

Capstone MovieLens Project

Yu Shing Lui

11 June 2020

1.Introduction

The MovieLens project is using the MovieLens 10M dataset to predict movie ratings in the validation set which containing part of MovieLens data. The main purpose of this project about the method to create and test the algorithm and predict movie ratings using the validation set by RMSE.

The key steps performed as shown below:

1.1 MovieLens 10M dataset.

```
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")

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")
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")
```

1.2 Validation set will be 10% of MovieLens data

```
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,]
```

1.3 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")
```

1.4 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)
```

2.Data exploration

2.1 Find overall data in the dataset.

```
head(edx)
```

```
##      userId movieId rating timestamp      title
## 1:      1      122      5 838985046      Boomerang (1992)
## 2:      1      185      5 838983525      Net, The (1995)
## 3:      1      292      5 838983421      Outbreak (1995)
## 4:      1      316      5 838983392      Stargate (1994)
## 5:      1      329      5 838983392 Star Trek: Generations (1994)
## 6:      1      355      5 838984474      Flintstones, The (1994)
##
##              genres
## 1:      Comedy|Romance
## 2:      Action|Crime|Thriller
## 3: Action|Drama|Sci-Fi|Thriller
## 4:      Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:      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
##
##
##
```

2.2 In the edx set, there are 9000055 rows and 6 columns.

```
dim(edx)
```

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

2.3 No movies have a rating of 0 and 2121240 movies have a rating of 3 can be found.

```
edx %>% filter(rating == 0) %>% tally()
```

```
##      n
## 1 0
```

```
edx %>% filter(rating == 3) %>% tally()
```

```
##      n
## 1 2121240
```

2.4 The edx dataset is including 10677 movies and 68978 users.

```
data.frame(Movies=n_distinct(edx$movieId), Users=n_distinct(edx$userId))
```

```
##      Movies Users
```

```
## 1 10677 69878
```

2.5 The number of movie rating are in each of the following genres in the edx dataset.

```
edx %>% separate_rows(genres, sep = "\\|") %>%  
  group_by(genres) %>% summarize(count = n()) %>% arrange(desc(count))
```

```
## # A tibble: 20 x 2  
##   genres      count  
##   <chr>      <int>  
## 1 Drama      3910127  
## 2 Comedy     3540930  
## 3 Action     2560545  
## 4 Thriller   2325899  
## 5 Adventure  1908892  
## 6 Romance    1712100  
## 7 Sci-Fi     1341183  
## 8 Crime      1327715  
## 9 Fantasy     925637  
## 10 Children  737994  
## 11 Horror     691485  
## 12 Mystery    568332  
## 13 War        511147  
## 14 Animation  467168  
## 15 Musical    433080  
## 16 Western    189394  
## 17 Film-Noir  118541  
## 18 Documentary 93066  
## 19 IMAX       8181  
## 20 (no genres listed) 7
```

2.6 It shows the top ten movies of rating and the movie “Pulp Fiction” has the greatest number of ratings.

```
edx %>% group_by(movieId, title) %>% summarize(count = n()) %>% arrange(desc(count))
```

```
## # A tibble: 10,677 x 3  
## # Groups:   movieId [10,677]  
##   movieId title      count  
##   <dbl> <chr>      <int>  
## 1 296 Pulp Fiction (1994) 31362  
## 2 356 Forrest Gump (1994) 31079  
## 3 593 Silence of the Lambs, The (1991) 30382  
## 4 480 Jurassic Park (1993) 29360  
## 5 318 Shawshank Redemption, The (1994) 28015  
## 6 110 Braveheart (1995) 26212  
## 7 457 Fugitive, The (1993) 25998  
## 8 589 Terminator 2: Judgment Day (1991) 25984  
## 9 260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672  
## 10 150 Apollo 13 (1995) 24284  
## # ... with 10,667 more rows
```

2.7 The top five ratings are shown as below, which rating of 4 is the most and 2 is the least given ratings.

```
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5) %>% arrange(desc(count))
```

```
## # A tibble: 5 x 2  
##   rating count  
##   <dbl> <int>
```

```
## 1    4    2588430
## 2    3    2121240
## 3    5    1390114
## 4    3.5  791624
## 5    2    711422
```

