MovieLens Rating Prediction Project

Introduction

On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem to many Internet users. Recommender systems solve this problem by searching through large volume of dynamically generated information to provide users with personalized content and services.

The aim of this project is to build a movie recommendation system to predict movies ratings by different users. Our target is to reach RMSE value of less than or equals 0.87750.

Steps followed to reach raget: 1- Create data set 2- Data Exploration 3- Test 3 different approaches

Dataset

The MovieLens dataset we are using is provided by GroupLens, a research lab in the Department of Computer Science and Engineering at the University of Minnesota.

GroupLens has collected and made available rating datasets from their website (https://grouplens.org/datasets/movielens/).

This data set contains 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users

The dataset contains following variables for each rating:

userId, movieId, rating, timestamp, title, genres

rating is from 0 to 5 with step = 0.5

genres are separated by |

Training and validation sets are generated using following code

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

# Note: this process could take a couple of minutes

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

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

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

```
                                           title = as.character(title),
                                           genres = as.character(genres))

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

# Validation set will be 10% of MovieLens data

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)
```

edx set will be used to create the model and validation set will be used to test the model and calculate RMSE.

Data Exploration, visualization and modeling

This function is used to calculate RMSE

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

Exploring training set (edx)

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

```
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
##
##
##
```

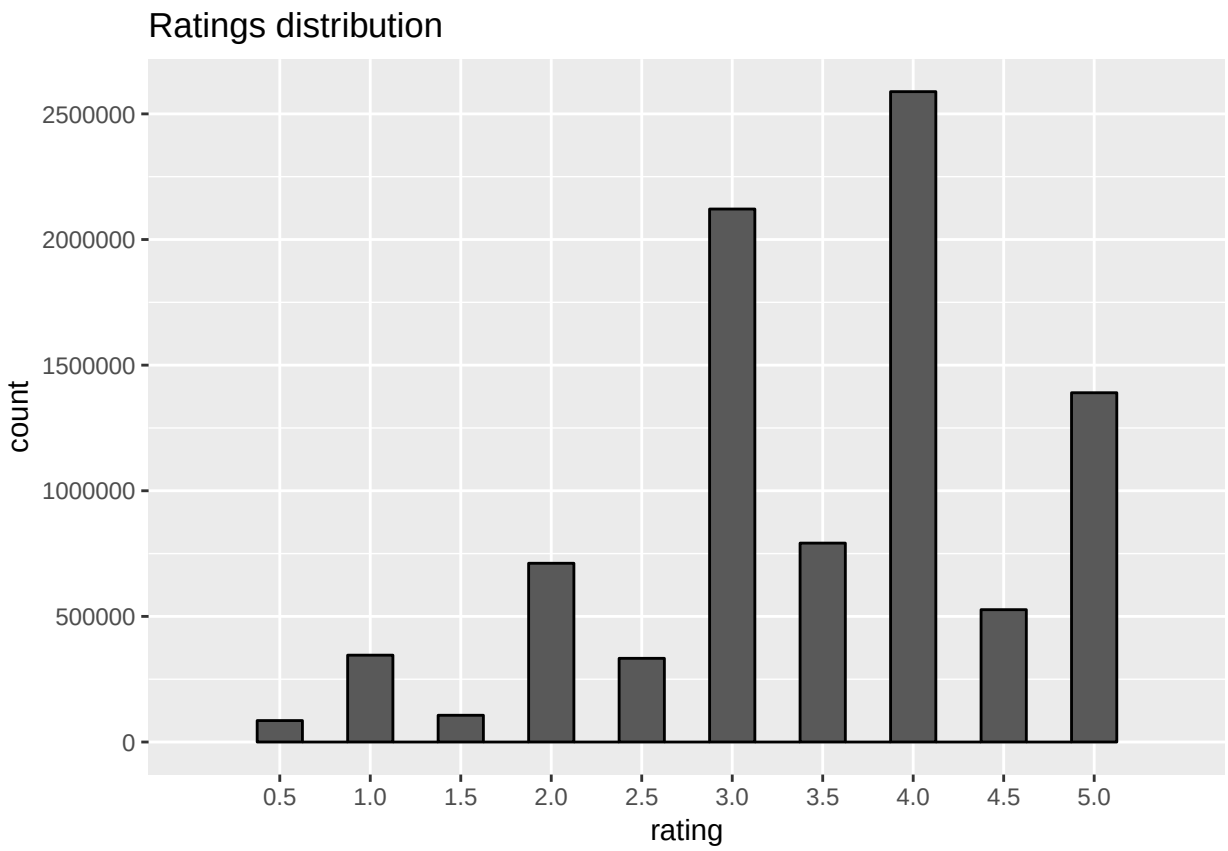
Number of unique movies and unique users in training (edx) dataset

```
edx %>%
  summarize(unique_users = n_distinct(userId),
            unique_movies = n_distinct(movieId))
```

```
##   unique_users unique_movies
## 1        69878         10677
```

