HarvardX Data Science MovieLens project

Omid Airom

10/14/2021

Movie Lens

This is the Report for the Movie Lens project in the last course of the Harvard University Data Science Professional Certificate in he edx educational platform.

The aim of the project is to use the dataset provided by the course including 10 million rows of data and 6 features to build a model to predict the ratings of the movies in our dataset with a RMSE at least lower than 0.89999. To reach this goal we need to first have an exploration and analysis on the data and then build models until we get the required RMSE.

RMSE is the root mean squared error that shows us how well our model works.

Introduction

Recommendation systems use ratings that users have given to items to make recommendations due to the ratings gathered by the other people who used these products. A lot of online stores make it available for costumers to rate the products they have buyed and collec these data to make recommendations to other costumers.

Grouplens research has done the same thing and collected a dataset from movie ratings that includes about 10 million rows and 6 feature for each row and we are using this dataset and machine learning methods to build a recommendation system to predict the movie ratings.

Dowloading Data

Downloading and using the dataset :

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

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

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

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

# changing timeout to more than 60 seconds to prevent error
options(timeout = 3600)



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 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")
```

Like every machine learning project we need two sets of data: one to be used to train the model and another one to test how good our model works. With the code below we split our downloaded data to the edx training dataset and the validation dataset to test our models.

```
# Validation set will be 10% of MovieLens data
set.seed(1) # 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")
```

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

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

Exploring the Data

Data Analysis

To see the dataset, we find the first 5 rows of "edx" data as below. "userID", "movieID", "rating", "timestamp", "title", and "genres" are the features in each row and in each row we can find the rating of a user for a single movie. We can have multiple rows for each movie and user.

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     231      5 838983392          Dumb & Dumber (1994)
## 4:      1     292      5 838983421               Outbreak (1995)
## 5:      1     316      5 838983392               Stargate (1994)
## 6:      1     329      5 838983392 Star Trek: Generations (1994)
##                            genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:                         Comedy
## 4:  Action|Drama|Sci-Fi|Thriller
## 5:       Action|Adventure|Sci-Fi
## 6: Action|Adventure|Drama|Sci-Fi
```

Checking for missing values :

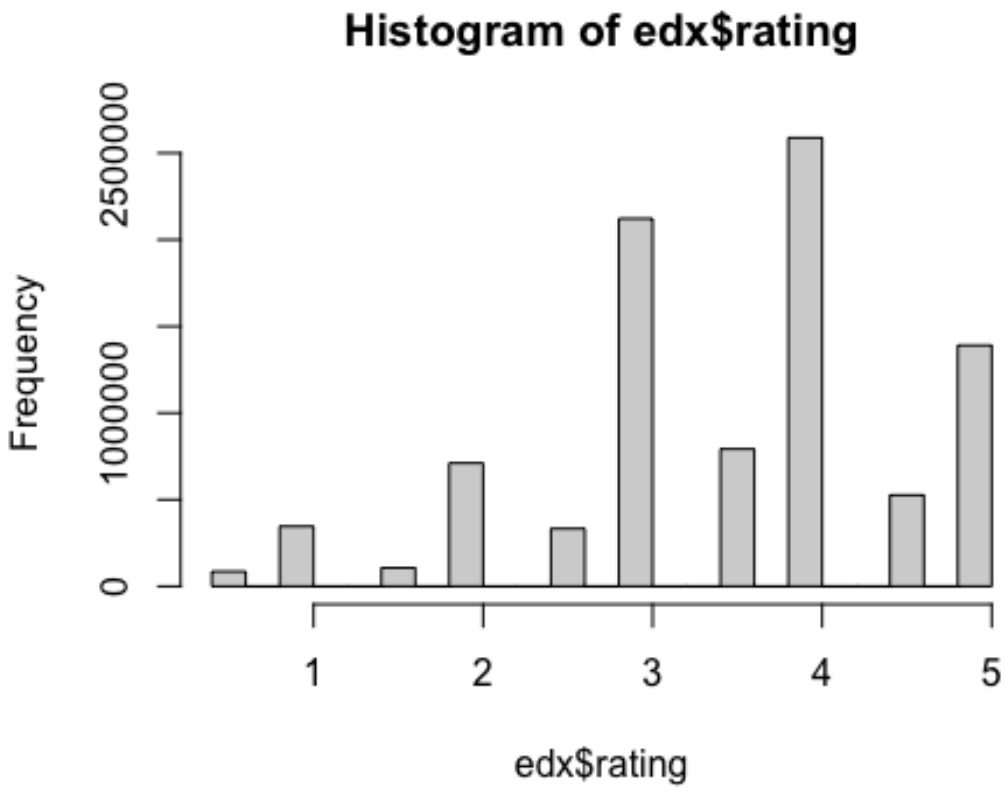
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000061     Length:9000061
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##

