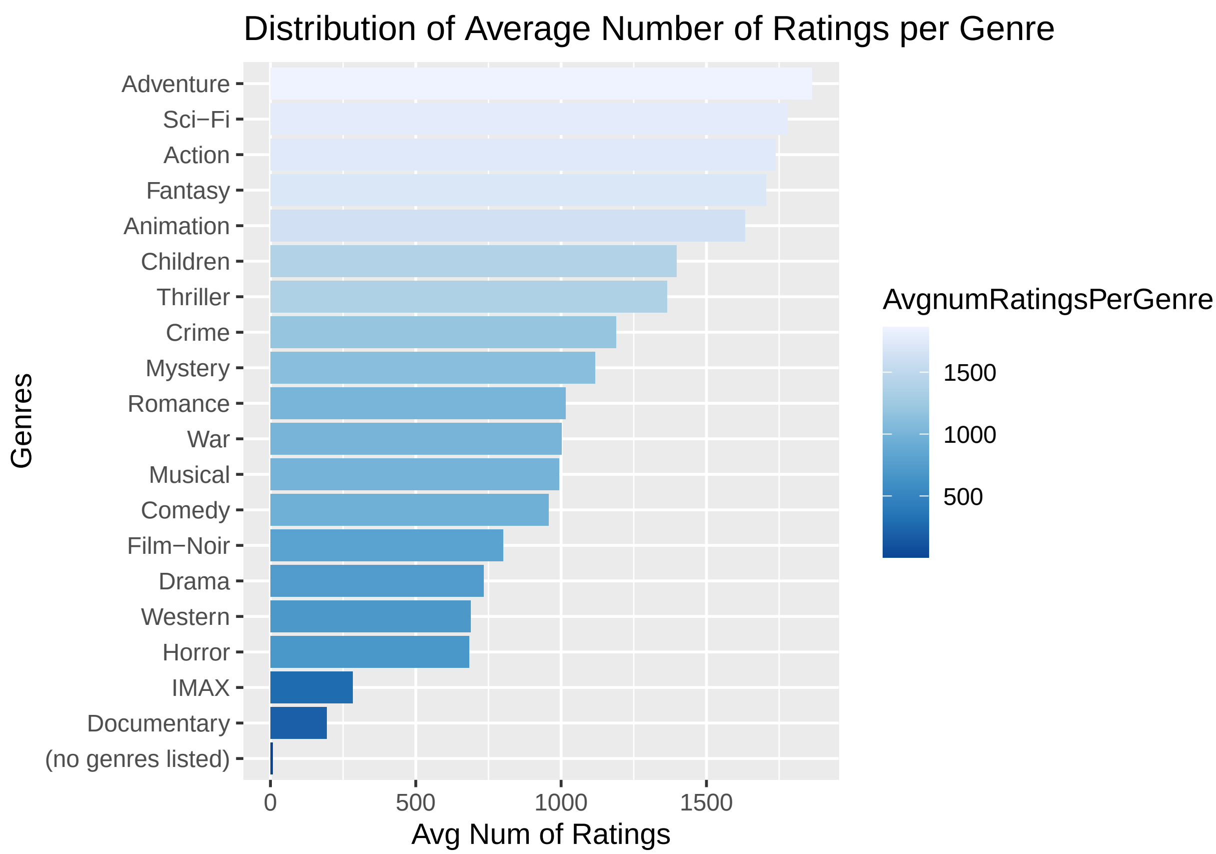
```
#Split the genres
Movie_per_Genres_group <- Genres_split %>%
  group_by(genres) %>%
  summarize( numMovies = n_distinct(movieId))
Movie_per_Genres_group %>%
  ggplot(aes(reorder(genres, numMovies), numMovies, fill= numMovies)) +
  geom_bar(stat = "identity") + coord_flip() +
  scale_fill_distiller(palette = "Blues") +
  labs(y = "Num of Movies Per Genre", x = "Genres") +
  ggtitle("Distribution of Number of Movies per Genre")
```



From the distribution above, it is observed that Drama, Comedy have the most number of movie titles respectively. This explains why Drama and Comedy has the most number of ratings respectively.