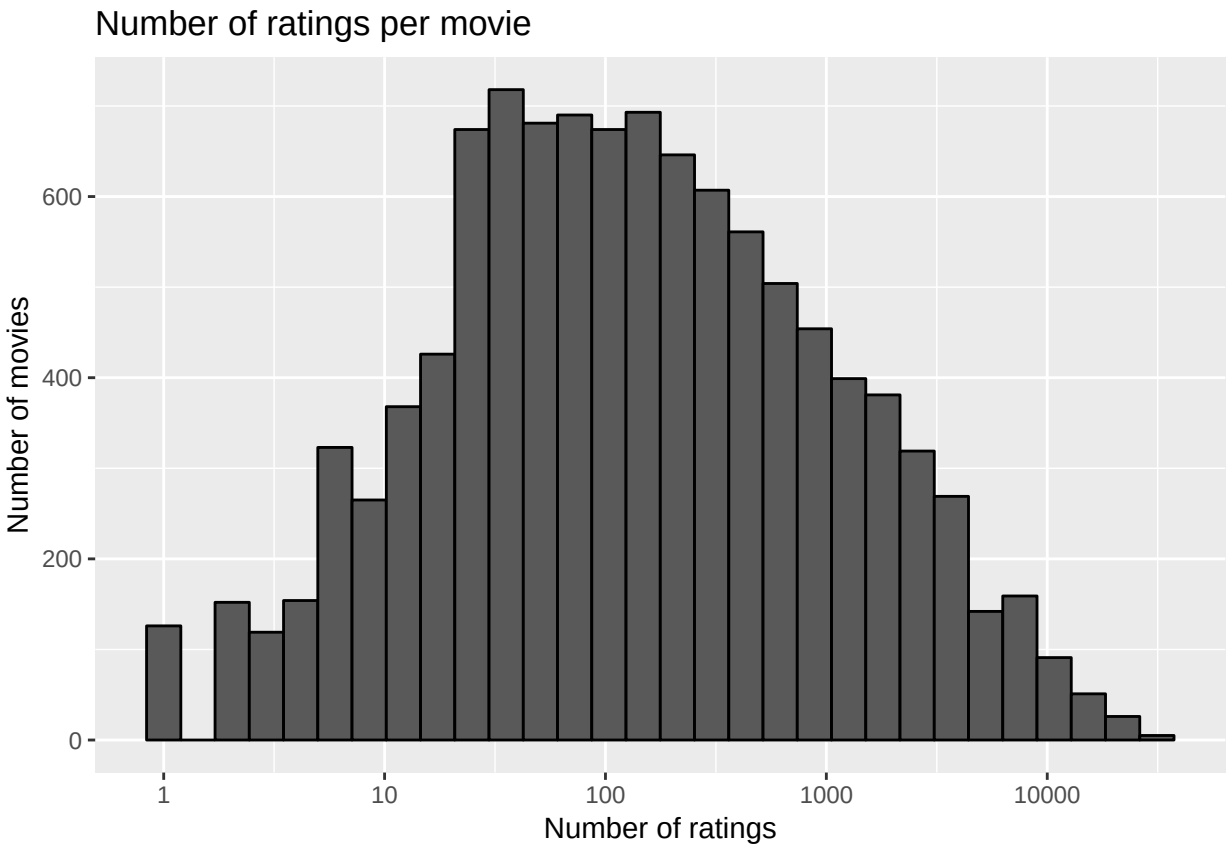
Ratings distribution

```
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "black") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ggtitle("Ratings distribution")
```



