HarvardX PH125.9X Data Science Capstone MovieLens Project

David Wong

June, 2020

Contents

Section 1: Introduction

1.1 Background

This is the first of the two Capstone projects that learners need to complete in the final module of the HarvardX Data Science Professional Certification series.

1.2 Objective

The key objective is to build a movie recommendation system using the smaller version of the MovieLens dataset which contains 10 million rating records.

1.3 Guidelines and rules

Data preparation script is provided by the course provider. The script pre-splits the data set into two data sets: "edx" and "validation".

The "validation" data set is STRICTLY used for final validation ONLY; whereas the "edx" data set is used for training and testing.

1.4 Key steps

Data Preparation: in addition to the initial data preparation script provided by the course provider, we will further split the "edx" data set into "train_set" and "test_set". As the names suggest, "train_set" and "test_set" will be used for training and testing the various algorithms/models built, and after each test, we will determine if the result has improved from the previous model and if it has hit the target goal.

Data Exploration: we will use a combination of simple scripts and plots to provide facts and statistics to support our rationale behind using each model.

Building & testing the models: this step is where we build our model based on the observation and principles determined in the Data Exploration step; train, test and predict result for each model.

Result for each test will be recorded and appended to a data frame and presented at the end of each test.

1.5 Goal

The goal is very focused and straightforward: achieve a target RMSE of <0.86490.

1.6 R script

The code snipplets presented in this report are extracted partially from the R script submitted separately as part of the project submission. Consolidation of these code snipplets does not make it whole, therefore, please always refer to the R script for a successful full run.

Section 2: Analysis

2.1 Data Preparation

Data preparation codes provided by the course provider edX will be run first as pre-requisite. As the dataset provided is already quite clean and neat, the only extra step required is to split the "edx" data set further into a "train_set" and a "test_set" for training and testing as the names suggest.

2.2 Data Exploration - Round 1

A movie recommendation system is something that is very close to our daily life. Common sense tells us a few things:

1. Not every user could have rated every movie.
2. Some movies are being rated more than the others such as the blockbusters - which leads us to "movie bias".
3. Some users are more active than the others - which leads us to "user bias"
4. Some movies are being rated once or only a few times - which leads us to the need of regularization on the biases.

Instead of just using common sense, we will observe the edx dataset and use facts and statistics to support the rationale behind them.

2.2.1 Generation observation

Before further exploration, we will use tibble to confirm the data looks clean and the number of rows and columns match the expectation for all the four data sets.

```
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|Thriller
##  3      1     292      5 838983421 Outbreak (1995)        Action|Drama|Sci-Fi|T~
##  4      1     316      5 838983392 Stargate (1994)        Action|Adventure|Sci-~
##  5      1     329      5 838983392 Star Trek: Generation~ Action|Adventure|Dram~
##  6      1     355      5 838984474 Flintstones, The (199~ Children|Comedy|Fanta~
##  7      1     356      5 838983653 Forrest Gump (1994)    Comedy|Drama|Romance|~
##  8      1     362      5 838984885 Jungle Book, The (199~ Adventure|Children|Ro~
##  9      1     364      5 838983707 Lion King, The (1994)  Adventure|Animation|C~
## 10      1     370      5 838984596 Naked Gun 33 1/3: The~ Action|Comedy
## # ... with 9,000,045 more rows
```

```
validation %>% as_tibble()
```

```
## # A tibble: 999,999 x 6
##    userId movieId rating  timestamp title                    genres
##     <int>   <dbl>  <dbl>      <int> <chr>                    <chr>
```

```
##  1      1     231    5    838983392 Dumb & Dumber (1994)      Comedy
##  2      1     480    5    838983653 Jurassic Park (1993)      Action|Adventure|~
##  3      1     586    5    838984068 Home Alone (1990)         Children|Comedy
##  4      2     151    3    868246450 Rob Roy (1995)            Action|Drama|Roma~
##  5      2     858    2    868245645 Godfather, The (1972)     Crime|Drama
##  6      2    1544    3    868245920 Lost World: Jurassic Par~ Action|Adventure|~
##  7      3     590    3.5 1136075494 Dances with Wolves (1990) Adventure|Drama|W~
##  8      3    4995    4.5 1133571200 Beautiful Mind, A (2001)  Drama|Mystery|Rom~
##  9      4      34    5    844416936 Babe (1995)               Children|Comedy|D~
## 10      4     432    3    844417070 City Slickers II: The Le~ Adventure|Comedy|~
## # ... with 999,989 more rows
```

```
train_set %>% as_tibble()
```

```
## # A tibble: 7,200,043 x 6
##    userId movieId rating timestamp title                  genres
##     <int>   <dbl>  <dbl>     <int> <chr>                  <chr>
##  1      1     185      5 838983525 Net, The (1995)        Action|Crime|Thriller
##  2      1     292      5 838983421 Outbreak (1995)        Action|Drama|Sci-Fi|T~
##  3      1     316      5 838983392 Stargate (1994)        Action|Adventure|Sci-~
##  4      1     329      5 838983392 Star Trek: Generation~ Action|Adventure|Dram~
##  5      1     355      5 838984474 Flintstones, The (199~ Children|Comedy|Fanta~
##  6      1     356      5 838983653 Forrest Gump (1994)    Comedy|Drama|Romance|~
##  7      1     362      5 838984885 Jungle Book, The (199~ Adventure|Children|Ro~
##  8      1     364      5 838983707 Lion King, The (1994)  Adventure|Animation|C~
##  9      1     370      5 838984596 Naked Gun 33 1/3: The~ Action|Comedy
## 10      1     377      5 838983834 Speed (1994)           Action|Romance|Thrill~
## # ... with 7,200,033 more rows
```

```
test_set %>% as_tibble()
```

```
## # A tibble: 1,799,967 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     594    5    838984679 Snow White and the Sev~ Animation|Children|~
##  3      2     376    3    868245920 River Wild, The (1994)  Action|Thriller
##  4      2     733    3    868244562 Rock, The (1996)        Action|Adventure|Th~
##  5      2    1210    4    868245644 Star Wars: Episode VI ~ Action|Adventure|Sc~
##  6      2    1356    3    868244603 Star Trek: First Conta~ Action|Adventure|Sc~
##  7      2    1391    3    868246006 Mars Attacks! (1996)    Action|Comedy|Sci-Fi
##  8      3    1252    4   1133571071 Chinatown (1974)        Crime|Film-Noir|Mys~
##  9      3    1408    3.5 1133571145 Last of the Mohicans, ~ Action|Romance|War|~
## 10      3    1552    2   1133571139 Con Air (1997)          Action|Adventure|Th~
## # ... with 1,799,957 more rows
```

