Movielens Report

by Lam Vu

5/16/2019

SUMMARY

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set.

In this exercise, MovieLens 10M dataset was used to build movie rating preditions. Then the predictions will be compared to the true ratings in the validation set using RMSE.

The **objectives** of this exercise are to gain insights with **movielens** dataset through exploration, and visualization, and modeling approach to achieve the **RMSE** <= 0.87750.

Dataset

The following code is used to generate datasets. Algorithm was develop using **edx** set. For a final test of the algorithm, predict movie ratings in the **validation** set as if they were unknown.

```
#### INTRODUCTION ####

###################################
# Create edx set and validation 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-proje
ct.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(movieI
d))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
#used caret pkg
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, l
ist = 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)
```

DATA ANALYSIS

Explorations that didn't lead to any insights are not included in this report.

```
#exploring edx dataset
class(edx)

## [1] "data.frame"

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
```

```
#number of rows and columns in edx dataset
dim(edx)

## [1] 9000055       6

#number of ratings were given as zero(0) or three(3)
edx %>% filter(rating == 0) %>% tally()

##   n
## 1 0

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

##         n
## 1 2121240
```

Number of distinct userIds and movieIds

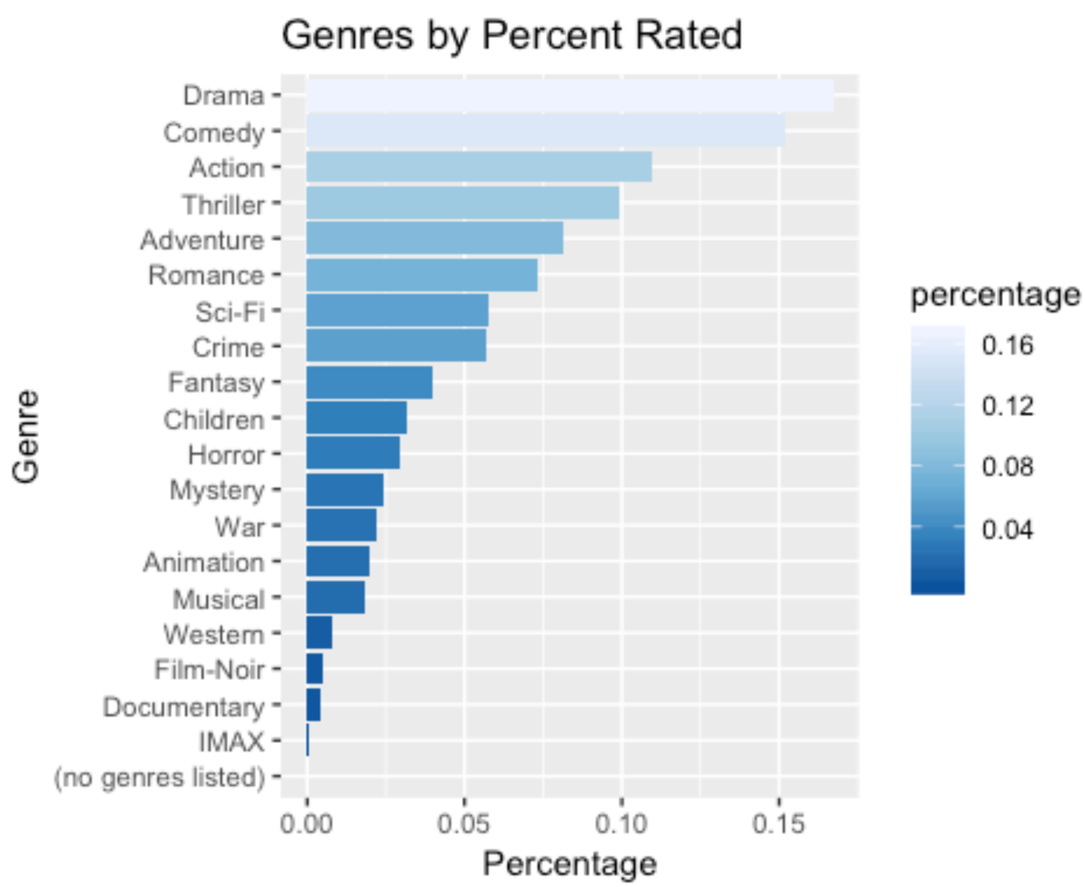
```
edx %>% summarize(n_users = n_distinct(userId),
                  n_movies = n_distinct(movieId))

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

Table of number of ratings by genres after separating combined genres.