5.11 Exploring average number of ratings per genre

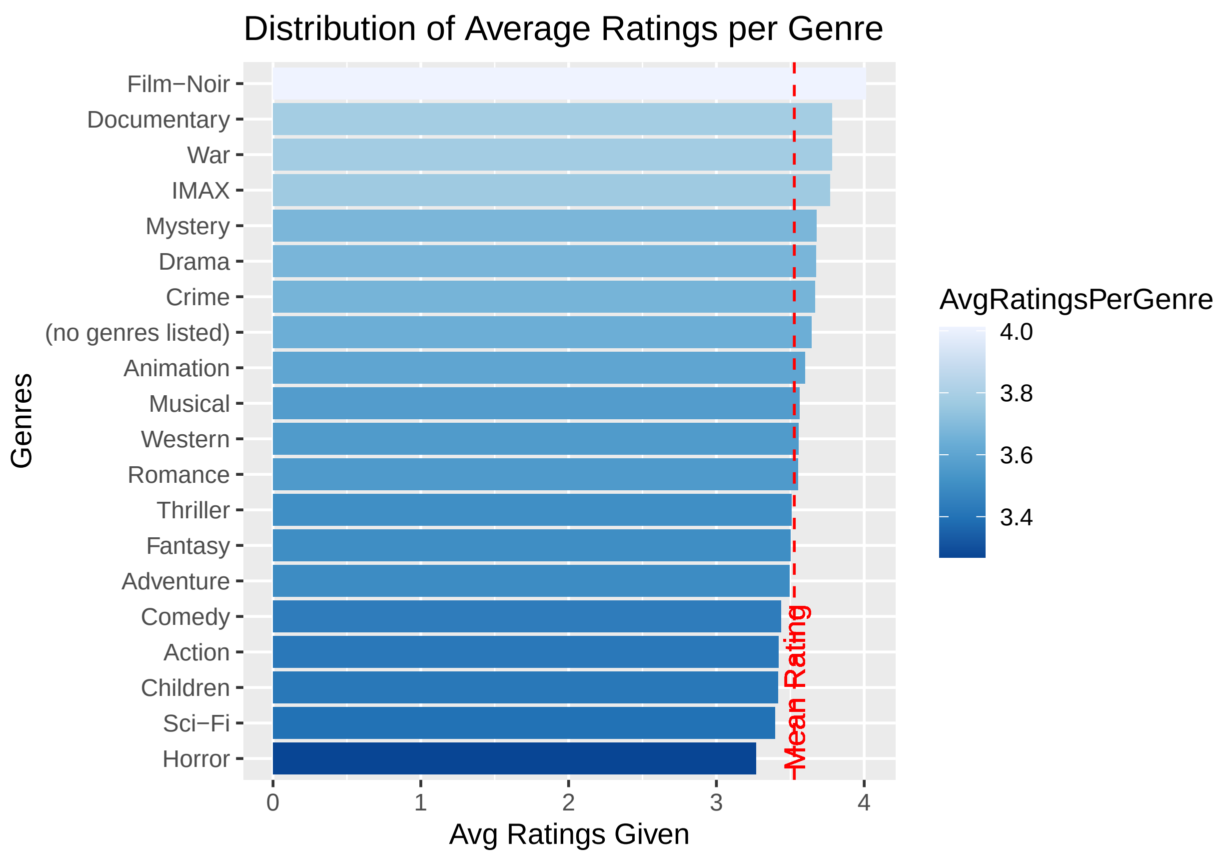
```
AvgNumRating_per_Movie_group <- Genres_split %>%
  group_by(genres) %>%
  summarize( AvgnumRatingsPerGenre = n()/n_distinct(movieId))
AvgNumRating_per_Movie_group %>%
  ggplot(aes(reorder(genres, AvgnumRatingsPerGenre),
             AvgnumRatingsPerGenre, fill= AvgnumRatingsPerGenre)) +
  geom_bar(stat = "identity") + coord_flip() +
  scale_fill_distiller(palette = "Blues") +
  labs(y = "Avg Num of Ratings", x = "Genres") +
  ggtitle("Distribution of Average Number of Ratings per Genre")
```



From the above distribution, it is observed that Adventure, Sci-Fi and Action are most frequently rated respectively despite them having lower number of movie titles.

5.12 Exploring average ratings given per genre

```
AvgRating_per_Genre_group <- Genres_split %>% group_by(genres) %>%
  summarize( AvgRatingsPerGenre = sum(rating)/n())
AvgRating_per_Genre_group %>%
  ggplot(aes(reorder(genres, AvgRatingsPerGenre), AvgRatingsPerGenre, fill= AvgRatingsPerGenre)) +
  geom_bar(stat = "identity") + coord_flip() +
  geom_hline(yintercept= mean(Genres_split$rating), linetype="dashed", color = "red") +
  geom_text(aes(y=mean(Genres_split$rating), label="Mean Rating", x=3.0), colour="red", angle=90) +
  scale_fill_distiller(palette = "Blues") + labs(y = "Avg Ratings Given", x = "Genres") +
  ggtitle("Distribution of Average Ratings per Genre")
```



From the above distribution, it is observed that the average rating given for each genre does not vary much from the mean rating of 3.51. The genres effect seems to be rather minor.

6 Evaluating the models using Residual Mean Squared Error (RMSE)

A typical performance measure for regression model is the Root Mean Square Error (RMSE). It gives an idea of how much error the model typically makes in its predictions, with higher weight for large errors.

```
#Define the Loss function
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
}
#Create a RMSE results dataframe to store and compare the results.
RMSE_results <- tibble()
```

7 Building a Regression model

A regression model can be derived to predict the rating by assuming the same rating for all movies and users with all the differences explained by random variation.

7.1 Baseline Model using the average movie rating

The most fundamental approach to predict a user's rating for a is to use the average

```
#Assumes the same rating for all movies and users with all the differences explained by random variatio
#Predicting using mu_hat
mu_hat <- mean(edx$rating)
```

```
rmse <- RMSE(validation$rating, mu_hat)
#Add result to the results dataframe
RMSE_results <- tibble(Model = "Mean_hat",
                       Desciption = "baseline (using mean hat only)",
                       RMSE_value = rmse)
RMSE_results
```

```
## # A tibble: 1 x 3
##   Model    Desciption                     RMSE_value
##   <chr>    <chr>                               <dbl>
## 1 Mean_hat baseline (using mean hat only)       1.06
```