Multiplying the unique number of movies and users which gives us a result of more than 746M records, proves that not every user has rated every movie in our data set.

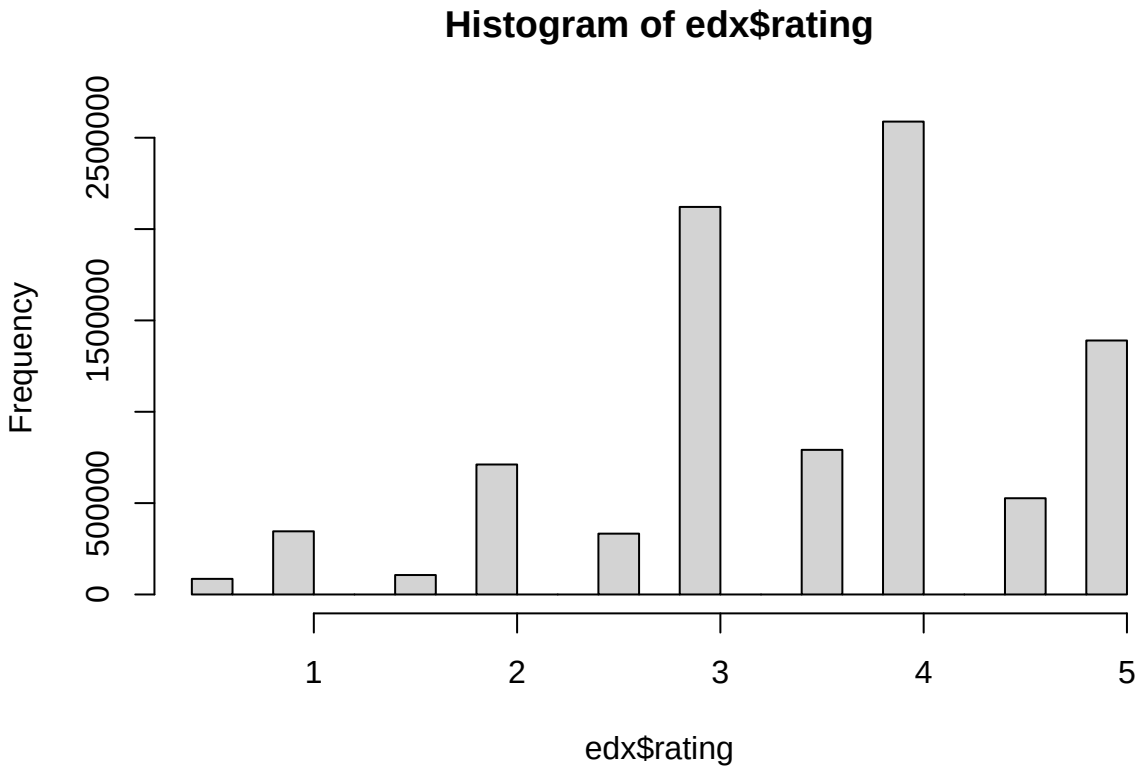
```
# Determine number of unique users and movies and confirm that not all users have rated all movies
unique_user <- n_distinct(edx$userId)
unique_movie <- n_distinct(edx$movieId)
unique_user * unique_movie
```

```
## [1] 746087406
```

