Movie Recommendation System
Limbrit
17 Feb 2019

Introduction

In this project, I demonstrate a film recommender system capable of recommending movies to users based on a rating scale. We will use the following libraries:

```
# Load required library
library(anytime)
library(data.table)
library(scales)
library(doParallel)
cl <- makeCluster(4)
registerDoParallel(cl)

# Time stamp as date factor
edx$timestamp<-anydate(edx$timestamp)
edx$date<-as.factor(format(edx$timestamp, "%Y-%m"))
edx$date<-as.factor(edx$date)
```

Data Loading

We develop the algorithm using the edx set, and have a final test with the validation set. These datasets are extracted with the project's code available at https://courses.edx.org/.

Data summary

```
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
```

```
head(validation)
```

```
##   userId movieId rating timestamp
## 1      1     231      5 838983392
## 2      1     480      5 838983653
## 3      1     586      5 838984068
## 4      2     151      3 868246450
## 5      2     858      2 868245645
## 6      2    1544      3 868245920
##                                                 title
## 1                                   Dumb & Dumber (1994)
## 2                                   Jurassic Park (1993)
## 3                                      Home Alone (1990)
## 4                                         Rob Roy (1995)
## 5                                    Godfather, The (1972)
## 6 Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
```

```
##                                       genres
## 1                                     Comedy
## 2          Action|Adventure|Sci-Fi|Thriller
## 3                           Children|Comedy
## 4                  Action|Drama|Romance|War
## 5                             Crime|Drama
## 6 Action|Adventure|Horror|Sci-Fi|Thriller
```