7.2 Modeling movie effect

From the exploration of datasets, it is observed that there are movies that are very frequently rated and there are some, which much less frequent. It is also observed that the popular movies are typically given higher ratings than those less popular. To account for this observation in the baseline model, the movie effects, which is calculated using the chunk below, is added to improve the RMSE.

```
#Compute the movie effect
Movie_effects <- edx %>% group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat))
#Examine the movie effects
Movie_effects %>% ggplot(aes(x = b_i)) +
  geom_histogram(bins = 30, fill = "blue", color = "grey") +
  ggtitle("Distribution of Movie effect")
```

7.3 Predicting rating with movie effect

```
#Predicting Baseline + movie effect
Predicted_ratings <- mu_hat + validation %>%
  left_join(Movie_effects, by='movieId') %>%  pull(b_i)
```

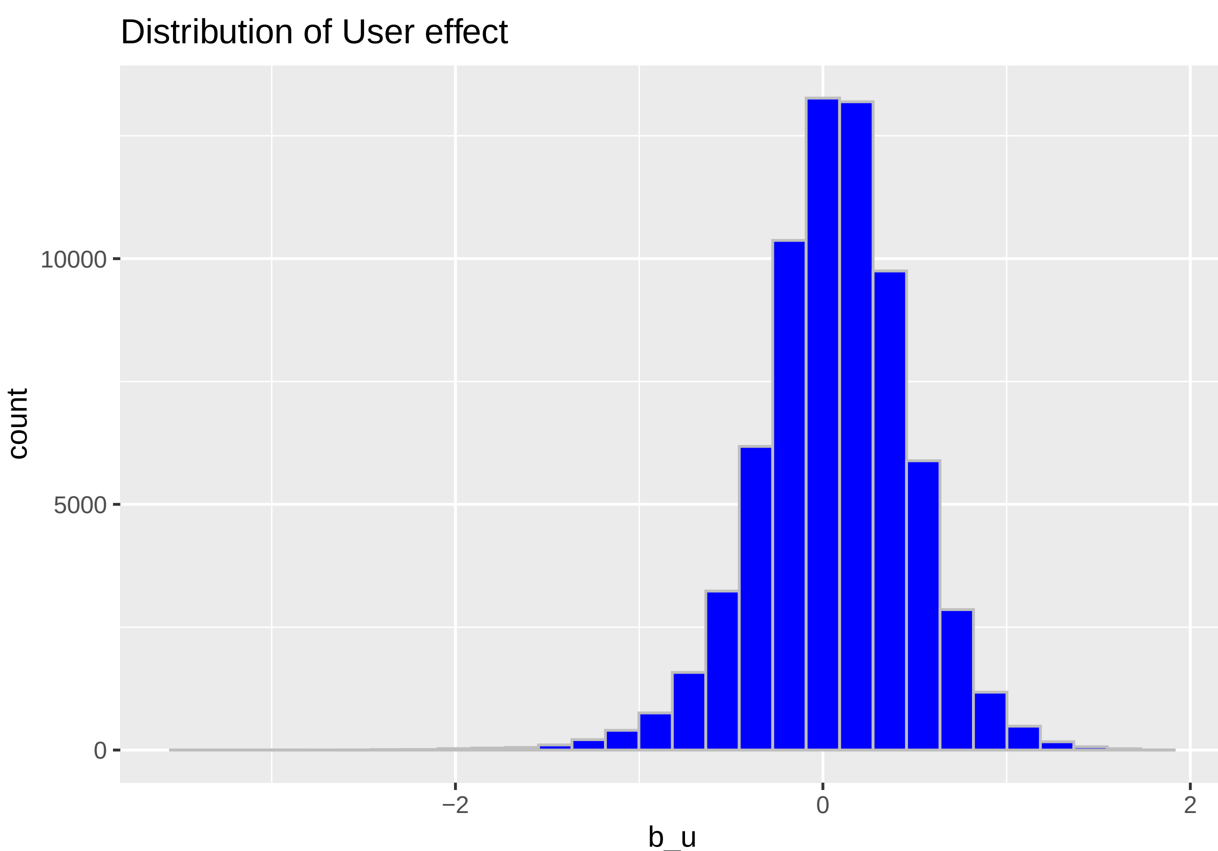
```
rmse <- RMSE(Predicted_ratings, validation$rating)
RMSE_results <- RMSE_results %>%
  add_row(Model = "baseline+b_i",
          Desciption = "baseline with movie effect",
          RMSE_value = rmse)
RMSE_results
```

```
## # A tibble: 2 x 3
##   Model        Desciption                           RMSE_value
##   <chr>        <chr>                                     <dbl>
## 1 Mean_hat     baseline (using mean hat only)            1.06
## 2 baseline+b_i baseline with movie effect                0.944
```

7.4 Modeling User effect

There is a wide spread how the users rated the movies. Those with high expectation likely to give a lower ratings compare to those with lower expectation for the movies they watched. Some users are active in rating the movies they watched, while some are not. It is observed the users are more inclined to give higher ratings too. To account these observations in the baseline model with movie effect added, the user effects, which is calculated using the chunk below, is added to further improve the RMSE.

```
#Compute the user effects
User_effects <- edx %>% left_join(Movie_effects, by='movieId') %>%
  group_by(userId) %>% summarize(b_u = mean(rating - mu_hat - b_i))
#Examine the user effects
User_effects %>% ggplot(aes(x = b_u)) +
  geom_histogram(bins = 30, fill = "blue", color = "grey") +
  ggtitle("Distribution of User effect")
```