2.2.2 Pave the way for movie bias

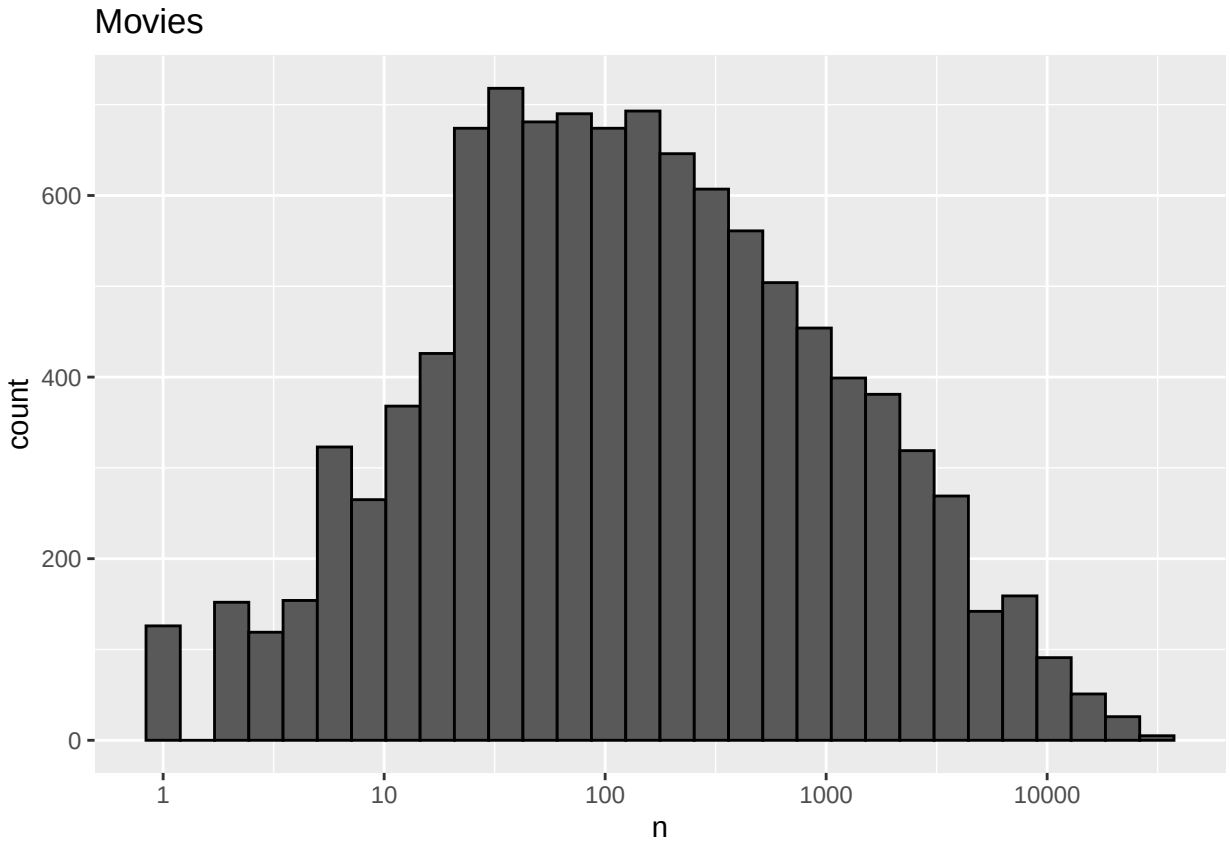
A simple histogram below shows the distribution of ratings regardless of movie or user.

```
# Observe ratings distribution
hist(edx$rating)
```



The diagram below plots the number of ratings per movie. We can see from the distribution that some movies get rated more often than the others and they vary a lot.

