MovieLens Report

Akram Zalloum
December 2021

Introduction

This recommendation system is used to predict the ratings that will be assigned to the movies based on the historical users' ratings. Based on the competition announced by Netflix in 2006 to solve a challenge in building a model used to enhance the prediction on users' ratings and predict what rate each movie will get. This happened and the model enhance the prediction by 10% which worth a lot of money, in 2009 winner announced who build this model. We will try to reflect this in below report.

Data set

The MovieLens Data set collected by GroupLens Research available in (http://movielens.org). The data will be split in training and testing sets where 10% will be used for validation and evaluation.

```r
##########################################################
# 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)
# 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 <- 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 3.6 or earlier:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(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")
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # 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")
# 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)
```

Understand data

Based on the data structure we have 6 columns (userId,movieID,rating,timestamp,title,genres).

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action
```

```
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
##
##
```

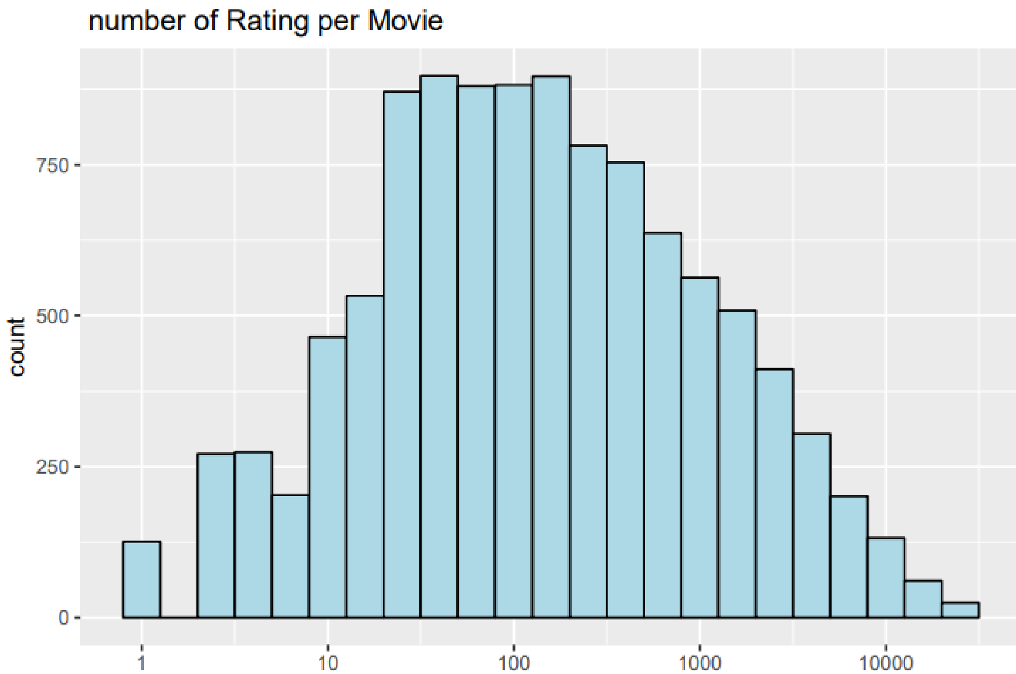
Regarding the ratings statistics we have Min = 1, Max = 5, Mean = 3.51 and Mode = 4.0.