2.8 The following code is shown that half star ratings are less common than whole star ratings.

```
edx %>% group_by(rating) %>% summarize(count = n())
```

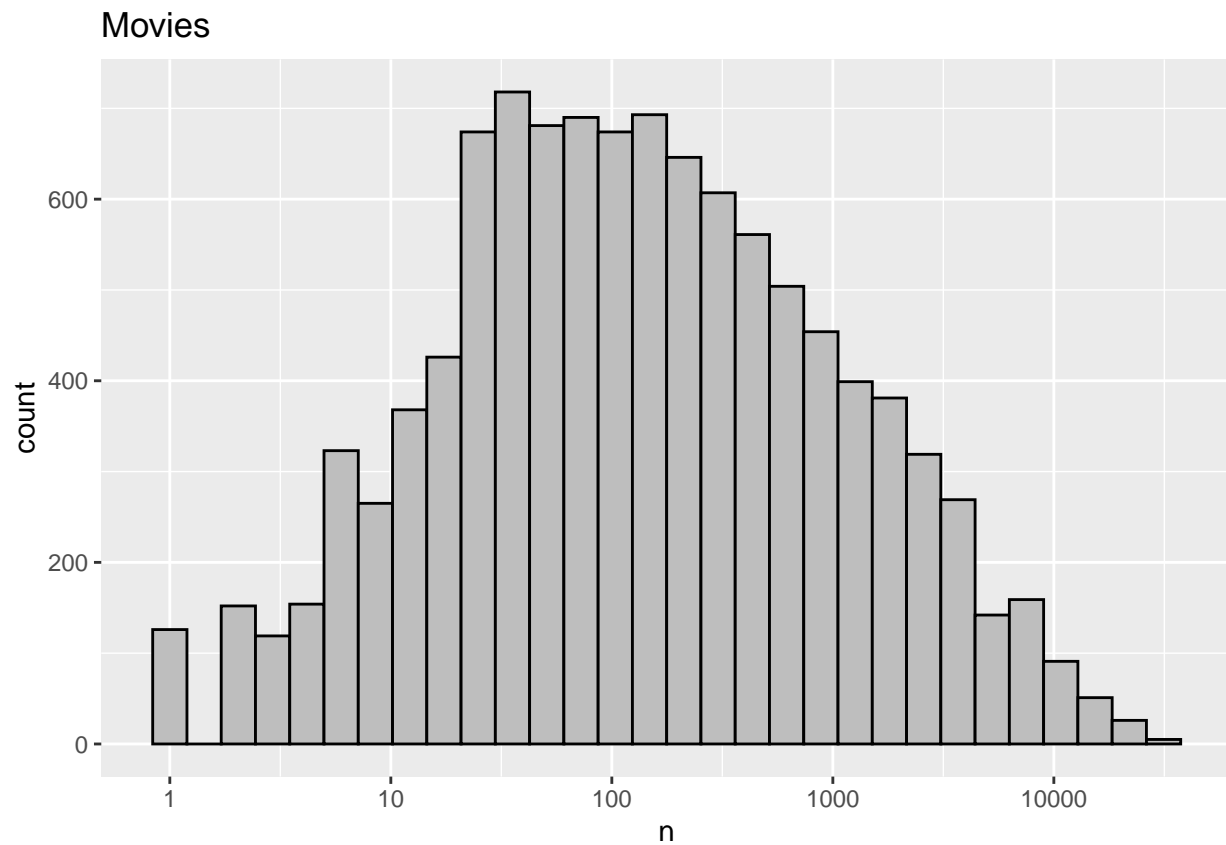
```
## # A tibble: 10 x 2
##   rating count
##   <dbl> <int>
## 1  0.5   85374
## 2  1     345679
## 3  1.5  106426
## 4  2     711422
## 5  2.5  333010
## 6  3     2121240
## 7  3.5  791624
## 8  4     2588430
## 9  4.5  526736
## 10 5     1390114
```

The basic information is shown above and we start to analyze the information in the following section.