## [1] 0
```

we have 69878 unique movies and 10677 users in the edx data :

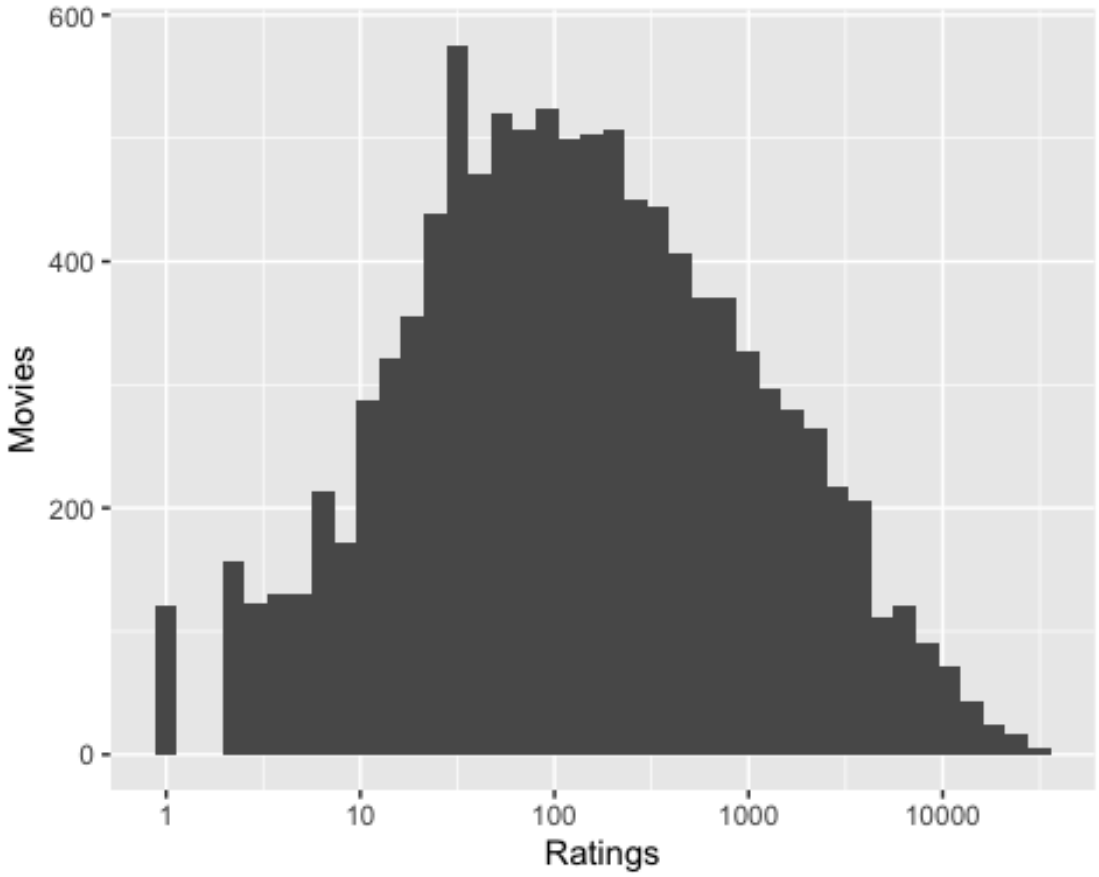
```
##   n_users n_movies
## 1   69878    10677
```

With the histogram of the rating data we can see that most of the audiences prefer to rate the movies with high number rather than the lowers. Most of the movies have a rating near to 4 and then in the next steps 3 and 5 and a few people have rated movies with low marks.



We can see that some movies have more ratings, while some others have fewer. We have to know exactly this amount because some movies with fewer ratings can change the power of prediction in our models.

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 40) +
  scale_x_log10() +
  xlab("Ratings") +
  ylab("Movies")
```




```
edx %>% group_by(title) %>%
  summarise(count = n()) %>%
  slice_max(count, n=10)
```

