Movielens

Fahmi Harum

5/24/2020

1. Introduction

Data has been essential for nowadays growth, data can be manipulated and elaborated by using statistical and analytics knowledge. Through this process data can be one of the most precious things in the world. This project has required me to create a movie recommendation system.

In details, the project specified the task to predict the rating a user will give a movie in a validation set based on a given set of users and movie ratings. A machine learning algorithm using the inputs in one subset will be developed to predict the movie ratings in the validation set. RMSE would be used to test the accuracy of the algorithm.

The data provided is Movielens 10M is further created and validated using EDX preprocessing syntaxes. The dataset can be acquired online in the 'dslabs' containing 9000055 tuples and 8 attributes in total.

This project will analyze all the data, gather insights from the analysis, create a prediction model and conclude a prediction results from the models.

Used Dataset

- [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/ (https://grouplens.org/datasets/movielens/10m/)
- [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip (http://files.grouplens.org/datasets/movielens/ml-10m.zip)

Data Loading & validation

```
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")

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
```

Used Libraries

The following libraries were used in this report:

```
library(ggplot2)
library(lubridate)
library(dplyr)
library(stringr)
library(tidyr)
```

Aim & Objectives

The aim of this project is to develop a machine learning algorithm by making use of inputs acquired from the dataset stated above in Datasets that will be able to forecast movie ratings based on the validated datasets.

Some external libraries will be used in data exploration to acquire crucial insights and trends from the dataset and the factors affecting users' voting. Four models will be created and assessed by comparing them through the RMSE results. An optimal model out of four models will be used for the prediction.

2. Methodology & Analysis

Data Pre-Processing & Gaining Insights(EDA)

RMSE evaluation

Measurement of the accuracy of the model will be assessed by the Root Mean Square Error(RMSE). Compared with Mean Average Error(MAE), RMSE would be more suitable to this project because it penalizes larger errors stronger and hence suitable for cases that consider minor prediction errors are not the priority.

```
# function to calculate the RMSE values
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2,na.rm = T))
}
```

Training & Testing Datasets

Before proceeding further with this project, training and testing datasets will need to be acquired from the original dataset as a prerequisite to develop prediction models. The edx partitioned set will be used for training the algorithm and validation dataset will be used for testing. The ratio for the partitioning will be 10% for testing and 90% for training.

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

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

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
# Validation dataset can be further modified by removing rating column
validation_CM <- validation
validation <- validation %>% select(-rating)

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

Modifying the Year & Genre

Dependencies in between the rating and release year can only be used by including the release year in a separated column, so we need to extract the year release from the title column. This is also done to the genre to extract to be attributes to that will be used for analysis

```
# lets modify the columns to suitable formats that can be further used for analysis
# Modify the year as a column in the edx & validation datasets
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
# Modify the genres variable in the edx & validation dataset (column separated)
split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
split_valid <- validation   %>% separate_rows(genres, sep = "\\|")
split_valid_CM <- validation_CM  %>% separate_rows(genres, sep = "\\|")
```

Data Exploration & Visualization

Summary Statistics/General Data Information

The dataset is already cleaned and well organized. However, it will be further validated and preprocessed through the preprocessing process to do data cleaning and produce training and testing dataset. This project will breakdown the dataset into much precise form to gather valuable analysis and insights.

The datasets initially come with a tuple containing the name and the year of the movie. The year is extracted from the column to give a better analysis and ease the predictions process. And this process will need the newly extracted attribute to append with the validation dataset.

```
# The 1st rows of the edx & split_edx datasets are presented below:
head(edx)
```

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

```
head(split_edx)
```

```
## # A tibble: 6 x 7
##   userId movieId rating timestamp title            genres    year
##    <int>   <dbl>  <dbl>     <int> <chr>            <chr>    <dbl>
## 1      1     122      5 838985046 Boomerang (1992) Comedy    1992
## 2      1     122      5 838985046 Boomerang (1992) Romance   1992
## 3      1     185      5 838983525 Net, The (1995)  Action    1995
## 4      1     185      5 838983525 Net, The (1995)  Crime     1995
## 5      1     185      5 838983525 Net, The (1995)  Thriller  1995
## 6      1     292      5 838983421 Outbreak (1995)  Action    1995
```

```
# edx Summary Statistics
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              year
##  Length:9000055     Length:9000055     Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1994
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

```
# Number of unique movies and users in the edx dataset
edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

Then we extracted and separated the movies according to the genres in the split_edx, which a movie may have multiple genres. So, this split_edx will have recurring titles with unique genres classification. And it will be validated using the same method base in the EDX preprocessing scripts.

After genre splitting, we can acquire the total numbers of the movies and the numbers of the users of the dataset. Insights acquired are; a total of 69878 users has rated for 10677 movies listed in the datasets. Belows are some insights gathered from the analysis:

####Total Observation

```
#Getting observation count
length(edx$rating) + length(validation$rating)
```

```
## [1] 9000055
```

####Total Movie and User

```
#getting user count and movie count
edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

####Top 10 Movies Ranked

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

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                     count
##      <dbl> <chr>                                                     <int>
##  1     296 Pulp Fiction (1994)                                       31362
##  2     356 Forrest Gump (1994)                                       31079
##  3     593 Silence of the Lambs, The (1991)                          30382
##  4     480 Jurassic Park (1993)                                      29360
##  5     318 Shawshank Redemption, The (1994)                          28015
##  6     110 Braveheart (1995)                                         26212
##  7     457 Fugitive, The (1993)                                      25998
##  8     589 Terminator 2: Judgment Day (1991)                         25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10     150 Apollo 13 (1995)                                          24284
## # ... with 10,667 more rows
```

####Total Movie Ratings per Genre

```
#getting rating counts per genre
split_edx%>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
## 20 (no genres listed)       7
```