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



The script below shows some examples of movies that are rated only once, which are, as expected, obscure ones.

```
edx %>%
  group_by(movieId) %>%
  summarize(count = n()) %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(rating = rating, n_rating = count) %>%
  slice(1:20) %>%
  knitr::kable()
```

title	rating	n_rating
1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)	2.0	1
100 Feet (2008)	2.0	1
4 (2005)	2.5	1
Accused (Anklaget) (2005)	0.5	1
Ace of Hearts (2008)	2.0	1
Ace of Hearts, The (1921)	3.5	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di...) (1971)	1.5	1
Africa addio (1966)	3.0	1
Aleksandra (2007)	3.0	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Battle of Russia, The (Why We Fight, 5) (1943)	3.5	1

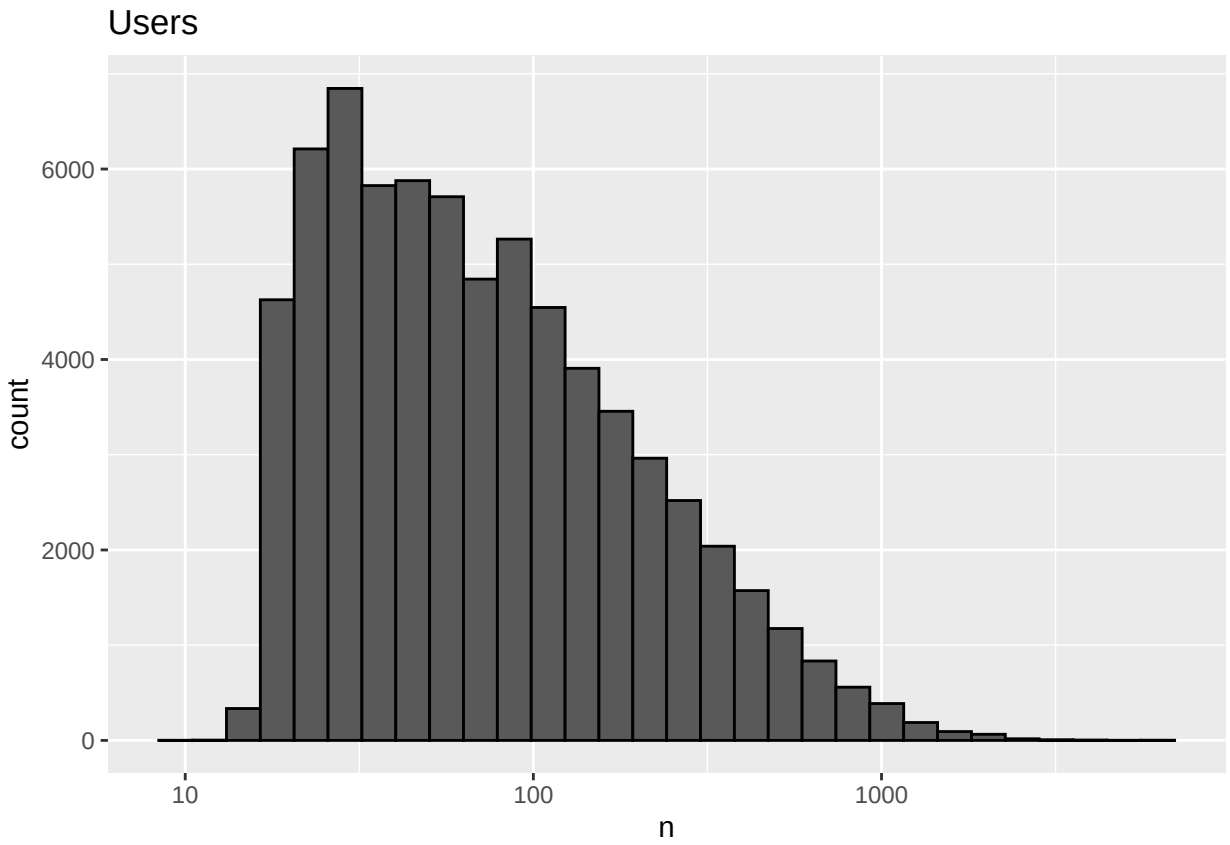
title	rating	n_rating
Bellissima (1951)	4.0	1
Big Fella (1937)	3.0	1
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3.0	1
Blind Shaft (Mang jing) (2003)	2.5	1
Blue Light, The (Das Blaue Licht) (1932)	5.0	1
Borderline (1950)	3.0	1
Brothers of the Head (2005)	2.5	1
Chapayev (1934)	1.5	1
Cold Sweat (De la part des copains) (1970)	2.5	1

Therefore, we will need to take movie bias into consideration of our modeling.

2.2.3 Pave the way for user bias

The diagram below plots the number of ratings per user and confirms that some users are more active than the others.

