HarvardX-PH125.9x-Capstone-Project-MovieLens-Report

Jollanda Shara

June, 2021

Contents

1 Introduction

In this assignment we will use the MovieLens extensive dataset, the 10M dataset from the GroupLens research lab. MovieLens is a web-based recommender system and virtual community that recommends movies for its users to watch, based on their film preferences using collaborative filtering of members' movie ratings and movie reviews. The MovieLens data set contains 10000054 rows, 10677 movies, 797 genres and 69878 users. MovieLens was created in 1997 by GroupLens Research, a research lab in the Department of Computer Science and Engineering at the University of Minnesota, in order to gather research data on personalized recommendations. A recommender system, or a recommendation system (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. During the last few decades, with the rise of Youtube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries).

1.1 Objective

The purpose for this project is creating a recommender system using MovieLens dataset and its evaluation based on the RMSE - Root Mean Squared Error. The formula for the RMSE we will use in this assignment is: The function that computes the RMSE for vectors of ratings and their corresponding predictors will be

the following:

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

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

The goal is to make a recommendation system with the lowest possible RMSE. It should be at least lower than 0.87750.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

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

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

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

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
## Warning: package 'forcats' was built under R version 4.0.5
```

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

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

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

```
## The following object is masked _by_ '.GlobalEnv':
##
##     RMSE
```

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

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

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

```
## Warning: package 'data.table' was built under R version 4.0.5
```

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

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

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

```
library(tidyverse)
library(caret)
library(data.table)
```

2 Data Analysis

2.1 The edx and validation datasets

The MovieLens dataset will be splitted into 2 subsets that will be the "edx", a training subset to train the algorithm, and "validation" a subset to test the movie ratings.
The edx validation dataset both contain approximately 69.000 different users and 10.000 different movies. There are no NA-values in both datasets. The users gave these movies a rating on a scale between 0 and 5 (with steps of 0.5), with 0 the worst rating and 5 the best rating for a movie. The edx dataset, which is our training dataset, contains 69878 users and 10677 movies. The validation dataset, which is our test dataset, contains 68534 users and 9809 movies.

Both datasets have six columns/variables: . userId (the identification number for each user) . movieId (the identification number for each movie), . rating (the rating of a movie by a user), . timestamp (the timestamp of the rating provided by a user), . title (the title of each movie including the year of release), . genres (a list of genres for each movie).

```
# 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)

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 3.6 or earlier:
# movies <- as.data.frame(movies) %>% mutate(movieId =
# as.numeric(levels(movieId))[movieId],                                          #title = as.characte
# 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")
```

```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)`
```

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

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

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

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

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

```r
edx <- rbind(edx, removed)

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

```r
# Head
head(edx) %>%
  print.data.frame()
```

```
##   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
```

Total unique movies and users

```r
# Total unique movies and users
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
```

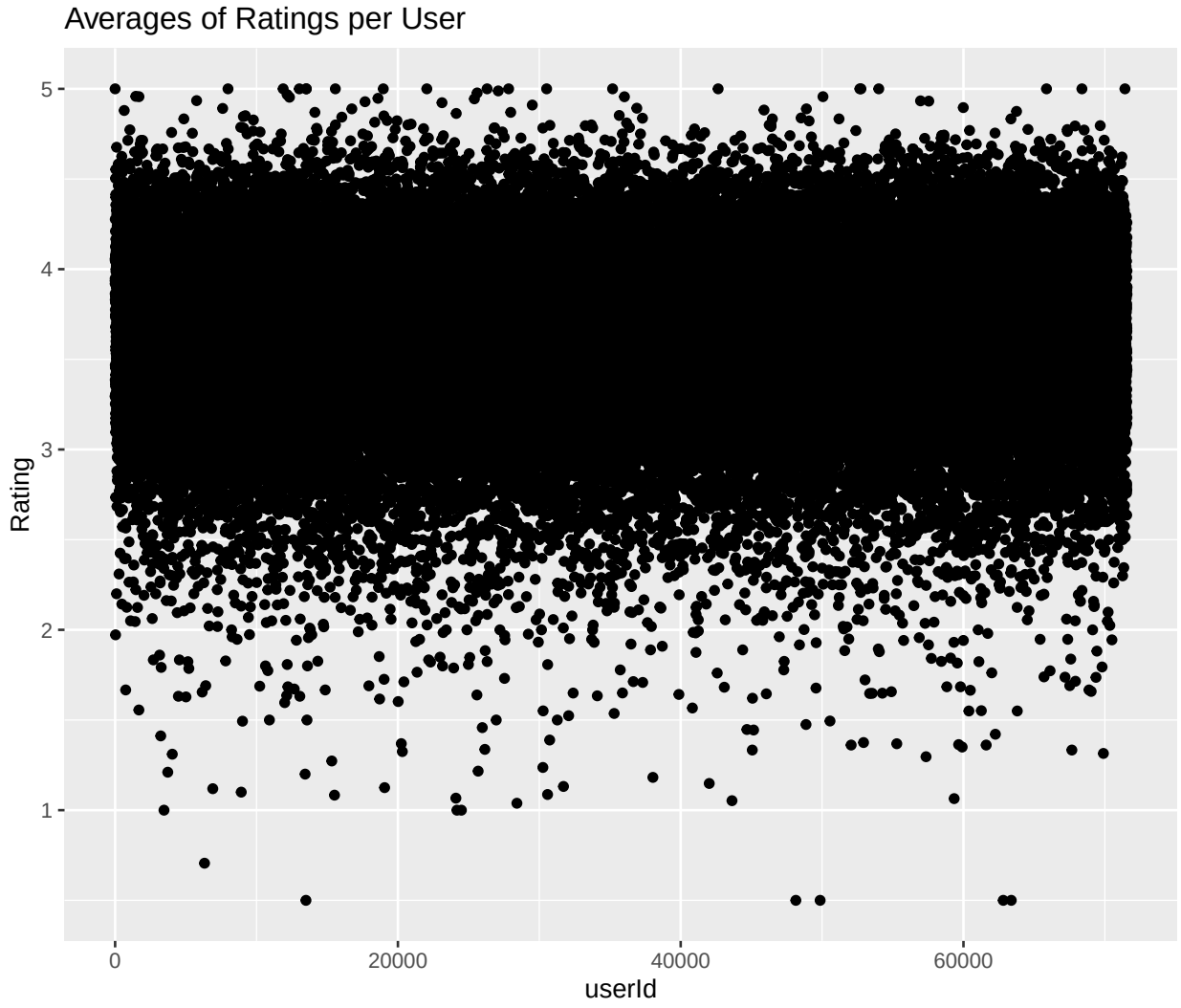
```
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Users' rating characteristics

```
# Users' rating characteristics

            users_rating_char <- edx %>%
              group_by(userId) %>%
              summarize(count = n(), avg = mean(rating), median = median(rating), std = sd(rating)) %>%
              arrange(desc(count))
```

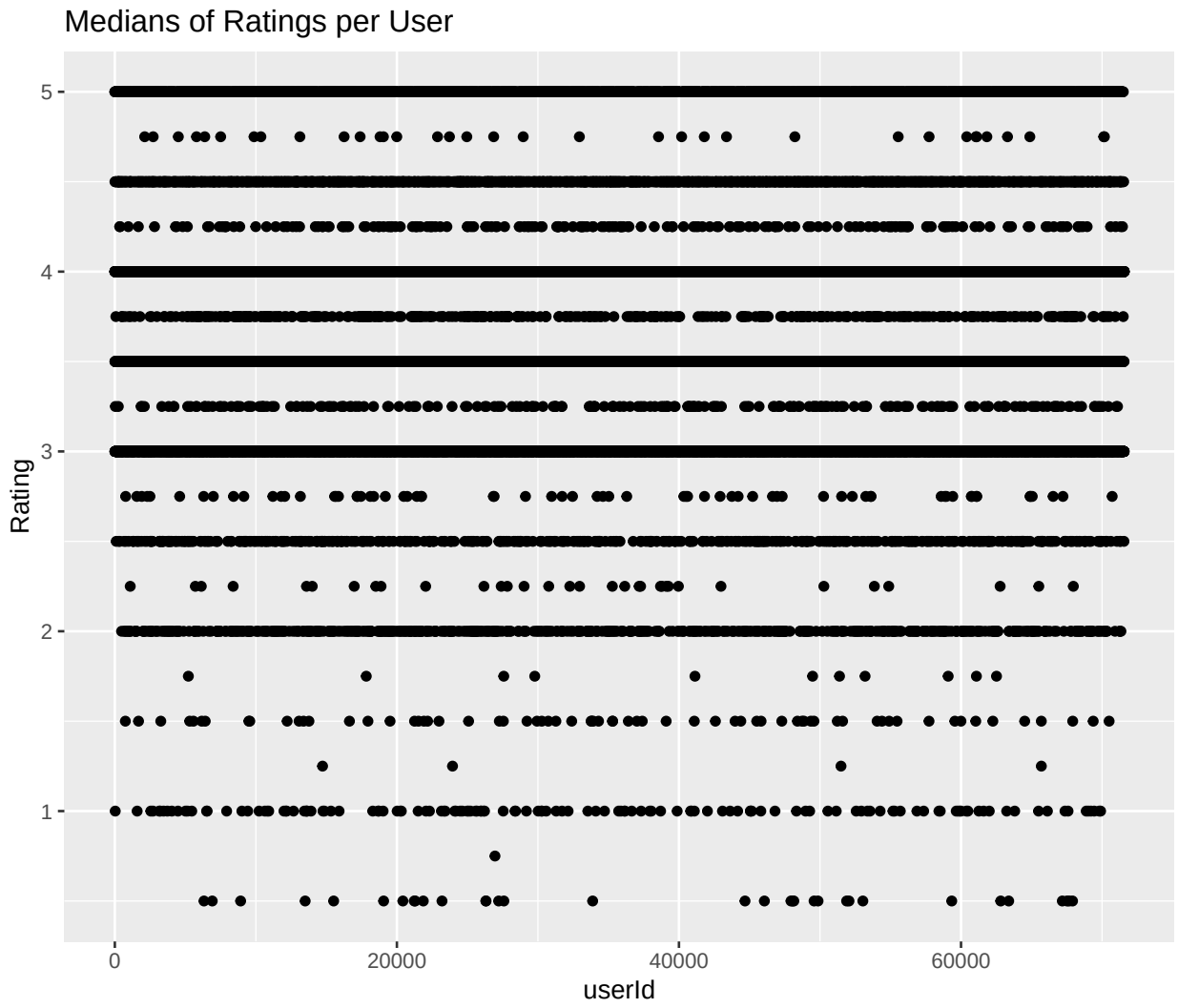
The averages of ratings per user