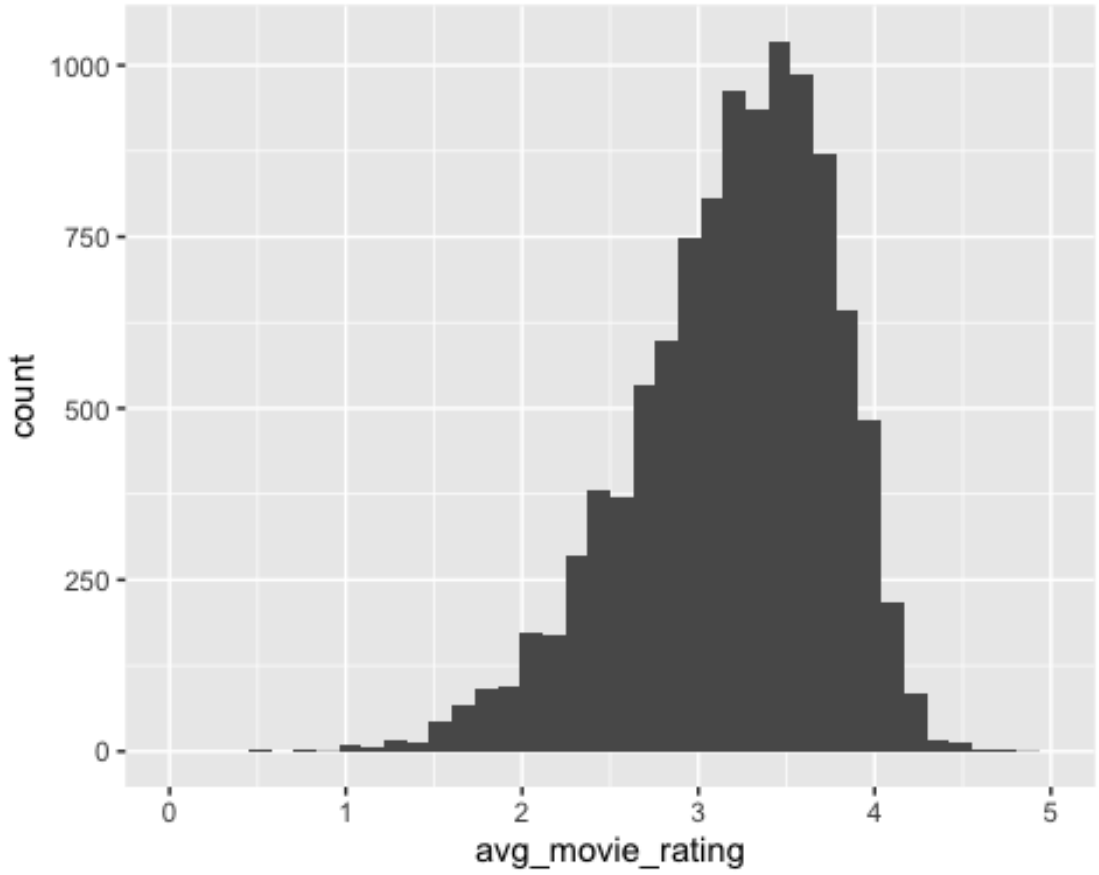
```
## # A tibble: 10 × 2
##    title                                                     count
##    <chr>                                                     <int>
##  1 Pulp Fiction (1994)                                       31336
##  2 Forrest Gump (1994)                                       31076
##  3 Silence of the Lambs, The (1991)                          30280
##  4 Jurassic Park (1993)                                      29291
##  5 Shawshank Redemption, The (1994)                          27988
##  6 Braveheart (1995)                                         26258
##  7 Terminator 2: Judgment Day (1991)                         26115
##  8 Fugitive, The (1993)                                      26050
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25809
## 10 Batman (1989)                                             24343
```

Most of the users in this dataset have rated 30 up to 100 movies.

