edx HarvardX Data Science Project Report - Movielens

Li Chen

6/9/2019

1. Introduction

Recommendation systems, which provide suggestions for items to a user, are one of the most widespread applications of machine learning. On October 2006, Netflix offered a challenge of improving the recommendation algorithm by 10% and winning a million dollars. In this course project a movie recommendation system will be created using the MovieLens dataset (https://grouplens.org/datasets/movielens/latest/) and tools we have learned throughout the courses in the edX HarvardX Data Science series (https://www.edx.org/professional-certificate/harvardx-data-science).

1.1 Movielens Data

Since the Netflix data is not publicly available, the data used for the movie recommendation system is the Movielens data generated by the GroupLens research lab. In this project the 10M version of the MovieLens dataset is used to make the computation a little easier. We download the MovieLens data from the Grouplens website (https://grouplens.org/datasets/movielens/latest/) and run the following code (provided by the course HarvardX: PH125.9x Data Science: Capstone (https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+2T2018/course/) to generate the training ("edx") and validation ("validation") datasets. As shown in the code, the validation dataset is 10% of the movielens data.

```r
# Create edx set and validation set

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, sample.kind = "Rounding") # R 3.6.0: set.seed(1, sample.kind = "Rounding")
```

```
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)

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

1.2 Objectives

The objectives of this project is to develop machine learning algorithms using the edx set, predict movie ratings in the validataion set as if they were unknown, and test the algorithm and measure the effectiveness of the recommendation system.

2. Methodology and Analysis

In this section, we first explore the movielens data and then different regression models for the movie recommendation system, including the simplest naive average model, movie effect model, user effect model, ane regularized movie and user effect model, will be studied.

2.1 Data Exploratory Analysis

Let's first look at the movielens data (edx dataset). It has 6 columns (userId, movieId, rating, timestamp, title and genres) and thousands of rows, each of which represents a rating of one movie given by one user.

```
library(tidyverse)
library(caret)

