MovieLens Project

Aravind Akella

12/5/2019

Introduction

The MovieLens project is part of the Capstone assignment in the "HarvardX: PH125.9x Data Science: Capstone" course.

Recommendation systems use ratings that users have given items to make specific recommendations. The **Netflix Prize** announced in 2016 is one of the best known open competitions offered by companies that sell many products to many customers and permit these customers to rate their products. Netflix uses a recommendation system to predict how many stars a user gives to a specific movie. One star suggests it is not a good movie, whereas five stars suggest an excellent movie. Netflix came up with a million dollar offer challenging the data science community to improve the Netflix inhouse recommendation algorithm by 10% and win a million dollars. In September 2009, the prize was awarded to the team that bested Netflix's own algorithm for predicting ratings by 10.06%.

This capstone project is based on the winning team algorithm and is a part of the edX course. The requirement is to create a movie recommendation system using a mini version of the **MovieLens** dataset (**movielens**) and is provided by edX to make the computations a little easier. The aim is to develop a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set. Here, after analyzing the dataset substantially, I attempted to construct the algorithms, in an iterative manner, calculating RMSE based on two basic parameters (userId and movieId). As I achieved the required lowest RMSE value by the fourth iteration, no further attempts were made to include other parameters (e.g., year, genre). The results were finally compared for maximum possible accuracy in prediction with the lowest RMSE being 0.8649.

Dataset

The following code to generate the training and test datasets is provided by edX in the course module. Also included is the code to include libraries for the data anlysis and visualization.

```
################################
# Create edx set, validation set
################################

library("tidyverse")
library("data.table")
library("caret")
library("dplyr")

# libraries needed for data analysis and visualization
library(stringr)
library(lubridate)
library(ggplot2)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip
```

```r
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")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
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)
```

Data Pre-processing

Evaluate the dataset features and perform the needed cleanup of the data with the following code.

```r
# general stats
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
```

```r
head(validation)
```

```
##   userId movieId rating timestamp
## 1      1     231      5 838983392
## 2      1     480      5 838983653
## 3      1     586      5 838984068
## 4      2     151      3 868246450
## 5      2     858      2 868245645
## 6      2    1544      3 868245920
##                                                      title
## 1                                     Dumb & Dumber (1994)
## 2                                     Jurassic Park (1993)
## 3                                        Home Alone (1990)
## 4                                           Rob Roy (1995)
## 5                                    Godfather, The (1972)
## 6 Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                      genres
## 1                                    Comedy
## 2        Action|Adventure|Sci-Fi|Thriller
## 3                           Children|Comedy
## 4                  Action|Drama|Romance|War
## 5                               Crime|Drama
## 6 Action|Adventure|Horror|Sci-Fi|Thriller
```

```r
# take care of any missing values in the datasets
edx %>% sapply(., function(x) sum(is.na(x)))
```

```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

```r
validation %>% sapply(., function(x) sum(is.na(x)))
```

```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

```r
# extract year from title and clean title
edx <- edx %>%
  mutate(title = str_trim(title)) %>%
  # split title to title, year
  extract(title, c("title_tmp", "year"),
          regex = "^(.*) \\(([0-9 \\-]*)\\)$", remove = F) %>%
  mutate(year = if_else(str_length(year) > 4,
                        as.integer(str_split(year, "-", simplify = T)[1]),
                        as.integer(year))) %>%
  # replace title NA's with original title
  mutate(title = if_else(is.na(title_tmp), title, title_tmp)) %>%
  # drop title_tmp column
  select(-title_tmp)
```

```
validation <- validation %>%
  mutate(title = str_trim(title)) %>%
  extract(title, c("title_tmp", "year"), regex = "^(.*) \\(([0-9 \\-]*)\\)$",
          remove = F) %>%
  mutate(year = if_else(str_length(year) > 4,
                        as.integer(str_split(year, "-", simplify = T)[1]),
                        as.integer(year))) %>%
  mutate(title = if_else(is.na(title_tmp), title, title_tmp)) %>%
  select(-title_tmp)
```

It appears that no additional cleanup of the datasets is needed. We are now ready to perform data analysis to gather some useful insights on the dataset variables.