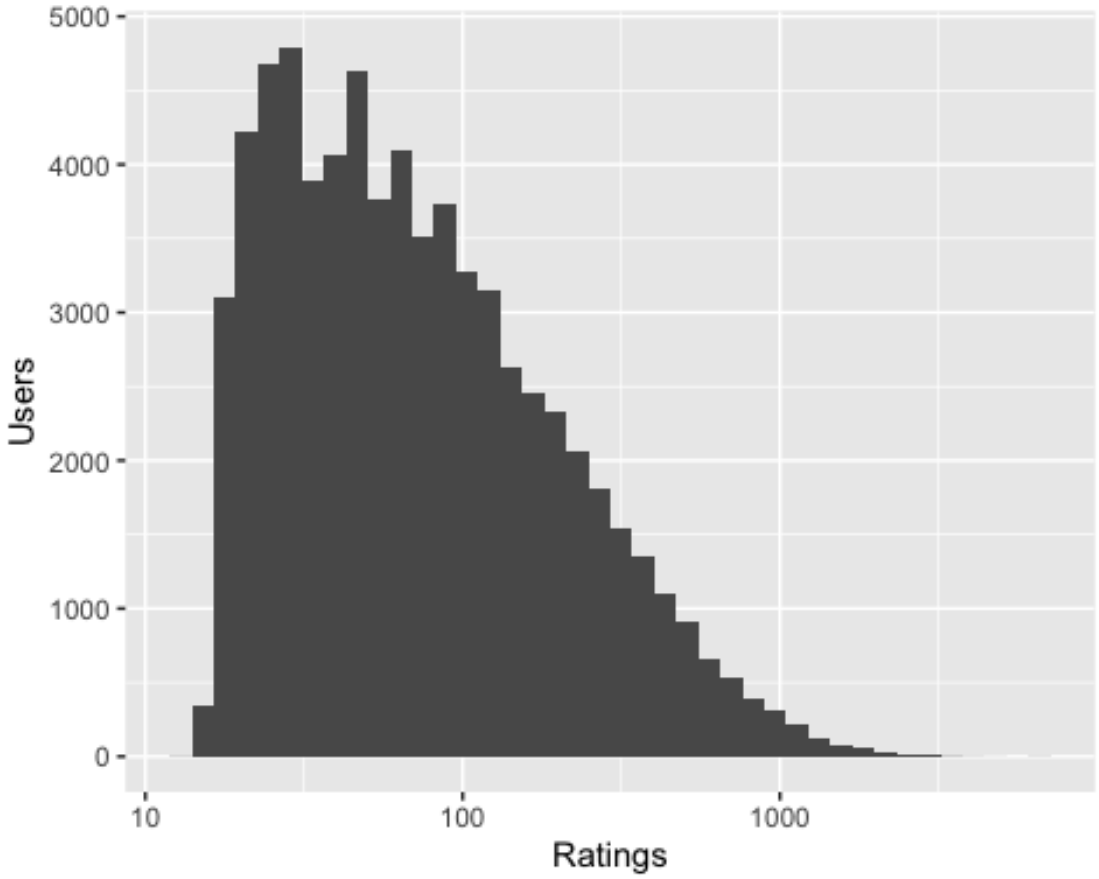
```
movies_by_title <- edx %>% group_by(title)
avg_of_movies <- movies_by_title %>% summarize(count = n()
                                               , avg_movie_rating =
mean(rating))
```

```
avg_of_movies %>%
  ggplot(aes(avg_movie_rating)) +
  geom_histogram(bins = 40) +
  xlim(0, 5)

## Warning: Removed 2 rows containing missing values (geom_bar).
```