# Columns and Rows of edx dataset
dim(edx)
```

```
## [1] 9000055       6
```

```
edx %>% as_tibble()
```

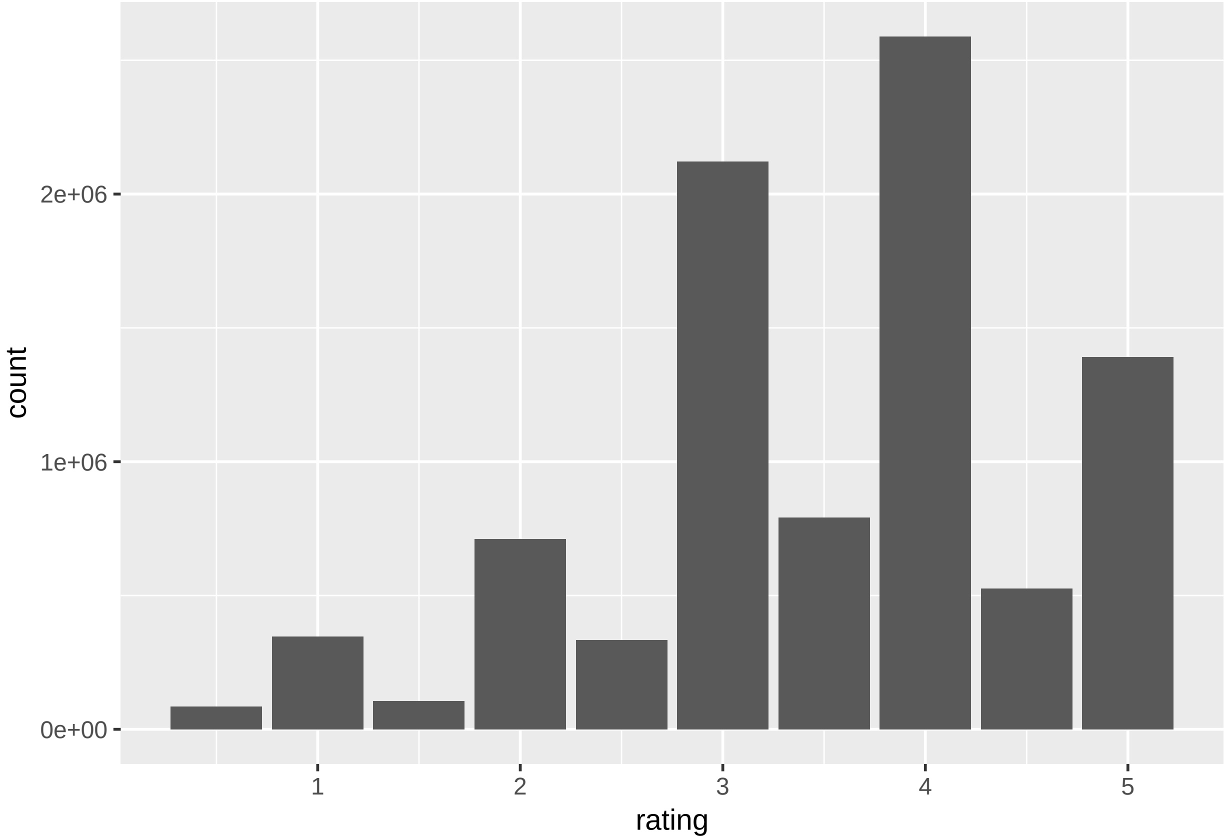
```
## # A tibble: 9,000,055 x 6
##    userId movieId rating timestamp title              genres
##     <int>   <dbl>  <dbl>     <int> <chr>              <chr>
##  1      1     122      5 838985046 Boomerang (1992)   Comedy|Romance
##  2      1     185      5 838983525 Net, The (1995)    Action|Crime|Thrill~
##  3      1     292      5 838983421 Outbreak (1995)    Action|Drama|Sci-Fi~
##  4      1     316      5 838983392 Stargate (1994)    Action|Adventure|Sc~
##  5      1     329      5 838983392 Star Trek: Generat~ Action|Adventure|Dr~
##  6      1     355      5 