Movie Recommendation System

Alexis Koffi

07/02/2019

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:

```
library(tidyverse)
library(caret)
library(ggplot2)
```

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/.

Methods

Data summary

edx and *validation* are both in tidy format

```
# Subset of edx and validation data
head(edx[, 1:3])
```

```
##   userId movieId rating
## 1      1     122      5
## 2      1     185      5
## 4      1     292      5
## 5      1     316      5
## 6      1     329      5
## 7      1     355      5
```

```
head(validation[, 1:3])
```

```
##   userId movieId rating
## 1      1     231      5
## 2      1     480      5
## 3      1     586      5
## 4      2     151      3
## 5      2     858      2
## 6      2    1544      3
```

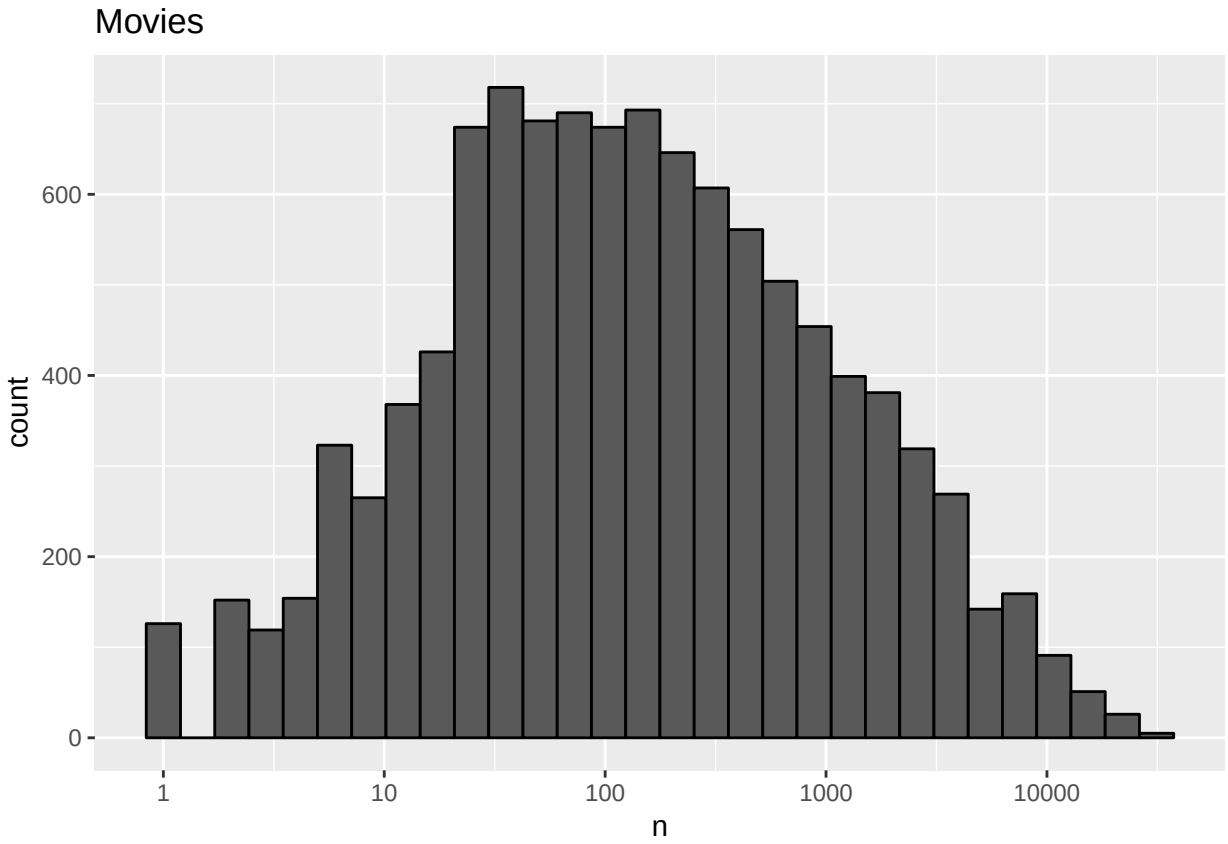
Number of unique users that provided ratings and for how many unique movies they provided :