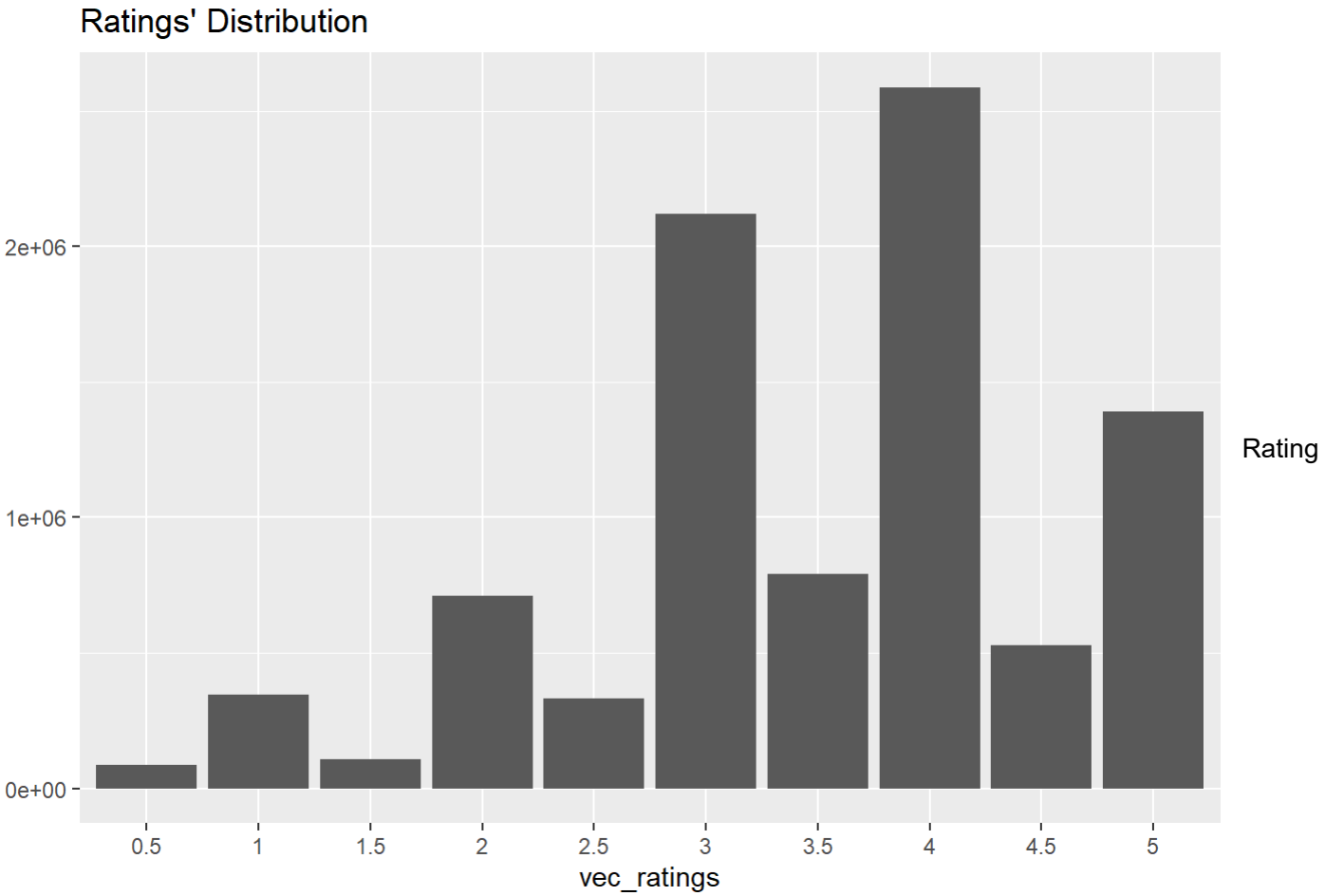
Ratings Distribution

The rating attribute in the validated dataset will be used to measure the models predictions. The rating holds value of 0.5 to 5.0 sequentially increasing by 0.5 which gives a better classification by doubling the range for better precisions. The movie has been splitted by the given ratings and has been classified uniquely to avoid redundant data.

```
#converting to vectors
vec_ratings <- as.vector(edx$rating)
unique(vec_ratings)
```

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

```
#plotting rating histogram
vec_ratings <- vec_ratings[vec_ratings != 0]
vec_ratings <- factor(vec_ratings)
qplot(vec_ratings) +
  ggtitle("Ratings' Distribution")
```

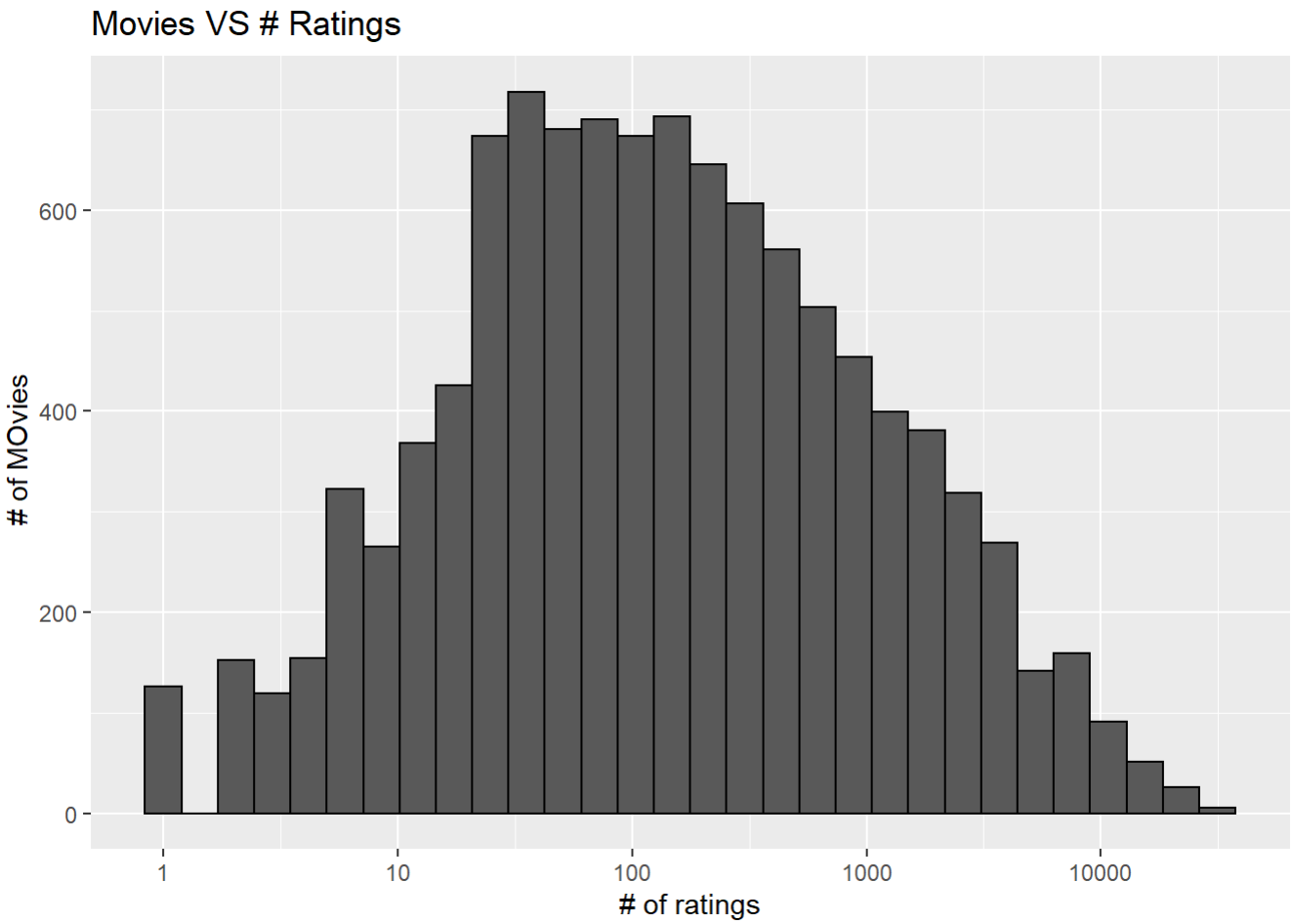


distribution has been acquired and in general, it can see that users tend to vote between 3.0 and 4.0 rate. To make a better predictions model, it is needed to explore the different features in the data set.