```
# The averages of ratings per user
            ggplot(users_rating_char) +
              geom_point(aes(x = userId, y = avg)) +
              labs(title = "Averages of Ratings per User", x = "userId", y = "Rating")
```



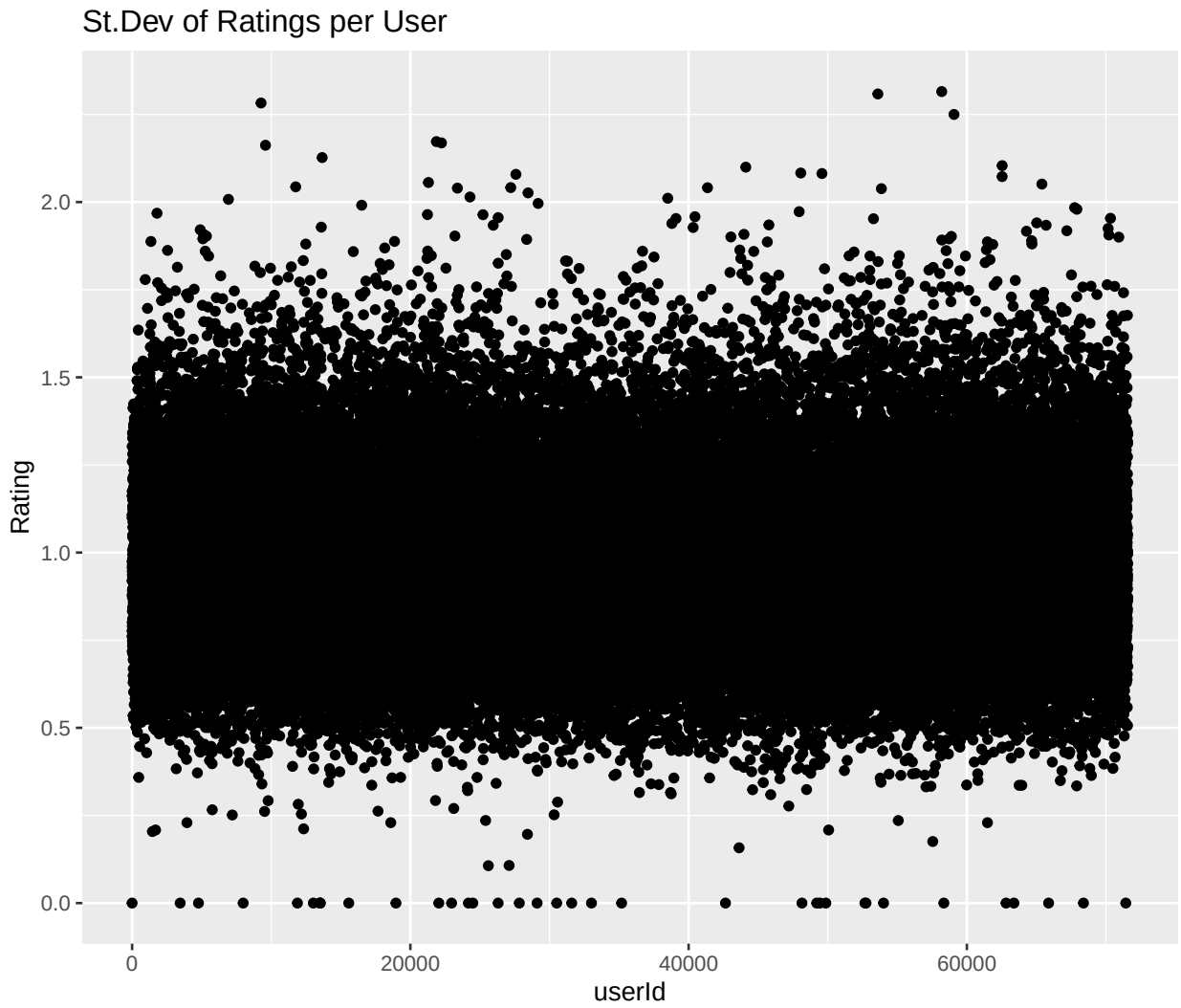
The medians of ratings per user

```
# The medians of ratings per user
            ggplot(users_rating_char) +
              geom_point(aes(x = userId, y = median)) +
              labs(title = "Medians of Ratings per User ", x = "userId", y = "Rating")
```



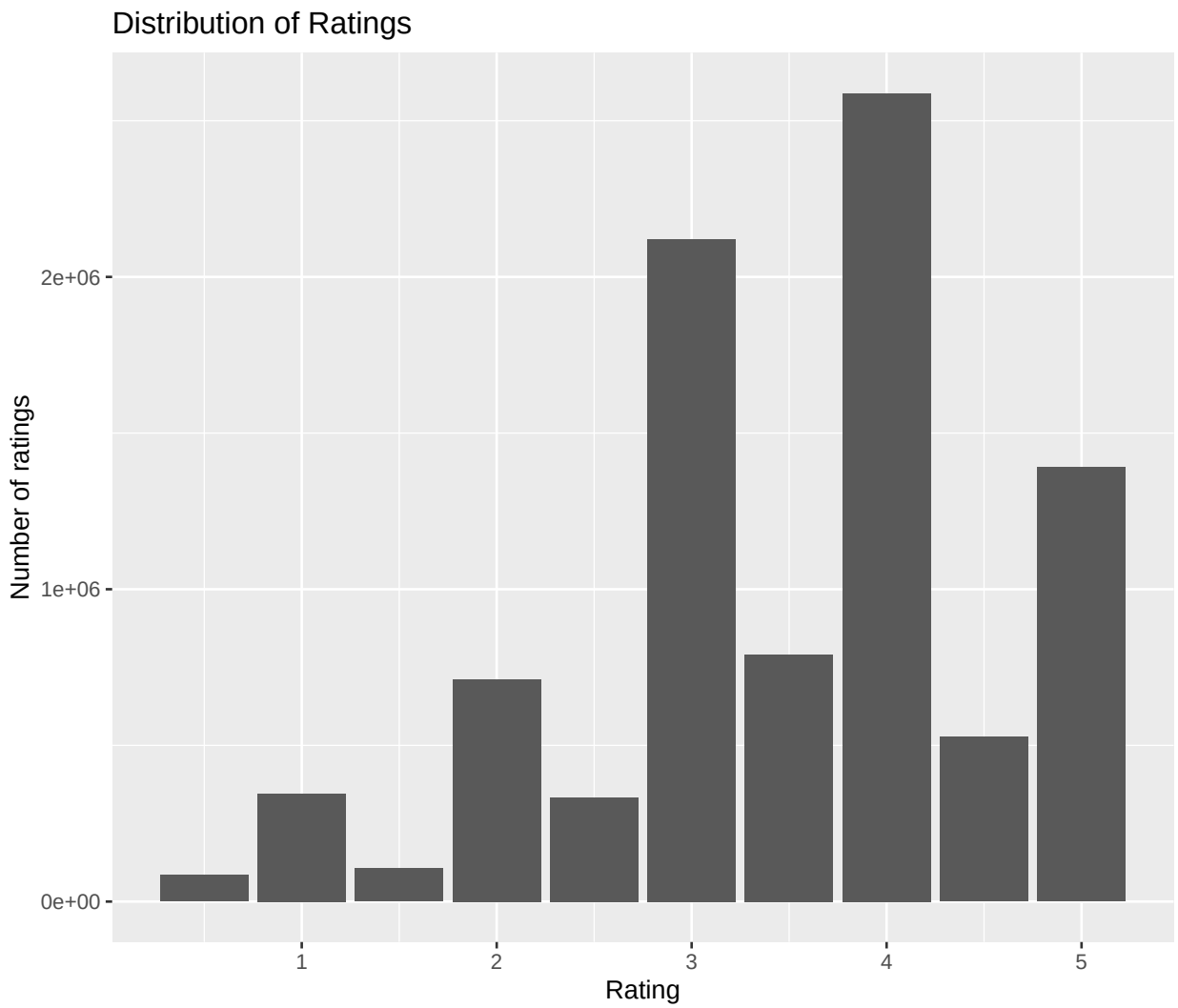
The standard deviations of ratings per user

