Re: Referencing the MovieLens Dataset to Predict Movie Ratings

Jan2, 2020

Contents

PHY125.9X | CAPSTONE PROJECT: MovieLens Data
By Nik S.

Reference List: Irizarry, A. R. (2015). Introduction to Data Science

1. Introduction

NOTE: GitHub repository is here: https://github.com/nik-labs/MovieLens

Background:
The Netflix data is not publicly available, but the GroupLens research lab generated their own database with over 20 million ratings for over 27K movies by more than 138K users.

Project Objective:
The focus is to create a movie recommendation system which will reduce the Root Mean Squared Error (RMSE) to less than or equal to 0.8649. To clarify, RMSE measures accuracy by stating differences between prediction values and observed values.
This initiative will leverage a small subset of the entire data population use in the Netflix Prize competition; specifically, the validation set will be 10% of MovieLens data due to the sheer volume of the database population. There was a skeleton framework of code provided to generate the intended datasets. The developed algorithm leverages the edx set. Additionally, the final test of the algorithm predicts movie ratings in the validation set as if they were unknown. RMSE will be used to evaluate how close the predictions are to the true values in the validation set.

The MovieLens dataset zip file which is downloaded includes the following code:

```
#################################
#Project MovieLens
#Create edx set, validation set
#################################

######################################################################
# Introduction - MovieLens Dataset
######################################################################
## Aim is to train a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set
## Note: this process could take a couple of minutes because it is loading tidyverse and caret packages

# Package Instals
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")

#Libraries Referenced
library(tidyverse)
library(caret)


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

#Check download
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")
```

 Reference List: Irizarry, A. R. (2015). Introduction to Data Science

To make an accurate prediction of what users will review a movie that they have not seen the movie yet, the MovieLens dataset is segmented into a training subset called “edx” to train the algorithm, and a “validation” subset to test the movie ratings. Testing will be performed on the “edx” subset whereas the “validation” subset will be the final algorithm test.

```
# Validation set will be 10% of MovieLens data
# Using R 3.6.2 version
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,]


# 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)


save(edx, validation, file = datafile)

} else {
  load(datafile)
}
```