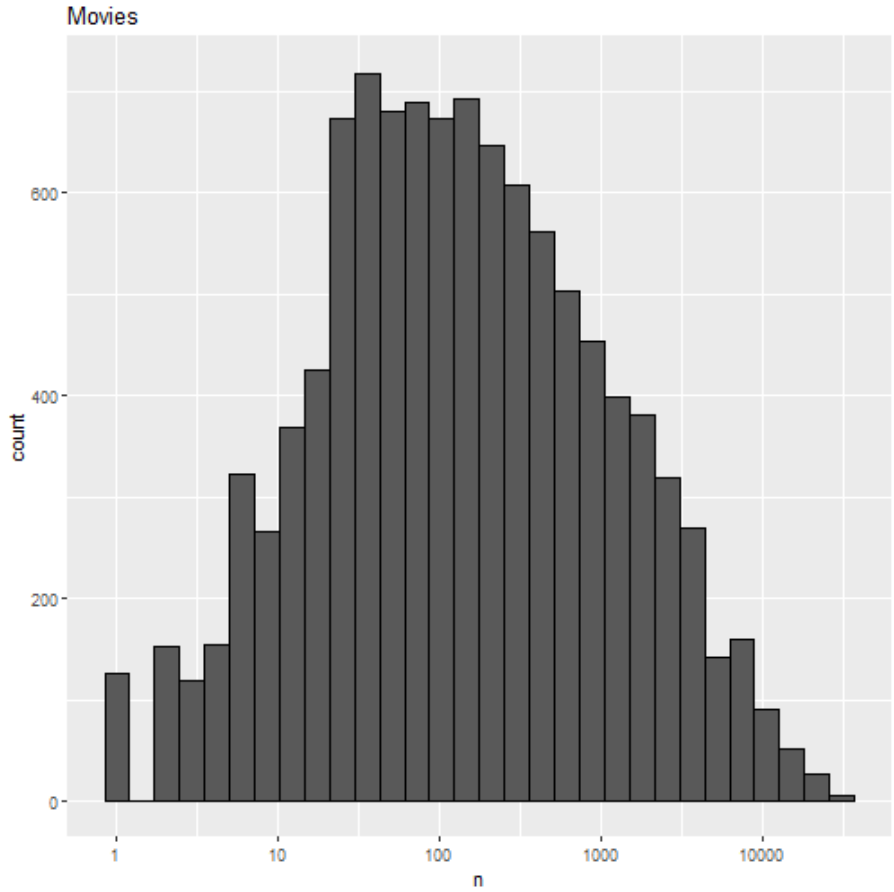
Number of unique users that provided ratings and for how many unique movies they provided :

```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

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

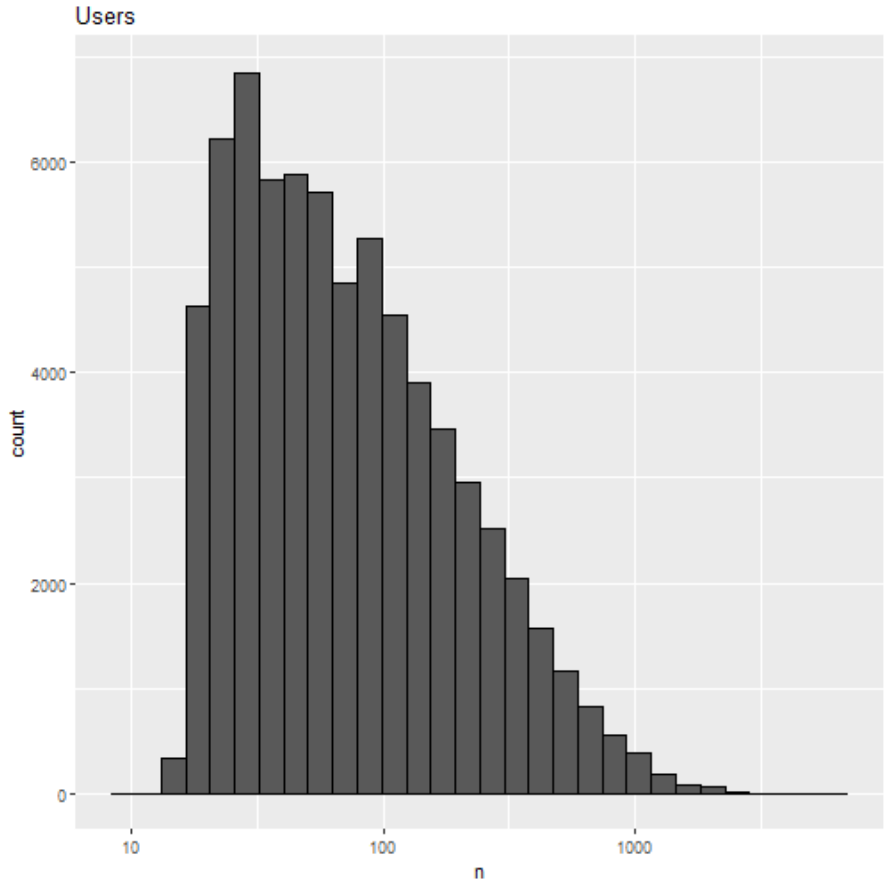
The first thing we notice is that some movies get rated more than others. Here is the distribution :

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



Our second observation is that some users are more active than others at rating movies :

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



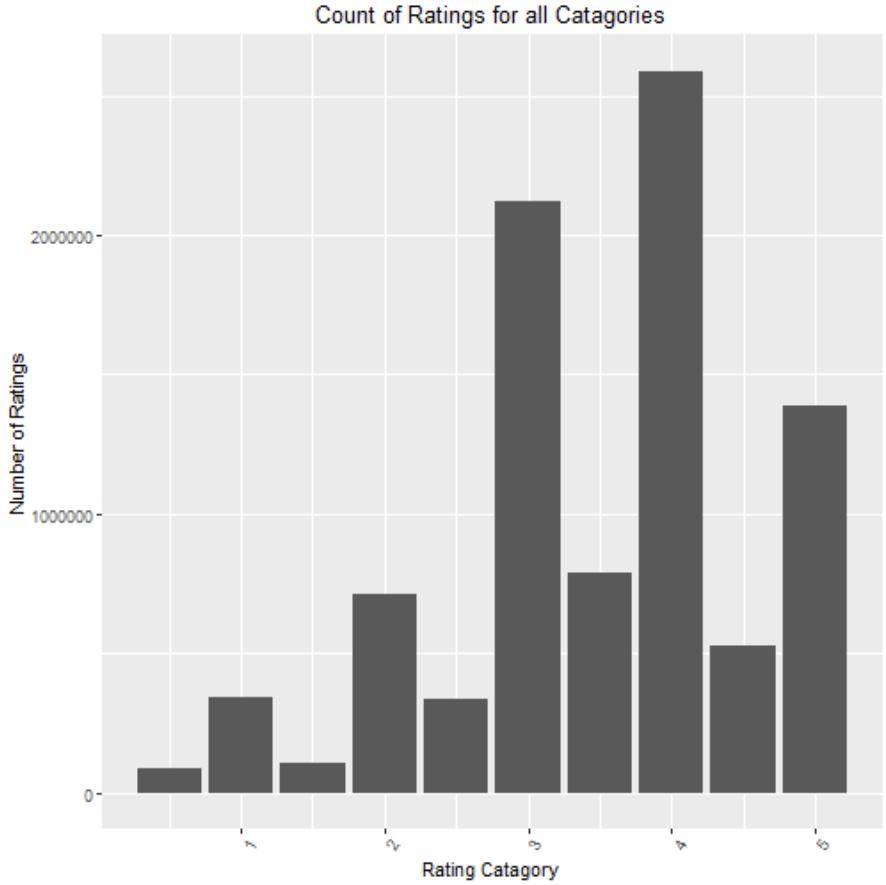
Rating Distribution

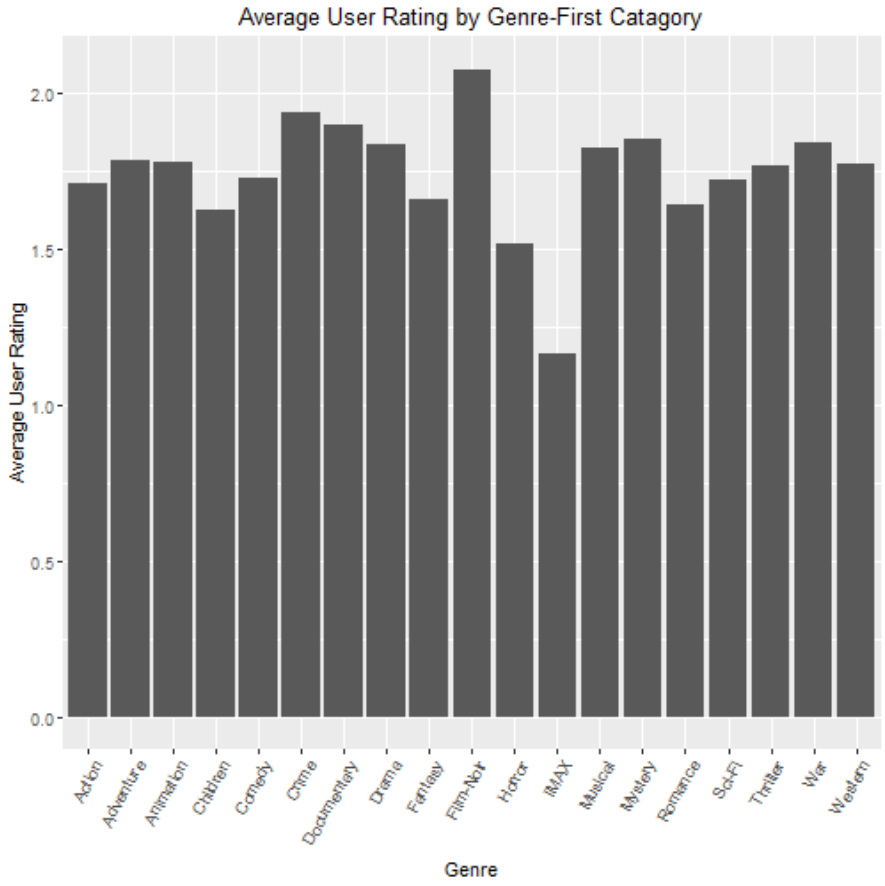
```
aa<-as.data.frame(edx %>%
                    group_by(rating) %>%
                    summarise (n = n()) %>%