```
#table of number of ratings by genres after separating combined genres
library(tidyr)
g_ratings <- edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n(), avg = mean(rating)) %>%
  arrange(desc(count))

## # A tibble: 20 x 3
##    genres               count   avg
##    <chr>                <int> <dbl>
##  1 Drama              3910127  3.67
##  2 Comedy             3540930  3.44
```

```
##   3 Action             2560545   3.42
##   4 Thriller           2325899   3.51
##   5 Adventure          1908892   3.49
##   6 Romance            1712100   3.55
##   7 Sci-Fi             1341183   3.40
##   8 Crime              1327715   3.67
##   9 Fantasy             925637   3.50
##  10 Children            737994   3.42
##  11 Horror              691485   3.27
##  12 Mystery             568332   3.68
##  13 War                 511147   3.78
##  14 Animation           467168   3.60
##  15 Musical             433080   3.56
##  16 Western             189394   3.56
##  17 Film-Noir           118541   4.01
##  18 Documentary          93066   3.78
##  19 IMAX                  8181   3.77
##  20 (no genres listed)       7   3.64
```

Distribution of Genres by Percent Rated



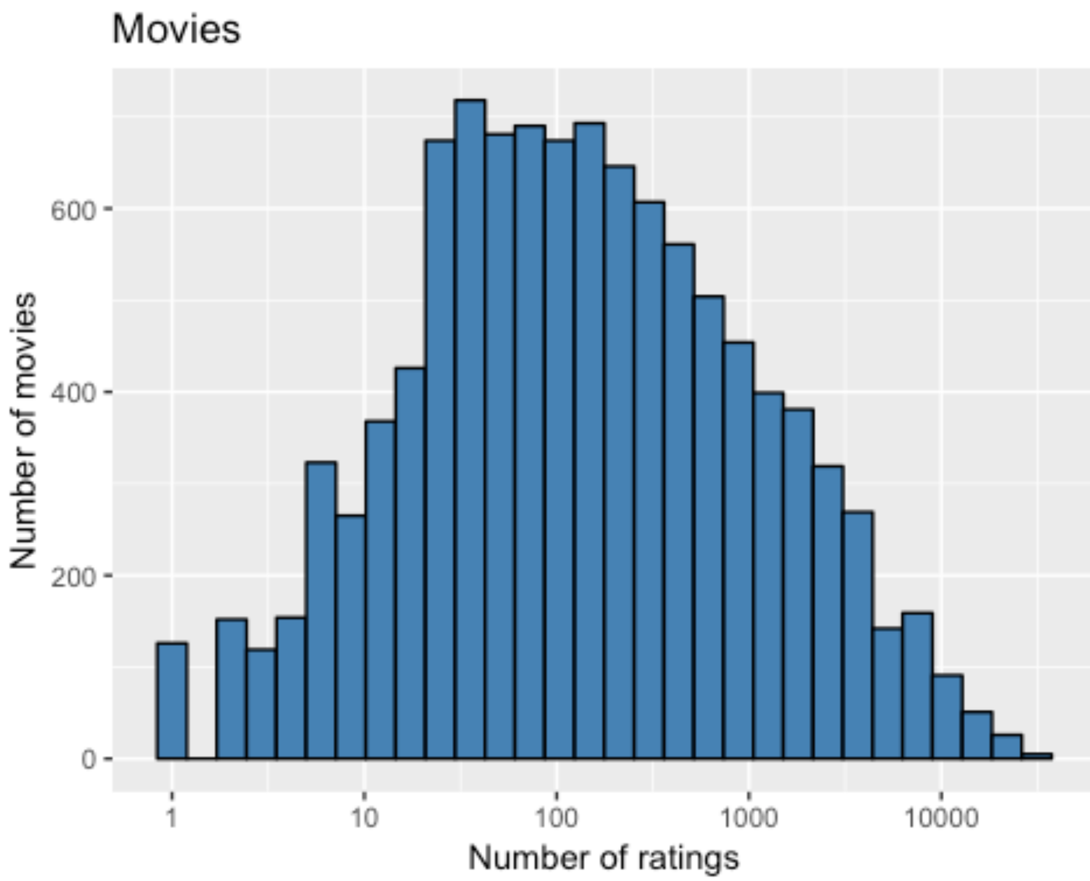
Here are the top 10 movie titles with the greatest number of ratings including average rating.