838984474 Flintstones, The (~ Children|Comedy|Fan~
##  7      1     356      5 838983653 Forrest Gump (1994) Comedy|Drama|Romanc~
##  8      1     362      5 838984885 Jungle Book, The (~ Adventure|Children|~
##  9      1     364      5 838983707 Lion King, The (19~ Adventure|Animation~
## 10      1     370      5 838984596 Naked Gun 33 1/3: ~ Action|Comedy
## # ... with 9,000,045 more rows
```

Following shows the number of unique users in the dataset that provided ratings and the number of unique movies that were rated:

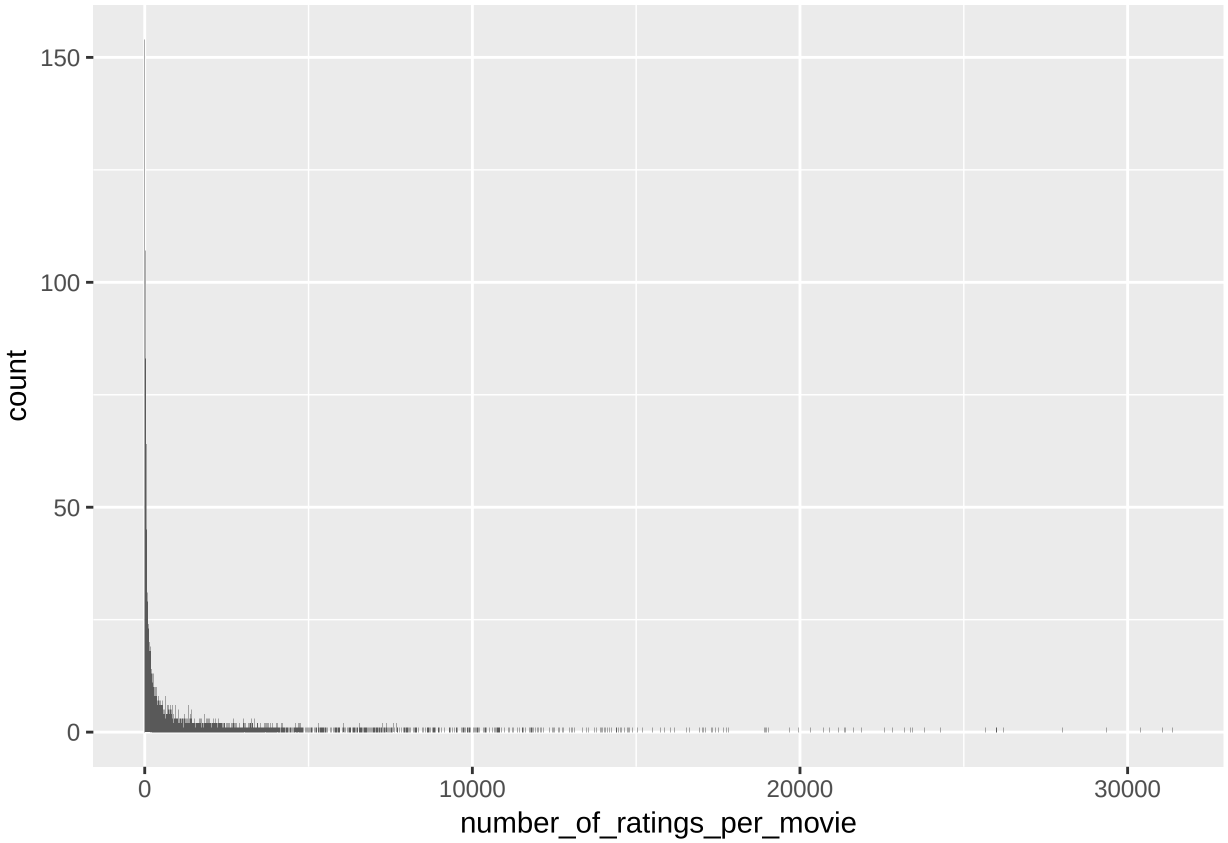
```
# Number of unique movies and number of unique users
edx %>%
  summarize(num_movies = n_distinct(movieId),
            num_users = n_distinct(userId))
```