                    mutate(percent = n / sum(n)))
aa$percent<-percent(aa$percent)
print(aa)
```

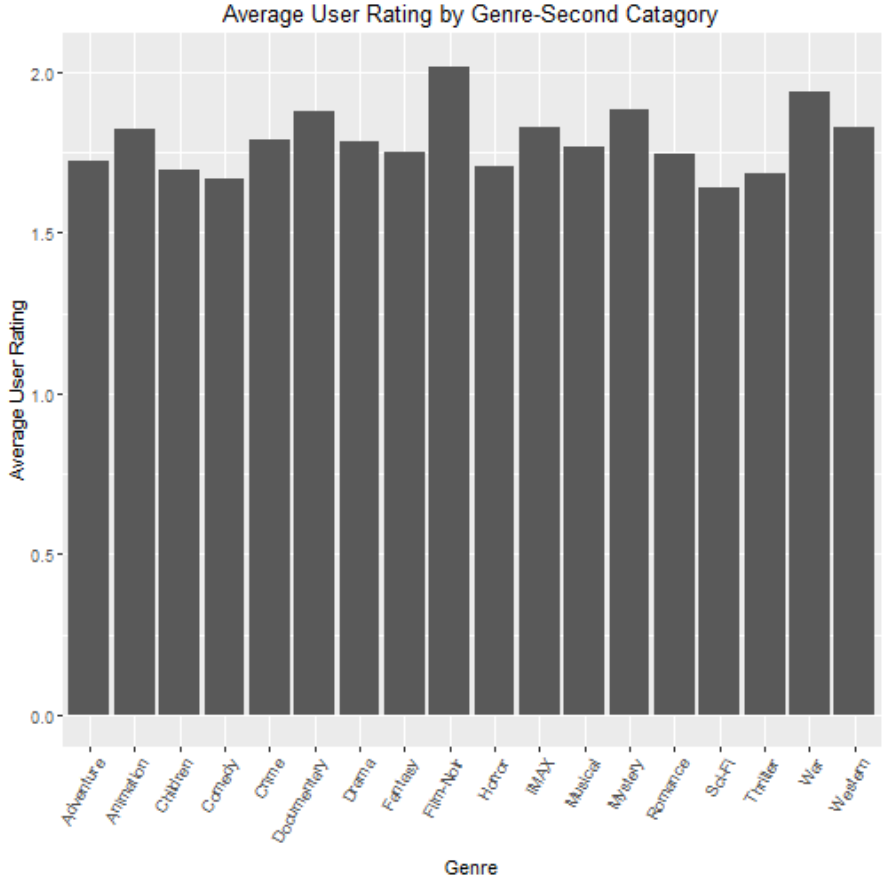
```
##    rating       n percent
## 1     0.5   85374    0.9%
## 2     1.0  345679    3.8%
## 3     1.5  106426    1.2%
## 4     2.0  711422    7.9%
## 5     2.5  333010    3.7%
## 6     3.0 2121240   23.6%
## 7     3.5  791624    8.8%
## 8     4.0 2588430   28.8%
## 9     4.5  526736    5.9%
## 10    5.0 1390114   15.4%
```

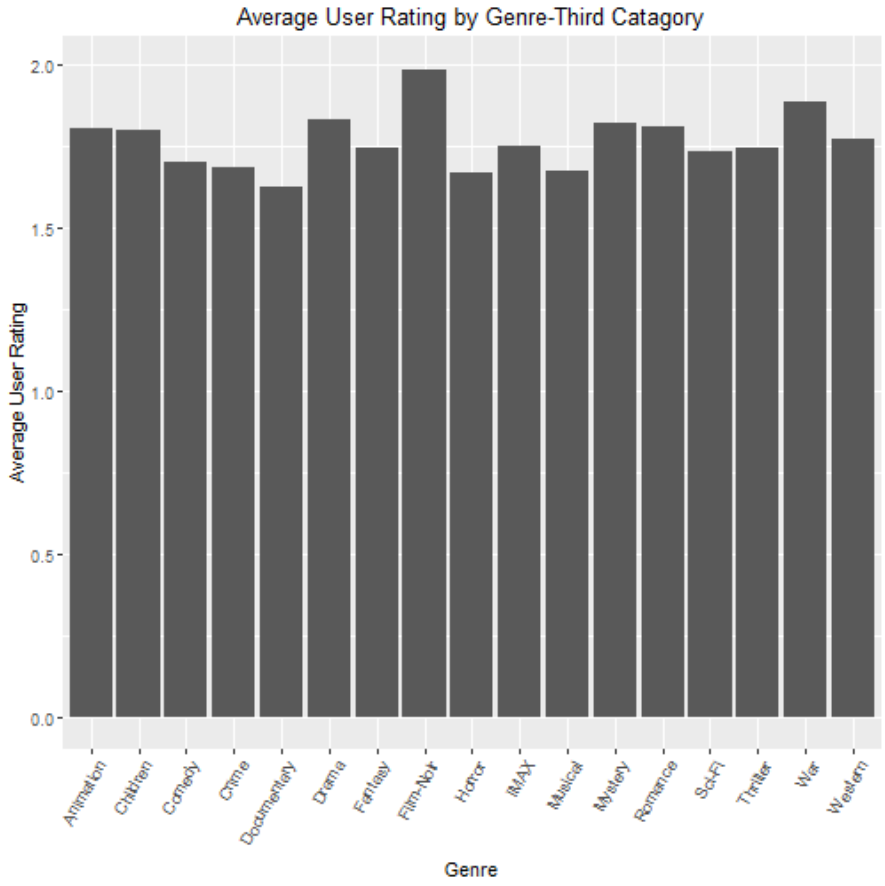
Summary-The ratings show that the largest number of reviewers gave ratings of 3(23.6%), 4 (28.8%) or 5 (15.4%). The lower ratings or x.5 ratings were all under 10% or the ratings given.

Plots-Data Visualization




Average User Rating by Genre-First Catagory
Average User Rating
2.0
1.5
1.0
0.5
0.0
Action
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film-Noir
Horror
IMAX
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western
Genre


Average User Rating by Genre-Second Catagory
Average User Rating
2.0
1.5
1.0
0.5
0.0
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film-Noir
Horror
IMAX
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western
Genre



Summary-The "Count of Ratings for all Categorizes" mirrors the proportion table given in the Process Data Chunk. The next three bar graphs show the first three Categories broken out from the genres variable. The first category had The Film-Noir, Mystery and Crime having the highest average rating, the second Film-Noir, War and Documentary with the highest average rating and the third category had Film-Noir and War as the highest category.

Training and Testing