```
#top 10 movie titles with the greatest number of ratings
top_titles <- edx %>%
  group_by(movieId) %>%
  summarize(n = n(),
            title = title[1],
            avg = mean(rating)) %>%
  top_n(10, n) %>%
  arrange(desc(n))
top_titles
```

```
## # A tibble: 10 x 4
##    movieId     n title                                                   av
g
##      <dbl> <int> <chr>                                                 <dbl
>
##  1     296 31362 Pulp Fiction (1994)                                     4.1
5
##  2     356 31079 Forrest Gump (1994)                                     4.0
1
##  3     593 30382 Silence of the Lambs, The (1991)                        4.2
0
##  4     480 29360 Jurassic Park (1993)                                    3.6
6
##  5     318 28015 Shawshank Redemption, The (1994)                        4.4
6
##  6     110 26212 Braveheart (1995)                                       4.0
8
##  7     457 25998 Fugitive, The (1993)                                    4.0
1
##  8     589 25984 Terminator 2: Judgment Day (1991)                       3.9
3
##  9     260 25672 Star Wars: Episode IV - A New Hope (a.k.a. Star War…  4.2
2
## 10     150 24284 Apollo 13 (1995)                                        3.8
9
```

Distribution of Movies

```
#Distribution of Movies
edx %>% group_by(movieId) %>%
  summarize(n = n())%>%
  ggplot(aes(n)) +
  geom_histogram(fill = "steelblue", color = "black", bins = 30) +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of movies") +
  ggtitle("Movies")
```



Distribution of Users

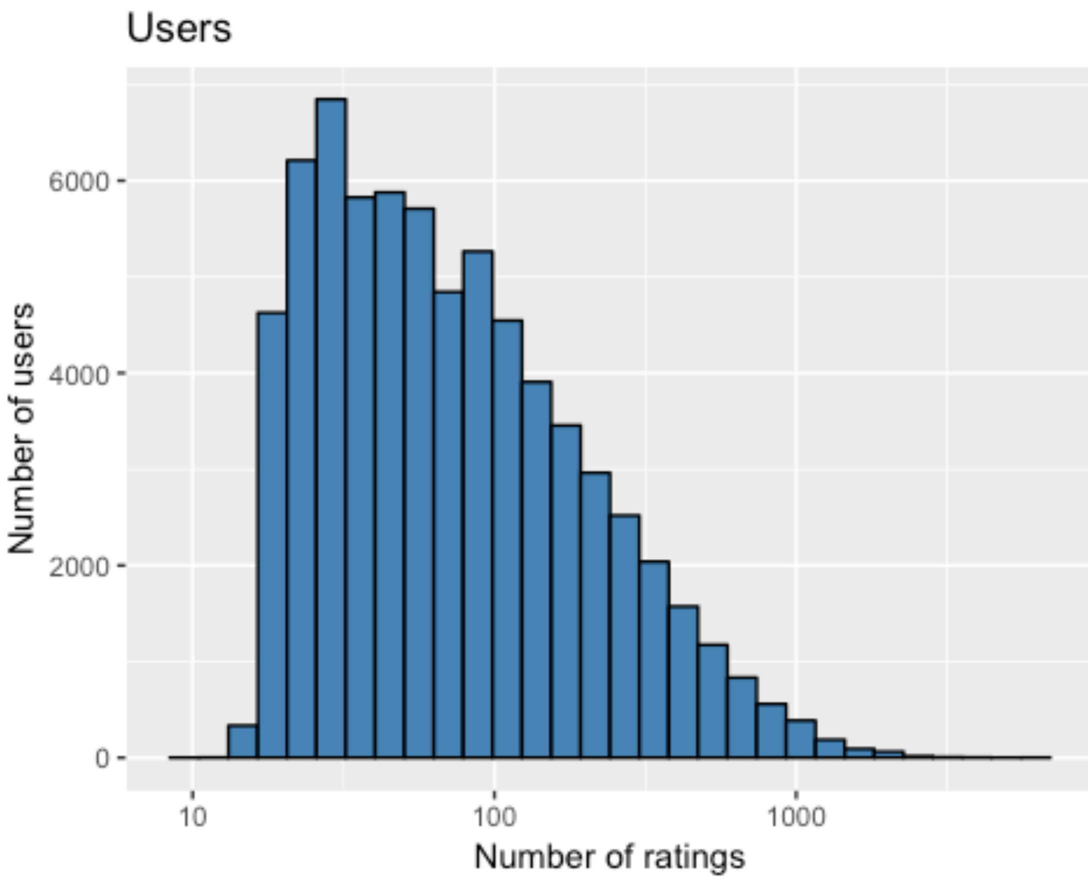
Top ten (10) users with the most ratings