```
## # 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
```

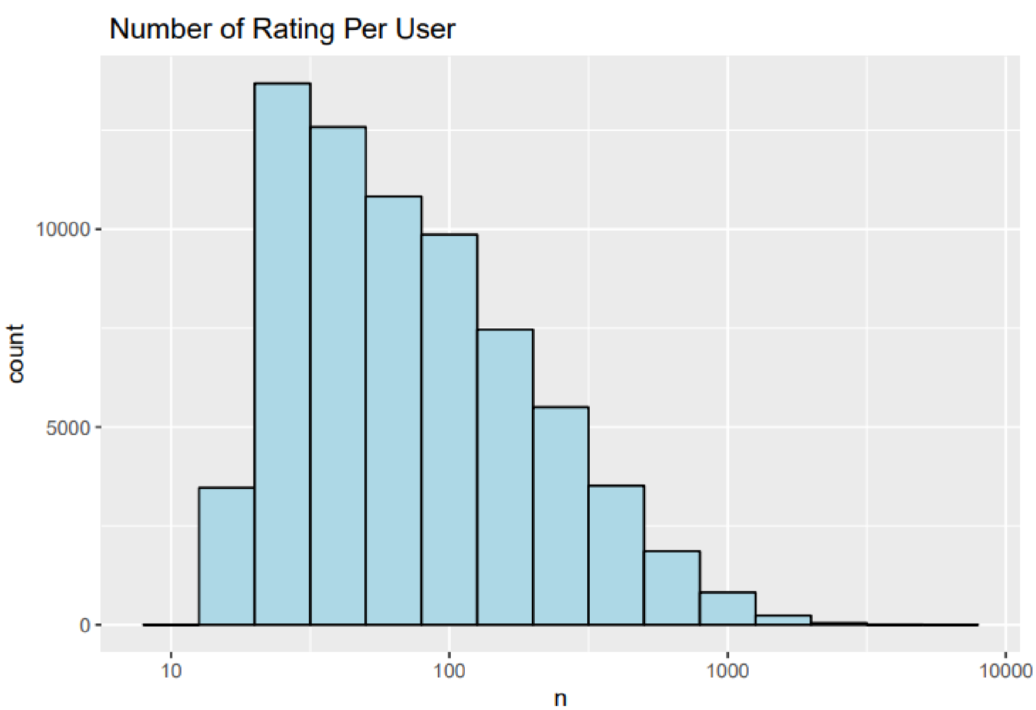
To get number of unique movies and users, we use the code:

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

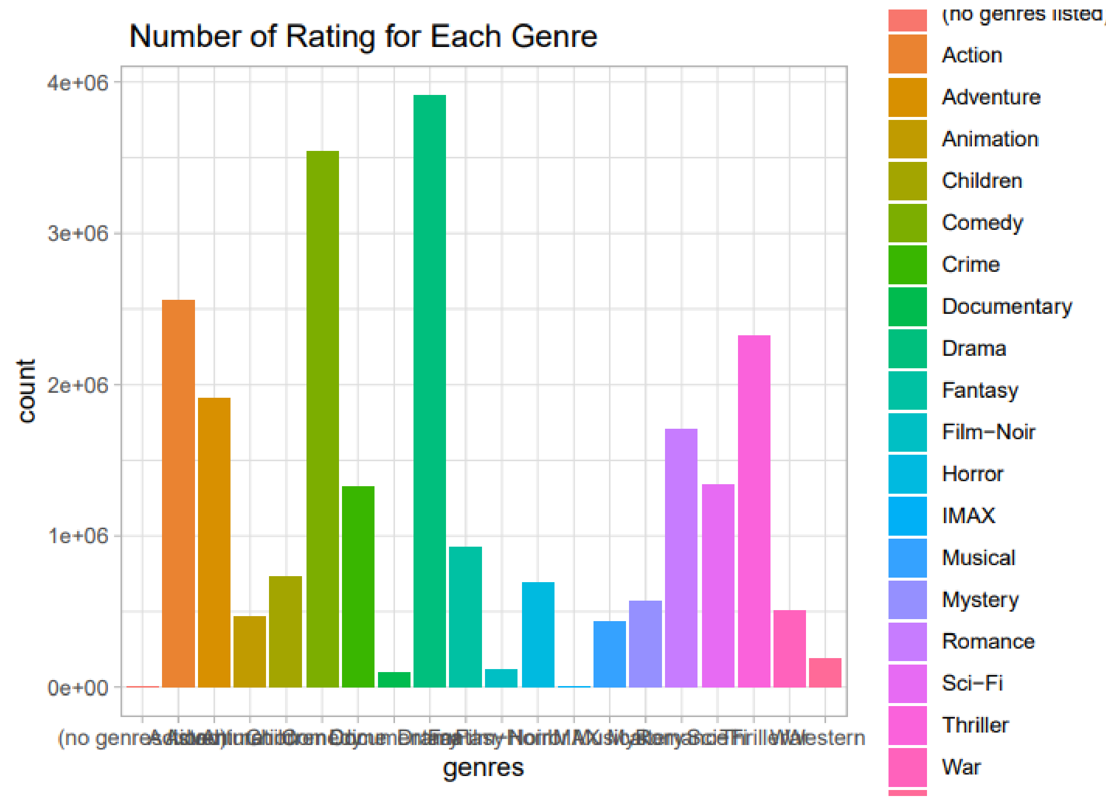
The histogram below to get number of ratings by movie



To get number of ratings by user



Visualization below to show the rating by movie genre



Will list below the top 10 genres