```
# Define test and train datasets using edx: 80% sample for training, and 20% sample for testing.

set.seed(1)
train_index <- createDataPartition(y = edx$rating, times = 1, p = 0.8, 
list = FALSE)
train_set <- edx[train_index, ]
temp <- edx[-train_index, ]

# Make sure userId and movieId in test set are also in train set

test_set <- temp %>% 
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")


# Add rows removed from test set back into train set

removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)
```

```
# For the recommendation systems, we will use three approaches and choose
the best model with the lowest RMSE

# Root Mean Square Error function

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

In this first model, we predict the same rating for all movies regardless of user :

```
# Raw mean of the training dataset

mu_hat <- mean(train_set$rating)
mu_hat

## [1] 3.512349

model_1_rmse <- RMSE(test_set$rating, mu_hat)
model_1_rmse

## [1] 1.059735
```

As we go along, we will be comparing two others approaches. Let's start by creating a results table with this naive approach :

```
rmse_results <- data_frame(method = "Just the average", RMSE =
model_1_rmse)
rmse_results%>%knitr::kable()
```

method	RMSE
Just the average	1.059735

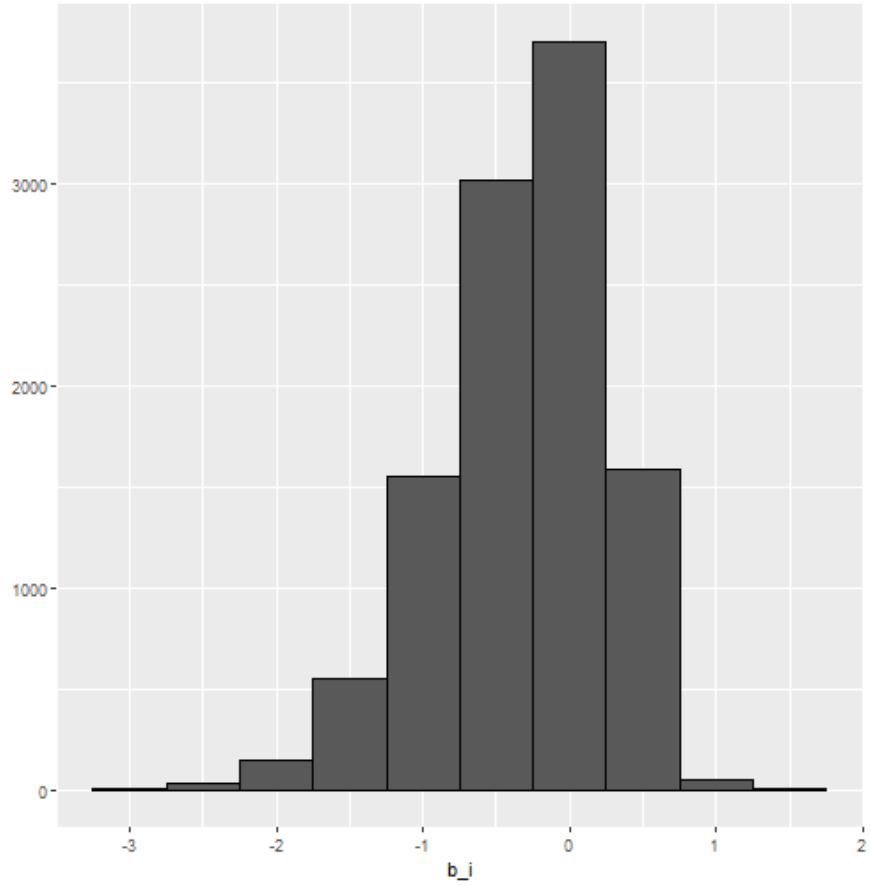
I consider the movie effects in the second model.

```
# fit <- lm(rating ~ as.factor(userId), data = movielens)
# the lm() function will be very slow here because there are thousands of
bias, each movie gets one
# I use instead the least square estimate :

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

These estimates vary substantially