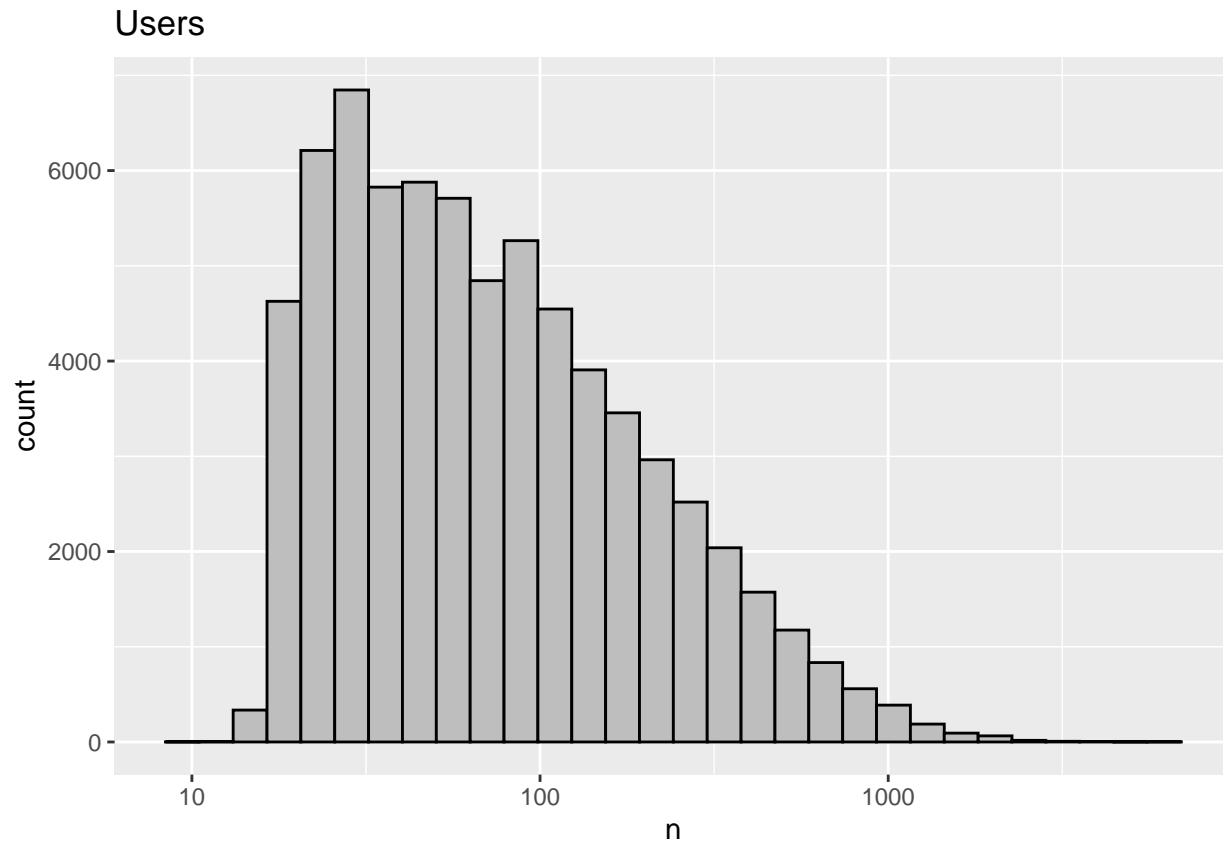
3. Analysis and Result

3.1 The general properties of the Movie and User data.

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



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



3.2 Computing the RMSE for vectors of ratings and their corresponding predictors.

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

3.3 Creating the average of all ratings model.

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

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

3.4 Predicting all unknown ratings with mu we obtain the following RMSE

```
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse
```

```
## [1] 1.061202
```