Data Analysis and Visualization

Before jumping into model building, it is essential to gain as much familiarity with the dataset variables. The following code is used for that purpose. The below code also includes plotting certain dataset variables to gain some insights into the dataset parametrs.

```
# summary stats
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                year          genres
##  Length:9000055     Min.   :1915   Length:9000055
##  Class :character   1st Qu.:1987   Class :character
##  Mode  :character   Median :1994   Mode  :character
##                     Mean   :1990
##                     3rd Qu.:1998
##                     Max.   :2008
```

```
summary(validation)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title                year          genres
##  Length:999999      Min.   :1915   Length:999999
##  Class :character   1st Qu.:1987   Class :character
##  Mode  :character   Median :1994   Mode  :character
```

```
##                      Mean   :1990
##                      3rd Qu.:1998
##                      Max.   :2008
```

The summary data confirms that there are no missing values in the datasets. The data also shows that the parameters are distributed very similarly between the two datasets.

```r
# number of unique users and movies
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

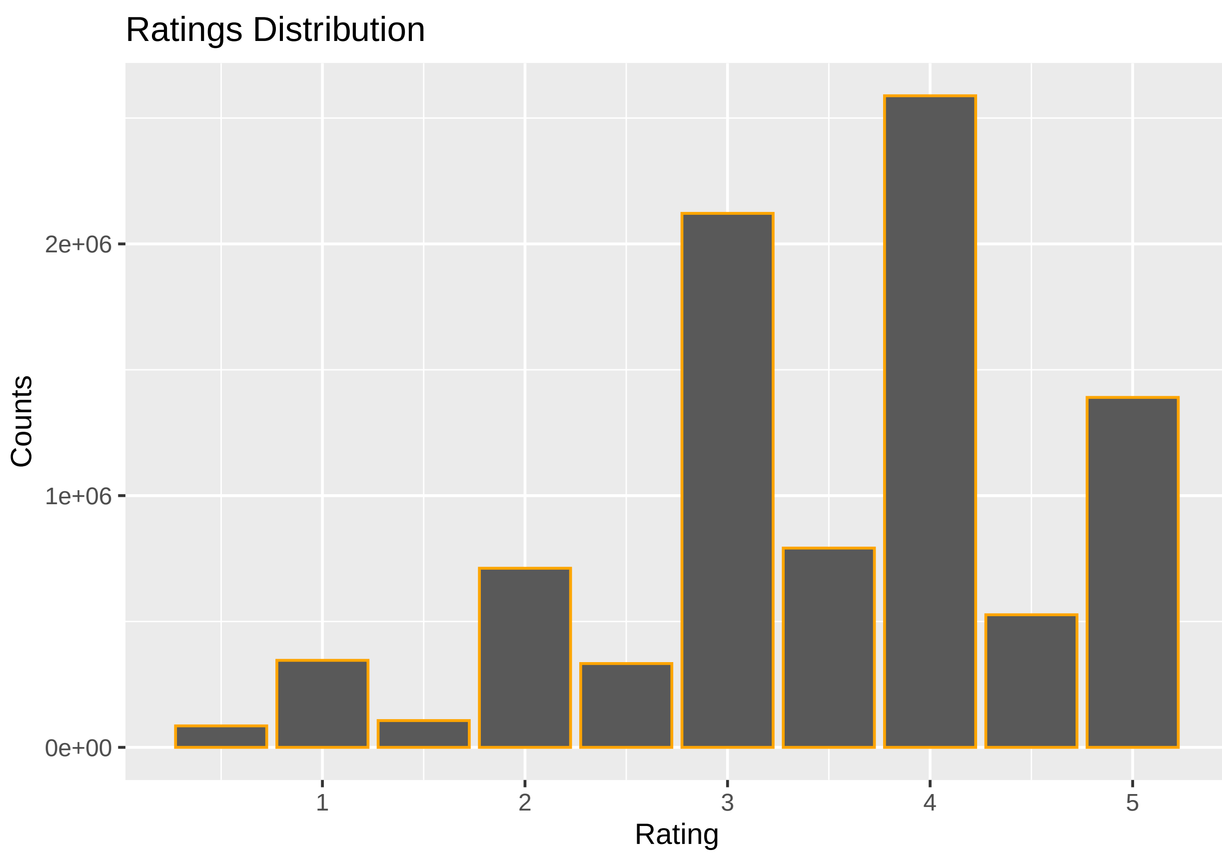
```
##   n_users n_movies
## 1   69878    10677
```

A little less than 70,000 users rated more than 10,000 movies at an average of seven movies per user.