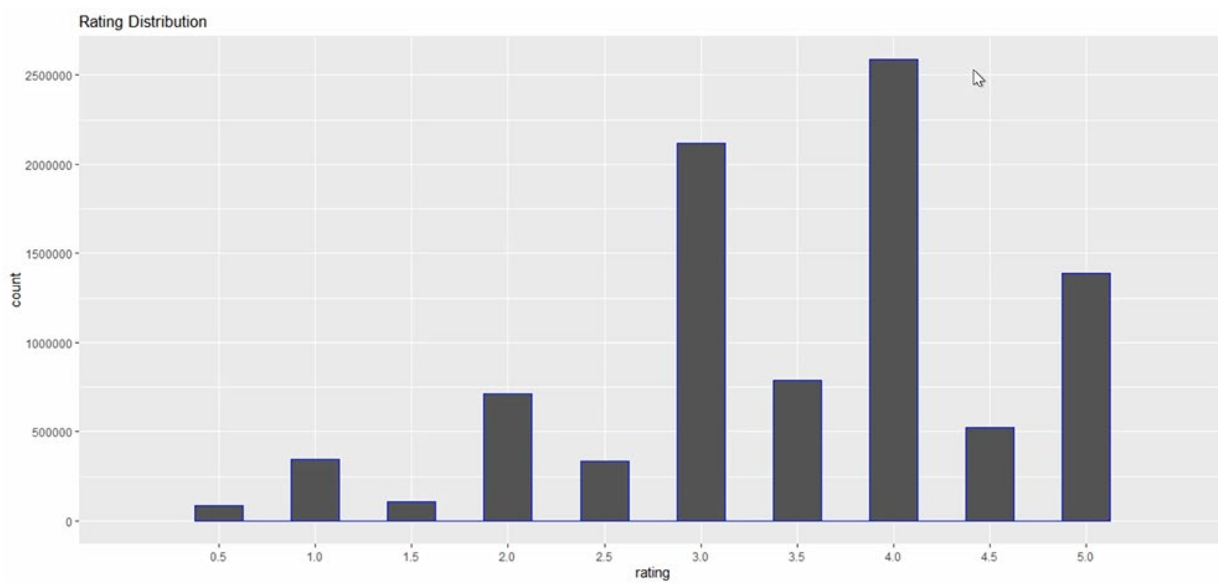
2. Methods / Analysis

Upon review of the edx subset, there are six columns that denote the following – userId, moveId, timestamp, title, and genres. Each entry illustrates a under rating for a specific movie (e.g., row#1 illustrates user rating for the movie Boomerang from 1992).

```
  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
> 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
```

```
# Ratings Mean
mean(edx$rating)

# Histogram: Ratings distribution in blue color font
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "blue") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ggtitle("Rating Distribution")
```