##Features Elaboration From the insights gathered in the processed data and EDA, it has been found that; some movies are rated more often than others, users has personal preferences to particular movies which will have varies review on the movies, popularity of the movie genre may depends on the preferred trends of the year and the perspective of a movies may evolve over the time. This project will focus on these features to support the model building :

Movie Bias

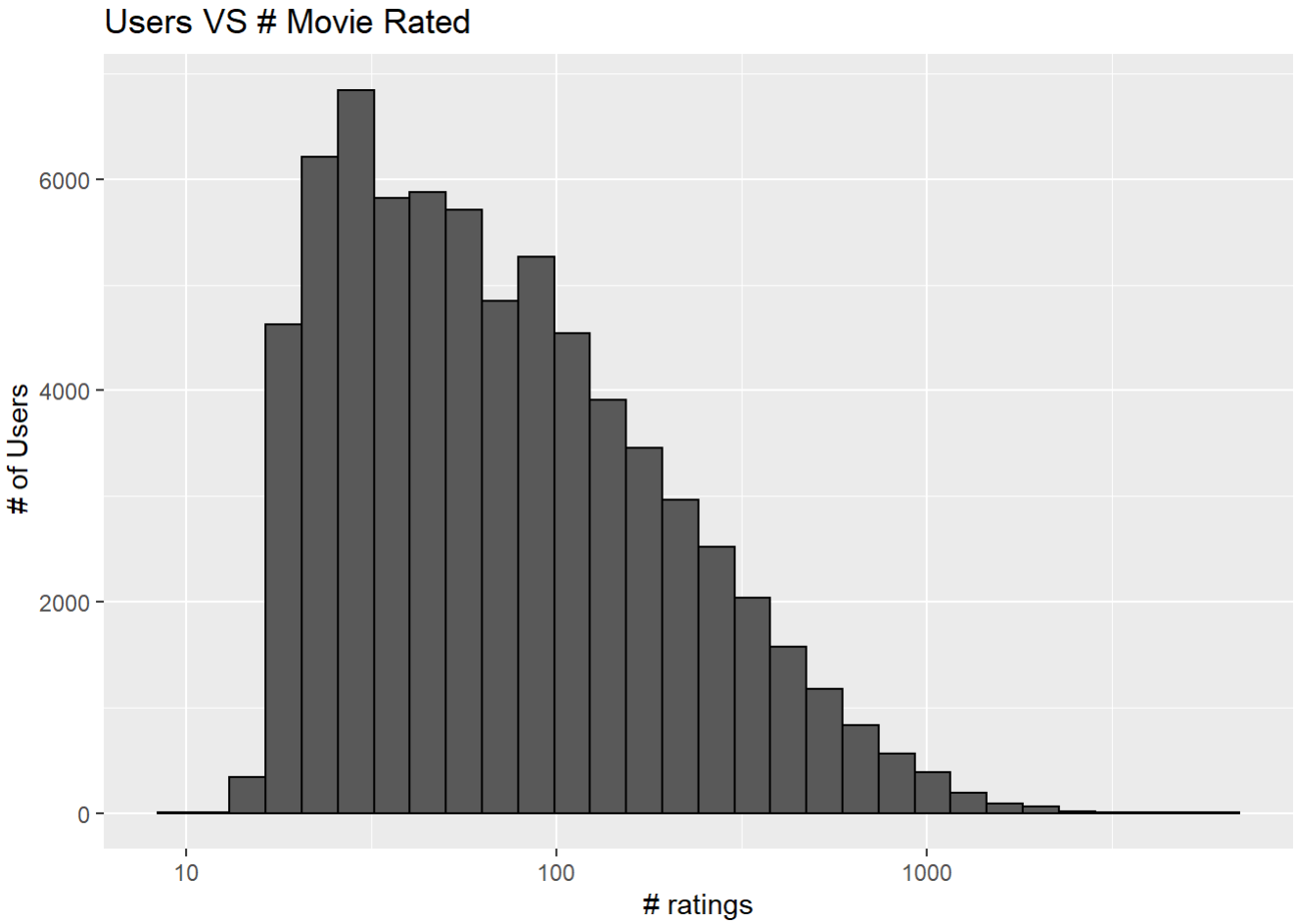
```
#plotting # movies & #  rated to explore user bias

edx %>%
    count(movieId) %>%
    ggplot(aes(n)) +
    geom_histogram(bins = 30, color = "black") +
    scale_x_log10() +
    ggtitle("Movies VS # Ratings")+
    labs(y=" # of MOvies", x = "# of ratings")
```



From the graph, it can be seen that movies have variation of rating given which can be inferred that less rated movies should be given lower importance in movie prediction.

User Bias

```
#plotting # user & # movie rated to explore user bias
edx %>% count(userId) %>%
    ggplot(aes(n)) +
    geom_histogram(bins = 30, color = "black") +
    scale_x_log10() +
    ggtitle("Users VS # Movie Rated")+
    labs(y=" # of Users", x = "# ratings")
```