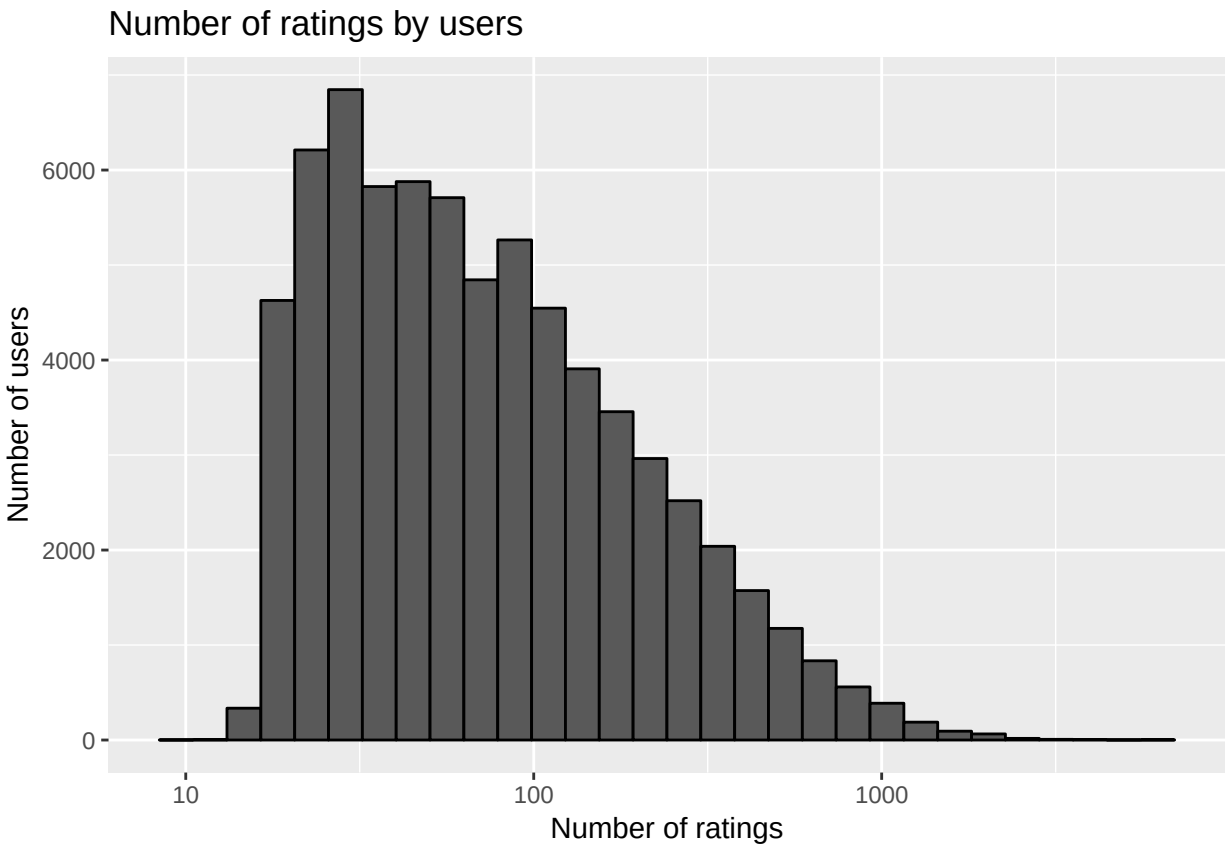
Ratings per movie

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of movies") +
  ggtitle("Number of ratings per movie")
```



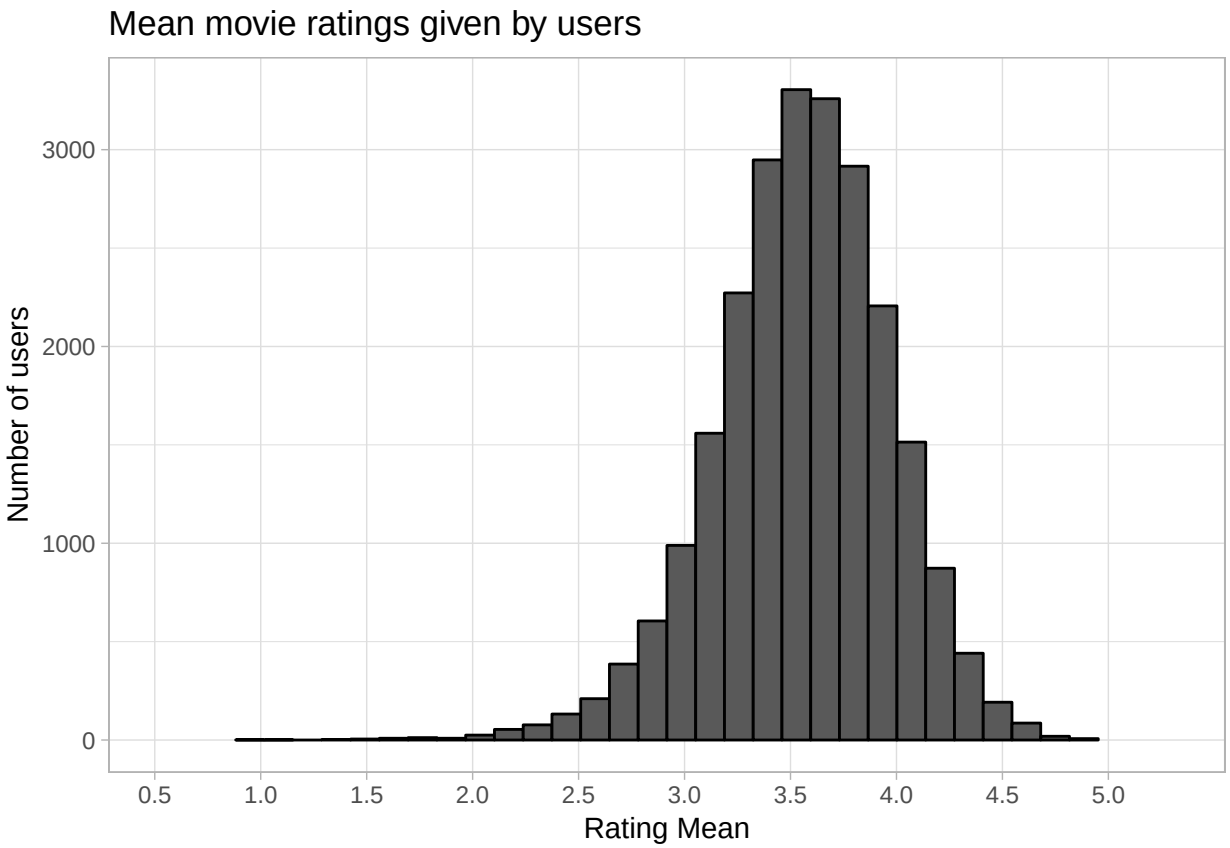
Ratings given by users

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of users") +
  ggtitle("Number of ratings by users")
```



Mean movie ratings given by users

```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Rating Mean") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings given by users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



Modeling

1- Basic average movie rating model

Compute dataset's mean rating

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

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

Calculate basic average movie rating model RMSE

```
basic_prediction_rmse <- RMSE(validation$rating, mu)
cat("Basic average movie rating RMSE = ", basic_prediction_rmse)
```