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

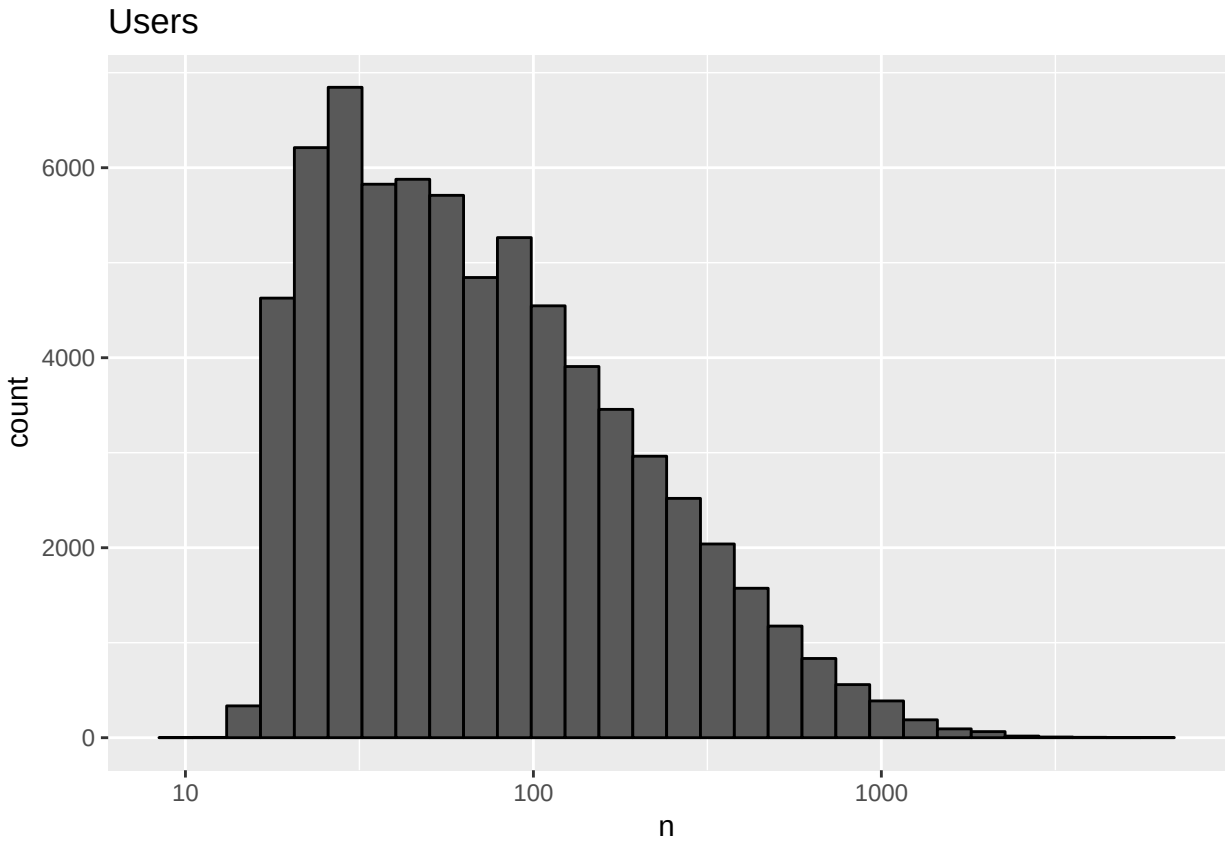
```
##   n_users n_movies
## 1   69878    10677
```

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 :



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)

rm(temp, removed) # remove temporary datasets
```

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

RMSE Calculations