```r
# ratings distribution - check unique values and plot the distirbutions
sort(unique(edx$rating))
```

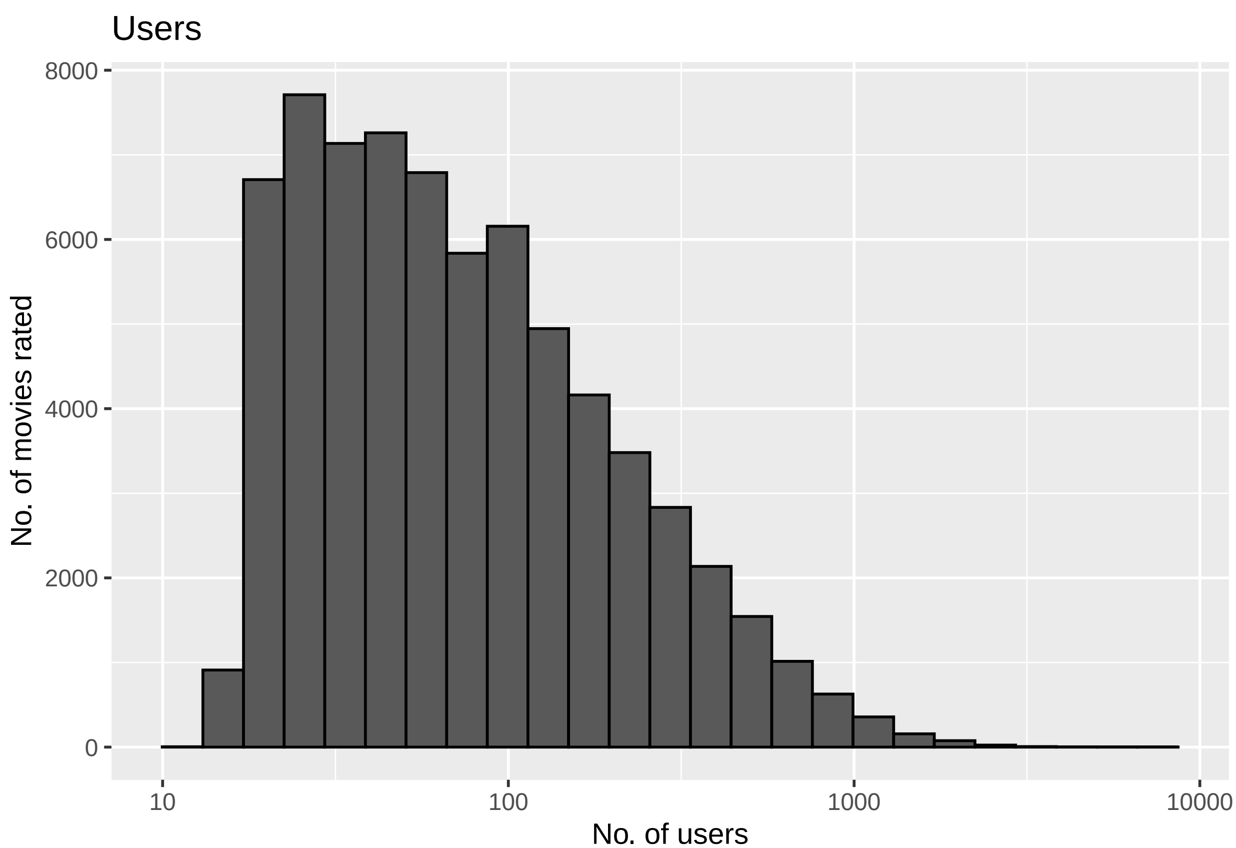
```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```r
ggplot(edx, aes(rating)) +
  geom_bar(color="orange") +