3.5 Creating a results table with this naive approach

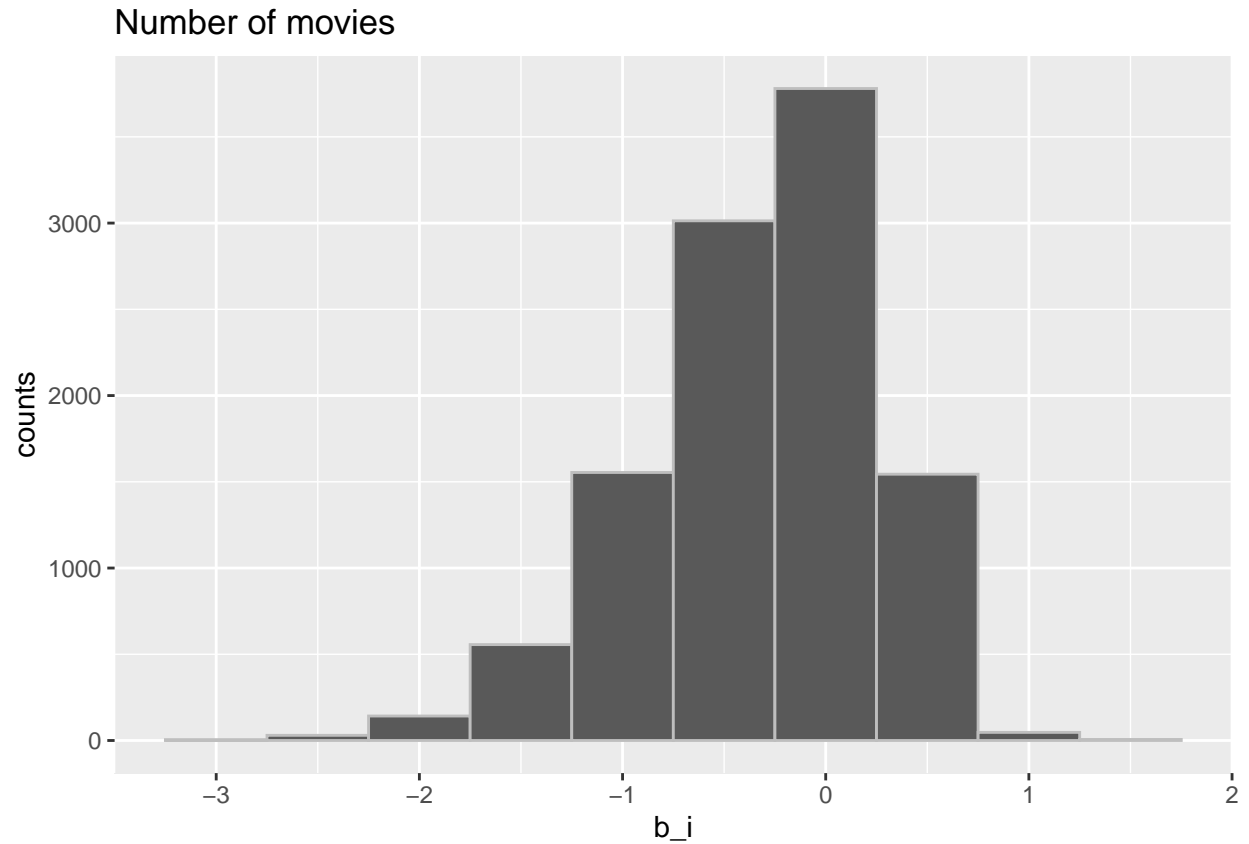
```
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)
rmse_results
```

```
## # A tibble: 1 x 2
##   method      RMSE
##   <chr>      <dbl>
## 1 Just the average 1.06
```

The result of this model of the average rating is 3.51, and RMSE is 1.06. So we can definitely do better!