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

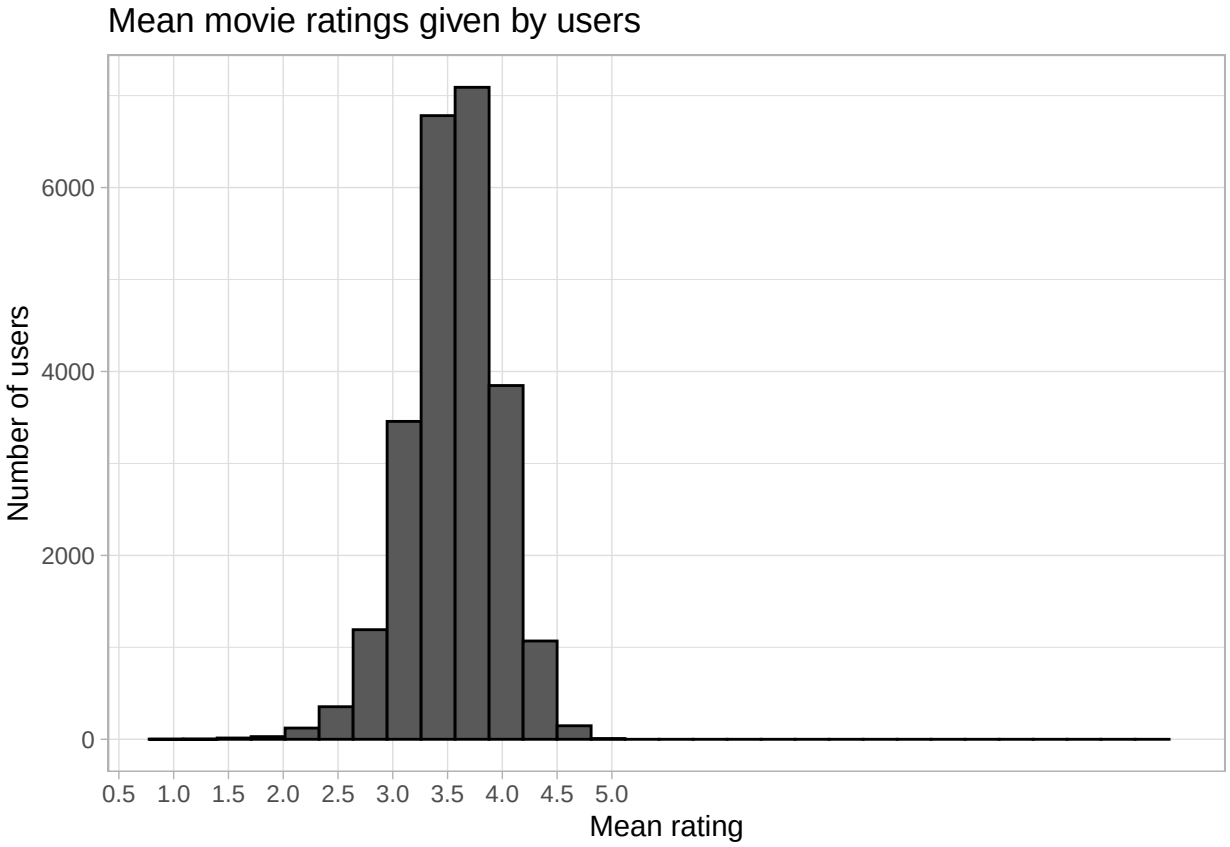


The diagram below plots the mean movie ratings given by users who have rated more than 100 movies

and there is substantial variability across users as well. Some users are very critical and stringent on their ratings, and some users are more forgiving and love every movie.

This implies that we need to take user bias into consideration of our modeling later.

```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings given by users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



2.3 Building & testing the models - Round 1: Simple average, movie and user effects

2.3.1 Model 1: Average movie rating

The first model we are building is simply to predict the same rating for all movies regardless of user using the average movie rating.

The result is expected to be nothing but the standard deviation and the resulted RMSE is about 1 which is far from our target (RMSE<0.86490).

```
# Compute the mean
mu <- mean(train_set$rating)

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

```
## [1] 1.060704
```

A data frame called "rmse_results" is created to store the test results. Each test result will be appended at the end of the table as the tests are conducted throughout the project. Below shows the first entry of the test result of our first model.

```
# Create a data frame and Save result of the first model
rmse_results <- data_frame(method = "Average movie rating model (test_set)", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
rmse_results
```

```
## # A tibble: 1 x 2
##   method                                  RMSE
##   <chr>                                  <dbl>
## 1 Average movie rating model (test_set)  1.06
```

2.3.2 Model 2: Movie Effect model

As confirmed by the data exploration earlier, some movies are rated higher and more frequently than others.

The second model is to enhance the first model by adding the term b_i to represent average ranking for movie i. We describe this as movie bias or movie effect.