```
# 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

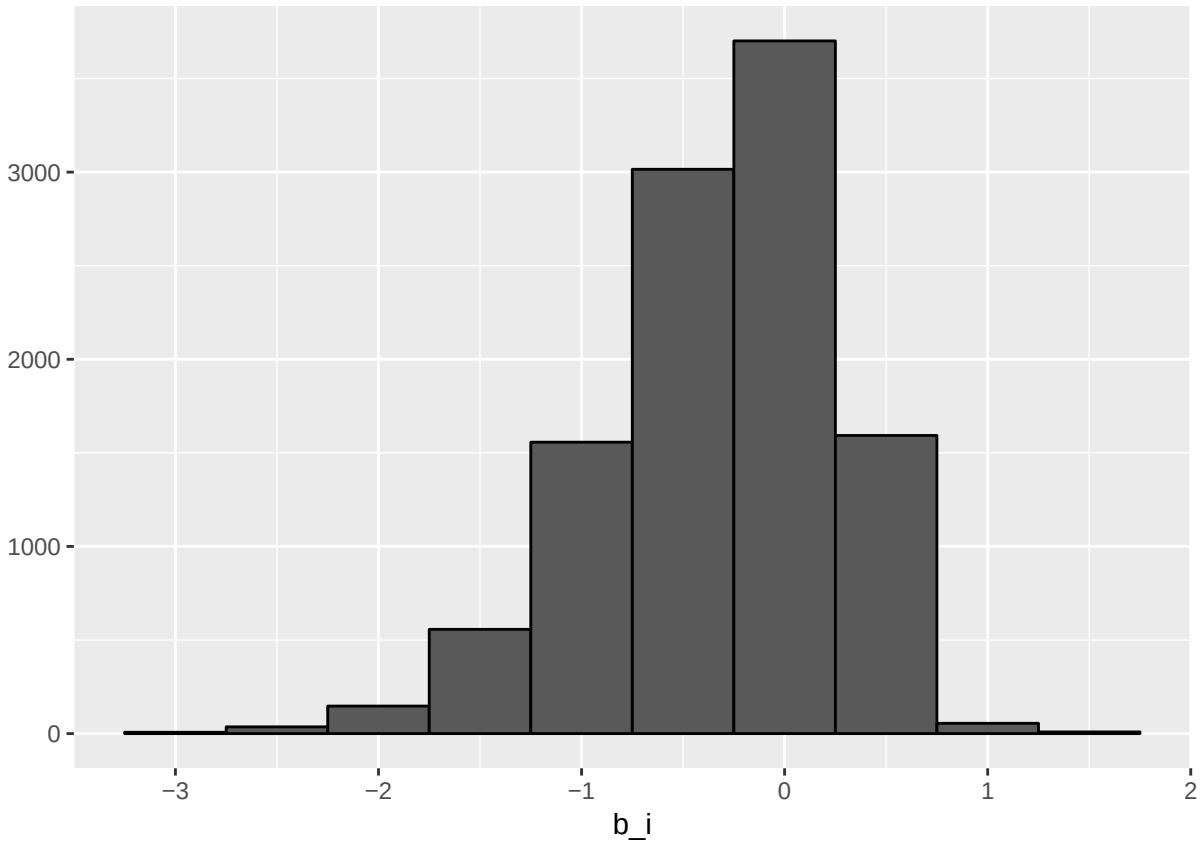
I consider the movie effects in the second model.

```
# 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 :

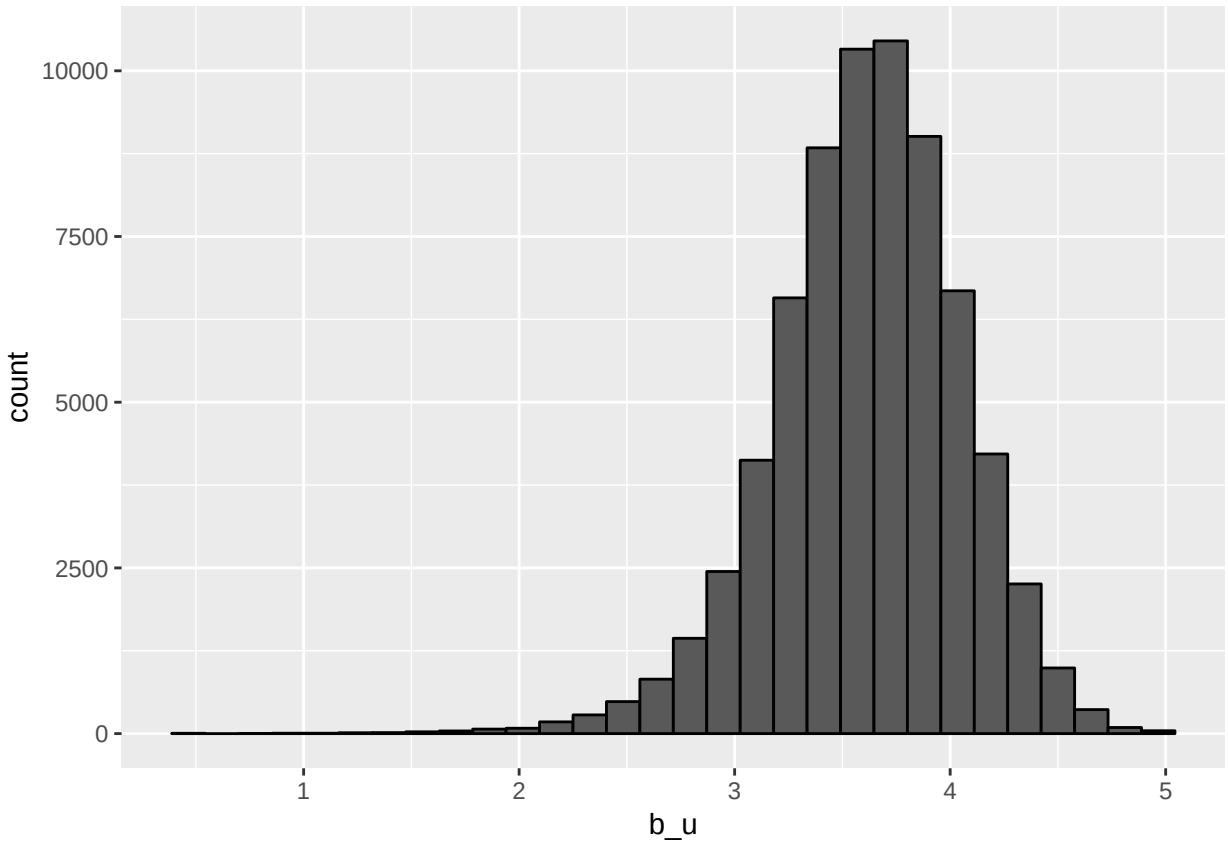
```
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

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")
```




```r
# 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 :

```r
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 analyzed to aspire to lower the RMSE and develop a better predictive method.