```
# The standard deviations of ratings per user
            ggplot(users_rating_char) +
              geom_point(aes(x = userId, y = std)) +
              labs(title = "St.Dev of Ratings per User ", x = "userId", y = "Rating")
```



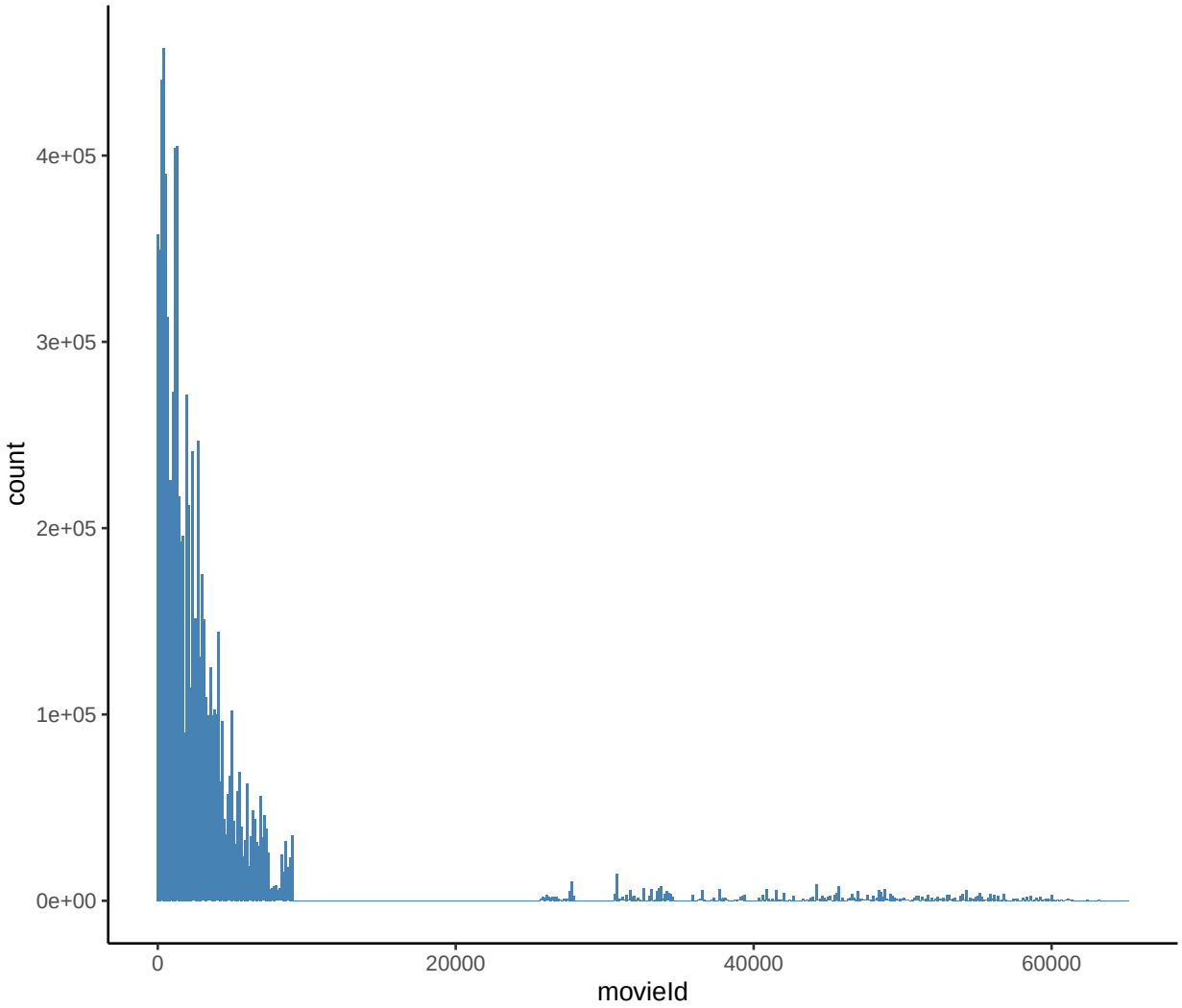
Distribution of ratings

```
# Distribution of ratings
            ggplot(data = edx, aes(x = rating)) +
              geom_bar() +
              labs(title = "Distribution of Ratings", x = "Rating", y = "Number of ratings")
```



Numbers of Ratings per Movie

```
# Numbers of Ratings per Movie

             ggplot(edx, aes(movieId)) +
             theme_classic()  +
             geom_histogram(bins=500, fill = "steel blue")
```



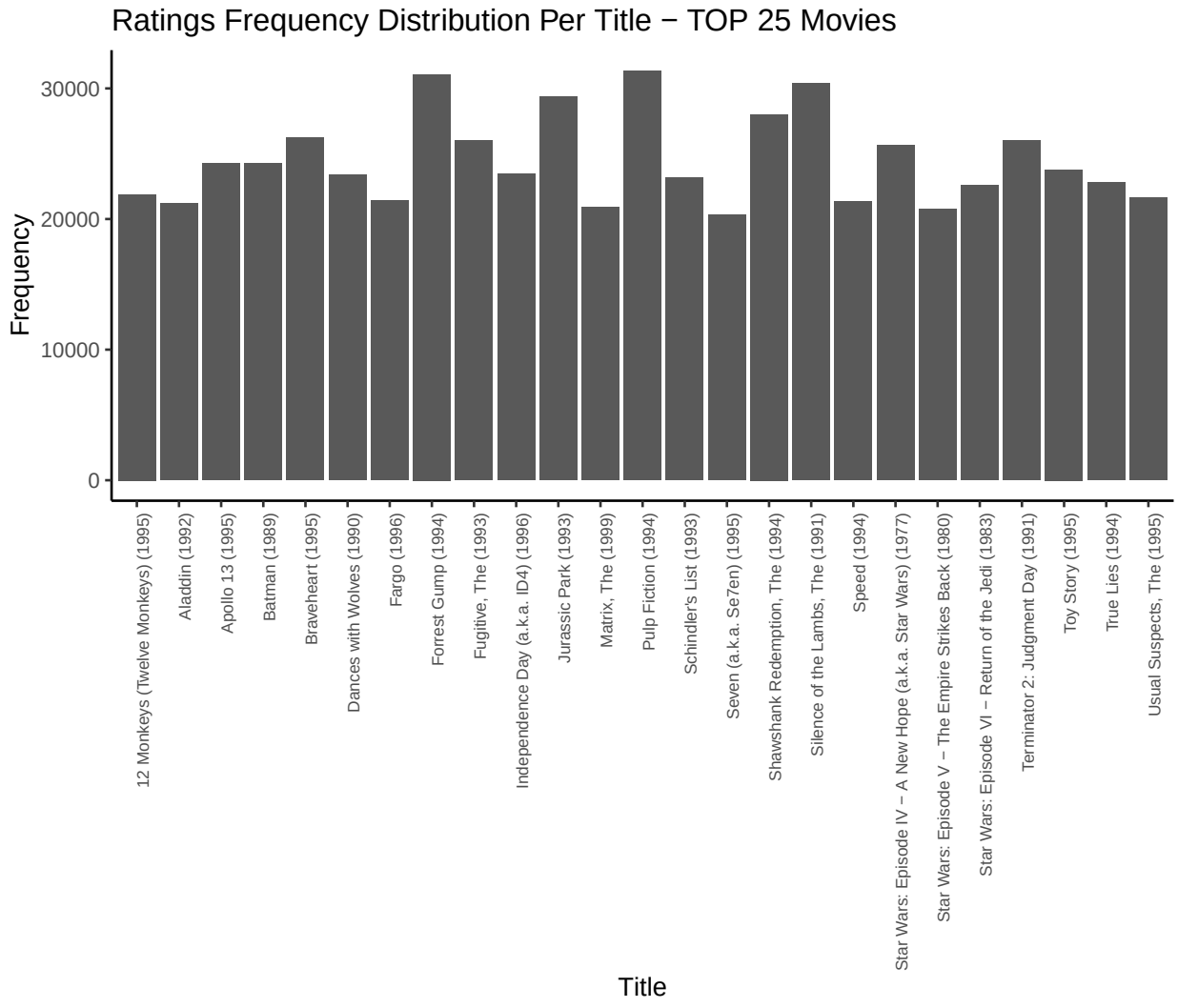