The result has improved but is not meeting our target (RMSE<0.86490) yet.

```
# Enhance Model 1 by adding the movie bias b_i which represents average ranking for movie i
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```
# Test results based on movie effect model
predicted_ratings <- mu +  test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

movie_effect_rmse <- RMSE(predicted_ratings, test_set$rating)

movie_effect_rmse
```

```
## [1] 0.9437144
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model (test_set)",
                                     RMSE = movie_effect_rmse ))

# Check results and confirm there is an improvement using movie bias
rmse_results
```

```
## # A tibble: 2 x 2
##   method                                 RMSE
##   <chr>                                 <dbl>
## 1 Average movie rating model (test_set) 1.06
## 2 Movie effect model (test_set)         0.944
```

2.3.3 Model 3: User Effect model

As confirmed by the data exploration earlier, we need a model that takes user effect into consideration, so that, for example, if a great movie is rated by a critical user, the movie and user effects will counter each other and give a more accurate prediction.

The third model is therefore to enhance the previous model by adding the term b_u to represent average rating for user u. We describe this as user bias or user effect.

The result has improved again and is very close to our target (RMSE<0.86490).

```
# Enhance Model 2 by adding the user bias b_u
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))


# Test results based on user effect model
predicted_ratings <- test_set%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

user_effect_rmse <- RMSE(predicted_ratings, test_set$rating)

user_effect_rmse
```

```
## [1] 0.8661625
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and user effect model (test_set)",
                                     RMSE = user_effect_rmse))

# Check results and confirm there is an improvement using user bias on top of movie bias
rmse_results
```

```
## # A tibble: 3 x 2
##   method                                 RMSE
##   <chr>                                 <dbl>
## 1 Average movie rating model (test_set)  1.06
## 2 Movie effect model (test_set)          0.944
## 3 Movie and user effect model (test_set) 0.866
```

2.4 Data Exploration - Round 2

This section is to confirm our intuition mentioned earlier that there would be a possibility that some movies are rated by very few users. We will need to add a penalty to target especially when the sample size is small. Before building the model, we will explore the data further.

The following code snipplet will show you the "best" and "worst" movies are the obscure ones, based on movie effect only.

```
# Create a list of Movie IDs and Titles
movie_titles <- edx %>%
  select(movieId, title) %>%
  distinct()

# Find out the 10 best movies according to our estimate - obscure movies!
print("Top 10 best movies")
```

```
## [1] "Top 10 best movies"
```

```
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  slice(1:10) %>%
  pull(title)
```

```
##  [1] "Hellhounds on My Trail (1999)"
##  [2] "Shanghai Express (1932)"
##  [3] "Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)"
##  [4] "Fighting Elegy (Kenka erejii) (1966)"
##  [5] "Sun Alley (Sonnenallee) (1999)"
##  [6] "Bullfighter and the Lady (1951)"
##  [7] "Blue Light, The (Das Blaue Licht) (1932)"
##  [8] "Human Condition II, The (Ningen no joken II) (1959)"
##  [9] "Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)"
## [10] "Life of Oharu, The (Saikaku ichidai onna) (1952)"
```

```
# Find out the 10 worst movies according to our estimate - also obscure movies!
print("Top 10 worst movies")
```

```
## [1] "Top 10 worst movies"
```

```
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  slice(1:10) %>%
  pull(title)
```

```
##  [1] "Besotted (2001)"
##  [2] "Grief (1993)"
##  [3] "Confessions of a Superhero (2007)"
##  [4] "War of the Worlds 2: The Next Wave (2008)"
##  [5] "Disaster Movie (2008)"
##  [6] "SuperBabies: Baby Geniuses 2 (2004)"
##  [7] "From Justin to Kelly (2003)"
```

```
##  [8] "Hip Hop Witch, Da (2000)"
##  [9] "Criminals (1996)"
## [10] "Mountain Eagle, The (1926)"
```

The reason for the above is that these obscure movies were rated by very few people, a lot of them are rated by one user in fact, according to the following codes.

```
#Find out how often these top (best and worst) movies are rated
train_set %>% count(movieId) %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  slice(1:10) %>%
  pull(n)
```

```
##  [1] 1 1 2 1 1 1 1 3 4 2
```

```
train_set %>% count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  slice(1:10) %>%
  pull(n)
```

```
## Joining, by = "movieId"
```

```
##  [1]   2   1   1   1  30  40 161  10   2   1
```

2.5 Building & testing the models - Round 2: Regularization

2.5.1 Model 4: Regularized Movie and User Effect

After the second round of data exploration, we can confirm the need of finding a means to constrain the total variability of the effect sizes. In other words, we need to add a penalty "lambda" that targets particularly when the sample size is small - for example, movies rated by only 1 or few users. Such penalty will not affect much for the case of a large sample size which gives us table estimate.