TABLE 1: Rating Distribution

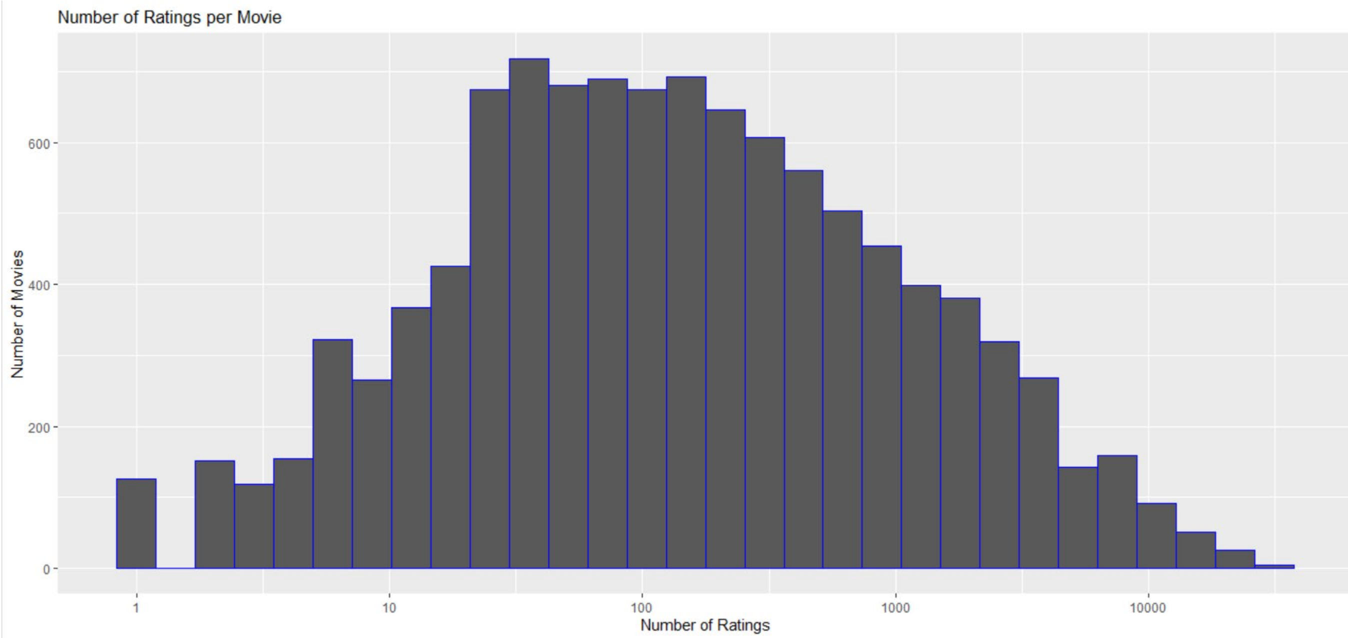


The number of ratings per movie varies and some so much so that movies which have received limited number of ratings may not be ones that can be relied upon. Specifically, movies which received a rating a piece tally up to 125 movies and should be candidates that should be carved out of reliable data. To account for such anomalies in the observed dataset below, regularization is employed to reduce fitting closely to skewed functions. Moreover, having the error function incorporate a penalty component allows for discounting atypical data.

```
# Plot number of ratings per movie in blue color font
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue") +
  scale_x_log10() +
  xlab("Number of Ratings") +
  ylab("Number of Movies") +
  ggtitle("Number of Ratings per Movie")
```


Reference List: Irizarry, A. R. (2015). Introduction to Data Science

TABLE 2: Number of movies per user rating



Looking at Table 3, it does not seem warranted to include this set of titles to base predictions, as we see that the list of these 20 movie titles only received one user rating each respectively.

```
# Plot Ratings Users - Number of Ratings in blue color font
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  xlab("Number of Ratings") +
  ylab("Number of Users") +
  ggtitle("User provided Number of Ratings")
```

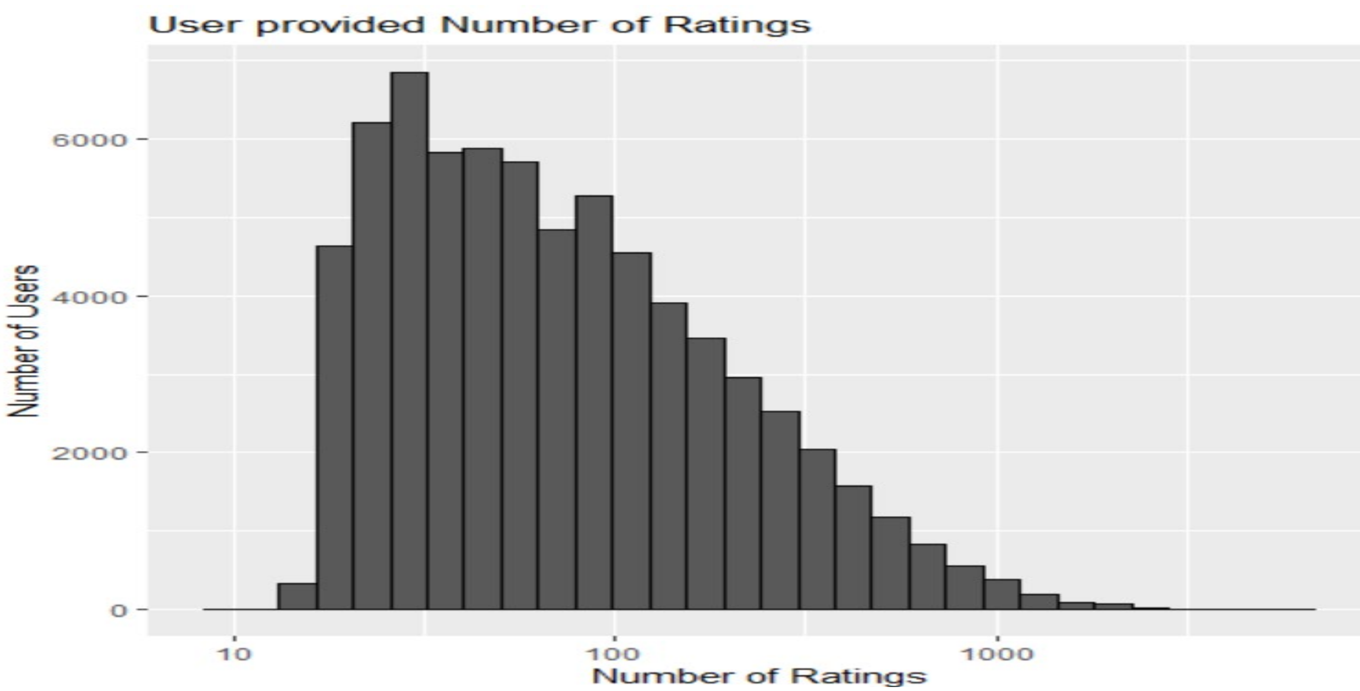
TABLE 3: Movies receiving only one user rating should be treated as anomalies