7.5 Predicting rating with movie and user effects

```
#Predicting Baseline + movie effects + user effects
Predicted_ratings <- validation %>% left_join(Movie_effects, by='movieId') %>%
  left_join(User_effects, by='userId') %>%
  mutate(pred = mu_hat + b_i + b_u) %>% pull(pred)
```

```
rmse <- RMSE(Predicted_ratings, validation$rating)
RMSE_results <- RMSE_results %>%
  add_row(Model = "baseline+b_i+b_u",
          Desciption = "baseline with movie+user effects",
```

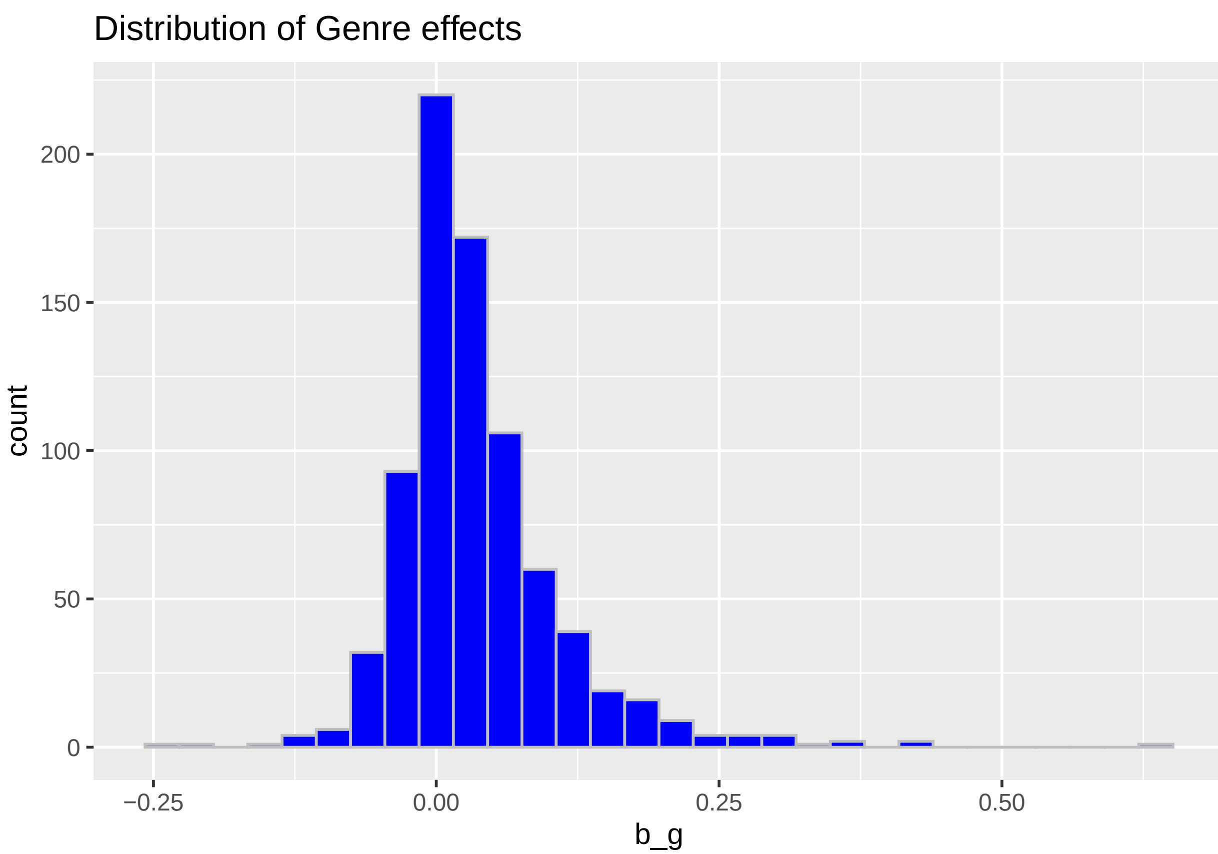
```
         RMSE_value = rmse)
RMSE_results
```

```
## # A tibble: 3 x 3
##   Model             Desciption                        RMSE_value
##   <chr>             <chr>                                  <dbl>
## 1 Mean_hat          baseline (using mean hat only)         1.06
## 2 baseline+b_i      baseline with movie effect             0.944
## 3 baseline+b_i+b_u baseline with movie+user effects      0.865
```

7.6 Modeling genre effect

It is observed that top 3 genres that have most number of ratings are Drama, Comedy and Action. These top genre preference did not vary much from year to year too. However, Adventure, Sci-Fi and Action are most frequently rated respectively despite they have lower number of movie titles. The average rating given for each genre also does not vary much from the mean rating. This indicates genres effect is likely to be minor.

```
#Compute the genre effects
Genre_effects <- edx %>% left_join(Movie_effects, by='movieId') %>%
  left_join(User_effects, by='userId') %>%
  group_by(genres) %>% summarize(b_g = mean(rating - mu_hat - b_i - b_u))
#Examine the genre effects
Genre_effects %>% ggplot(aes(x = b_g)) +
  geom_histogram(bins = 30, fill = "blue", color = "grey") +
  ggtitle("Distribution of Genre effects")
```