```
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color =
I("black"))
```



Let's see how much the RMSE improves with this second model :

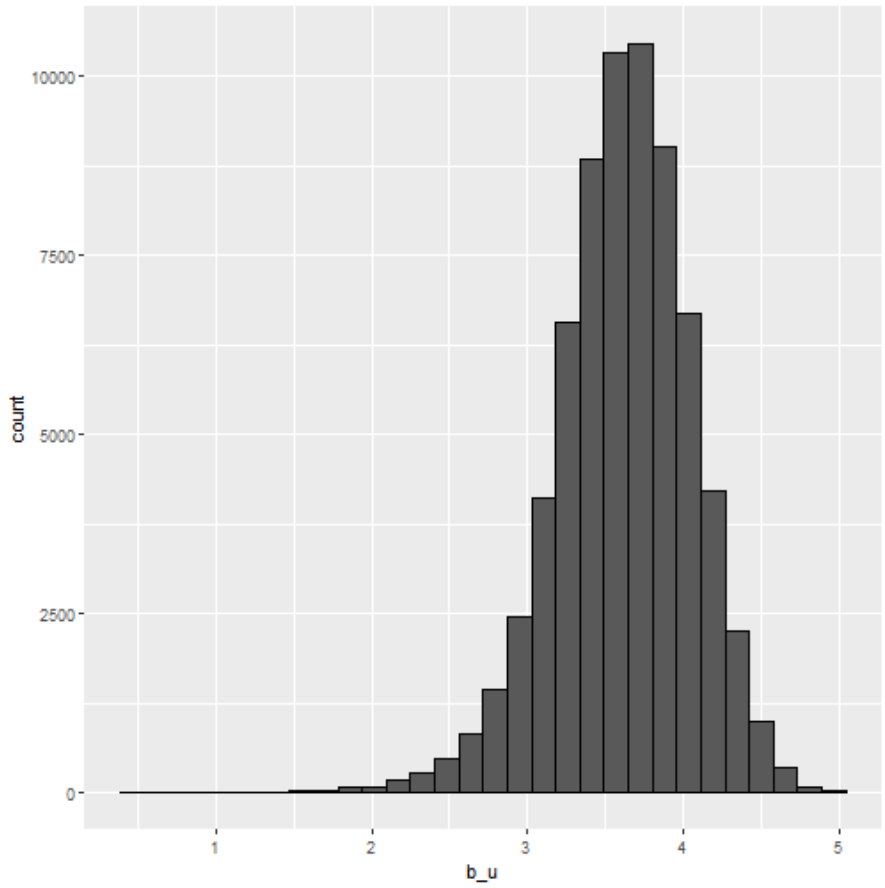
```
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.059735
Movie Effect Model	0.943203

In the third model, I apply the user effects. Let's compute the average rating for user u for those that have rated over 100 movies. Notice that there is substantial variability across users as well: some users are very cranky and others love every movie.

```
train_set %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```




```
# User-specific effect model : lm(rating ~ as.factor(movieId) +
as.factor(userId))

# We will compute an approximation instead for the reasons described earlier
in 2nd model


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


We can now construct predictors and see how much the RMSE improves :

predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_3_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_3_rmse ))
```

Results

Models RMSE The 3rd model has the lowest RMSE and will be used for the final testing of the validation set

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

method	RMSE
Just the average	1.0597347
Movie Effect Model	0.9432030
Movie + User Effects Model	0.8426298

RMSE of the validation set

```
## Validation test
# We compute first the user effect for validation set

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

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

model_rmse_validation <- RMSE(predicted_ratings, validation$rating)
model_rmse_validation

## [1] 0.8294231
```

Conclusion and discussion

In this project, I have developed and evaluated the naive approach, the movie effects and the user effects for recommending movies. The movie effects and user effects bring big improvements. The dataset provided, also includes the ratings timestamp, and movies genres. To go further, theses variables could be analysed to aspire to lower the RMSE and develop a better predictive method.