  ggtitle("Ratings Distribution") +
  xlab("Rating") + ylab("Counts")
```

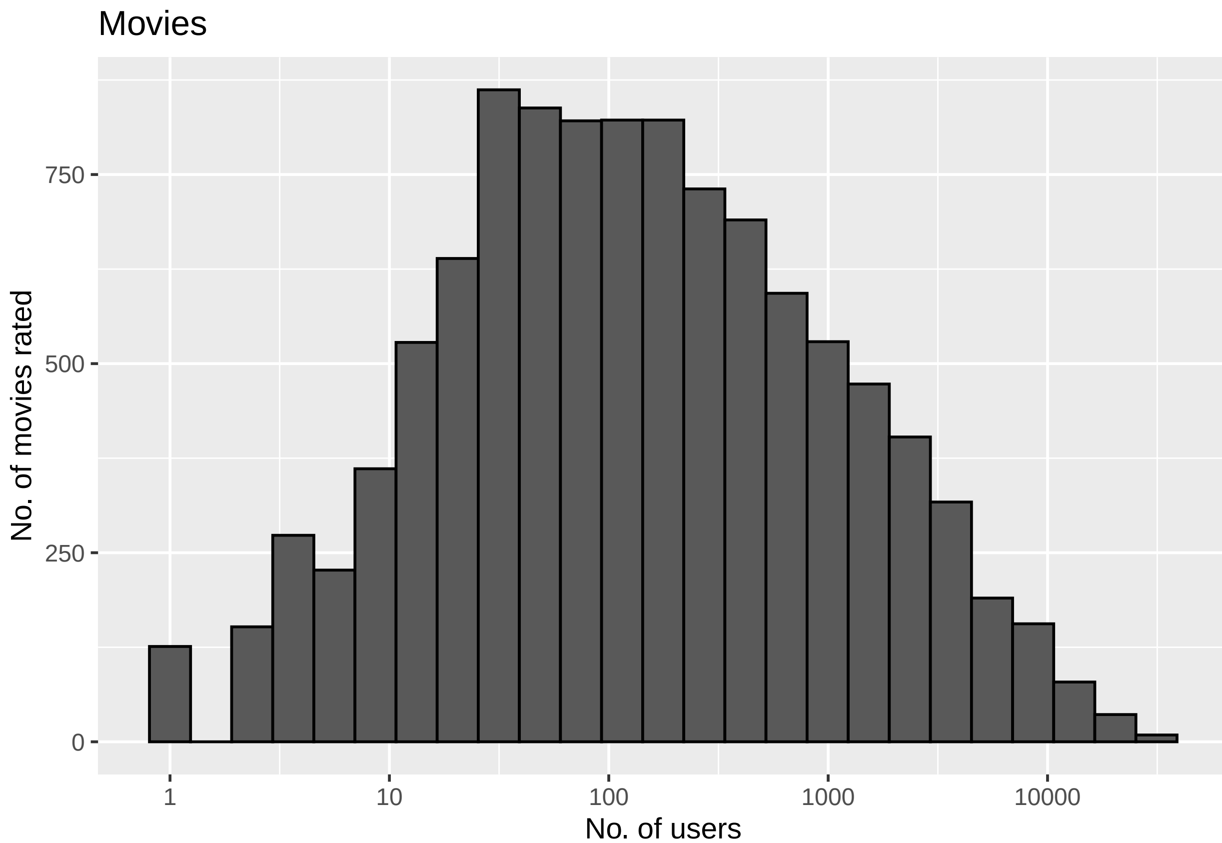


The data shows there are no movises with a zero(0) rating. It can be seen from the plot that most movies receive a rating of three (3) or above.