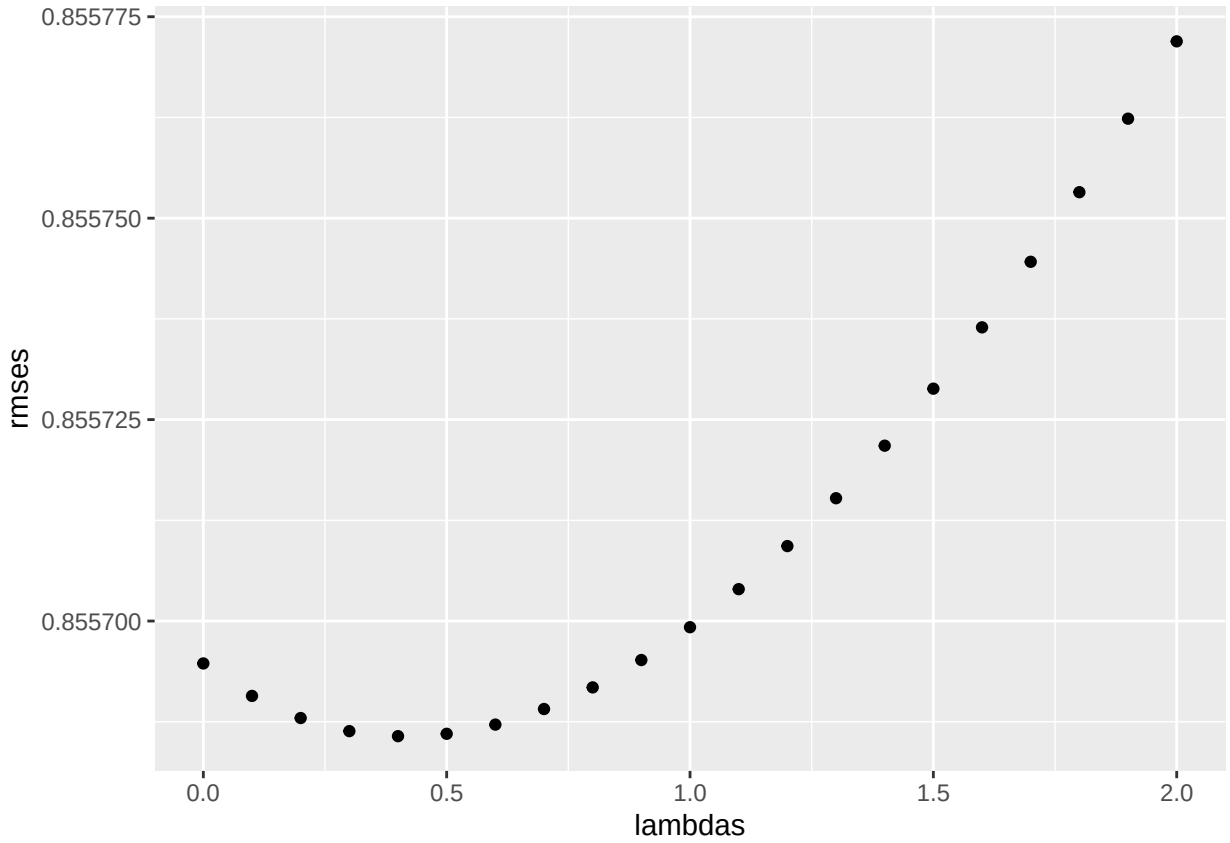
The next step is to find out the optimal of "lambda".

2.5.2 Choosing the right lambda

lambda is a tuning parameter and it takes a bit of "trial-and-error" and we use cross-validation to choose it.

The following codes and plot help us identify the optimal value.

```
# The penalized estimate, lambda, is a tuning parameter, therefore, use cross-validation to choose it
lambdas <- seq(0, 2, 0.10)

# For each lambda,find b_i & b_u, followed by rating prediction & testing

############################
#                          #
# Use train_set data only  #
#                          #
############################

rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

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

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

  predicted_ratings <-
    train_set %>%
    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, train_set$rating))
})

# Plot rmses vs lambdas to select the optimal lambda
qplot(lambdas, rmses)
```




```
# Determine the optimal lambda which gives the smallest RMSE value
lambda <- lambdas[which.min(rmses)]
```

2.5.3 Test results using Regularized model on both train_set and test_set

The result on "train_set" can be obtained by getting the minimum "rmses" value from the earlier code snipplet. We can see that it has already hit our target (RMSE<0.86490).

```
min(rmses)
```

```
## [1] 0.8556857
```

```
# Test and append result to the existing result data frame
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model (train_set)",
                                     RMSE = min(rmses)))

# Check results and confirm there is an improvement and also meets the target
rmse_results
```

```
## # A tibble: 4 x 2
##   method                                              RMSE
##   <chr>                                              <dbl>
## 1 Average movie rating model (test_set)              1.06
```

```
## 2 Movie effect model (test_set)                          0.944
## 3 Movie and user effect model (test_set)                 0.866
## 4 Regularized movie and user effect model (train_set) 0.856
```