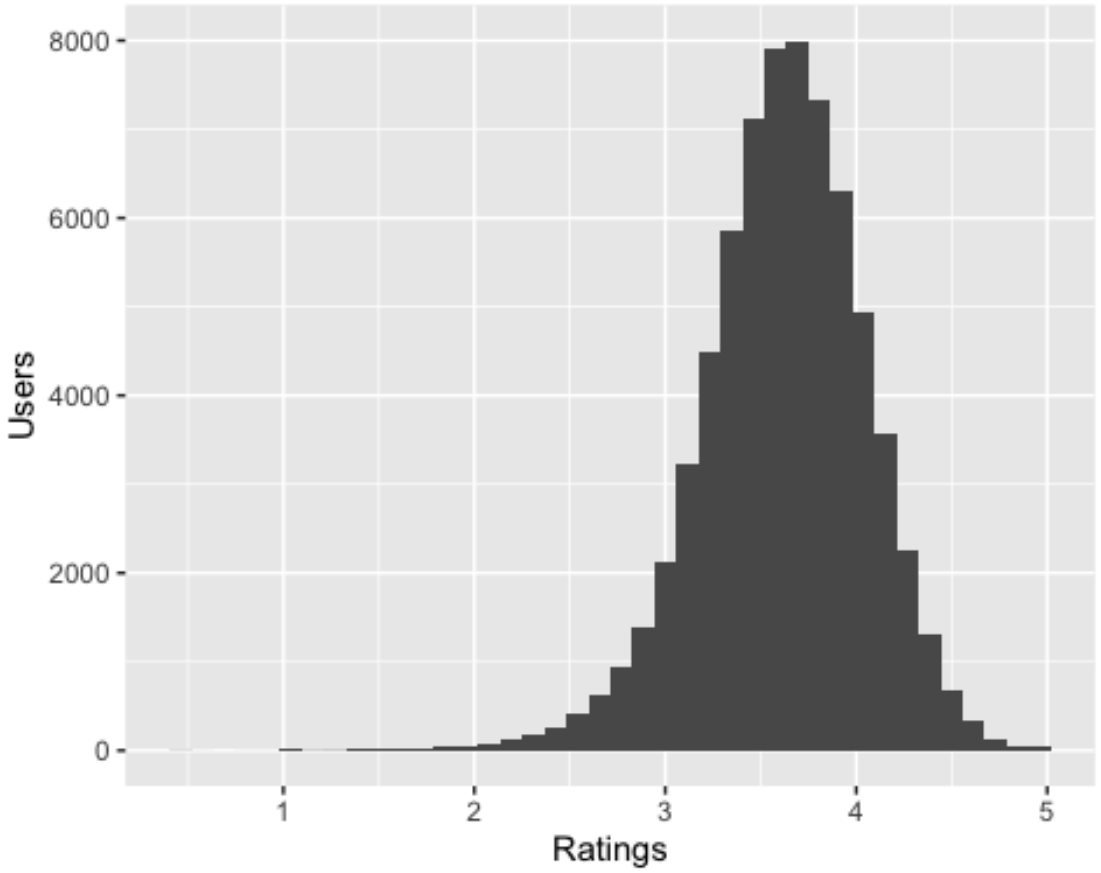



```
users <- edx %>% count(userId)
users %>% ggplot(aes(n)) +
  geom_histogram(bins = 40) +
  scale_x_log10() +
  xlab("Ratings") +
  ylab("Users")
```



We also can see and use the average of ratings given to movies by each user to se the average rating of the to all the movies they have seen.

```
edx %>%
  group_by(userId) %>%
  summarize(avg_ur = mean(rating)) %>%
  ggplot(aes(avg_ur)) +
  geom_histogram(bins = 40) +
  xlab("Ratings") +
  ylab("Users")
```