```
#Top ten (10) users with the most ratings
edx %>% group_by(userId) %>% summarize(n = n())%>%top_n(10, n) %>%
  arrange(desc(n))
```

```
## # A tibble: 10 x 2
##    userId     n
##     <int> <int>
##  1  59269  6616
##  2  67385  6360
##  3  14463  4648
##  4  68259  4036
##  5  27468  4023
##  6  19635  3771
##  7   3817  3733
##  8  63134  3371
##  9  58357  3361
## 10  27584  3142
```

Distribution of users vs. ratings

```
#Distribution of Users
edx %>% group_by(userId) %>% summarize(n = n()) %>%
  ggplot(aes(n)) + geom_histogram(fill = "steelblue", color = "black", bins =
30) +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of users") +
  ggtitle("Users")
```



Top 10 movie titles with the greatest average rating

```
#top 10 movie titles with the greatest average rating
top_avg <- edx %>%
  group_by(movieId) %>%
  summarize(n = n(),
            title = title[1],
            avg = mean(rating)) %>%
  top_n(10, n) %>%
  arrange(desc(avg))
top_avg
```

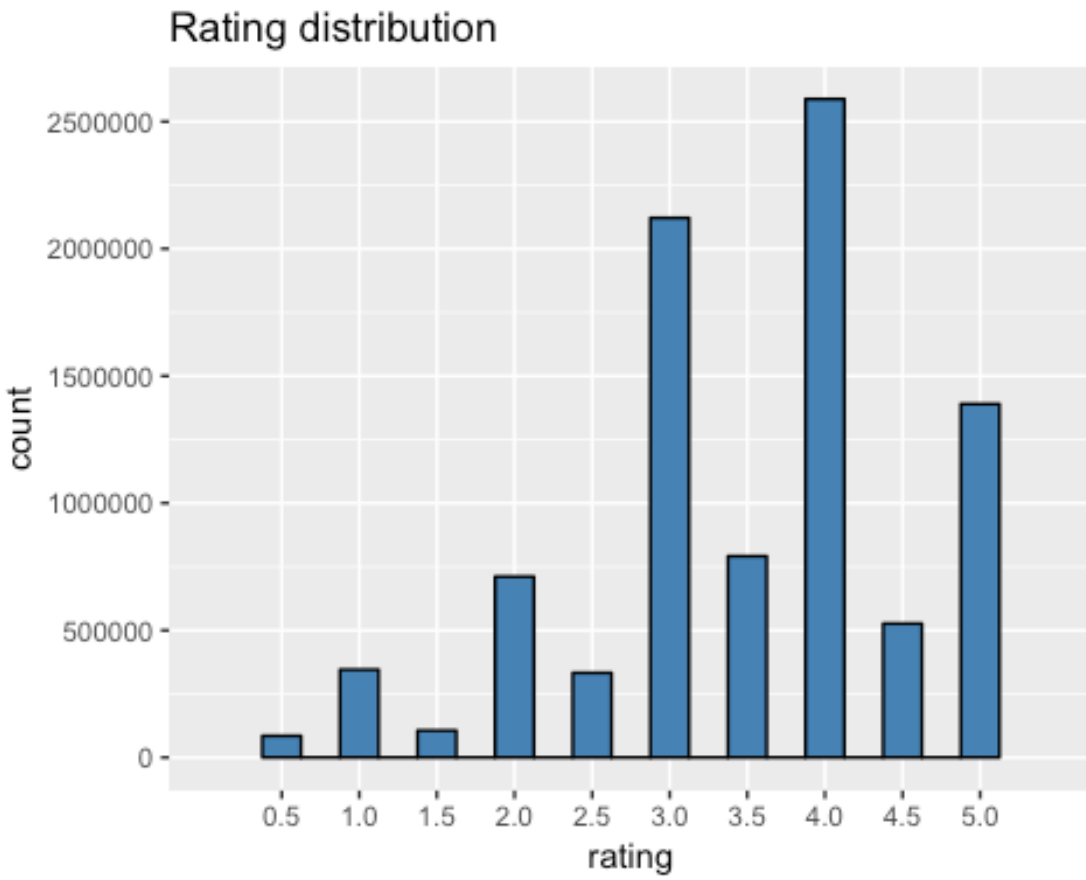
```
## # A tibble: 10 x 4
##    movieId     n title                                                      av
g
##      <dbl> <int> <chr>                                                    <dbl
>
##  1     318 28015 Shawshank Redemption, The (1994)                           4.4
6
##  2     260 25672 Star Wars: Episode IV - A New Hope (a.k.a. Star War…  4.2
2
##  3     593 30382 Silence of the Lambs, The (1991)                           4.2
0
##  4     296 31362 Pulp Fiction (1994)                                        4.1
5
##  5     110 26212 Braveheart (1995)                                          4.0
8
##  6     356 31079 Forrest Gump (1994)                                        4.0
1
##  7     457 25998 Fugitive, The (1993)                                       4.0
1
##  8     589 25984 Terminator 2: Judgment Day (1991)                          3.9
3
##  9     150 24284 Apollo 13 (1995)                                           3.8
9
## 10     480 29360 Jurassic Park (1993)                                       3.6
6
```

In general, half star ratings are less common than whole star ratings. This can be observed at the table and graph below.