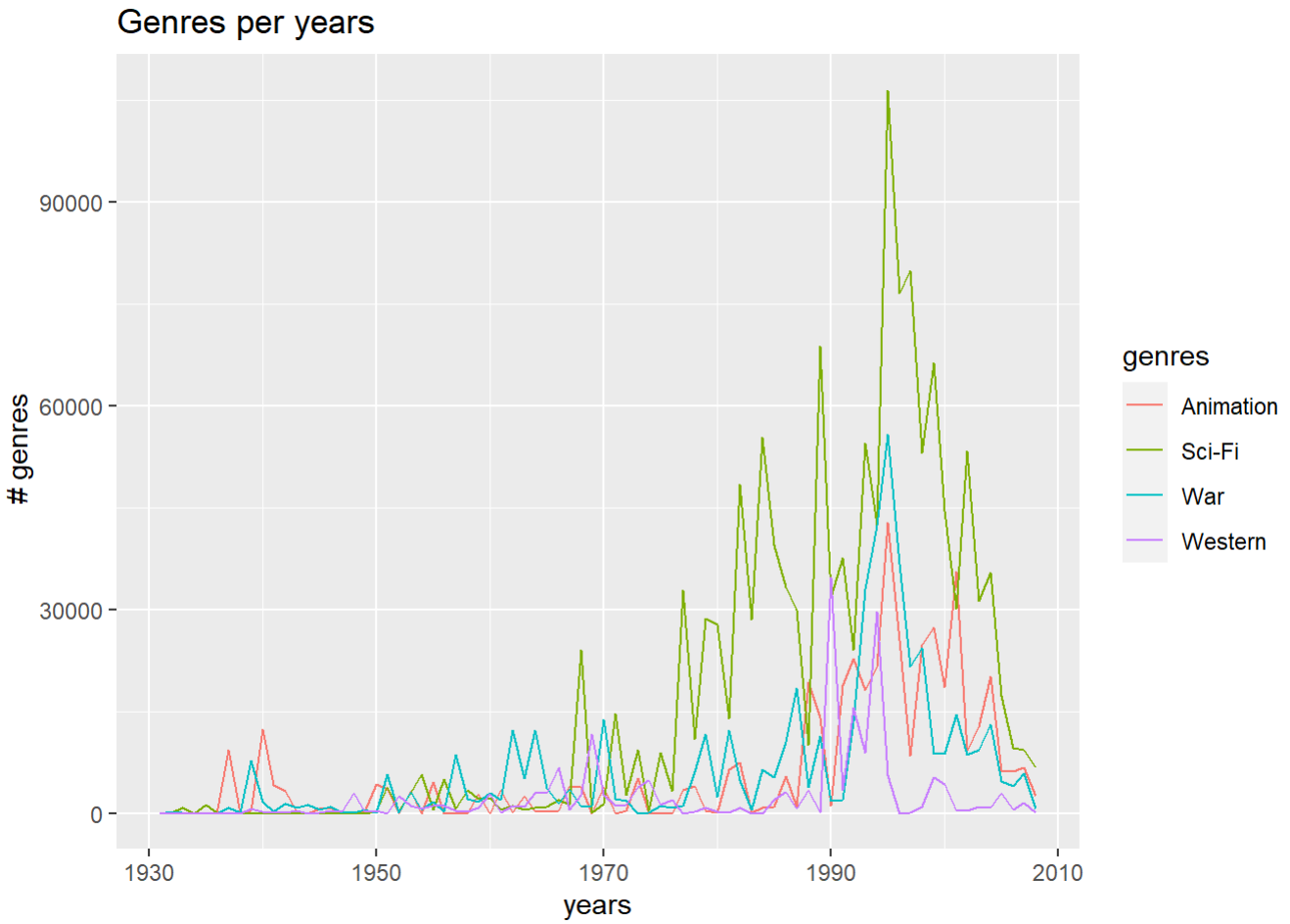


From the histogram, it can be concluded that not all users are equally active in giving ratings which result in bias from the users to the prediction results.

Genres Popularity per Year

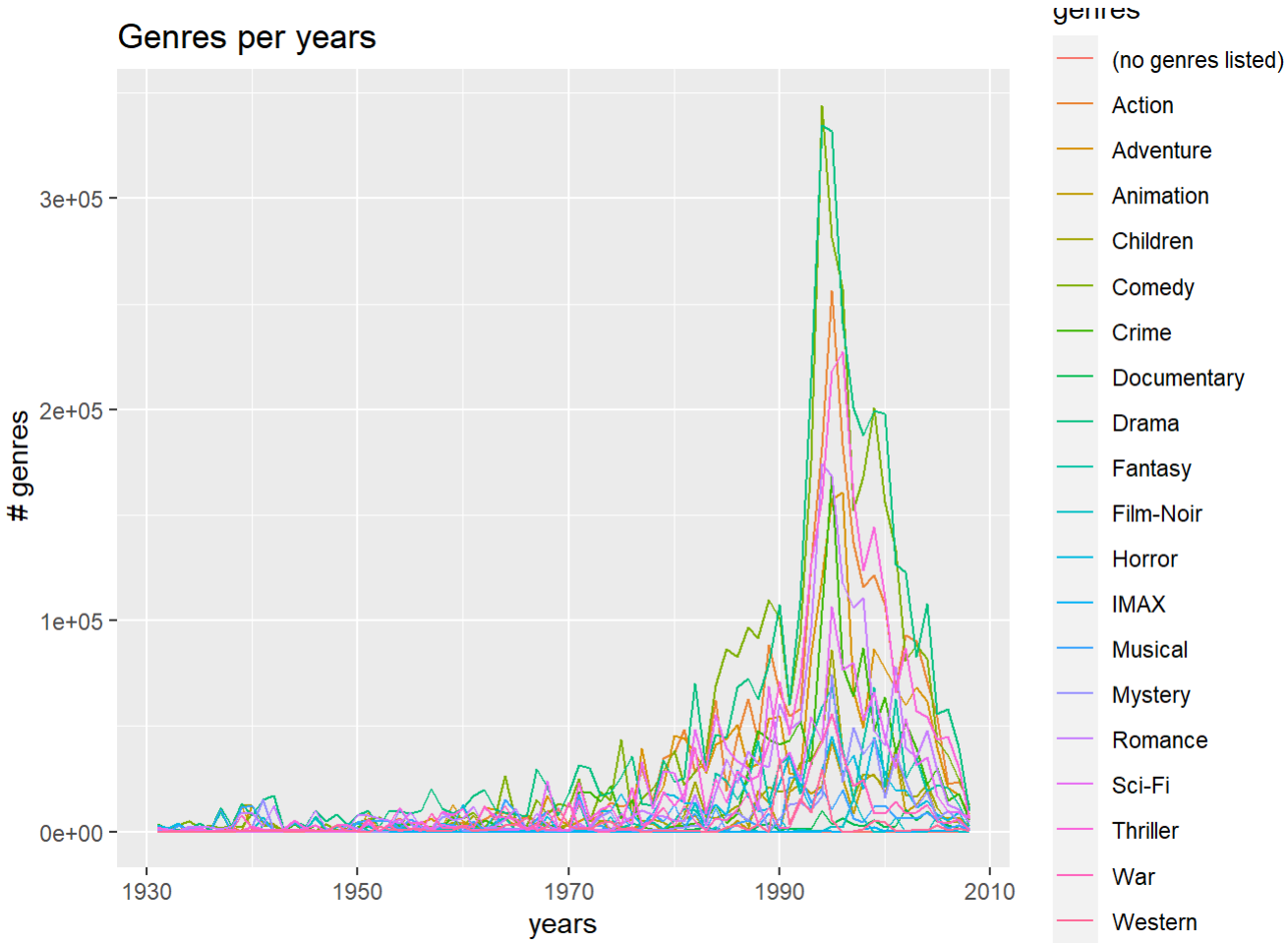
Genres vs year: 4 genres are chosen for readability: animation, sci-fi, war and western movies.

```
genres_popularity %>%
  filter(year > 1930) %>%
  filter(genres %in% c("War", "Sci-Fi", "Animation", "Western")) %>%
  ggplot(aes(x = year, y = number)) +
  geom_line(aes(color=genres)) +
  scale_fill_brewer(palette = "Paired")+
  labs(y=" # genres", x = "years")+
  ggtitle("Genres per years")
```



From the chart, some popular genres have been plotted to see the trends over the period of time. It can be seen that the most favored genres from 1970 to 2010 are the western genre movies. And the chart of all of the genres is below; the most preferred genre is the drama genre.

```
genres_popularity %>%
  filter(year > 1930) %>%
  ggplot(aes(x = year, y = number)) +
  geom_line(aes(color=genres)) +
  scale_fill_brewer(palette = "Paired")+
  labs(y=" # genres", x = "years")+
  ggtitle("Genres per years")
```