8000
6000
4000
2000
0
Users
1
2
3
4
5
Ratings

Models

The function used to calculate the predictions with the true ratings :

```
RMSE <- function(predictions, ratings){
  sqrt(mean((predictions - ratings)^2))}
```

Our goal is to minimize and lower the RMSE.

ratings average of the movies

The first model we build to predict the ratings in this project is just build by the average of the ratings for all of the movies, without paying attention to the users.

```
ratings_avg <- mean(edx$rating)
prediction <- RMSE(validation$rating, ratings_avg)
prediction

## [1] 1.060651

results <- tibble(model = "ratings average of the movies", RMSE = prediction)
results

## # A tibble: 1 × 2
##   model                          RMSE
##   <chr>                         <dbl>
## 1 ratings average of the movies  1.06
```

This is the first RMSE that we can use to predict this model to the next ones and see if we reach below 0.89999.

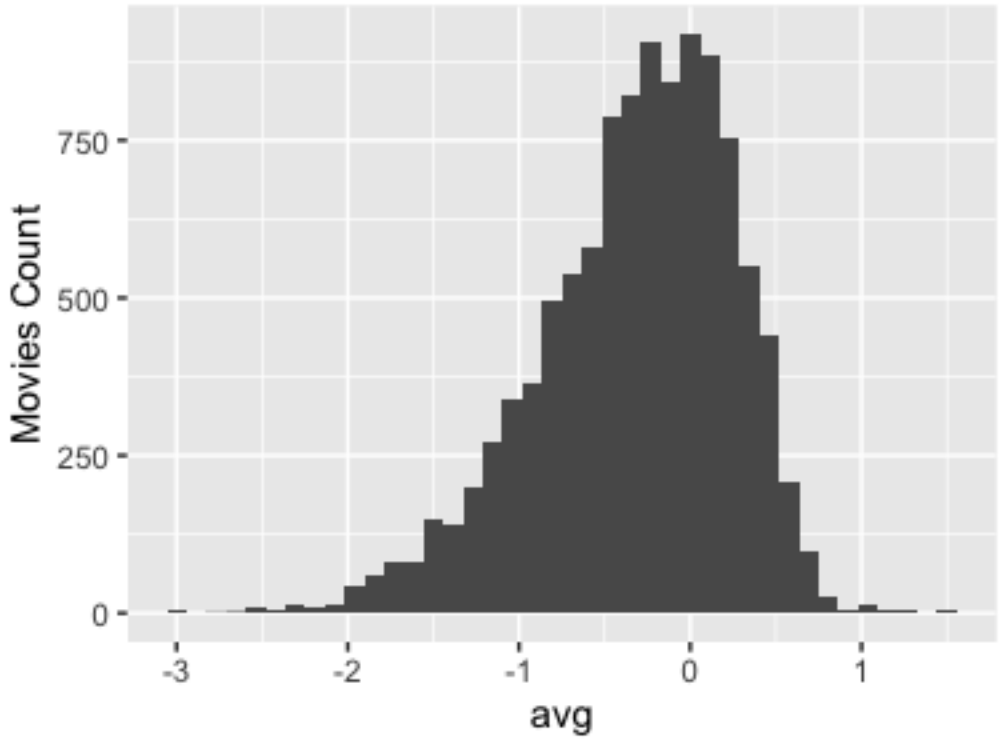
To improve the next models we use some of the information we got from the exploration & analysis on the dataset.

movie effect

To improve the last model we had we use the fact that some movies that are more popular among the audiences have better and higher ratings than others that are less popular. We calculate the difference between the mean rating of each movie and the mean of all the ratingsc (mean rating of all movies) which we calculated and used in the previous model.

```
movie_effect <- edx %>%
  group_by(movieId) %>%
  summarize(avg = mean(rating - ratings_avg))

movie_effect %>% ggplot(aes(avg)) +
  geom_histogram(bins = 40) +
  ylab("Movies Count")
```




```
predicted_ratings <- ratings_avg +  validation %>%
  left_join(movie_effect, by='movieId') %>%
  pull(avg)

head(predicted_ratings)

## [1] 3.669204 3.998851 2.943630 3.537779 4.072319 3.508718

new_prediction <- RMSE(predicted_ratings, validation$rating)
results <- bind_rows(results, data_frame(model = "movie effect"
                                         ,RMSE = new_prediction ))

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

results

## # A tibble: 2 × 2
##   model                          RMSE
##   <chr>                         <dbl>
## 1 ratings average of the movies 1.06
## 2 movie effect                  0.944
```