```
                        labs(title = "Ratings Frequency Distribution Per Title (MovieID)",
                  x = "Title (MovieID)",
                  y = "Frequency")
```

```
## $x
## [1] "Title (MovieID)"
##
## $y
## [1] "Frequency"
##
## $title
## [1] "Ratings Frequency Distribution Per Title (MovieID)"
##
## attr(,"class")
## [1] "labels"
```

Top Rated Movies

```
# Top Rated Movies

            edx %>%
```

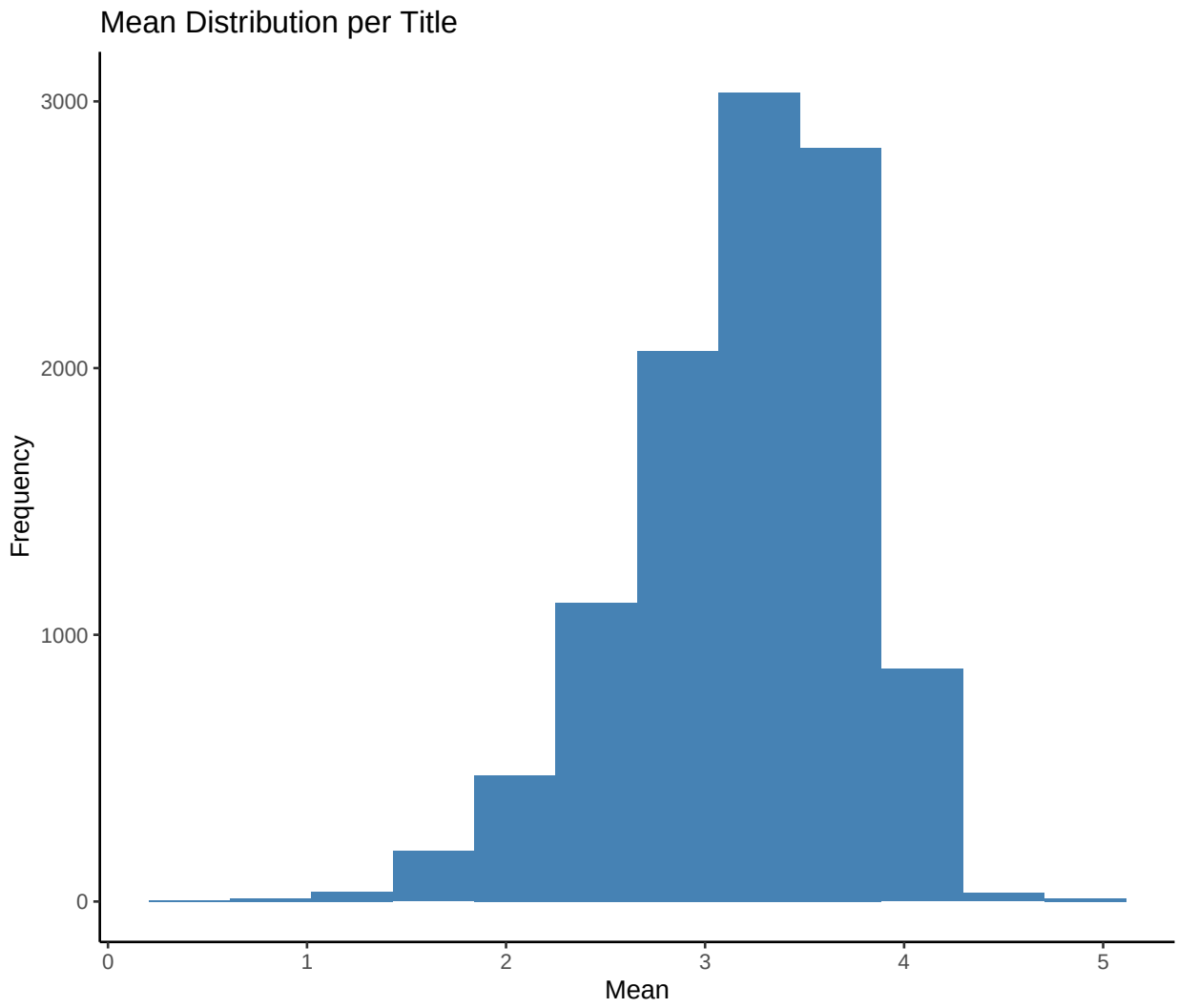
```
            group_by(title) %>%
            summarise(count = n()) %>%
            arrange(desc(count)) %>%
            head(n=25) %>%
            ggplot(aes(title, count)) +
            theme_classic()  +
            geom_col() +
            theme(axis.text.x = element_text(angle = 90, hjust = 1,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



Mean Distribution per Title (Movie ID)

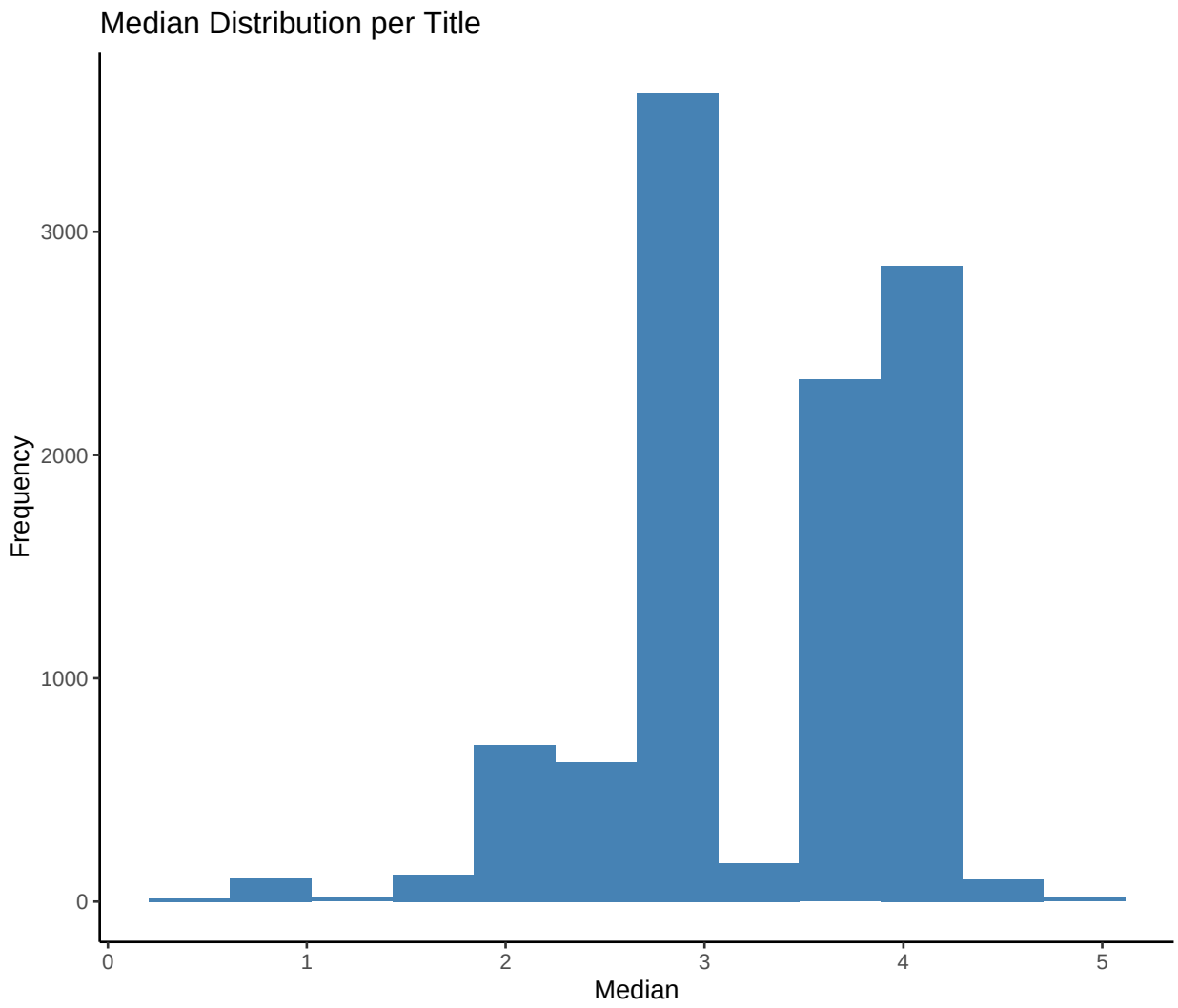
```
# Mean Distribution per Title (Movie ID)

          edx %>%
             group_by(title) %>%
             summarise(mean = mean(rating)) %>%
             ggplot(aes(mean)) +
```

```
  theme_classic()  +
  geom_histogram(bins=12, fill = "steel blue") +
  labs(title = "Mean Distribution per Title",
       x = "Mean",
       y = "Frequency")
```