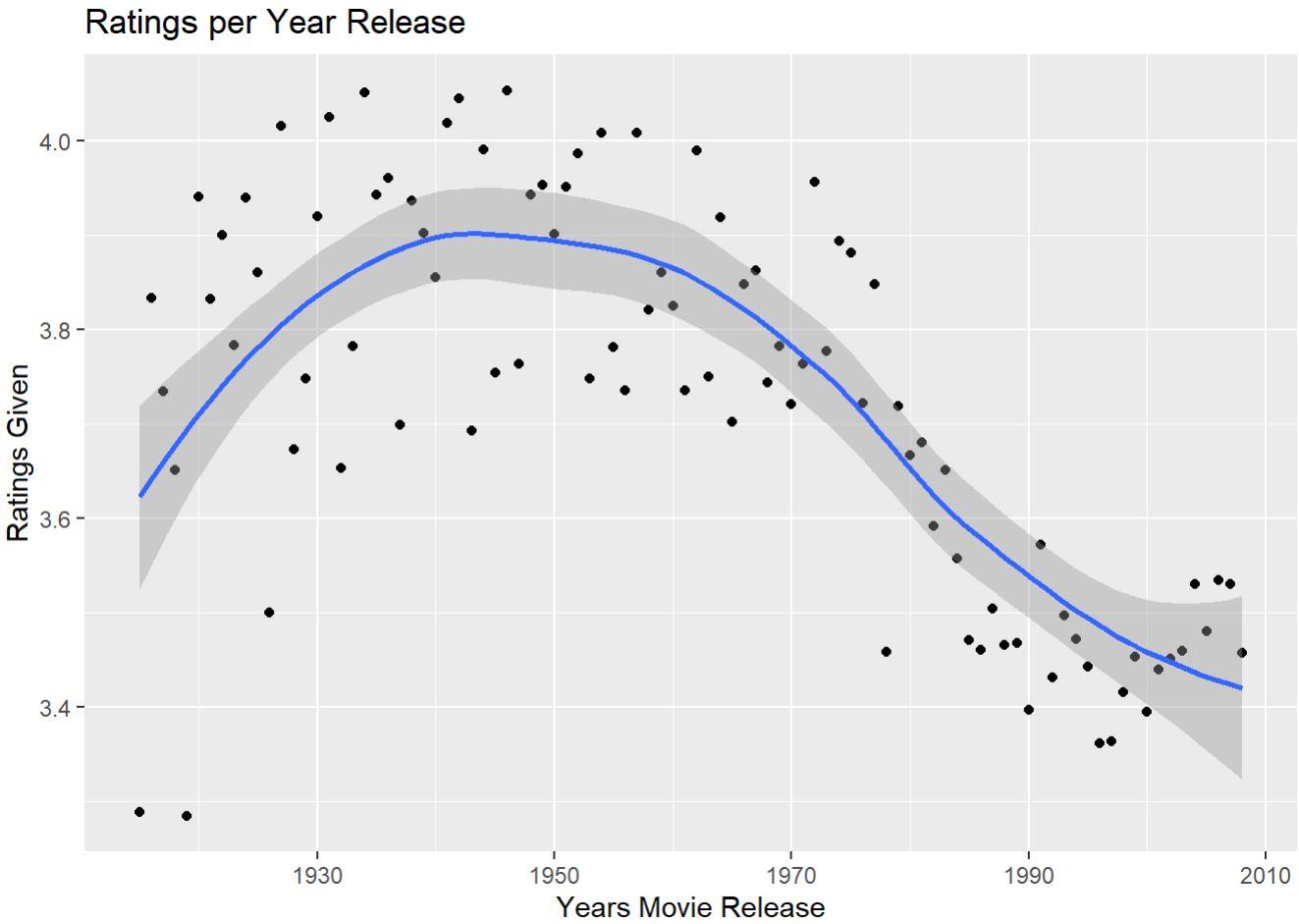
The Effects of Release Year & Genre on Ratings

Rating vs Release Year

A clear trend is shown in the below figure: the most recent years have in average lower rating than earlier years.

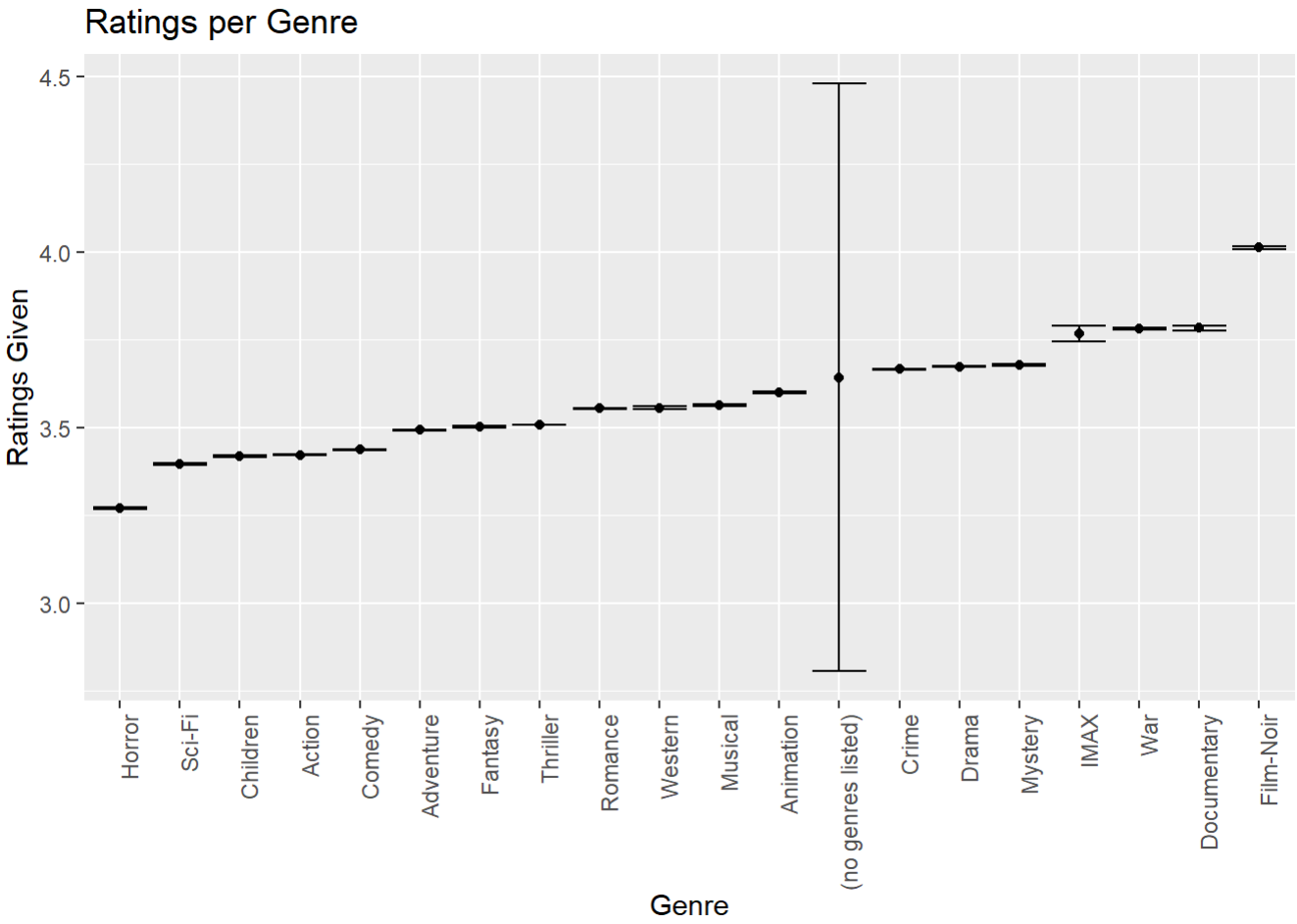
```
#plotting rating vs release year for the movies to see user rating trends.
edx %>% group_by(year) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(year, rating)) +
  geom_point() +
  geom_smooth()+
  ggtitle("Ratings per Year Release")+
  labs(y="Ratings Given",x="Years Movie Release")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The graph above depicts the trend of user ratings per year released, it can be inferred that in recent years, the lower ratings are given than the late 1930 years. It can be understood that movies at that time were of low quality as there are so many constraints at that moment.