We can see that we got a lower RMSE in this model than the previous model and the movie effect improved the model, but we can still improve this improved model by having the

same approach for the ratings of each user for the movies and combine the movie effect with the user effect. Also we have not yet reached our goal to get a RMSE lower than 0.89999 so we have to keep up improving our model.

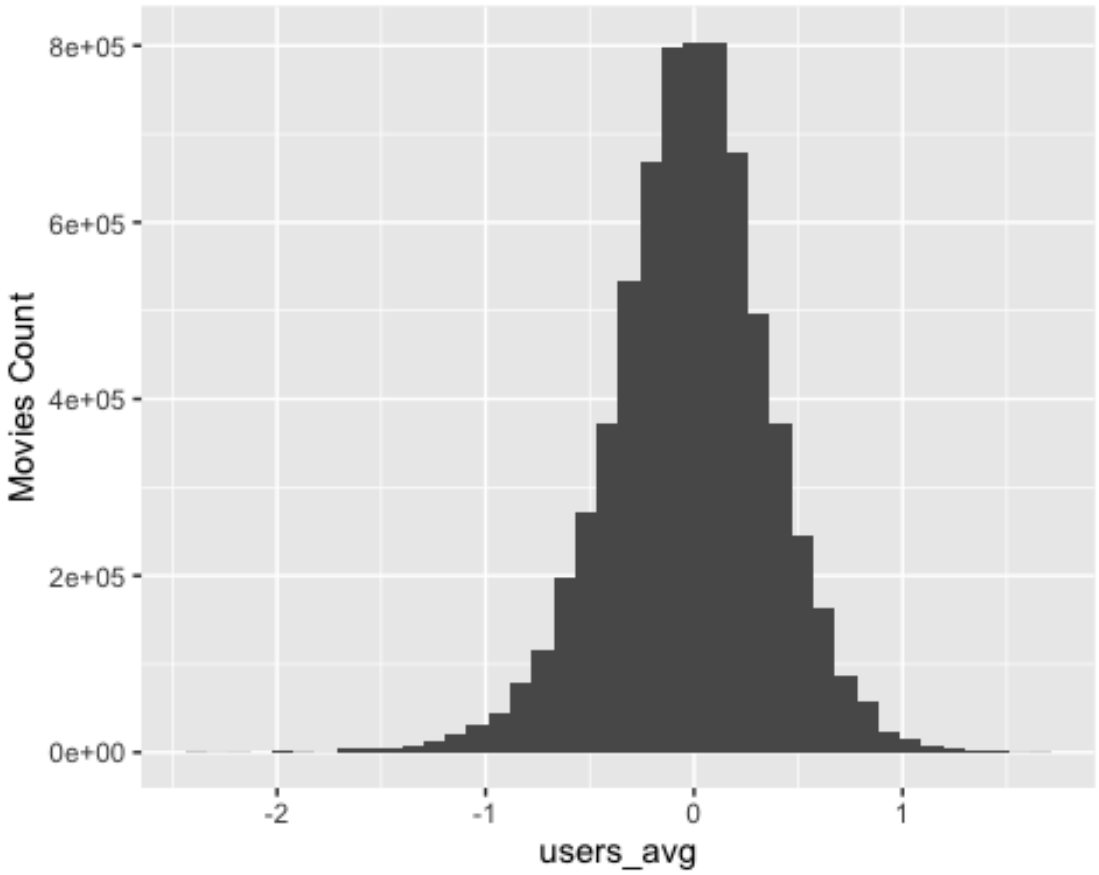
movie and user effect

In this model we have to compute the average rating for users.

```
users <- edx %>% left_join(movie_effect, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100)

users <- users %>% mutate(users_avg = mean(rating - ratings_avg - avg))

users %>% ggplot(aes(users_avg)) +
  geom_histogram(bins = 40) +
  ylab("Movies Count")
```




```
users <- edx %>%
  left_join(movie_effect, by='movieId') %>%
  group_by(userId) %>%
  summarize(users_avg = mean(rating - ratings_avg - avg))
```

```
head(movie_effect)
```

```
## # A tibble: 6 × 2
##   movieId    avg
##     <dbl>  <dbl>
## 1       1  0.414
## 2       2 -0.310
## 3       3 -0.362
## 4       4 -0.632
## 5       5 -0.434
## 6       6  0.299
```

```
head(users)
```

```
## # A tibble: 6 × 2
##   userId users_avg
##    <int>     <dbl>
## 1      1    1.71
## 2      2   -0.385
## 3      3    0.306
## 4      4    0.661
## 5      5    0.0190
## 6      6    0.322
```

```
a <- users %>% select(users_avg, userId)
head(a)
```

```
## # A tibble: 6 × 2
##   users_avg userId
##       <dbl>  <int>
## 1    1.71        1
## 2   -0.385       2
## 3    0.306       3
## 4    0.661       4
## 5    0.0190      5
## 6    0.322       6
```

```
nrow(a)
```

```
## [1] 69878
```

```
nrow(movie_effect)
```

```
## [1] 10677
```

```
new_preds <- validation %>%
  left_join(movie_effect, by='movieId')
new_preds <- new_preds %>%
  left_join(a, by='userId')
new_preds <- new_preds %>%
  mutate(pred = ratings_avg + avg + users_avg) %>%
```

```
  pull(pred)

head(new_preds)

## [1] 5.379742 3.613422 2.558201 3.843998 4.378538 3.814937

head(validation)