```
## Basic average movie rating RMSE =  1.061202
```

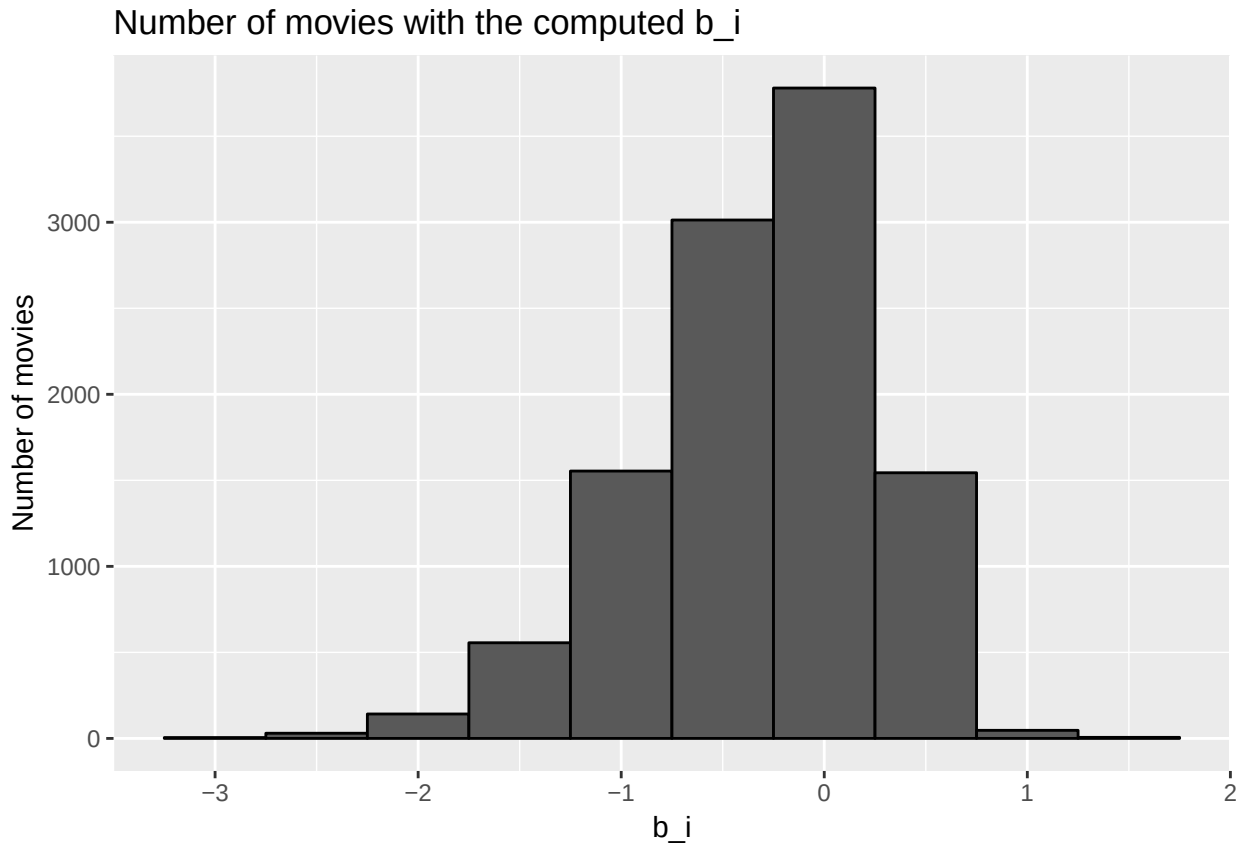
2- Movie effect model

Taking into consideration the movie effect b_i = mean(rating - mu)

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

Plot number of movies of the computed b_i

```
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
                     ylab = "Number of movies", main = "Number of movies with the computed b_i")
```



Predict ratings considering movies effect

```
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
```

Calculate movie effect model RMSE

```
movie_model_rmse <- RMSE(predicted_ratings, validation$rating)
cat("Movie effect model RMSE = ", movie_model_rmse)
```