7.7 Predicting rating with movie, user and genre effects

```
#Predicting Baseline + movie effect + user effect + genre effects
Predicted_ratings <- validation %>% left_join(Movie_effects, by='movieId') %>%
  left_join(User_effects, by='userId') %>% left_join(Genre_effects, by='genres') %>%
  mutate(pred = mu_hat + b_i + b_u + b_g) %>% pull(pred)
```

```
rmse <- RMSE(Predicted_ratings, validation$rating)
RMSE_results <- RMSE_results %>%
  add_row(Model = "baseline+b_i+b_u+b_g",
          Desciption = "baseline with movie+user+genres effects",
          RMSE_value = rmse)
RMSE_results
```

```
## # A tibble: 4 x 3
##   Model                Desciption                              RMSE_value
##   <chr>                <chr>                                        <dbl>
## 1 Mean_hat             baseline (using mean hat only)               1.06
## 2 baseline+b_i         baseline with movie effect                   0.944
## 3 baseline+b_i+b_u     baseline with movie+user effects             0.865
## 4 baseline+b_i+b_u+b_g baseline with movie+user+genres effects      0.865
```

8 Regularization based approach

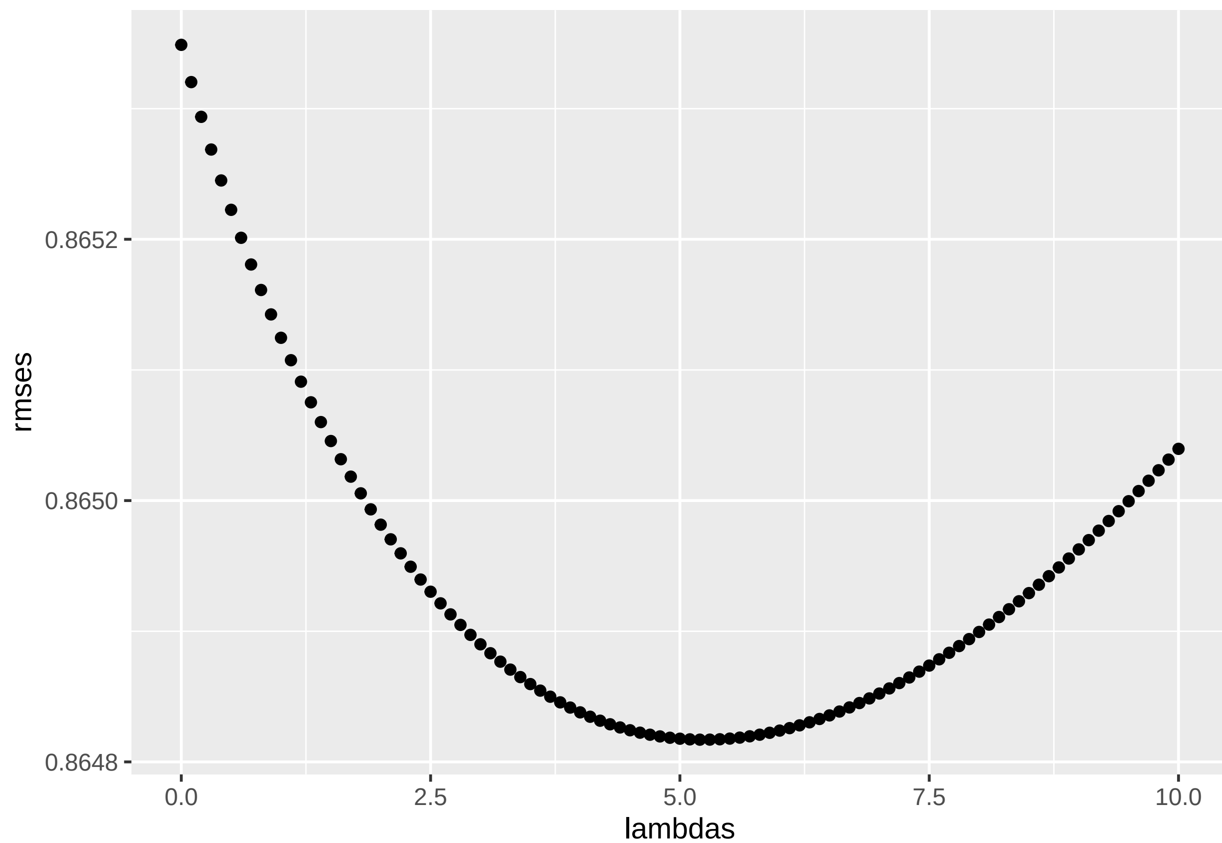
The above regression model has achieved a RMSE value of 0.8649. However, it has been observed that there is a group of users who are more active in rating the movies and vice versa, there is a group of users who are less active. Similarly, there are movies that were rated frequently, and there are movies that were rated only a few times. Therefore these can lead to errors in the estimates. Since RMSE is sensitive to large errors, the concept of regularization can be used to further improve the RMSE by adding a factor that penalized large estimates that are formed using small sample sizes.(Rafael, 2020)

8.1 Regularization of Baseline + movie effect + user effect

```
lambdas <- seq(0, 10, 0.1)
rmses <- sapply(lambdas, function(l){
     Movie_effects_reg <- edx %>% group_by(movieId) %>%
       summarize(b_i_reg = sum(rating - mu_hat)/(n()+l))
     User_effects_reg <- edx %>% left_join(Movie_effects_reg, by="movieId") %>%
       group_by(userId) %>%
       summarize(b_u_reg = sum(rating - mu_hat - b_i_reg)/(n()+l))

#Predicting Baseline + movie effect + user effect with regularization
     predicted_ratings <-
          validation %>%
          left_join(Movie_effects_reg, by = "movieId") %>%
          left_join(User_effects_reg, by = "userId") %>%
          mutate(pred = mu_hat + b_i_reg + b_u_reg) %>%
          pull(pred)
     return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```