```
# distribution of users rating movies
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins=25, color="black") +
  scale_x_log10() +
  ggtitle("Users") +
  xlab("No. of users") + ylab("No. of movies rated")
```




```
edx %>% count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins=25, color="black") +
  scale_x_log10() +
  ggtitle("Movies") +
  xlab("No. of users") + ylab("No. of movies rated")
```



It is clear from the above three plots that (1) some movies are rated more often than others and (2) most users rated between 30 and 100 movies. This type of variability warrants the need to intorduce the concept of regularization in our models. The general idea behind regularization is to constrain the total variability of the effect sizes by introducing penalty terms. We will see this in the respective models below.

Model Preparation

The following function is used to compute the RMSE for vectors of ratings and their corresponding predictors:

```
# RMSE function for vectors of ratings and corresponding pedictors
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Model Building

The simplest possible recommendation system is predicting the same rating for all movies regardless of user. Such an approach makes use of the mean (or average) of all ratings. The first prediction is run with this approach and the resulting RMSE value is input into a table to generate a summary table.

```
####################################
#  Baseline Model (Just the Average)
####################################
```

```
# get the average of all ratings and view
mu_hat <- mean(edx$rating)
mu_hat
```

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

```
# run the model
model_1_rmse <- RMSE(validation$rating, mu_hat)

# store the model outcome and view
rmse_results <- tibble(method = "Baseline Model",
                       RMSE = model_1_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Baseline Model	1.061202

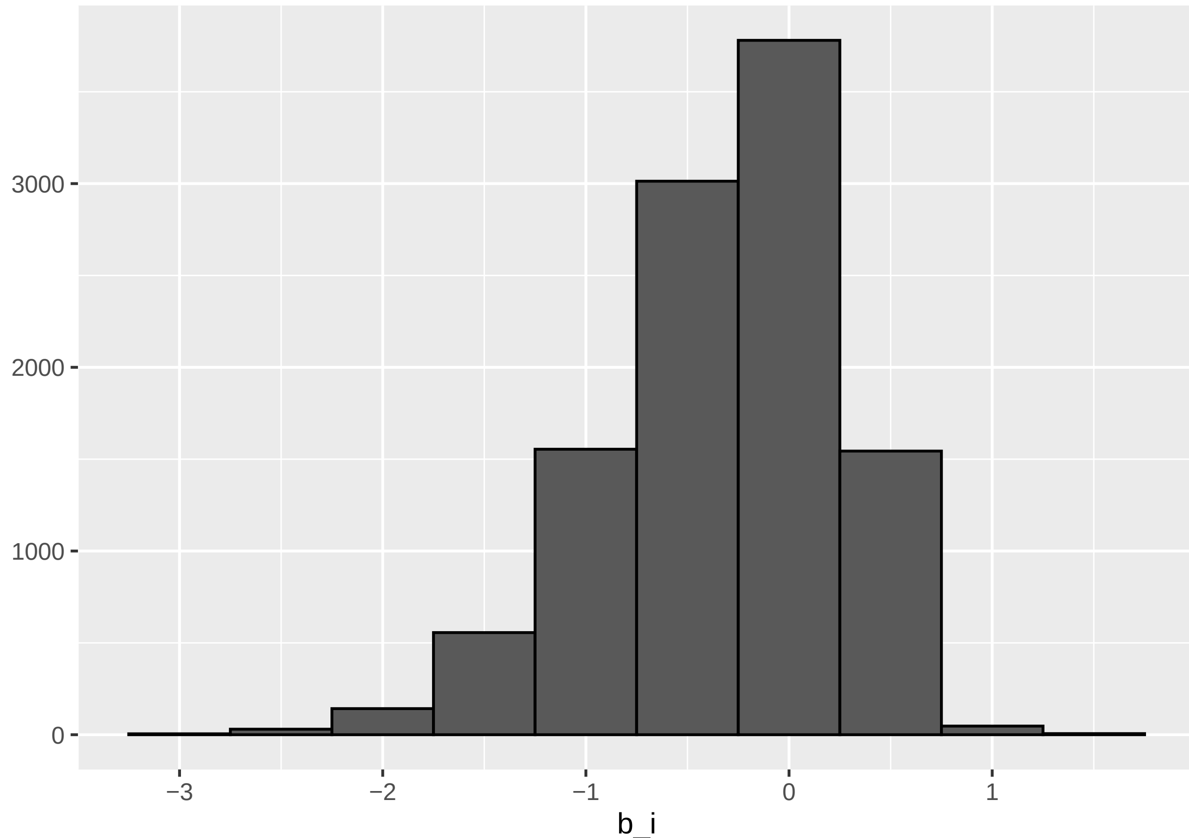
This model gave us baseline RMSE value on which improvements should be made as in the following.

Earlier analysis showed that some movies are rated higher than others. The above baseline model needs to be augmented by adding the bias term as in the below code.

```
####################
# Movie Effect Model
####################

# get the average rating
mu <- mean(edx$rating)
# get the least square estimates for each movie
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# view the variability of the estimates
qplot(b_i, data = movie_avgs, bins = 10, color = I("black"))
```




```
#generate the predictions set and run the model
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_2_rmse <- RMSE(predicted_ratings, validation$rating)

# store the model outcome and view
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Movie Effect Model",
                                 RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Baseline Model	1.0612018
Movie Effect Model	0.9439087

The prediction has significantly improved with the addition of a computed bias term to the mean. In the next iteration, we consider the individual user rating effect.

```
##############################
# Movie and User Effects Model
##############################

# compute the least square estimates for each user rating
```

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

# run the model
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# model outcome
model_3_rmse <- RMSE(predicted_ratings, validation$rating)

# store the model outcome and view
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Movie and User Effects Model",
                                 RMSE = model_3_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Baseline Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488

A further reduction in RMSE is now seen with the addition of individual user rating effect.

In the next iteration we use regularization to improve the predictions. Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients do not take extreme values.