```
##   num_movies num_users
## 1      10677     69878
```

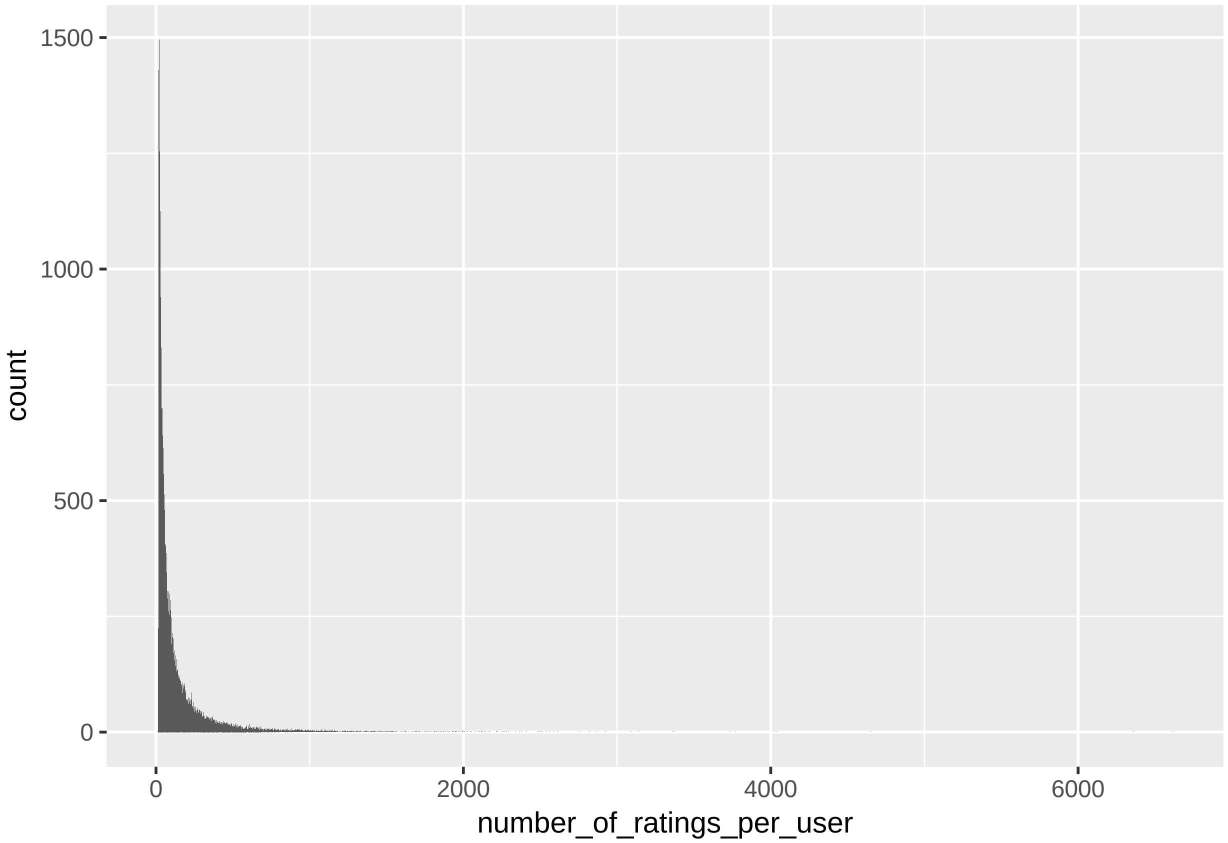
Following is the rating distribution of the movielens data.



When we look at the movie rating distribution we can see that some movies get rated more than others. This is expected as there are popular movies watched by millions while some other independent movies were watched by just a few.



We also look at the user rating distribution and we can see that some users are more active than others at rating movies:



2.2 Performance Evaluation - RMSE

One of the typical metrics to measure the effectiveness of the recommendation system is root mean squared error (RMSE). RMSE is defined as the square root of the average square error between the true rating and predicted rating.

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

It is used to evaluate how close the predictions are to the true values in the validation set: the smaller the error, the better the recommendation system is; if the error is larger than 1 star, then the recommendation system is not good. The R code for RMSE function is written as below:

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

2.3 Models for Recommendation System

2.3.1 Just the Average Model

The simplest possible recommendation system is predicting the same rating for all movies. This model assumes the same rating for all movies and users and that all the differences were explained by random variation ($\varepsilon_{u,i}$). The model can be written as follows:

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

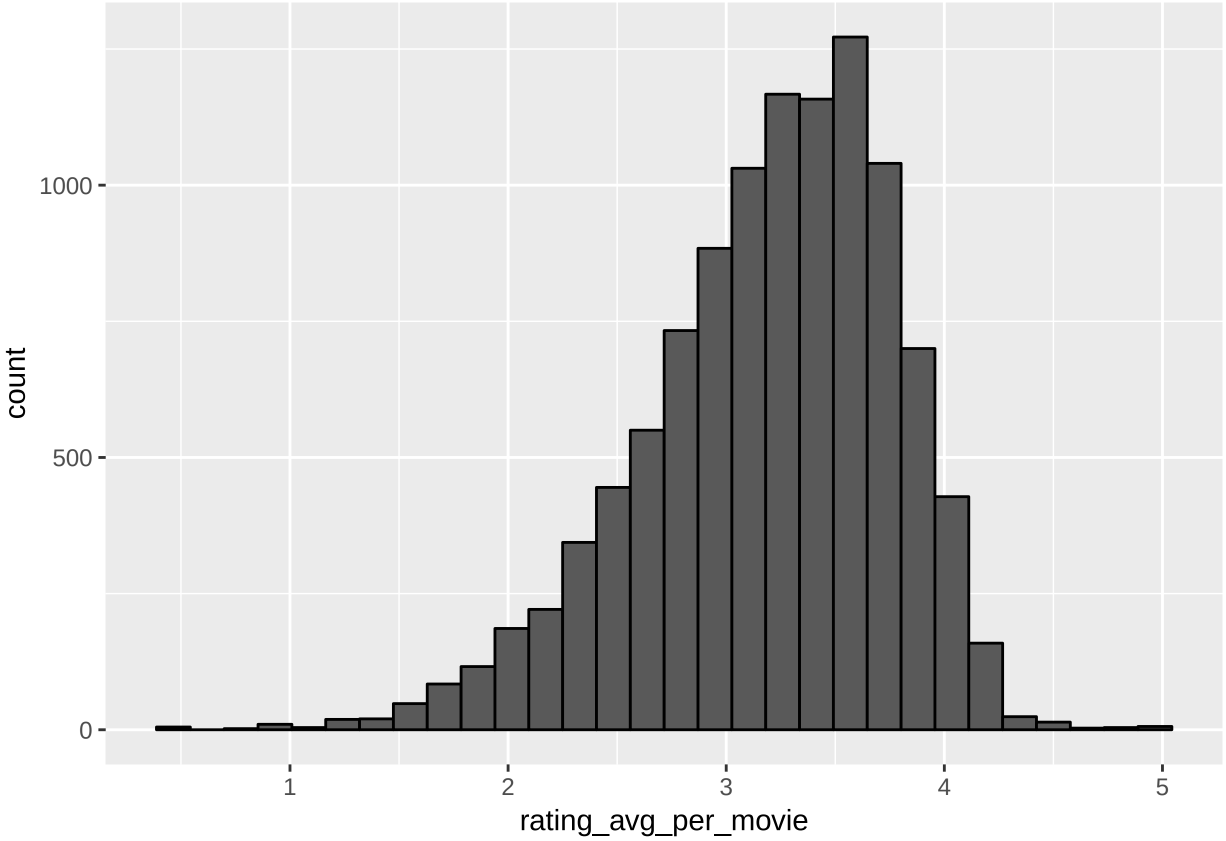
In this model, the estimated rating (μ) is the average of all ratings, which is the least squares estimate that minimizes RMSE. Here we obtained the naive RMSE of over 1, which means the average model is not performing well in predicting the movie ratings.