```
tibble(Lambda = lambdas[which.min(rmses)], RMSE = min(rmses))
```

```
## # A tibble: 1 x 2
##   Lambda  RMSE
##    <dbl> <dbl>
## 1    5.2 0.865
```

```
RMSE_results <- RMSE_results %>%
  add_row(Model = "baseline+b_i_reg+b_u_reg",
          Desciption = "Regularised with movie+user effects",
          RMSE_value = min(rmses))
RMSE_results
```

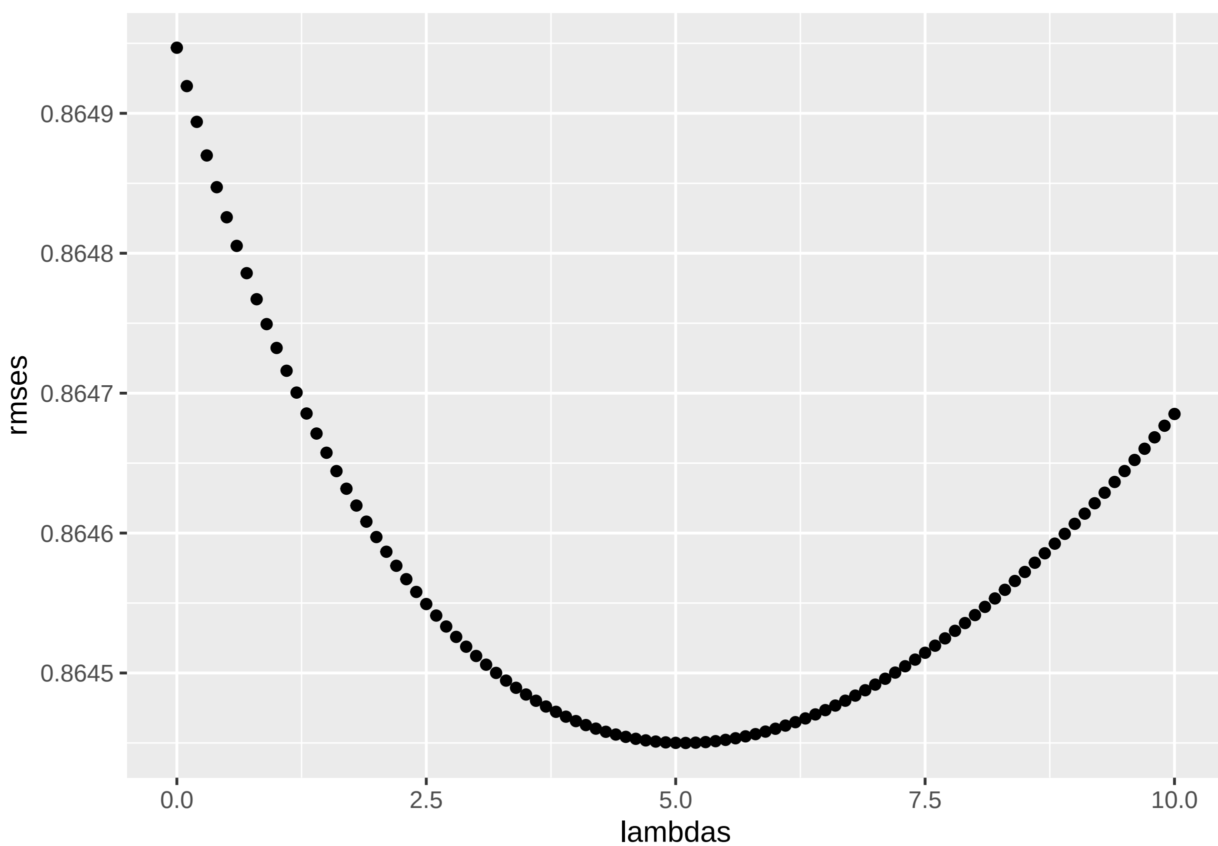
```
## # A tibble: 5 x 3
##   Model                    Desciption                              RMSE_value
##   <chr>                    <chr>                                        <dbl>
## 1 Mean_hat                 baseline (using mean hat only)               1.06
## 2 baseline+b_i             baseline with movie effect                   0.944
## 3 baseline+b_i+b_u         baseline with movie+user effects             0.865
## 4 baseline+b_i+b_u+b_g     baseline with movie+user+genres effects      0.865
## 5 baseline+b_i_reg+b_u_reg Regularised with movie+user effects          0.865
```

8.2 Regularization of Baseline + movie effect + user effect + genre effect

```
lambdas <- seq(0, 10, 0.1)
rmses <- sapply(lambdas, function(l){
     Movie_effects_reg <- edx %>% group_by(movieId) %>%
       summarize(b_i_reg = sum(rating - mu_hat)/(n()+l))
     User_effects_reg <- edx %>% left_join(Movie_effects_reg, by="movieId") %>%
       group_by(userId) %>% summarize(b_u_reg = sum(rating - mu_hat - b_i_reg)/(n()+l))
     Genre_effects_reg <- edx %>% left_join(Movie_effects_reg, by="movieId") %>%
       left_join(User_effects_reg, by='userId') %>% group_by(genres) %>%
     summarize(b_g_reg = sum(rating - mu_hat - b_i_reg - b_u_reg)/(n()+l))