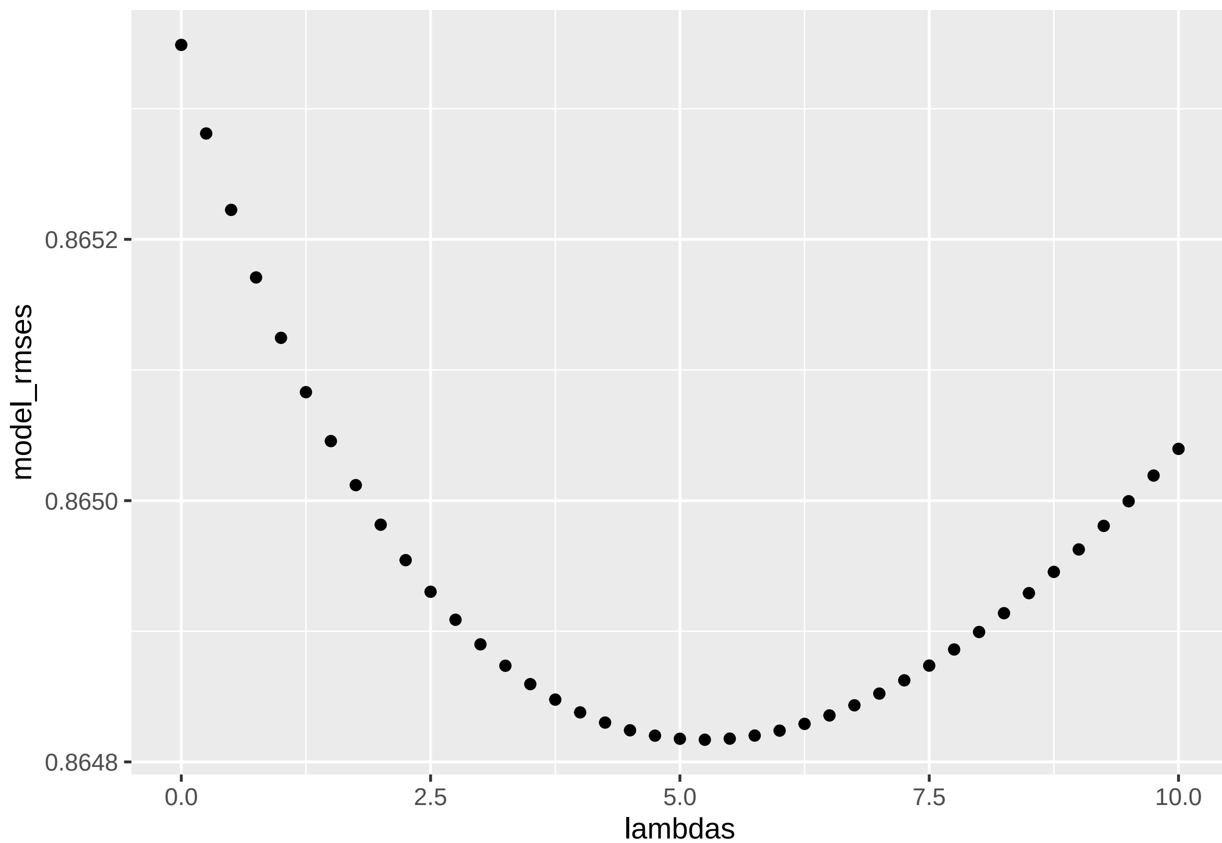
```r
##########################################
# Regularized Movie and User Effects Model
##########################################

# build a vector with tuning parameters
lambdas <- seq(0, 10, 0.25)

# run the model
model_rmses <- sapply(lambdas, function(l){
  # first comute the regularized estimates for movies and users
  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))
  # run the predictions
  predicted_ratings <-
```

```r
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  # run the model for each of the tuning parametrs
  return(RMSE(predicted_ratings, validation$rating))
})

# plot to view the results for all the tuning parametrs
qplot(lambdas, model_rmses)
```




```r
#chose the optimal tuning parameter for final compilation
lambda <- lambdas[which.min(model_rmses)]
lambda
```

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

```r
# store the model outcome and view
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Regularized Movie + User Effects Model",
                                 RMSE = min(model_rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Baseline Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488
Regularized Movie + User Effects Model	0.8648170

Results

The RMSE values of the respective models are as in the following table:

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


method	RMSE
Baseline Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488
Regularized Movie + User Effects Model	0.8648170

The lowest RMSE obtained therefore is **0.864817**

Conclusion

The RMSE table shows improvement of the model (as demonstrated in the lower RMSE values) over four different iterations. The baseline model 'Just the Average' estimates the RMSE to be slightly more than 1. Then incorporating 'Movie effect' and 'Movie and User effect' on model provide improvements (as in the reduced RMSE values). Finally, by introducing the regularization concept, the RMSE estimate turned out to be even lower (0.8649) and is just below the accepted minimum for this project.