Now, let's use the optimal lambda, run the algorithm with the test_set. We can see that the RMSE result using the test_set also meets our target (RMSE<0.86490).

```
########################################
#                                      #
# Use test_set data with optimal lambda #
#                                      #
########################################

mu <- mean(test_set$rating)

# Compute regularized movie bias using optimal lambda
b_i <- test_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

# Compute regularized user bias using optimal lambda
b_u <- test_set %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

# Test results based on Regularized Movie and User effect model
predicted_ratings <-
  test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

test_set_result <- RMSE(predicted_ratings, test_set$rating)

test_set_result
```

```
## [1] 0.8408758
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model (test_set)",
                                     RMSE = test_set_result))

# Check results and confirm there is an improvement on test_set and also meets the target
rmse_results
```

```
## # A tibble: 5 x 2
##   method                                                RMSE
##   <chr>                                                <dbl>
## 1 Average movie rating model (test_set)                1.06
## 2 Movie effect model (test_set)                        0.944
## 3 Movie and user effect model (test_set)               0.866
```

```
## 4 Regularized movie and user effect model (train_set) 0.856
## 5 Regularized movie and user effect model (test_set)  0.841
```

Section 3: Results

3.1 Final result using "validation" data set

Now, we are quite confident that the "Regularized movie and user effect model" will achieve our goal for hitting RMSE<0.86490, at least for both the train_set and test_set data.

The following is the final step of using the optimal lambda and run the same algorithm using the "validation" data pre-split by the code provided by the course.

As usual, the final result is appended to the result data frame and presented in the summary table.

```
##############################################################################
#                                                                            #
# Final test using Regularized Movie and User Effect model on Validation data set #
#                                                                            #
##############################################################################

mu <- mean(validation$rating)

# Compute regularized movie bias using optimal lambda
b_i <- validation %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

# Compute regularized user bias using optimal lambda
b_u <- validation %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

# Final test
predicted_ratings <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

final_va_result <- RMSE(predicted_ratings, validation$rating)

final_va_result
```

```
## [1] 0.825615
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Final: Regularized movie and user effect model on Valida
                                     RMSE = final_va_result))

# Check results and confirm project target is met!
rmse_results
```

```
## # A tibble: 6 x 2
```

```
##   method                                                                  RMSE
##   <chr>                                                                  <dbl>
## 1 Average movie rating model (test_set)                                  1.06
## 2 Movie effect model (test_set)                                          0.944
## 3 Movie and user effect model (test_set)                                 0.866
## 4 Regularized movie and user effect model (train_set)                    0.856
## 5 Regularized movie and user effect model (test_set)                     0.841
## 6 Final: Regularized movie and user effect model on Validation data set 0.826
```

To recap, the final result on Validation data set: RMSE = 0.825615 which is less than the target of 0.86490.

3.2 Interpretation

The findings of the results are quite straightforward to interpret: I started with an average movie rating model as the basis and added one bias after another - firstly, the movie bias and next the user bias - which presented improvement. To further fine-tune the model, we added a penalty term lambda to tackle the issue on small sample size such as certain movies were rated only by 1 or a few user, this further regulated the biases or the effects and hence produced a model that met our target goal.

In short, the more the regulated biases are introduced, the lower the RMSE result we get.

Section 4: Conclusion

The project is considered successful as it was completed with the target goal achieved and all the rules followed.

The approach used throughout is systematic and effective, albeit conservative; it is conservative in the sense that I have been using the same concept and technique by adding and regulating the biases or effects one step at a time and eventually achieve the goal. There is obviously still room for improvement and opportunities to explore in the future. For example, the existing model can be enhanced further by adding other biases or effects such as genre bias and time bias. To address genre bias, I would plan to normalize the genre values from multiple genre values in one column into one genre value per row or observation. To address time bias, I would plan to convert the timestamp value into a date format that is easy to analyse such as "YYYY-MM-DD". To go beyond biases and effects, in the future, we should also study further around the fact that groups of movies have similar rating patterns and groups of users have similar rating patterns as well, through Matrix factorization.

Final retrospective: as someone who is totally new to statistics, data science and R programming, I am generally pleased that I have completed this assignment without major issues. I can't thank Professor Rafael A. Irizarry, his team and everyone supporting edX platform for the great materials. The only mistake I could have made was probably signing up this Capstone module a bit too close to the deadline, otherwise I would have more time to develop and fine-tune the model by enhancing with what I suggested earlier.

THE END