# Predicting Baseline + movie effect + user effect + genre effect with regularization
     predicted_ratings <-
          validation %>%
          left_join(Movie_effects_reg, by = "movieId") %>%
          left_join(User_effects_reg, by = "userId") %>%
          left_join(Genre_effects_reg, by = "genres") %>%
          mutate(pred = mu_hat + b_i_reg + b_u_reg + b_g_reg) %>%
          pull(pred)
     return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```



```r
tibble(Lambda = lambdas[which.min(rmses)], RMSE = min(rmses))
```

```
## # A tibble: 1 x 2
##   Lambda  RMSE
##    <dbl> <dbl>
## 1    5.1 0.864
```

```r
RMSE_results <- RMSE_results %>%
  add_row(Model = "baseline+b_i_reg+b_u_reg+b_g_reg",
          Desciption = "Regularised with movie+user+genres effects",
          RMSE_value = min(rmses))
RMSE_results
```

```
## # A tibble: 6 x 3
##   Model                            Desciption                                  RMSE_value
##   <chr>                            <chr>                                            <dbl>
## 1 Mean_hat                         baseline (using mean hat only)                   1.06
## 2 baseline+b_i                     baseline with movie effect                       0.944
## 3 baseline+b_i+b_u                 baseline with movie+user effects                 0.865
## 4 baseline+b_i+b_u+b_g             baseline with movie+user+genres effe~            0.865
## 5 baseline+b_i_reg+b_u_reg         Regularised with movie+user effects              0.865
## 6 baseline+b_i_reg+b_u_reg+b_g~ Regularised with movie+user+genres e~              0.864
```

9 Matrix factorization with parallel stochastic gradient descent.

A regression model has been derived that accounts for the movie effects, user effects and genre effects. This model is further regularized to take into consideration the large estimates that are formed using small sample sizes. However, this model excluded an important source of variation related to the fact that that groups of movies have similar rating patterns and groups of users have similar rating patterns as well (Rafael, 2020). These patterns can be discovered by analyzing the residuals, which is derive from the difference between predicted ratings and the reality.

With the understanding that within in a rating matrix, many groups of users and movies are somehow correlated, Matrix Factorization is explored to improve the RMSE. The "recosystem" package, an R wrapper of the LIBMF library, is used.

9.1 Convert training and validation datasets into matrix

```r
#Convert training and validation datasets into Matrix for processing
edx_MF <- edx %>% select(movieId, userId, rating)
edx_MF <- as.matrix(edx_MF)
validation_MF <- validation %>% select(movieId, userId, rating)
validation_MF <- as.matrix(validation_MF)
```

```r
#Write to files
write.table(edx_MF, file = "edx_MF.txt", sep = " ",
            row.names = FALSE, col.names = FALSE)
write.table(validation_MF, file = "validation_MF.txt", sep = " ",
            row.names = FALSE, col.names = FALSE)
```

```r
set.seed(123, sample.kind="Rounding")
edx_MF_set <- data_file("edx_MF.txt")
validation_MF_set <- data_file("validation_MF.txt")
```

9.2 Create and optimize a recommender model

```r
#Create a recommender object
recommender_obj <-Reco()

#optimization
optimized_values <- recommender_obj$tune(edx_MF_set, opts =
                        list(dim = c(10, 20, 30),
                        lrate = c(0.05, 0.1, 0.2),
                        costp_l1 = 0, costq_l1 = 0,
                        nthread = 6, niter = 10))
head(optimized_values)
```

```
## $min
## $min$dim
## [1] 30
##
## $min$costp_l1
## [1] 0
##
## $min$costp_l2
## [1] 0.1
##
## $min$costq_l1
## [1] 0
##
## $min$costq_l2
## [1] 0.01
##
## $min$lrate
## [1] 0.1
##
## $min$loss_fun
## [1] 0.7965013
##
##
## $res
##    dim costp_l1 costp_l2 costq_l1 costq_l2 lrate  loss_fun
## 1   10        0     0.01        0     0.01  0.05 0.8283173
## 2   20        0     0.01        0     0.01  0.05 0.8074412
## 3   30        0     0.01        0     0.01  0.05 0.8034478
## 4   10        0     0.10        0     0.01  0.05 0.8363509
## 5   20        0     0.10        0     0.01  0.05 0.8309741
## 6   30        0     0.10        0     0.01  0.05 0.8238981
## 7   10        0     0.01        0     0.10  0.05 0.8292144
## 8   20        0     0.01        0     0.10  0.05 0.8052216
## 9   30        0     0.01        0     0.10  0.05 0.7988526
```

```
## 10   10         0      0.10         0      0.10   0.05 0.8472171
## 11   20         0      0.10         0      0.10   0.05 0.8456078
## 12   30         0      0.10         0      0.10   0.05 0.8439901
## 13   10         0      0.01         0      0.01   0.10 0.8254754
## 14   20         0      0.01         0      0.01   0.10 0.8081964
## 15   30         0      0.01         0      0.01   0.10 0.8141016
## 16   10         0      0.10         0      0.01   0.10 0.8287163
## 17   20         0      0.10         0      0.01   0.10 0.8024268
## 18   30         0      0.10         0      0.01   0.10 0.7965013
## 19   10         0      0.01         0      0.10   0.10 0.8262323
## 20   20         0      0.01         0      0.10   0.10 0.8024821
## 21   30         0      0.01         0      0.10   0.10 0.8023695
## 22   10         0      0.10         0      0.10   0.10 0.8359578
## 23   20         0      0.10         0      0.10   0.10 0.8293012
## 24   30         0      0.10         0      0.10   0.10 0.8278826
## 25   10         0      0.01         0      0.01   0.20 0.8165171
## 26   20         0      0.01         0      0.01   0.20 0.8688507
## 27   30         0      0.01         0      0.01   0.20 0.9700106
## 28   10         0      0.10         0      0.01   0.20 0.8209275
## 29   20         0      0.10         0      0.01   0.20 0.8017433
## 30   30         0      0.10         0      0.01   0.20 0.7978538
## 31   10         0      0.01         0      0.10   0.20 0.9137031
## 32   20         0      0.01         0      0.10   0.20 0.9105215
## 33   30         0      0.01         0      0.10   0.20 0.9109787
## 34   10         0      0.10         0      0.10   0.20 0.8396796
## 35   20         0      0.10         0      0.10   0.20 0.8263650
## 36   30         0      0.10         0      0.10   0.20 0.8267795
```