Median Distribution per Title (Movie ID)

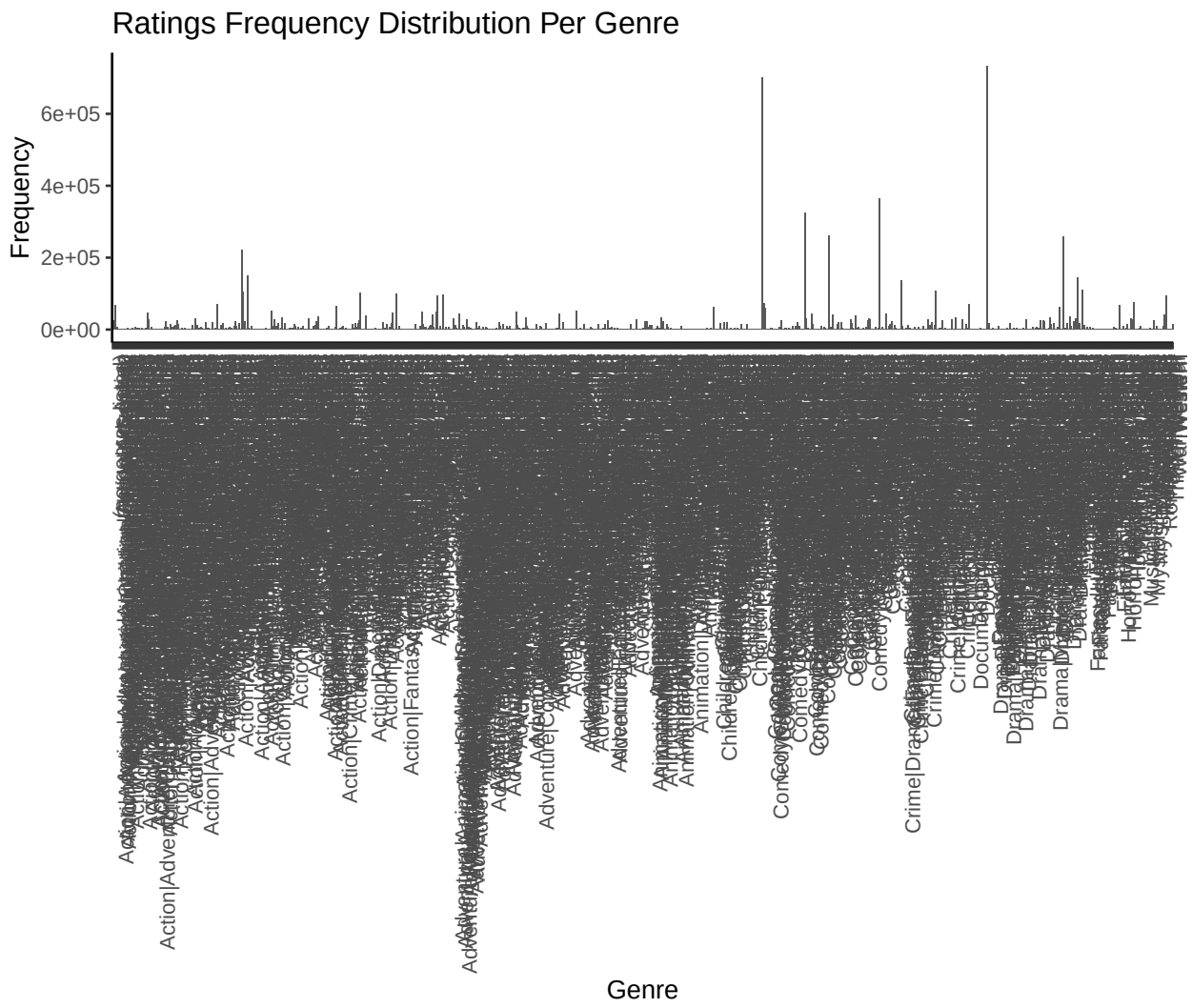
```
# Median Distribution per Title (Movie ID)

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



Genre Analysis

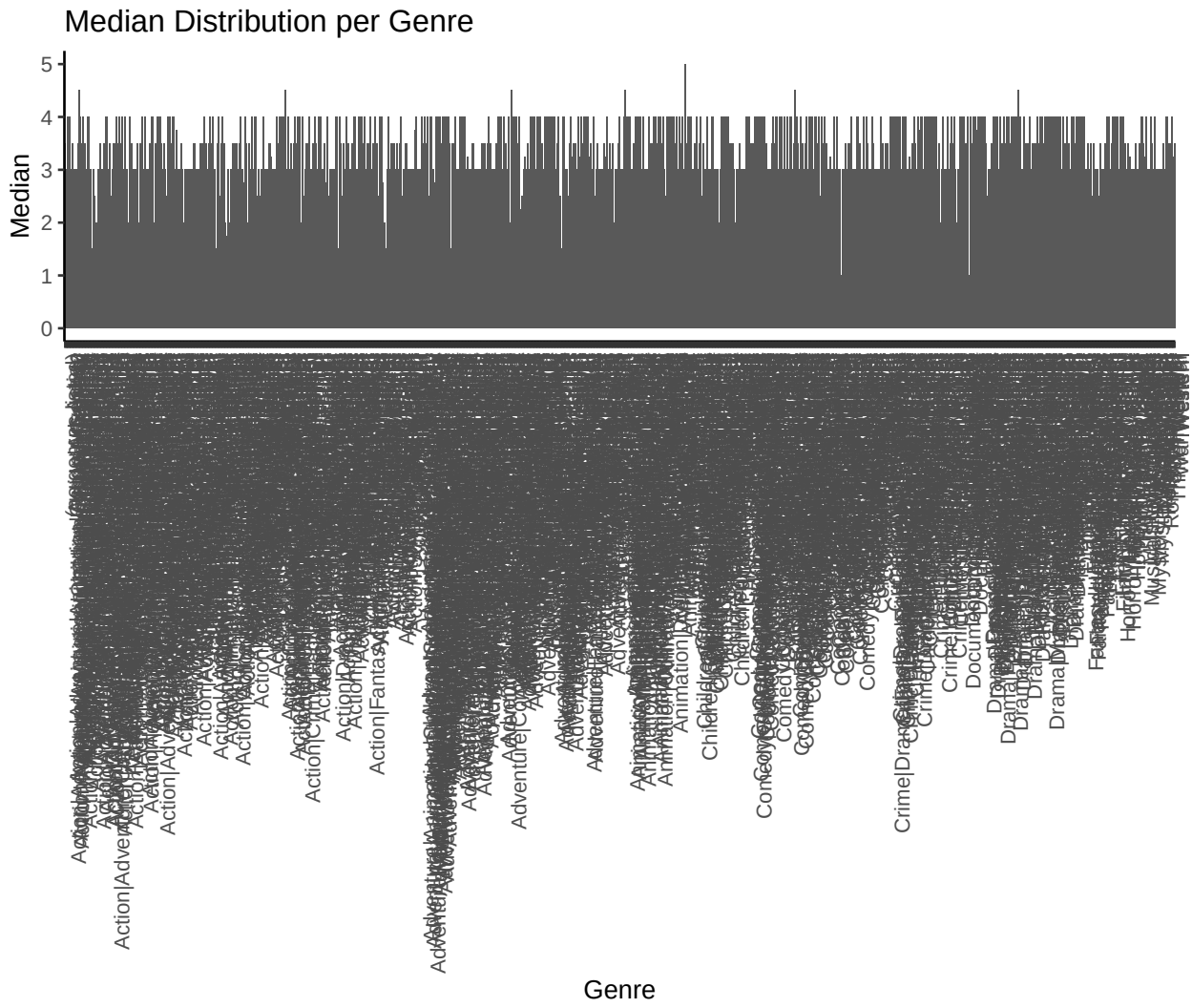
Rating Distribution per Genre

```
# Genre Analysis #

# Rating Distribution per Genre

            edx %>%
              group_by(genres) %>%
              summarise(count = n()) %>%
              ggplot(aes(genres, count)) +
              theme_classic()  +
              geom_col() +
              theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
              labs(title = "Ratings Frequency Distribution Per Genre",
                   x = "Genre",
                   y = "Frequency")
```



Median Distribution per Genre

```
# Median Distribution per Genre

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



3 Modeling of the recommendation system

As we mentioned above, with the following formula, we can compute the RMSE:

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

with N being the number of user/movie combinations and the sum occurring over all these combinations. The RMSE is our measure of model accuracy. We can interpret the RMSE similarly to a standard deviation: it is the typical error we make when predicting a movie rating. If its result is larger than 1, it means that our typical error is larger than one star, which is not a good result. The first basic model predicts the same rating for all movies, so we compute the dataset's mean rating.