##    userId movieId rating  timestamp
## 1:      1     588    5.0  838983339
## 2:      2    1210    4.0  868245644
## 3:      2    1544    3.0  868245920
## 4:      3     151    4.5 1133571026
## 5:      3    1288    3.0 1133571035
## 6:      3    5299    3.0 1164885617
##                                                    title
## 1:                                          Aladdin (1992)
## 2:       Star Wars: Episode VI - Return of the Jedi (1983)
## 3: Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
## 4:                                          Rob Roy (1995)
## 5:                               This Is Spinal Tap (1984)
## 6:                         My Big Fat Greek Wedding (2002)
##                                        genres
## 1: Adventure|Animation|Children|Comedy|Musical
## 2:                   Action|Adventure|Sci-Fi
## 3:     Action|Adventure|Horror|Sci-Fi|Thriller
## 4:                    Action|Drama|Romance|War
## 5:                              Comedy|Musical
## 6:                              Comedy|Romance

nrow(validation)

## [1] 999993
```

Now we make our predictions to see the RMSE improvement.

```
mu_model <- RMSE(new_preds, validation$rating)
mu_model

## [1] 0.8655329

results

## # A tibble: 2 × 2
##   model                          RMSE
##   <chr>                         <dbl>
## 1 ratings average of the movies 1.06
## 2 movie effect                  0.944

results <- bind_rows(results
                     ,data_frame(model="movie and user effect"
```

```
                    ,RMSE = mu_model))
results

## # A tibble: 3 × 2
##   model                          RMSE
##   <chr>                         <dbl>
## 1 ratings average of the movies 1.06
## 2 movie effect                  0.944
## 3 movie and user effect         0.866
```

Now we can see that by this model built on both the users effect and movie effect we reached the goal of the project the RMSE is equal to 0.8653488 and we can see that our improvent was successful and we have now a model more accurate than the two previous models.

Conclusion

We built a simple machine learning model to predict the rating of the movies and we reached the goal to have a model with a RMSE lower than 0.89999. But it is still possible to improve the last model by analyzing other features like genre , date & ... and see how their effect can change the RMSE of the new models built with them.