9.3 Train the recommender model

```
#training the recommender model
recommender_obj$train(edx_MF_set, opts = c(optimized_values$min, nthread = 6, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.9731   1.2024e+07
##    1       0.8730   9.9010e+06
##    2       0.8382   9.1655e+06
##    3       0.8163   8.7507e+06
##    4       0.8000   8.4622e+06
##    5       0.7875   8.2601e+06
##    6       0.7777   8.1026e+06
##    7       0.7697   7.9858e+06
##    8       0.7631   7.8915e+06
##    9       0.7574   7.8122e+06
##   10       0.7523   7.7448e+06
##   11       0.7479   7.6891e+06
##   12       0.7439   7.6410e+06
##   13       0.7401   7.5970e+06
##   14       0.7369   7.5615e+06
##   15       0.7337   7.5244e+06
##   16       0.7308   7.4938e+06
##   17       0.7283   7.4675e+06
```

```
## 18 0.7259 7.4438e+06
## 19 0.7236 7.4204e+06
```

9.4 Make prediction with trained recommender model

```r
# Make prediction on validation_MF_set:
Prediction_file <- tempfile()
recommender_obj$predict(validation_MF_set, out_file(Prediction_file))
```

```
## prediction output generated at C:\Users\lingl\AppData\Local\Temp\Rtmp0Q5cty\file56e4780e3771
```

```r
Validation_ratings <- read.table("validation_MF.txt", header = FALSE, sep = " ")$V3
Predicted_ratings <- scan(Prediction_file)
```

9.5 Compute the RMSE of the recommender model

```r
#calculate RMSE
rmse <- RMSE(Predicted_ratings, Validation_ratings)
RMSE_results <- RMSE_results %>%
          add_row(Model = "Matrix Factorization",
          Desciption = "Matrix Factorization with parallel stochastic gradient descent",
          RMSE_value = rmse)
RMSE_results
```

```
## # A tibble: 7 x 3
##   Model                   Desciption                                   RMSE_value
##   <chr>                   <chr>                                             <dbl>
## 1 Mean_hat                baseline (using mean hat only)                    1.06
## 2 baseline+b_i            baseline with movie effect                        0.944
## 3 baseline+b_i+b_u        baseline with movie+user effects                  0.865
## 4 baseline+b_i+b_u+b_g    baseline with movie+user+genres effects           0.865
## 5 baseline+b_i_reg+b_u_r~ Regularised with movie+user effects               0.865
## 6 baseline+b_i_reg+b_u_r~ Regularised with movie+user+genres effects        0.864
## 7 Matrix Factorization    Matrix Factorization with parallel stochas~       0.782
```

10 Conclusion

From the summary of the RMSE table above, the baseline model, which used the mean rating of all the movies, gave a RMSE value of 1.061. By taking movie and user effects into consideration in the baseline model, the RMSE value improved by 18.5% to 0.865. By further adding the genre effects to the latter model, the improvement in RMSE value is not significant. This is due to the average rating given for each genre does not vary much from the mean rating. To further improvement the RMSE value to achieve the target RMSE value less than 0.8649, the concept of regularization was introduced. With regularization, the lowest RMSE value obtained is 0.86445 (Model: baseline with Regularized movie, user and genres effects). This is less than target RMSE value of 0.8649.

With the understanding that this regularized regression model excluded an important source of variation related to the fact that that groups of movies have similar rating patterns and groups of users have similar rating patterns as well (Rafael, 2020), Matrix Factorization technique is explored and the RMSE value obtained is 0.78255. This is 9.5% improvement in RMSE value over the regularized regression model. As the lowest RMSE value obtained, the Matrix Factorization model is selected for movie recommendation system.

11 References

Rafael A. Irizarry, 2020, Introduction to Data Science: Data Analysis and Prediction Algorithms with R, Chapman and Hall/CRC.

Bradley, Boehmke & Brandon, Greenwell, 2020, Hands-On Machine Learning with R, Chapman and Hall/CRC.