```
## Movie effect model RMSE =  0.9439087
```

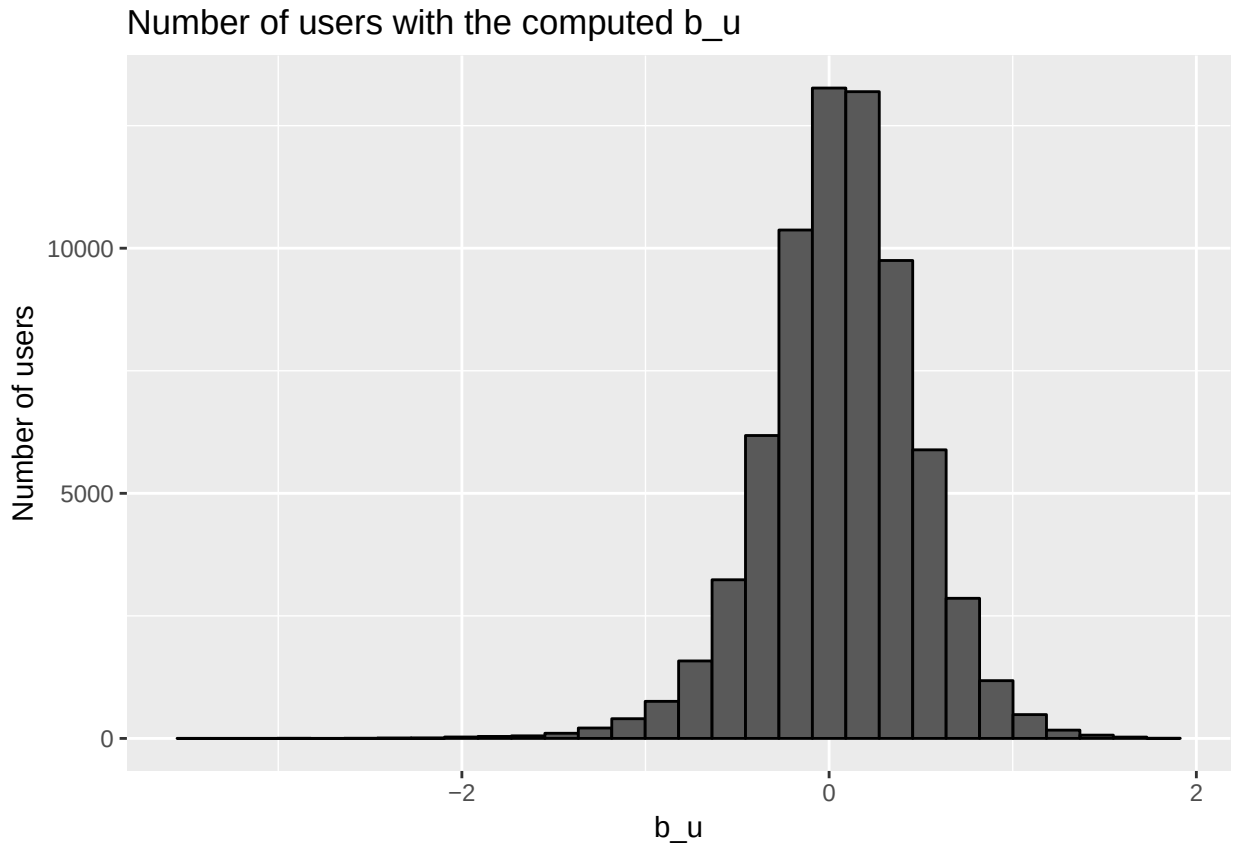
3- Movie and user model

Taking into account the user effect b_u = mean(rating - mu - b_i)

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

Plot number of users of the computed b_u

```
user_avgs%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"),
                   ylab = "Number of users", main = "Number of users with the computed b_u")
```



Predict ratings considering movies and users effect

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

Calculate movie & user effect model RMSE

```
movie_user_model_rmse <- RMSE(predicted_ratings, validation$rating)
cat("Movie & user effect model RMSE = ", movie_user_model_rmse)
```

```
## Movie & user effect model RMSE =  0.8653488
```

Results

The RMSE of all tested models are as follows:

```
cat("Basic average movie rating RMSE = ", basic_prediction_rmse)
```

```
## Basic average movie rating RMSE =  1.061202
```

```
cat("Movie effect model RMSE = ", movie_model_rmse)
```

```
## Movie effect model RMSE =  0.9439087
```

```
cat("Movie & user effect model RMSE = ", movie_user_model_rmse)
```

```
## Movie & user effect model RMSE =  0.8653488
```

The lowest RMSE was gotten when considering movie and the user in the model.

Conclusion

We have built a an algorithm to predict movie ratings.

We had to just use movie and user to get RMSE value of 0.8648170. We can also use other provided variables to improve results much more and get less RMSE value.