```
#table of frequency of star ratings from most to least
rating_frequency <- edx %>% group_by(rating) %>%
  summarize(n = n()) %>%
  arrange(desc(n))
rating_frequency
```

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

Rating distribution



MODELS

Root-mean-square error (RMSE) is used to measure the differences between values predicted by a model and the values observed.

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

Simple Model

Predicting same rating for all movies across all users

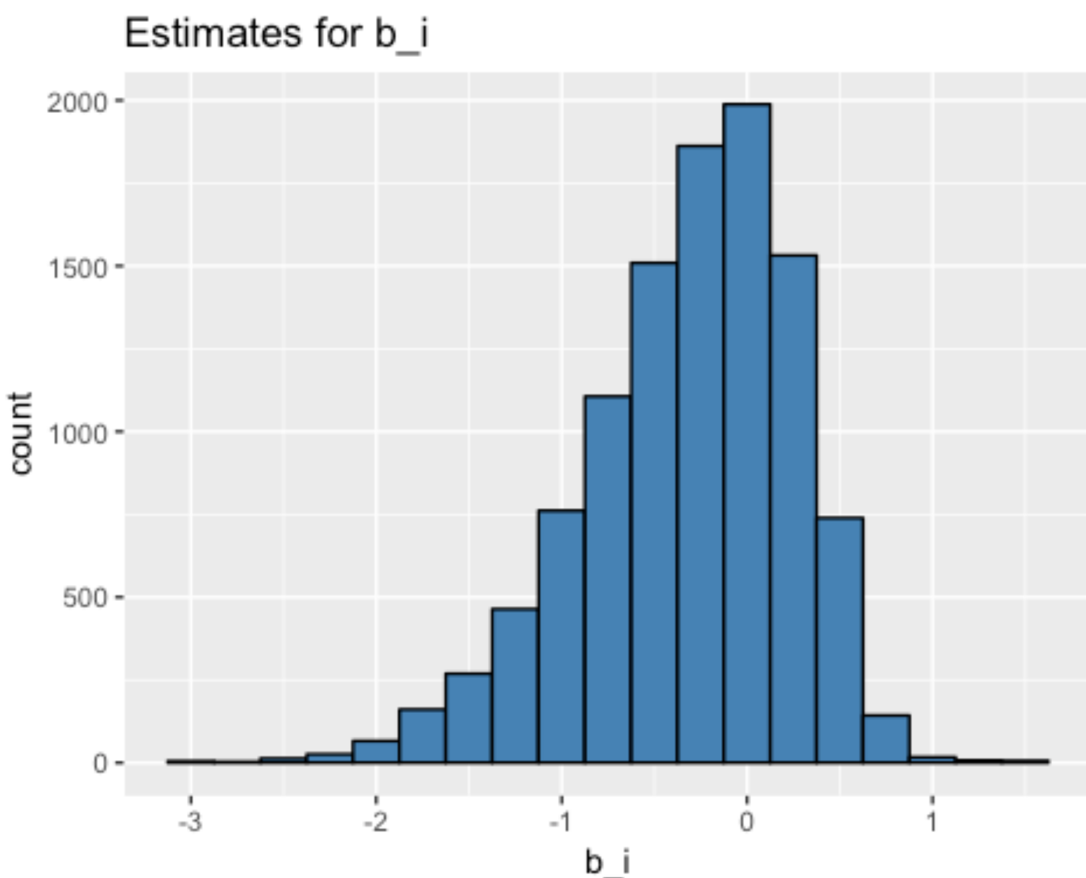
```
### Simplest Model: predict same rating for all movies across all users
#Yu,i = mu + Eu,i
mu <- mean(edx$rating)
mu