3.1 Naive Model

The first and simplest model we can build, is the so-called Naive Model. It predicts the same rating for all movies regardless of user who give it. A model based approach assumes the same rating for all movie with all differences explained by random variation :

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

with $\epsilon_{u,i}$ independent error sample from the same distribution centered at 0 and μ the "true" rating for all movies.

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

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

Compute the dataset's mean rating

```
# Compute the dataset's mean rating
            mu <- mean(edx$rating)
            mu
```

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

Test results based on simple prediction

```
# Test results based on simple prediction
            naive_rmse <- RMSE(validation$rating, mu)
            naive_rmse
```

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

Save prediction in data frame

```
# Save prediction in data frame
            results <- data_frame(method = "Average movie rating model", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

```
            results
```

```
## # A tibble: 1 x 2
##   method                     RMSE
##   <chr>                     <dbl>
## 1 Average movie rating model  1.06
```

The RMSE on the validation dataset is 1.06120181. This number is larger than 1, and as we stressed above this is not a good result. So we have to improve our model.

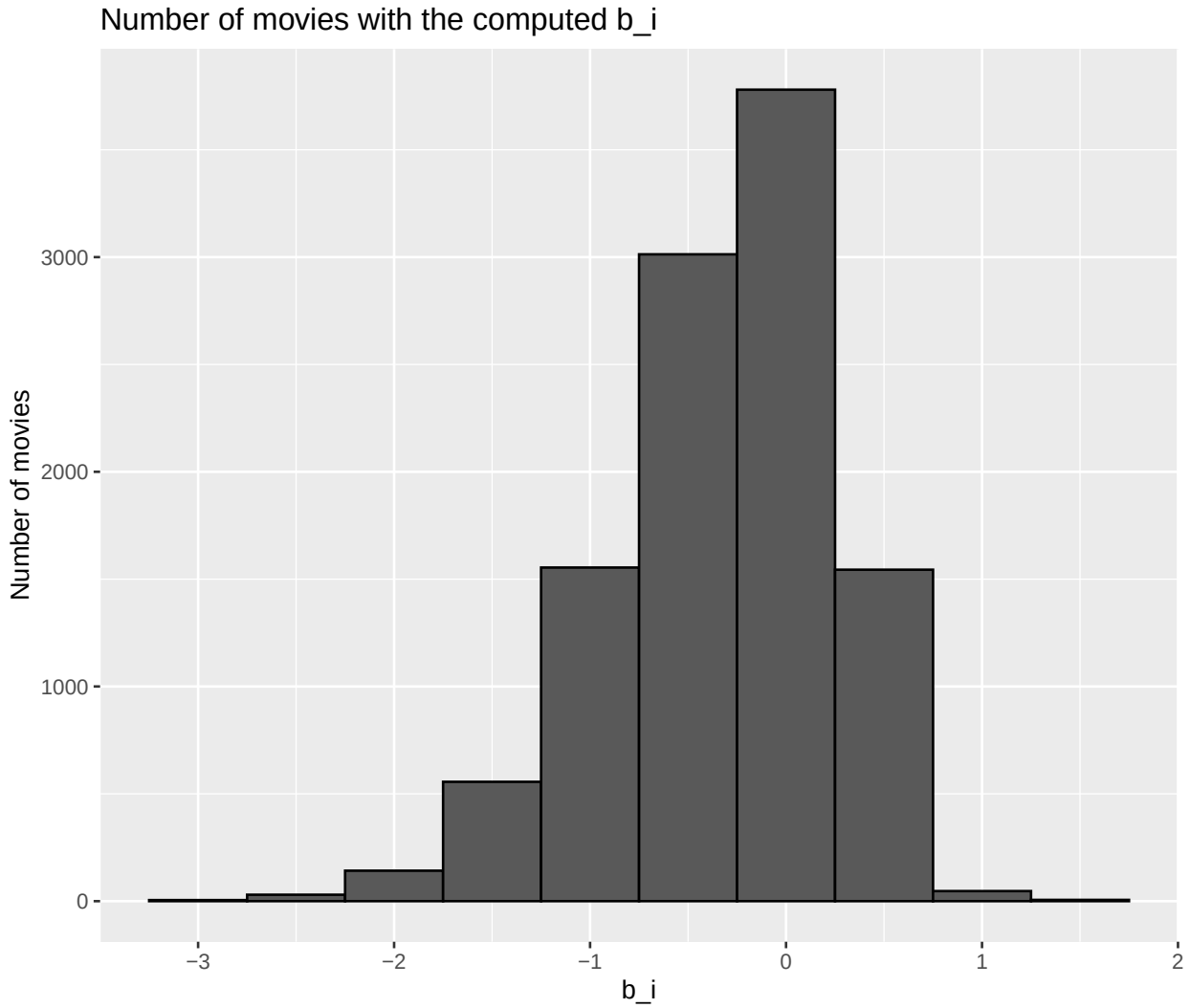
3.2 Movie effect model

Simple model taking into account the movie effect b_i. The improved formula is

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

Simple model taking into account the movie effect b_i

```
# Simple model taking into account the movie effect b_i
# Subtract the rating minus the mean for each rating the movie received
# Plot number of movies with the computed b_i
            movie_avgs <- edx %>%
              group_by(movieId) %>%
              summarize(b_i = mean(rating - mu))
            movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
                                 ylab = "Number of movies", main = "Number of movies with the computed
```



Test and save rmse results

```
predicted_ratings <- mu +  validation %>%
              left_join(movie_avgs, by='movieId') %>%
              pull(b_i)
            model_1_rmse <- RMSE(predicted_ratings, validation$rating)
            results <- bind_rows(results,
                                 data_frame(method="Movie effect model",
                                            RMSE = model_1_rmse ))
            results
```