```
## # 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
```

Prepare training and testing data sets (80% to 20%)

```r
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

Calculate RMSE to measure the accuracy of the model.

```r
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2, na.rm = TRUE))
}
```

Model 1

Predict the same rating for all movies regardless of the user and without any constraints.

```r
Mu_1 <- mean(train_set$rating)
Mu_1
```

[1] 3.512482

```r
naive_rmse <- RMSE(test_set$rating,Mu_1)
naive_rmse
```

[1] 1.059909

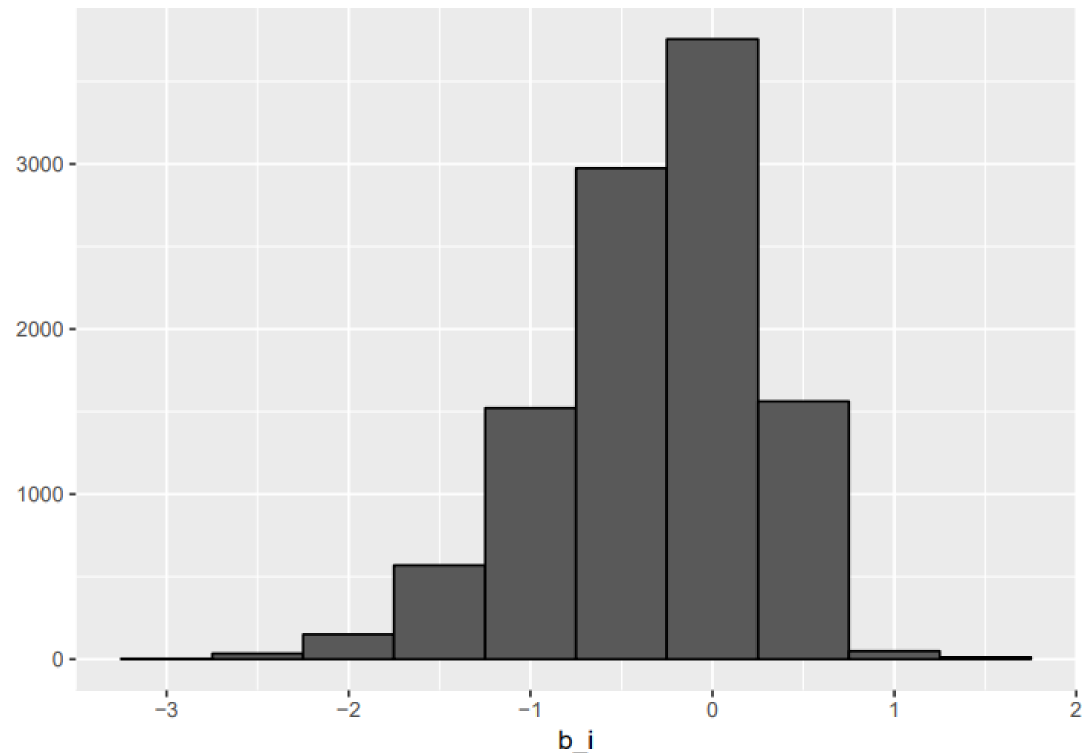
Get the RMSE and compare results.

```r
rmse_results <- data_frame(method = "The average", RMSE = naive_rmse)
rmse_results%>% knitr::kable()
```

Method	RMSE
The average	1.0599094

Model 2

Where some movies rated more than others and based on the previous model will add $b~i$ to represent the average ranking for movie i to show the biased, so we compute the average.