```
mu_hat <- mean(edx$rating)
predicted_ratings_1 <- mu_hat
```

2.3.2 Movie Effect Model

We know that the average rating for each movie will be quite different and it is confirmed by the following histogram of average rating for each movie that has been rated by no less than 100 users.

```
# Distribution of average rating per movie
edx %>%
  group_by(movieId) %>%
  summarize(rating_avg_per_movie = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(rating_avg_per_movie)) +
  geom_histogram(bins = 30, color = "black")
```



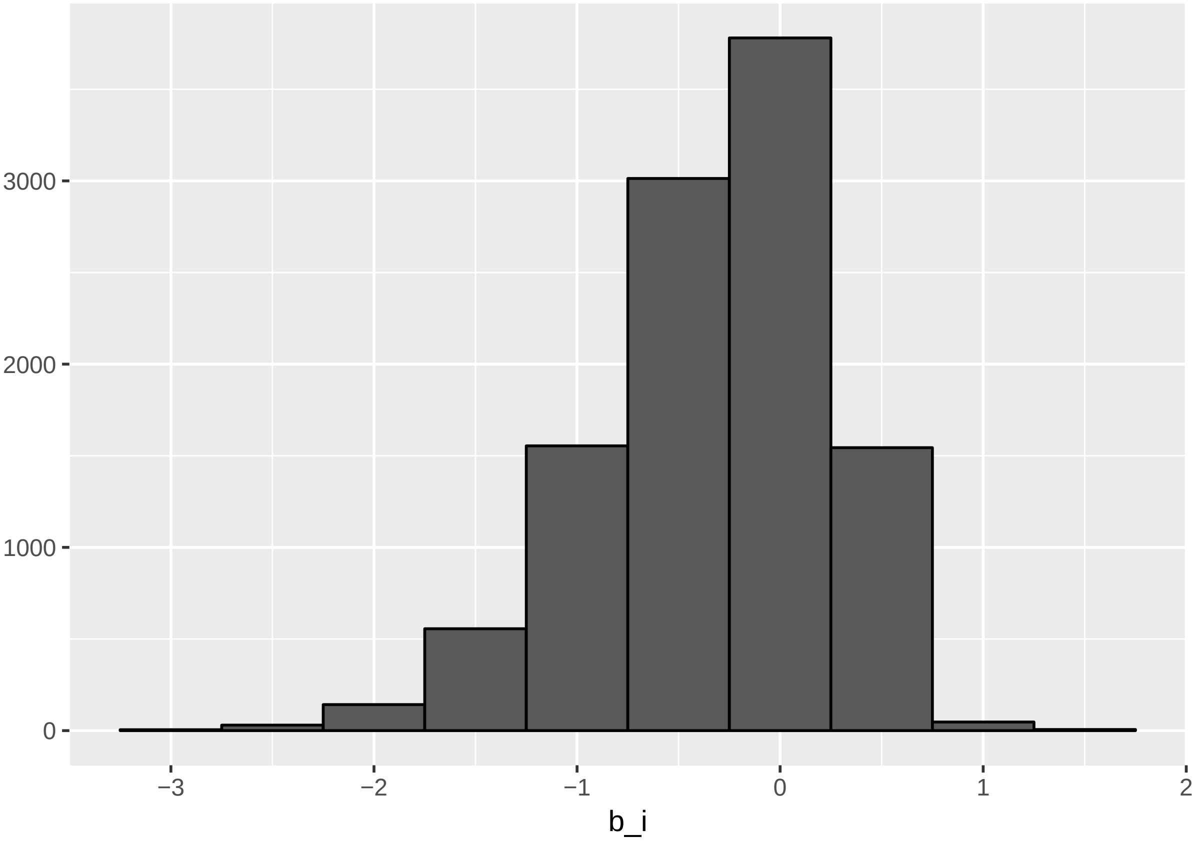
As different movies are rated differently, we can add this movie effect to our first average model and the model is written as below, where a term b_i is added to represent the average rating effect of movie i.

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

We can calculate the least square estimate of b_i as the average of $Y_{u,i} - \hat{\mu}$ for each movie i.

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

The histogram shows the distribution of average rating effect of each movie: for example, a negative value of b_i means the predited rating of the movie is lower than the average rating of all movies.



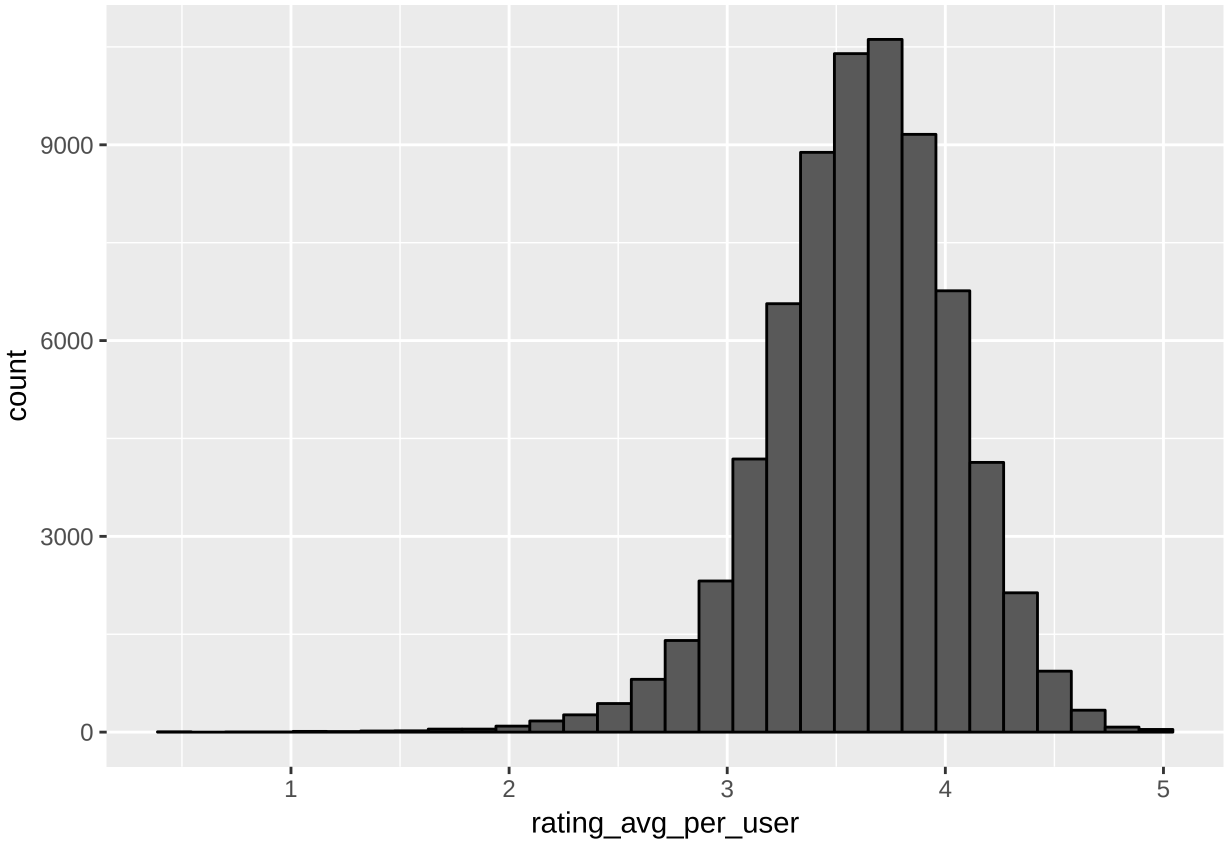
The predicted ratings considering movie effects are calculated as $\hat{\mu} + \hat{b}_i$:

```
# predicted ratings considering movie effects
predicted_ratings_2 <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
```

2.3.3 Movie and User Effect Model

Following is the distribution of the average rating for user u that have rated over 100 movies:

```
# Distribution of average rating per user
edx %>%
  group_by(userId) %>%
  summarize(rating_avg_per_user = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(rating_avg_per_user)) +
  geom_histogram(bins = 30, color = "black")
```

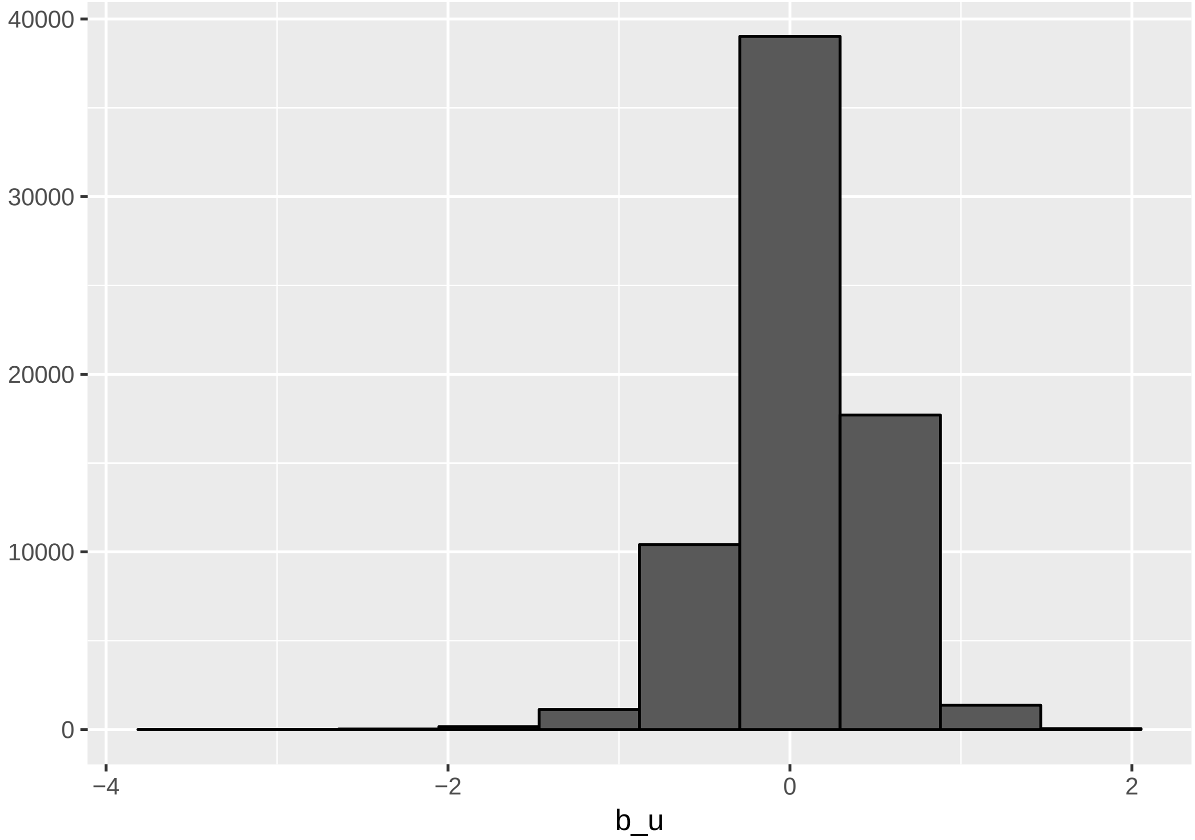


As there are variability across users and we can further improve the model by incorporating the user effect b_u:

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