```
#plotting rating vs release year for the movies to see user rating trends.
split_edx %>% group_by(genres) %>%
   summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
   mutate(genres = reorder(genres, avg)) %>%
   ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
   geom_point() +
   geom_errorbar() +
   theme(axis.text.x = element_text(angle = 90, hjust = 1))+
   ggtitle("Ratings per Genre")+
   labs(y="Ratings Given",x="Genre")
```



This depicts the rating given according to the genre. It can be seen on the histogram that the highest median is Film-Noir genre and the genre lowest median is Horror.

Model Development : Preparation

```
#Initiate RMSE results to compare various models
rmse_results <- data_frame()
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

Mean Ratings

The initial step is to compute the dataset's mean rating.

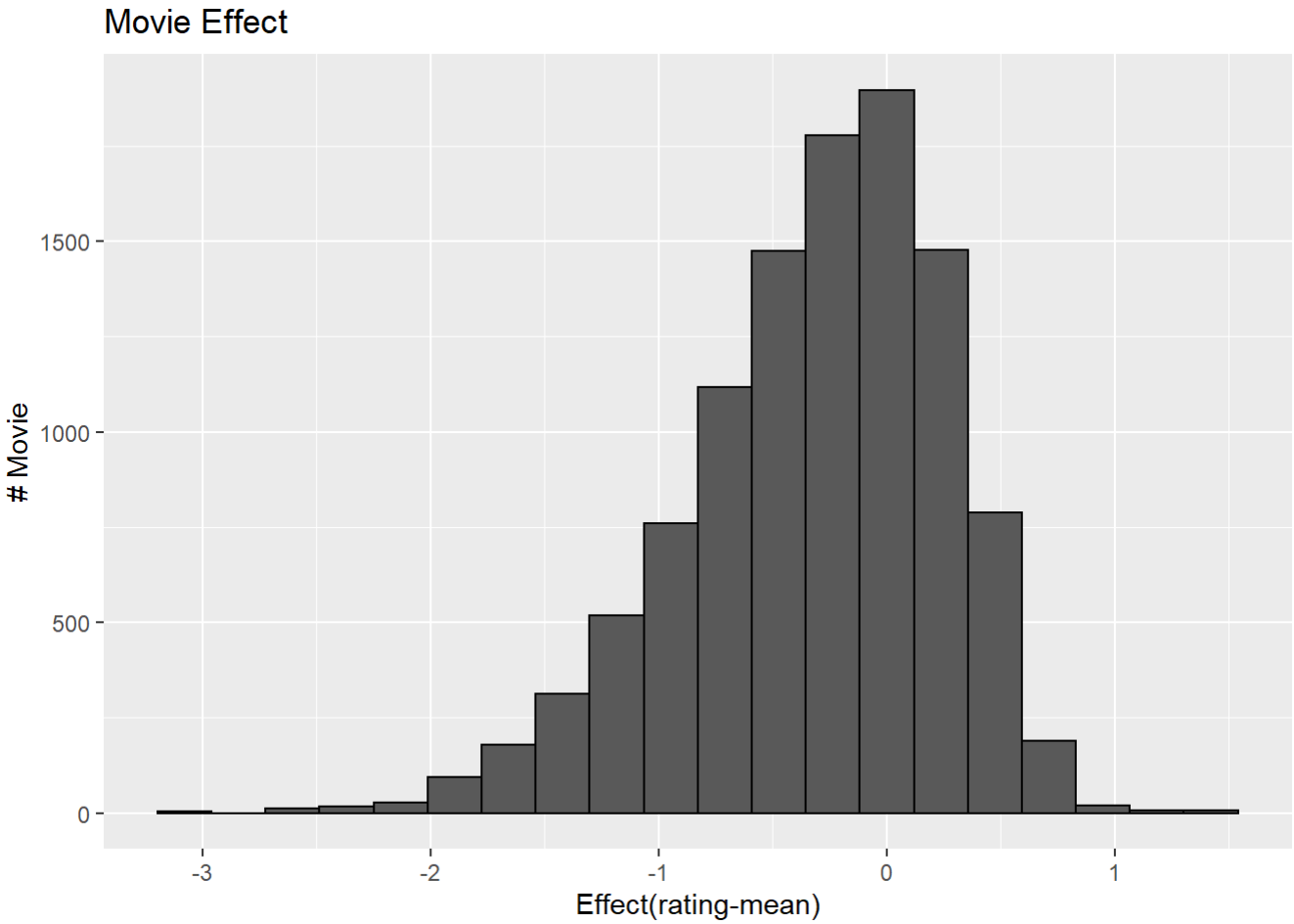
```
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

Movie Effect

In model development, the most basic step to start with is the mean, for this dataset the mean rating is 3.5(3.512465). Considering the movie bias, the histogram of the ratings distribution can be seen skewed because one of factors of the effect of the bias can be take into account by finding the penalty term: B_I = mean(rating-mean(rating))

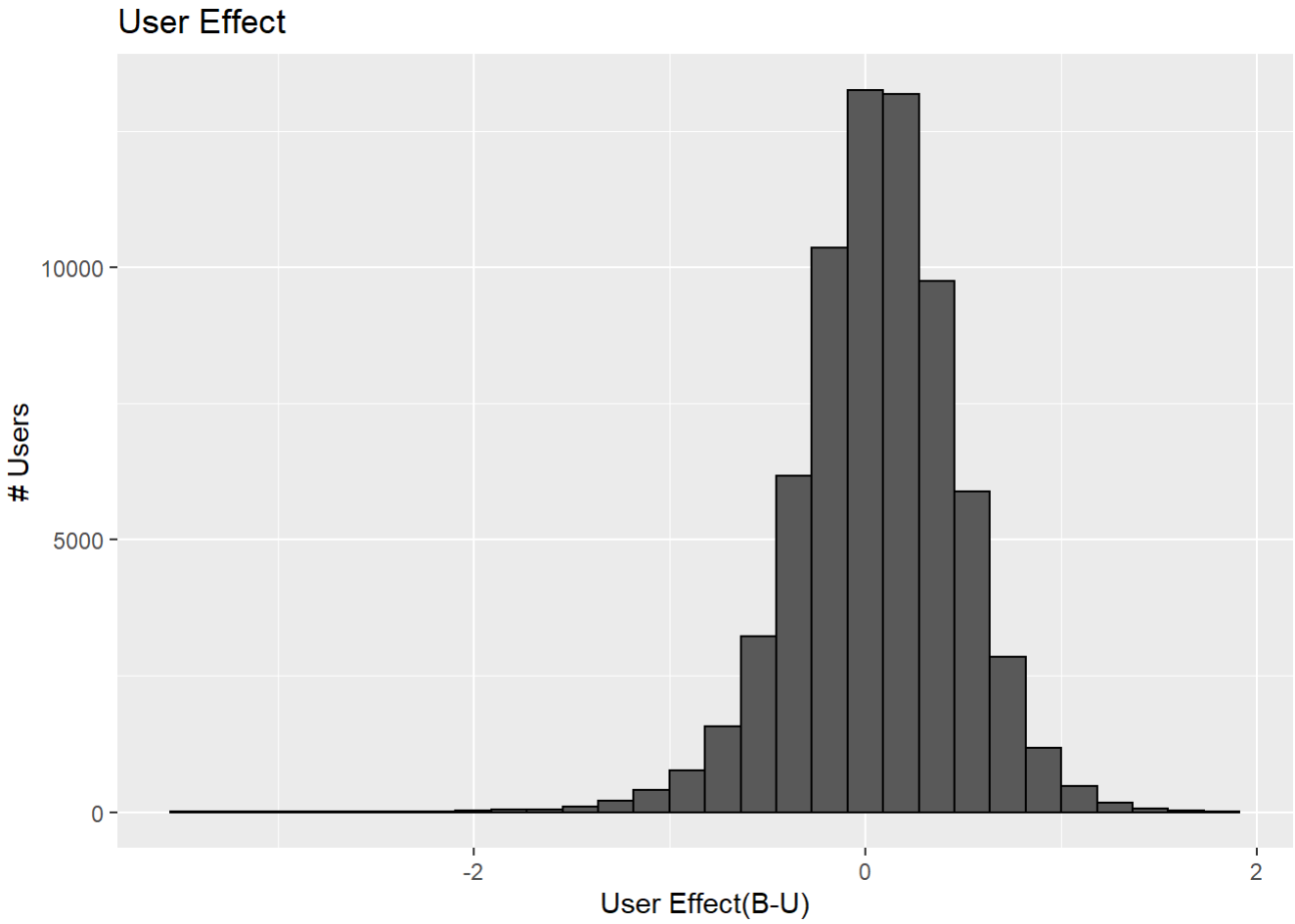
```
#taking into account the movie effect cause by the bias
movie_avgs_norm <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