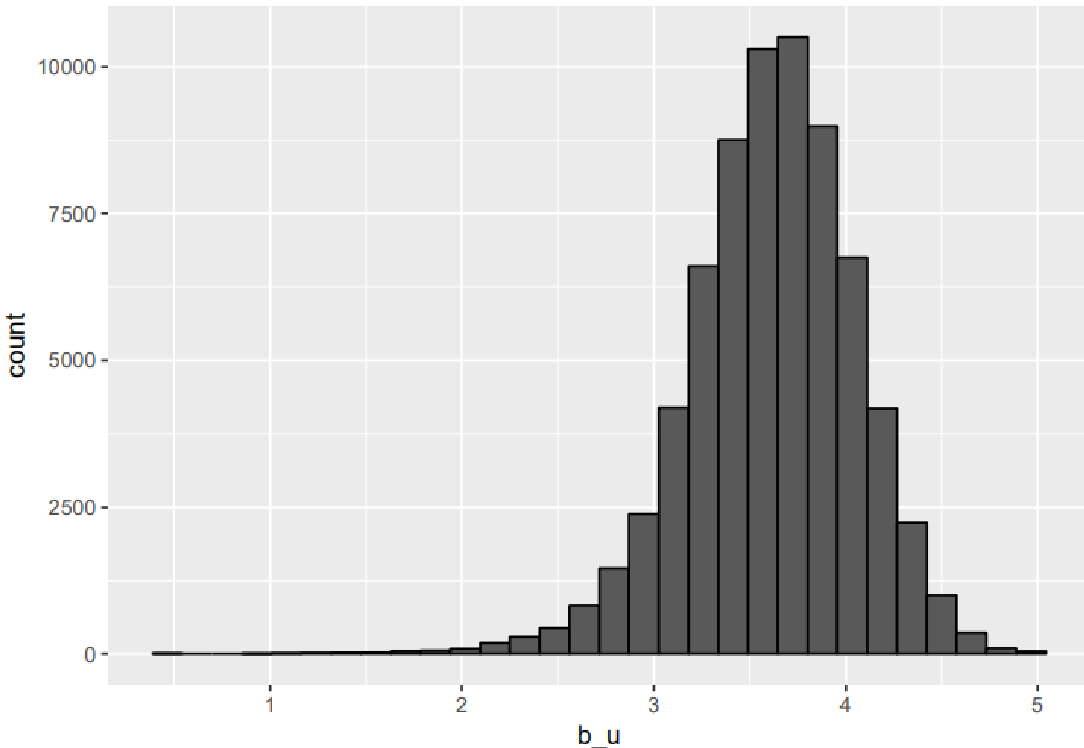



```
predicted_ratings <- Mu_2 + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
The average	1.0599094
Movie Effect Model	0.9437429

Model 3

Users who rate too many movies (>100)



Based on the clear gaps between ratings will get the mean

```
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - Mu_2 - b_i))
```

Will calculate RMSE again to see the enhancement

```
predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = Mu_2 + b_i + b_u) %>%
  pull(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))
rmse_results%>% knitr::kable()
```

Method	RMSE
The average	1.0599094
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659320

Apply RMSE for validation

```
valid_pred_rating <- validation %>%
  left_join(movie_avgs , by = "movieId" ) %>%
  left_join(user_avgs , by = "userId") %>%
  mutate(pred = Mu_2 + b_i + b_u ) %>%
  pull(pred)
model_3_valid <- RMSE(validation$rating, valid_pred_rating)
rmse_results <-  bind_rows( rmse_results, data_frame(Method = "Validation Results" , RMSE = model_3_valid))
rmse_results%>% knitr::kable()
```

Method	RMSE	Method
The average	1.0599094	NA
Movie Effect Model	0.9437429	NA
Movie + User Effects Model	0.8659320	NA
NA	0.8664515	Validation Results

Findings

Based on the methodology followed which is naive the 3rd model has best RMSE, after using the linear regression the findings that old movies have better prediction.