## [1] 3.512465

#test results based on simple prediction
naive_rmse <- RMSE(validation$rating, mu)
```

```
#create a table that's going to store the results
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.061202

Movie Effect Model

```
#Yu,i = mu + b_i + Eu,i where b_i = the average rating for movie i or as "bia
s
movie_avgs <- edx %>%
  group_by(movieId)%>%
  summarize(b_i = mean(rating - mu))

#plot movie averages b_i
movie_avgs %>%
  ggplot(aes(b_i )) +
  geom_histogram(fill = "steelblue",binwidth = .25, color = "black") +
  ggtitle("Estimates for b_i")
```



```
###Model
predicted_ratings <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i
model_1_rmse <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

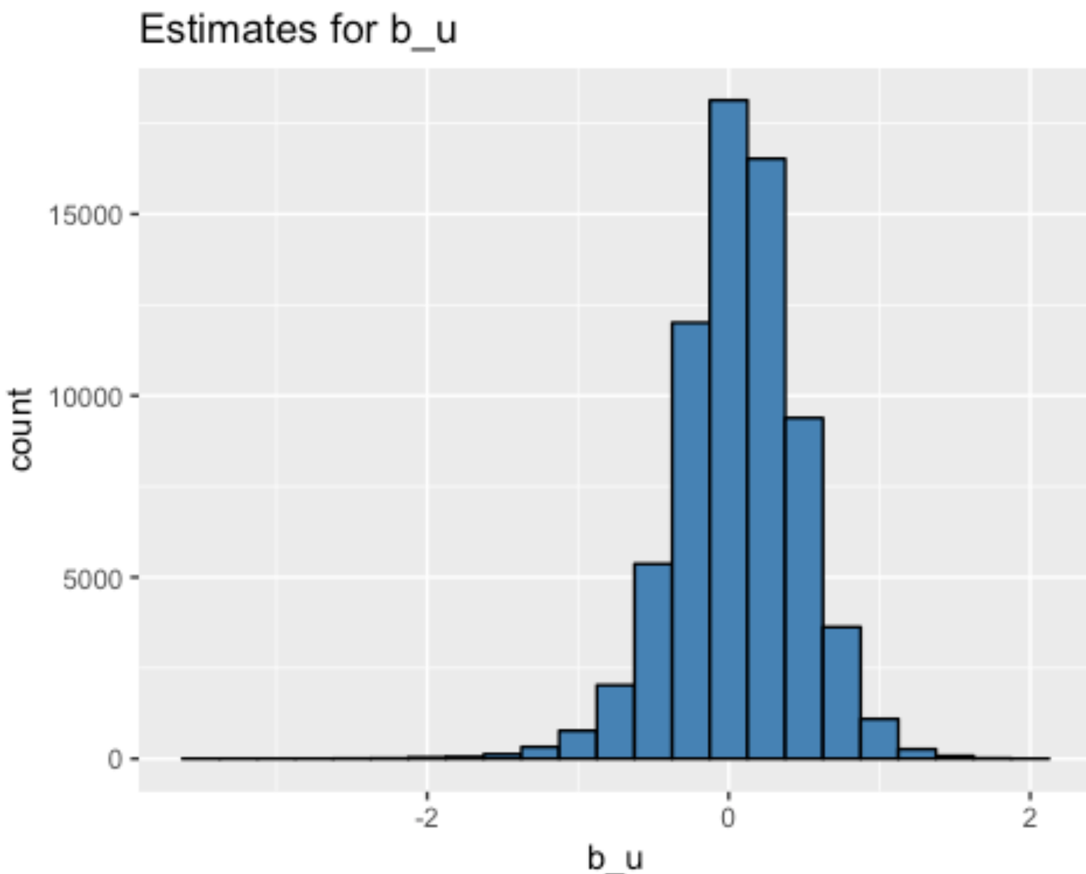
method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

Movie and User Effect Model

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

Plot user averages `b_u`