#plotting graph for movie effexts
movie_avgs_norm %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("black"))+
  ggtitle("Movie Effect")+
  labs(y="# Movie", x="Effect(rating-mean)")
```



###User Effect

Another feature discovered from the EDA is the user bias, different users rate movies differently according to your perspective. Picky users may rate a good movie movie badly or vice versa because they do not care for the assessment of the movie. The bias can be empirically calculated through the penalty term of: B_U = mean(rating - mu - B_I)

```
#taking user effects into account
user_avgs_norm <- edx %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

#plotting graph for user effects
user_avgs_norm %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))+
  ggtitle("User Effect")+
  labs(y="# Users", x="User Effect(B-U)")
```



Model Development: Creation

The quality of the model will be assessed by the RMSE (the lower the better).

Baseline Model

For model building process, RMSE will be used to measure the accuracy or quality of a model. It is expected that models with lower RMSE have better quality. First, a base model is developed to act as the foundation of models that are to be developed. Base model does a simple method by ignoring all features and takes mean into consideration as a prediction value.

```
#model creation part
#creating base model
#rmse(rating,mu)- taking mean as prediction
baseline_rmse <- RMSE(validation_CM$rating,mu)
## Test results based on simple prediction
baseline_rmse
```

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

```
## Check results
rmse_results <- data_frame(method = "Using mean only", RMSE = baseline_rmse)
rmse_results
```

```
## # A tibble: 1 x 2
##   method           RMSE
##   <chr>           <dbl>
## 1 Using mean only  1.06
```

Movie Effect Model

The next model will take into account the movie bias, Movie Effect model is an improved version of the baseline model which will make use of the b_i penalty described in the preparation stage. This model eliminated the bias from the movie. By using a simple math formula for predicting, adding mean with the penalty term.

```
# Movie effects only
predicted_ratings_movie_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  mutate(pred = mu + b_i)
model_movie_rmse <- RMSE(validation_CM$rating,predicted_ratings_movie_norm$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_movie_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087

```
rmse_results
```

```
## # A tibble: 2 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Using mean only    1.06
## 2 Movie Effect Model 0.944
```

Movie and User Effect Model

In this model, b_u and b_i are taken into consideration to eliminate the biases from both parties: user and model.

```
# Use test set,join movie averages & user averages
# Prediction equals the mean with user effect b_u & movie effect b_i
predicted_ratings_user_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  left_join(user_avgs_norm, by='userId') %>%
  mutate(pred = mu + b_i + b_u)


# test and save rmse results
model_mnu_rmse <- RMSE(validation_CM$rating,predicted_ratings_user_norm$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and User Effect Model",
                                     RMSE = model_mnu_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

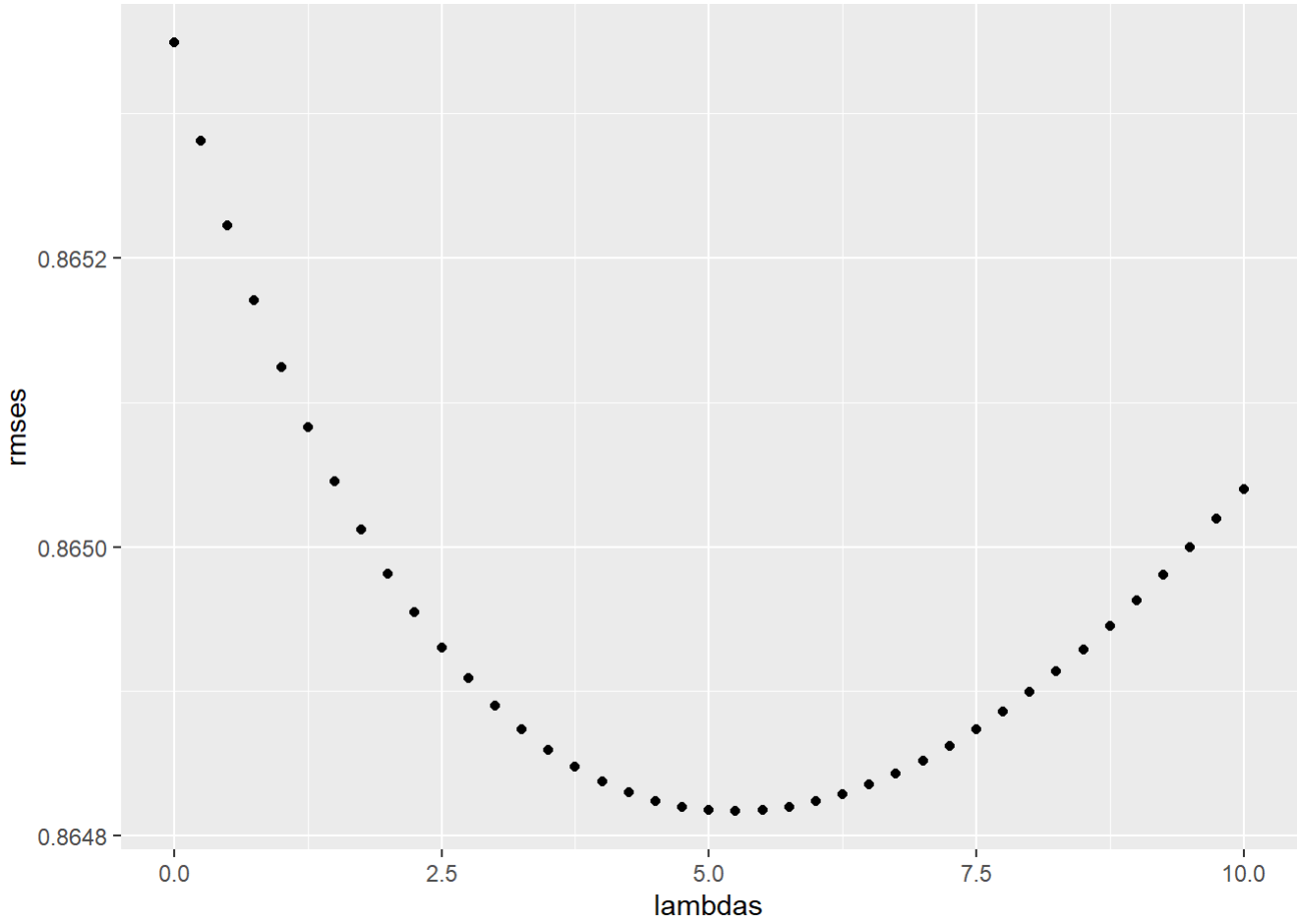
```
rmse_results
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Using mean only            1.06
## 2 Movie Effect Model         0.944
## 3 Movie and User Effect Model 0.865
```

Regularized Movie and User Effect Model

Using the concept of regularization, this model takes into account low ratings' number effects from users and movies. In the previous section, it showed that the ratings given are not consistent by the users, which will strongly affect the prediction, so regularization helps to reduce the overfitting effect.