```
## # A tibble: 2 x 2
##   method                     RMSE
##   <chr>                     <dbl>
## 1 Average movie rating model 1.06
## 2 Movie effect model         0.944
```

The RMSE on the validation dataset for the Movie-based Model is 0.9439. It is a better result than that of Naive Model, but not a good result, yet.

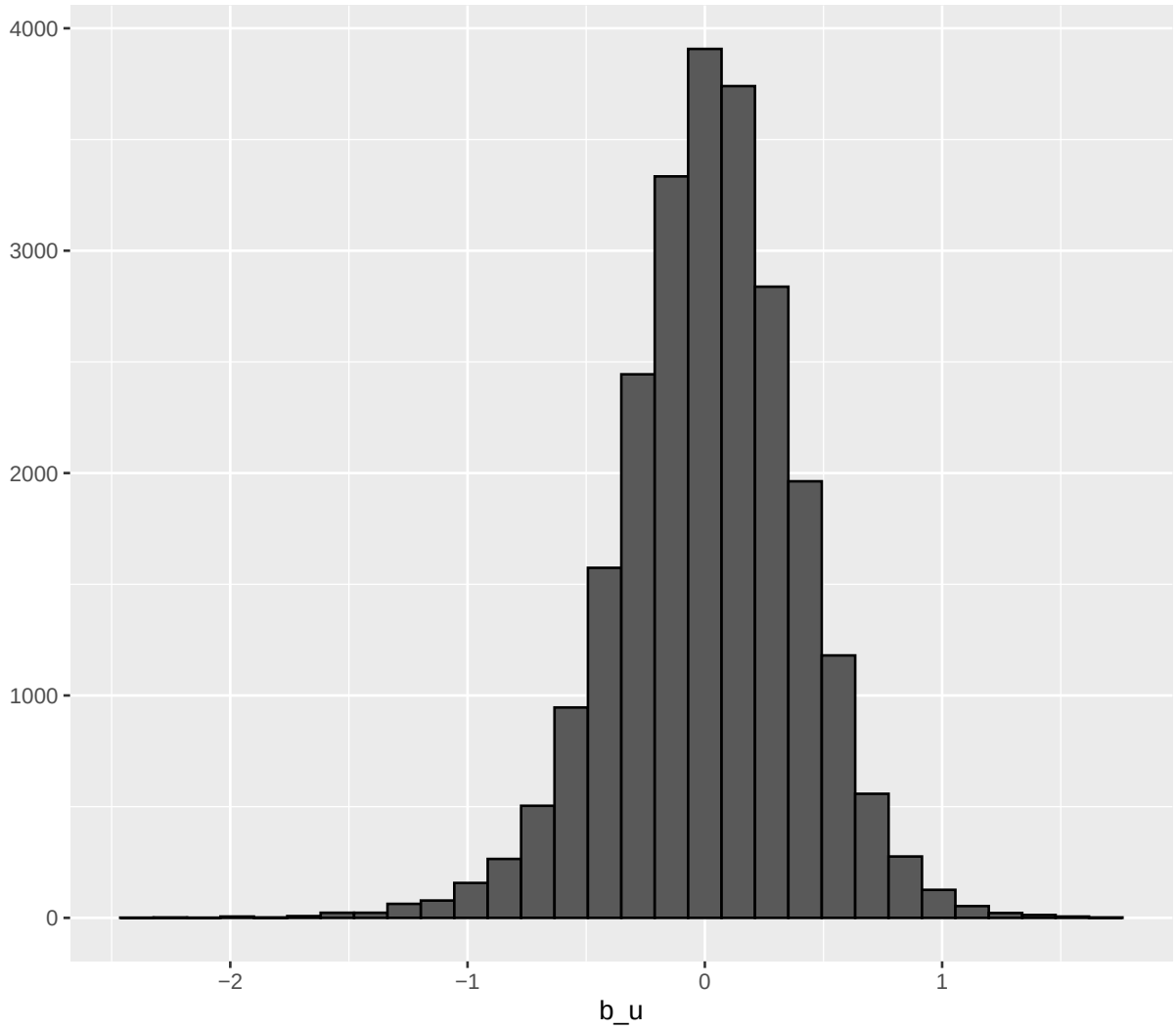
3.3 Movie and user effect model.

We now take into account the user effect, as well, using the formula:

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

Plot penaly term user effect

```
# Plot penaly term user effect
            user_avgs<- edx %>%
              left_join(movie_avgs, by='movieId') %>%
              group_by(userId) %>%
              filter(n() >= 100) %>%
              summarize(b_u = mean(rating - mu - b_i))
            user_avgs%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```




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

Test and save rmse results

```
# Test and save rmse results
            predicted_ratings <- validation%>%
              left_join(movie_avgs, by='movieId') %>%
              left_join(user_avgs, by='userId') %>%
              mutate(pred = mu + b_i + b_u) %>%
              pull(pred)

            model_2_rmse <- RMSE(predicted_ratings, validation$rating)
            results <- bind_rows(results,
                                 data_frame(method="Movie and user effect model",
                                            RMSE = model_2_rmse))
            results
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Average movie rating model  1.06
## 2 Movie effect model          0.944
## 3 Movie and user effect model 0.865
```

The RMSE on the validation dataset is 0.8653488 and this is a better result. This means that the combination of the movie effect and the user effect gives us a good prediction.

3.4 Movie, User and Genre-based Model

The formula we use in this model taking into account the genre effect is:

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

Calculate the average by movie

```
# Calculate the average by movie

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

Calculate the average by user

```
# Calculate the average by user

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

Calculate the average by genre

```
# Calculate the average by genre

            genre_avgs <- edx %>%
              left_join(movie_avgs, by="movieId") %>%
              left_join(user_avgs, by="userId") %>%
              group_by(genres) %>%
              summarize(b_u_g = mean(rating - mu - b_i - b_u))
```

Compute the predicted ratings for the movie effect, the user effect and the genre effect on the validation dataset

```
            # Compute the predicted ratings for the movie effect, the user effect and the genre effect
            predicted_ratings <- validation %>%
              left_join(movie_avgs, by="movieId") %>%
              left_join(user_avgs, by="userId") %>%
              left_join(genre_avgs, by="genres") %>%
              mutate(pred = mu + b_i + b_u + b_u_g) %>%
              pull(pred)
```

Predict the RMSE for the movie effect, the user effect and the genre effect on the validation dataset

```
# Predict the RMSE for the movie effect, the user effect and the genre effect on the validation dataset
            model_3_rmse <- RMSE(predicted_ratings, validation$rating)
            # Adding the result to the results dataset
            results <- bind_rows(results,
                                 data_frame(method="Movie +User + Genre  model",
                                            RMSE = model_3_rmse))
            results
```

```
## # A tibble: 4 x 2
##   method                       RMSE
##   <chr>                       <dbl>
## 1 Average movie rating model  1.06
## 2 Movie effect model          0.944
## 3 Movie and user effect model 0.865
## 4 Movie +User + Genre  model  0.865
```

The RMSE on the validation dataset is now 0.86494688 and this is still a good result.But the effect of the genres of the movies on the performance of the model is small, compared with the second model. We can improve this by using regularization.

3.5 Regularization

We introduce the concept of regularization, that permits to penalize large estimates that come from small sample sizes. The general idea is to add a penalty for large values of b_i to the sum of squares equation that we minimize. So having many large b_i, make it harder to minimize. Regularization is a method used to reduce the effect of overfitting.

3.5.1 Regularized Movie and User-based Model

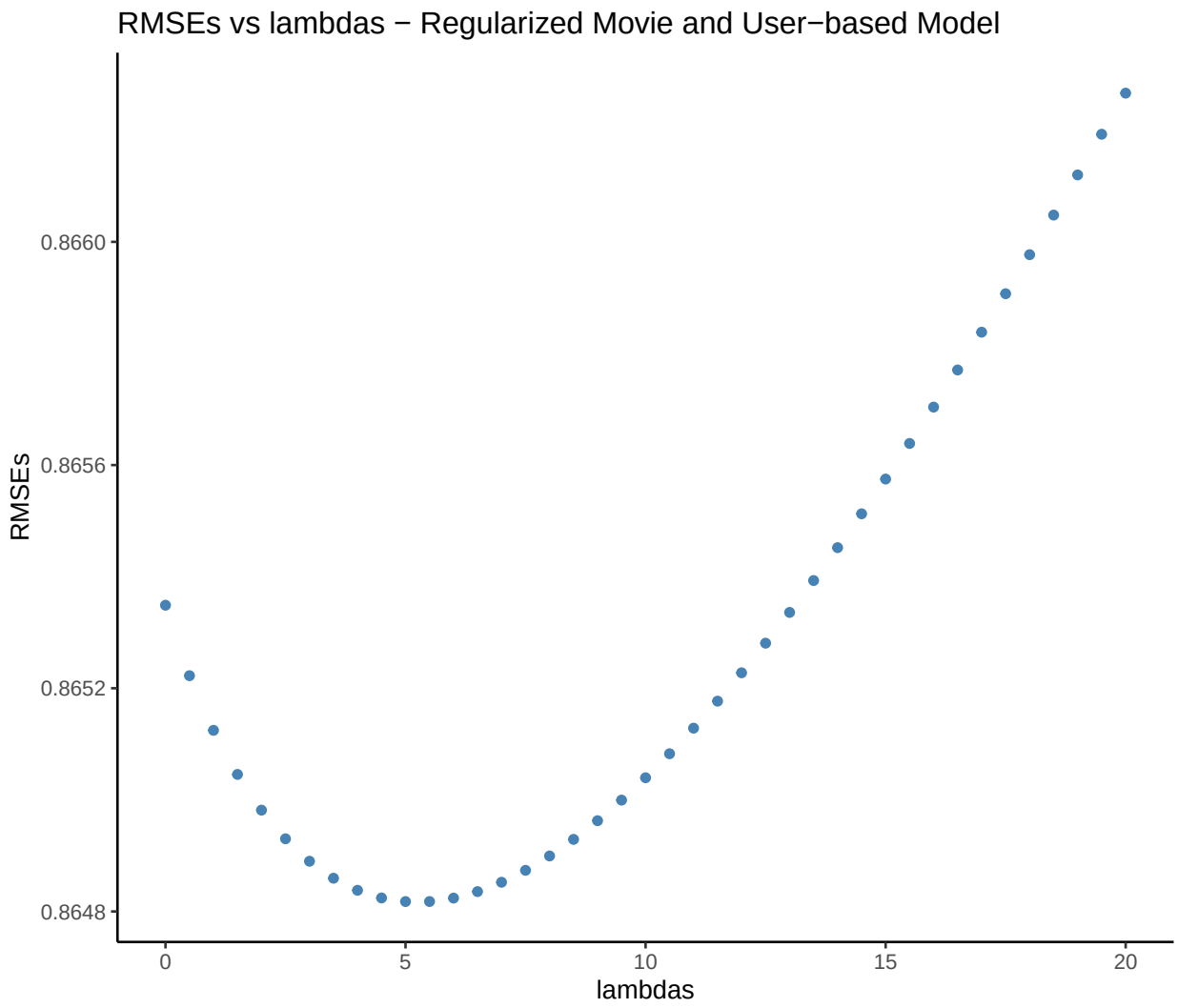
This regularized model takes into account the movie effect and the effect of the users.

```
 # Regularized movie and user effect model #
            lambdas <- seq(0, 20, 0.5)


            # For each lambda,find b_i & b_u, followed by rating prediction & testing
            # 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) %>%
    pull(pred)

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

Plot the lambdas vs the RMSEs