```
user_avgs %>%
  ggplot(aes(b_u )) +
  geom_histogram(fill = "steelblue", binwidth = .25, color = "black") +
  ggtitle("Estimates for b_u ")
```



Model

```
###Movie and User Effect Model
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_2_rmse <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effect Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

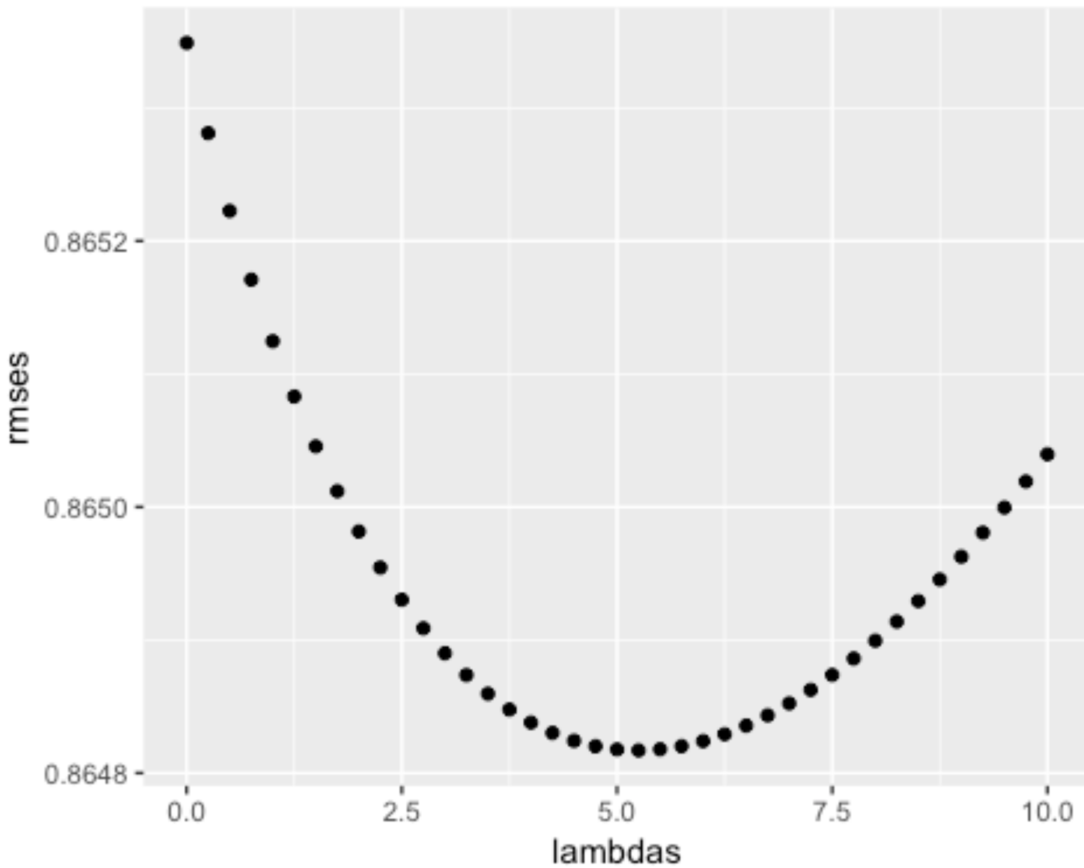
method	**RMSE**
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effect Model	0.8653488

Regularization Model

Regularization permits us to penalize large estimates that come from small sample sizes. This model includes the parameters for both **movie and user** effects. Cross-validation is also used to pick lambda.

```
# lambda is a tuning parameter
# use cross-validation to find the lambda with lowest rmse
lambdas <- seq(0, 10, 0.25)

# For each lambda, find b_i & b_u, followed by rating prediction & testing
rmses <- sapply(lambdas, function(lambda){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

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

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

Plot rmses vs lambdas to select the optimal lambda



Optimal lambda

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

RESULTS

Result of all models included in the table below.

method	RMSE
Just the average	**1.0612018**
Movie Effect Model	**0.9439087**
Movie + User Effect Model	**0.8653488**
Regularized Movie and User Effect Model	**0.8648170**

CONCLUSION

Residual mean squared error is used to evaluate how close the predictions are to the true values in the validation set. RMSE of the **Regularized Models** has improved from **0.8653488** to **0.8648170** compared to the **Movie and User Effect Model** . Both models meet the objective of this exercise of achieving RMSE <= 0.87750.