In this model, the user-specific effect will adjust the predicted rating: negative b_u means user u is usually rate lower for a specific movie i. We can calculate the estimate of b_u as the average of $y_{u,i} - hat\mu - hatb_i$:

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

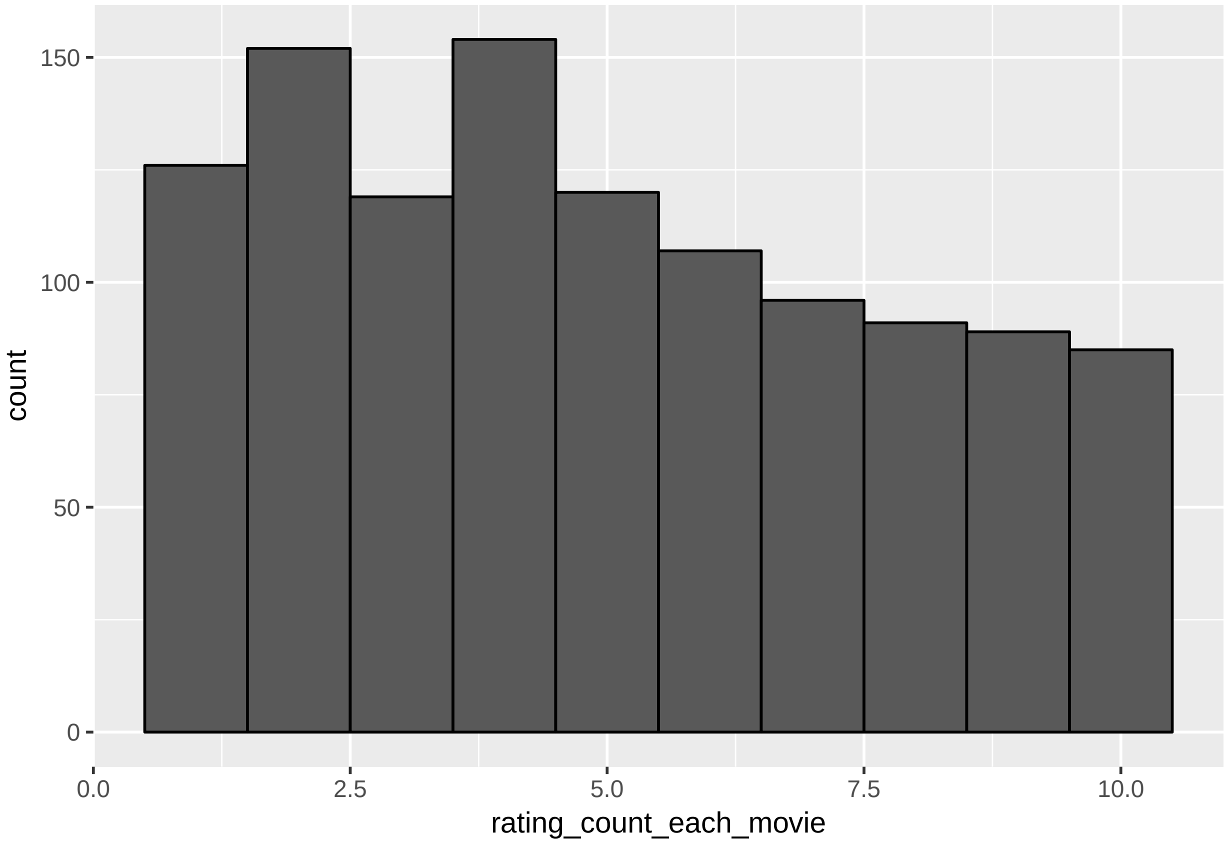


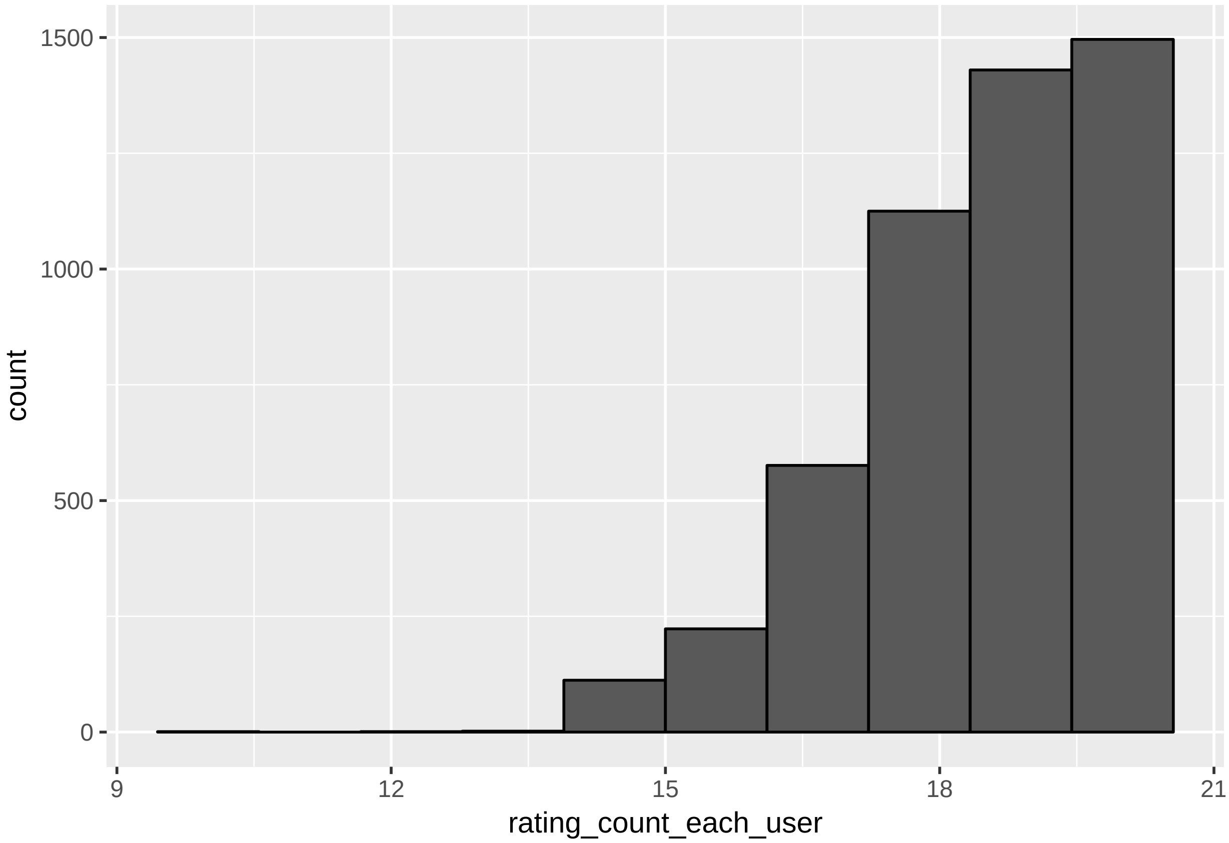
The predicted ratings considering both movie effects and user effects are calculated as $\hat{\mu} + \hat{b_i} + \hat{b_u}$:

```
# Predicted ratings considering both movie effects and user effects
predicted_ratings_3 <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

2.3.4 Regularized Movie and User Effect Model

In the previous movie and user effect models we ingore one of the possible problems: some movies have very few user ratings (as shown in the following histogram) and thus using the average movie effect b_i, which is based on the few observations, as the estimate is questionable. Similarly, some users only rated several movies (e.g., fewer than 15) and using the average user-specific effect b_u is also problematic.


150
100
50
0
count
0.0
2.5
5.0
7.5
10.0
rating_count_each_movie


1500
1000
500
0
count
9
12
15
18
21
rating_count_each_user

The idea of regularization is to penalize the effect sizes of ratings for movies and/or by users. We add a penalty term to the lease square equation, which gets larger when b_i and/or b_u are large:

$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i-b_u)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2)$$

The values that minize this equation are:

$$\hat{b_i}(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu})$$

$$\hat{b_u}(\lambda)=\frac{1}{\lambda+n_u}\sum_{i=1}^{n_u}(Y_{u,i}-\hat{\mu}-\hat{b_i})$$

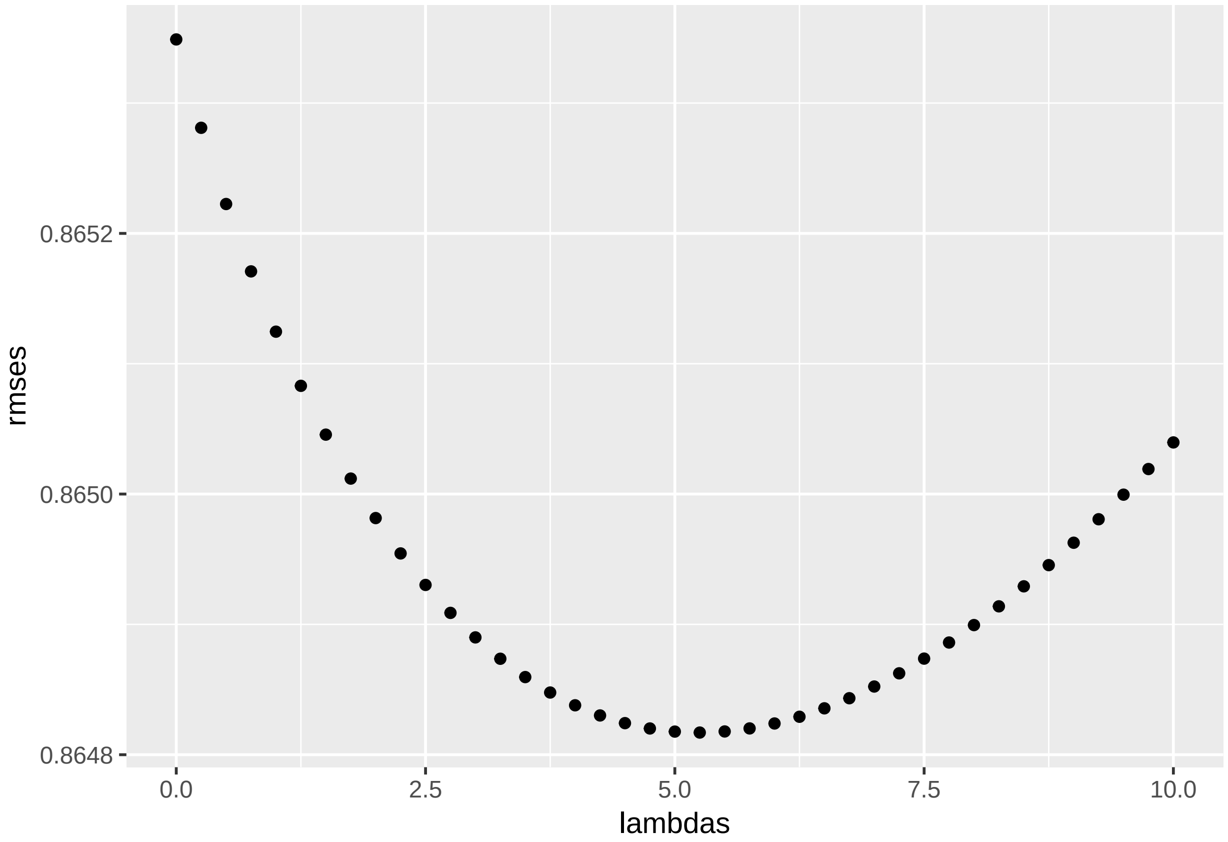
Here we can see that when the sample size is very large, the penalty is effectively ignored since $\lambda+n_i\approx n_i$ or $\lambda+n_u\approx n_u$. When the sample size is small, then the larger the parameter λ, the more the estimated $\hat{b_i}(\lambda)$ or $\hat{b_u}(\lambda)$ shrink towards 0.

Following shows the method we used to find the optimal tuning parameter λ which produces the smallest root mean square error.

```r
# Choosing the tuning parameter for regularized movie + user effect model
lambdas <- seq(0, 10, 0.25)

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))
})

qplot(lambdas, rmses)
```



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

```
## [1] 5.25
```

We then used the optimal tuning parameter λ to calculate the predicted rating as follows:

```
# Regularized Movie + User Effect Model using the optimal lambda
lambda <- 5.25
mu <- mean(edx$rating)
movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda)) #, n_i = n())
user_reg_avgs <- edx %>%
  left_join(movie_reg_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

# predicted rating for regularized movie and user effects model
predicted_ratings_4 <- validation %>%
  left_join(movie_reg_avgs, by = "movieId") %>%
  left_join(user_reg_avgs, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

3. Results

In the previous methodology section we have calculated the predicted ratings for the validation dataset based on four models: the naive average model, movie effect model, movie + user effecf model, and the regularized movie + user effect model. In this section, we compute the RMSE for each model and creating the results table as follows:

```
rmse_model_1 <- RMSE(predicted_ratings_1, validation$rating)
rmse_results <- tibble(Method = "Just the average",
                           RMSE = rmse_model_1)

rmse_model_2 <- RMSE(predicted_ratings_2, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Movie Effect Model",
                                     RMSE = rmse_model_2))

rmse_model_3 <- RMSE(predicted_ratings_3, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Movie + User Effects Model",
                                     RMSE = rmse_model_3))

rmse_model_4 <- RMSE(predicted_ratings_4, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Regularized Movie + User Effect Model",
                                     RMSE = rmse_model_4))

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

Method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie + User Effect Model	0.8648170

Just as we expected, the results shows that the naive average model has a RMSE of over 1, performing the worst; after considering the movie effects and user effects, the RMSEs improve significantly; the regularized movie + user effect model performs best and has the lowest RMSE of 0.8648. However, comparing the regularized model with the non-regularized movie + user effect model, the improvement in the RMSE is not so significant.

4. Conclusion

In this project we created a movie recommendation system using the regression models in machine learning techniques. We improve the predictions of movie ratings by modeling the movie and user effects and the sample size effects (regularization). The final RMSE from our model is 0.8648, achieved the goal of RMSE <= 0.87750 for this project.

References

Rafael A. Irizarry; Introduction to Data Science - Data Analysis and Prediction Algorithms with R; 2019-04-22 https://rafalab.github.io/dsbook/