```
# Use cross-validation to choose lmabda.
lambdas <- seq(0, 10, 0.25)
# For each lambda,find b_i & b_u, followed by rating prediction & testing
# note:the below code could take some time
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

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

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

  return(RMSE(validation_CM$rating,predicted_ratings))
})
# Plot rmses vs lambdas to select the optimal lambda
qplot(lambdas, rmses)
```




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

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

```
# Compute regularized estimates of b_i using lambda
movie_avgs_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())
# Compute regularized estimates of b_u using lambda
user_avgs_reg <- edx %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n())
# Predict ratin  gs
predicted_ratings_reg <- validation %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  left_join(user_avgs_reg, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
# Test and save results
model_regmnu_rmse <- RMSE(validation_CM$rating,predicted_ratings_reg)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie and User Effect Model",
                                     RMSE = model_regmnu_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018

method	RMSE
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170

```
rmse_results
```

```
## # A tibble: 4 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Using mean only                      1.06 
## 2 Movie Effect Model                   0.944
## 3 Movie and User Effect Model          0.865
## 4 Regularized Movie and User Effect Model 0.865
```

Regularized Movie, User, Year and Genre Effects Model

This model improved all the previous models and utilizes the genre and rating effects too

```
# b_y and b_g represent the year & genre effects, respectively
lambdas <- seq(0, 20, 1)
# Note: the below code could take some time
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

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

  b_y <- split_edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+lambda), n_y = n())

  b_g <- split_edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by = 'year') %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u - b_y)/(n()+lambda), n_g = n())
  predicted_ratings <- split_valid %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by = 'year') %>%
    left_join(b_g, by = 'genres') %>%
    mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
    .$pred

  return(RMSE(split_valid_CM$rating,predicted_ratings))
})
# Compute new predictions using the optimal lambda
# Test and save results
lamplot2 <- qplot(lambdas, rmses)

lambda_2 <- lambdas[which.min(rmses)]
lambda_2
```

```
## [1] 14
```

```
movie_reg_avgs_2 <- split_edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_2), n_i = n())
user_reg_avgs_2 <- split_edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda_2), n_u = n())
year_reg_avgs <- split_edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  group_by(year) %>%
  summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+lambda_2), n_y = n())
genre_reg_avgs <- split_edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  left_join(year_reg_avgs, by = 'year') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u - b_y)/(n()+lambda_2), n_g = n())
predicted_ratings <- split_valid %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  left_join(year_reg_avgs, by = 'year') %>%
  left_join(genre_reg_avgs, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
  .$pred


model_regmnug_rmse <- RMSE(split_valid_CM$rating,predicted_ratings)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Reg Movie, User, Year, and Genre Effect Model",
                                     RMSE = model_regmnug_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170
Reg Movie, User, Year, and Genre Effect Model	0.8623650

3. Results

RMSE overview

The RMSE values for the used models are shown below:

```
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018

method	RMSE
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170
Reg Movie, User, Year, and Genre Effect Model	0.8623650

Rating Prediction using Regularized MUG Model

This model results the lowest RMSE above other models, which give the most accurate result for final prediction. Since the ratings are continuous, all of the predictions will be rounded to nearest 0.5 values to give a better comprehension after the prediction is done.

```
lambda_3<-14
# Redo model 4 analysis
movie_reg_avgs_2 <- split_edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_3), n_i = n())
user_reg_avgs_2 <- split_edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda_3), n_u = n())
year_reg_avgs <- split_edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  group_by(year) %>%
  summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+lambda_3), n_y = n())
genre_reg_avgs <- split_edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  left_join(year_reg_avgs, by = 'year') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u - b_y)/(n()+lambda_3), n_g = n())
## Adding all effects to the validation set & predicting the ratings
## Group by userId & movieID
## Compute each prediction's mean
predicted_ratings <- split_valid %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  left_join(year_reg_avgs, by = 'year') %>%
  left_join(genre_reg_avgs, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
  group_by(userId,movieId) %>% summarize(pred_2 = mean(pred))
```

```
# Round predicted_ratings & confirm that they're between 0.5 & 5
predicted_ratings <- round(predicted_ratings*2)/2
predicted_ratings$pred_2[which(predicted_ratings$pred_2<1)] <- 0.5
predicted_ratings$pred_2[which(predicted_ratings$pred_2>5)] <- 5
```

4. Conclusion

The most accurate and precise model to predict the movie ratings is Regularized Movie,User, Year and Genre Model which has the lowest RMSE value (0.862) and it will be used for optimized results for the predictions for this project.

For further studies, maybe more features or attributes that may manipulate the ratings poll can be taken into consideration for model development, like age and genders may manipulate the genre preferences of a personnel. Moreover, the use of machine learning algorithms like Decision Trees or Neural Networks could improve the results accuracy. As for this project, few constraints like machine constraints and capacity size, has limited the ability to analyze deep further into this dataset and has been put for further future research.