3.6 Now we create movie effects model.

```
movie_avgs <- edx %>%  
  group_by(movieId) %>%  
  summarize(b_i = mean(rating - mu_hat))  
  
movie_avgs %>% qplot(b_i, geom = "histogram", bins = 10, data = ., color = I("grey"),  
  ylab = "counts", main = "Number of movies")
```



3.7 The predict result of movie effects model.

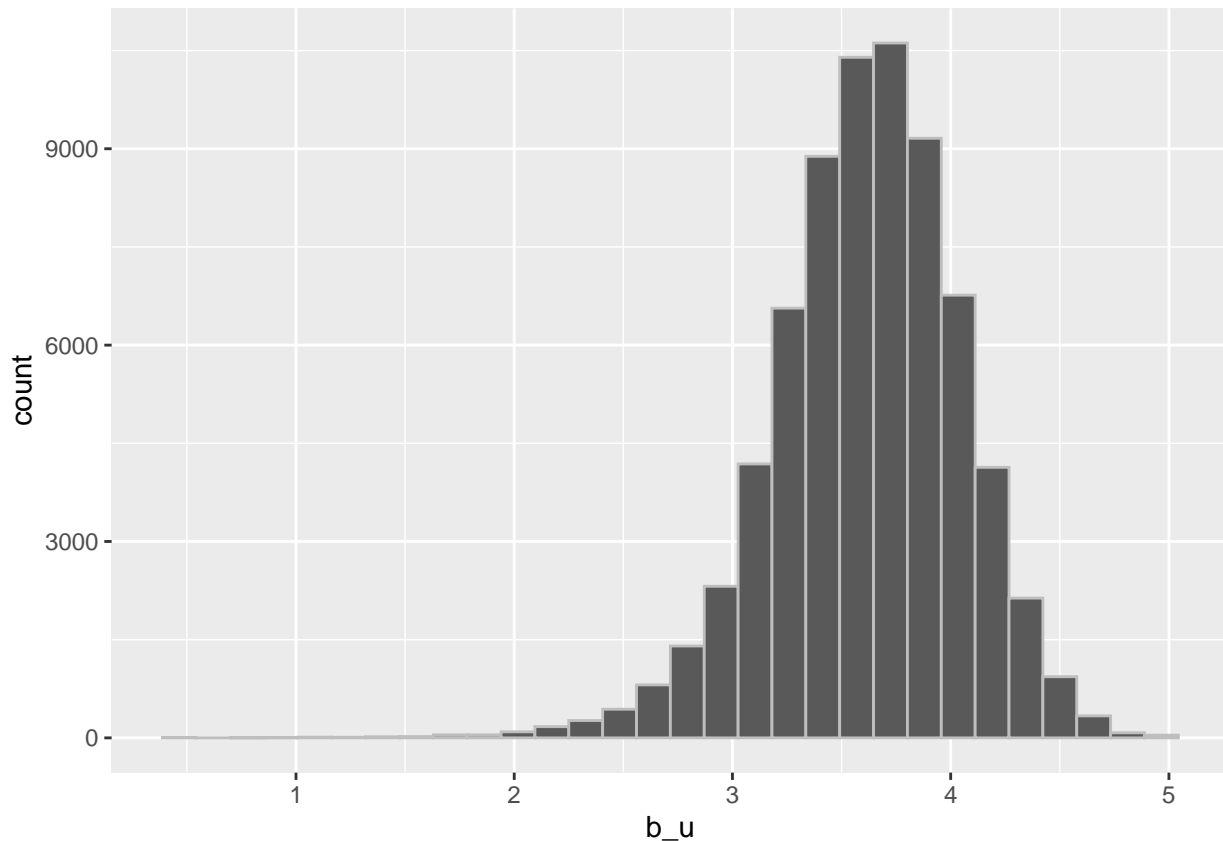
```
predicted_ratings <- mu_hat + validation %>%  
  left_join(movie_avgs, by='movieId') %>% pull(b_i)  
  
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.9439087
```

The predict result is 0.94 and we believe that we can do much better.

3.8 Creating user effect model and computing the average rating for user.

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



3.9 Computing an approximation by computing μ and b_i and estimating b_u as the average of $y_{u,i} - \mu - b_i$:

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

`summarise()` ungrouping output (override with `.groups` argument)

3.10 Constructing predictors and see how much the RMSE improves.

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

```
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8653488
```

The result of RMSE is 0.865.

Conclusion

As the result, the algorithm achieved a final RMSE value, which is fulfilled the requirement. It is a challenging project and learn a lot about the machine learning and build up the model to analysis the result.