```
# Make a plot of the lambdas vs the RMSEs

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

ggplot(rmse_lambda_movie_user, aes(lambdas, rmses)) +

  theme_classic() +

  geom_point(color="steel blue") +

  labs(title = "RMSEs vs lambdas - Regularized Movie and User-based Model",

       y = "RMSEs",

       x = "lambdas")
```



The optimal lambda is:

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

```
## [1] 5
```

Test and save results

```
# Test and save results
            results <- bind_rows(results,
                                 data_frame(method="Regularized movie and user effect model",
                                            RMSE = min(rmses)))
            results
```

```
## # A tibble: 5 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Average movie rating model           1.06
## 2 Movie effect model                   0.944
## 3 Movie and user effect model          0.865
```

```
## 4 Movie +User + Genre  model              0.865
## 5 Regularized movie and user effect model 0.865
```

3.5.2 Regularized Movie+User+Genre Model

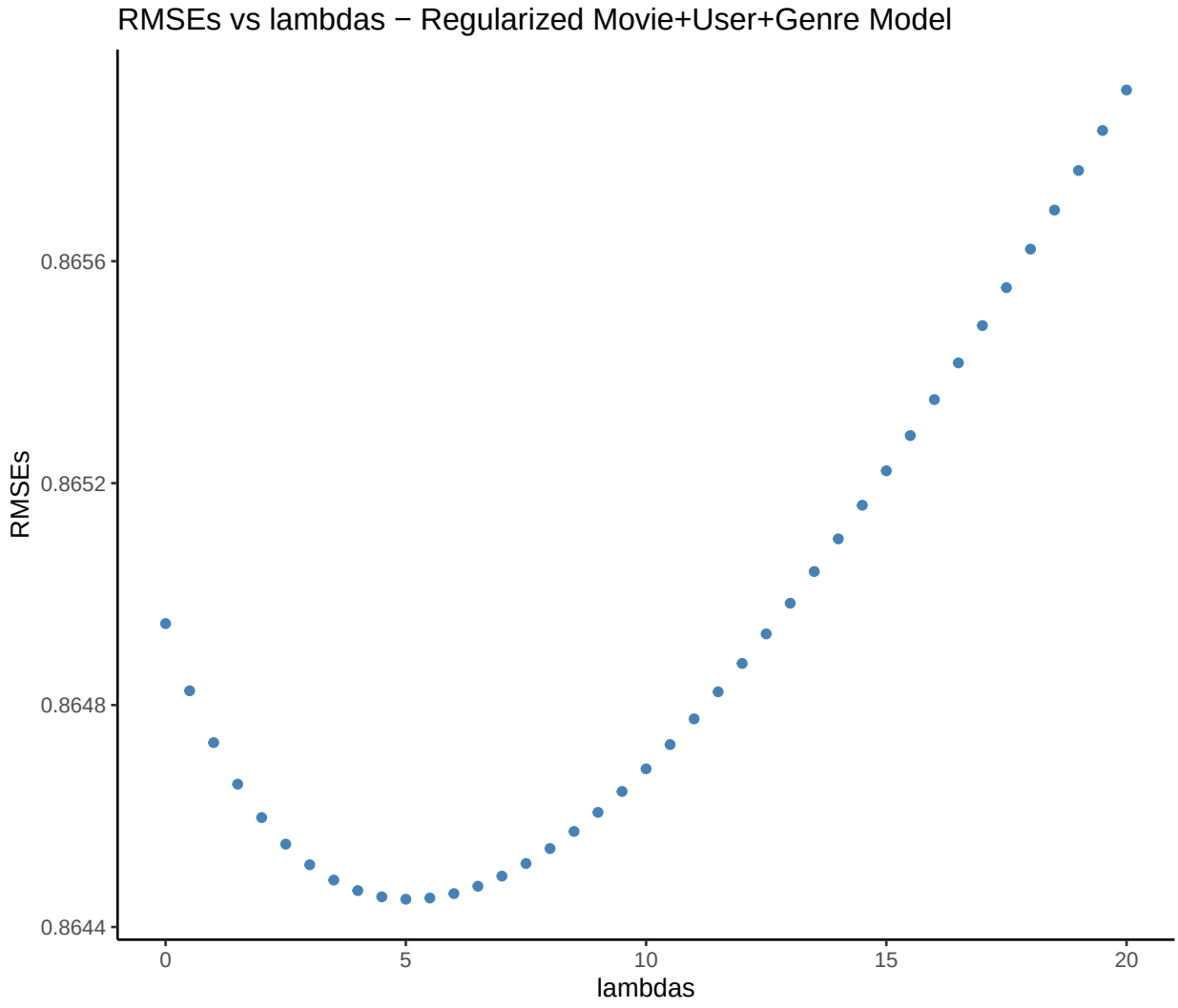
```
# Calculate the average of all movies

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

```
            # Define a table of lambdas

            lambdas <- seq(0, 20, 0.5)

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

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

              # Calculate the average by movie

              b_i <- edx %>%
                group_by(movieId) %>%
                summarize(b_i = sum(rating - mu) / (n() + l))
              # Calculate the average by user

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

              # Calculate the average by genre
              b_u_g <- edx %>%
                left_join(b_i, by="movieId") %>%
                left_join(b_u, by="userId") %>%
                group_by(genres) %>%
                summarize(b_u_g = sum(rating - b_i - mu - b_u) / (n() + l))

              # Calculate the predicted ratings on validation dataset
              predicted_ratings <- validation %>%
                left_join(b_i, by="movieId") %>%
                left_join(b_u, by="userId") %>%
                left_join(b_u_g, by="genres") %>%
                mutate(pred = mu + b_i + b_u + b_u_g) %>%
                pull(pred)

              # Predict the RMSE on the validation set

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

plot the result of lambdas

```
            # Make a plot of the lambdas vs the RMSEs

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

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



Get the lambda value that minimize the RMSE

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

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

```
## [1] 5
```

Predict the RMSE on the validation dataset

```
# Predict the RMSE on the validation set

            rmse_regularized_movie_user_genre_model <- min(rmses)
```

Test and save results

```
# Test and save results

            results <- bind_rows(results,
                                 data_frame(method=" Regularized Movie+User+Genre Based Model",
                                            RMSE = min(rmses)))
            results
```

```
## # A tibble: 6 x 2
##   method                                      RMSE
##   <chr>                                      <dbl>
## 1 "Average movie rating model"               1.06
## 2 "Movie effect model"                       0.944
## 3 "Movie and user effect model"              0.865
## 4 "Movie +User + Genre  model"               0.865
## 5 "Regularized movie and user effect model"  0.865
## 6 " Regularized Movie+User+Genre Based Model" 0.864
```

4 Results

In the above table are presented the results of our models which are trained on the edx dataset and validated on the validation dataset.

5 Conclusion

We have built a machine learning algorithm to predict movie ratings with MovieLens dataset. The regularized model including the effect of user is characterized by the lower RMSE value and it is, so, the optimal model to use for this project. The genre effect only makes a very small improvement. With regularization the models improve even more reaching the value 0.864 for RMSE.

6 References

1. Rafael A. Irizarry, "Introduction to Data Science Data Analysis and Prediction Algorithms with R", 2019.
2. https://en.wikipedia.org/wiki/MovieLens
3. https://grouplens.org/datasets/movielens/100k/
4. https://en.wikipedia.org/wiki/Recommender_system
5. https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
6. F.O. Isinkaye ,Y.O. Folajimi , B.A. Ojokoh, Recommendation systems: Principles, methods and Evaluation, Egyptian Informatics Journal, (2015) 16, 261-273.