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
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

Looking at Table 4, it can be observed that the vast number of users have rated movies within 35 and 100 movies. Since there is a wide range, a user penalty component will need to be factored into the prediction model.

```
# Plot Ratings Users - Number of Ratings in blue color font
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  xlab("Number of Ratings") +
  ylab("Number of Users") +
  ggtitle("User provided Number of Ratings")
```

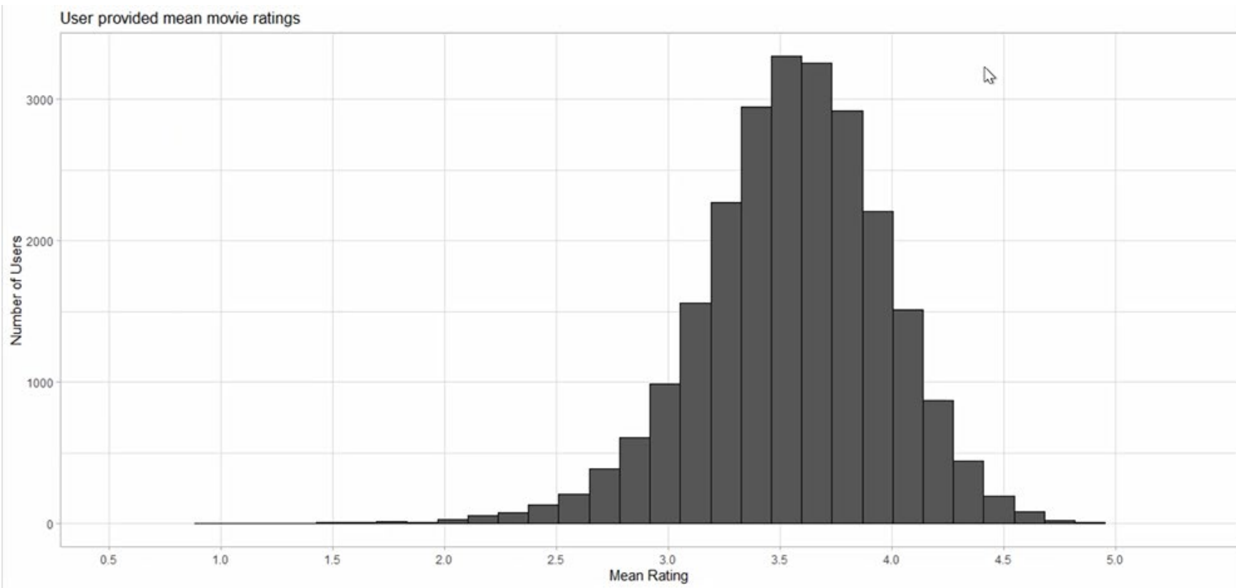
TABLE 4: Number of movie ratings provided by users



Looking at Table 5, this illustrates users whom have rated at least 100 movies or more. What can be concluded is that users are not consistent with providing movie ratings across the board of movies.

```
# Plot Ratings Users - Mean
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("User provided mean movie ratings") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```

TABLE 5: User provided mean movie ratings



We will first look at the average movie rating model (mean rating) which predicts the rating for all movies across the dataset. The expected rating of the dataset is between 3 and 4. This simple recommendations system works under the criteria that the same rating for all movies is predicted agnostic to the user.

A model based approach assumed the same rating for all movies with differences accounted by random variation:

$Y_{u,i} = mu + e_{u.i}$; where $e_{u.i}$ is independent error sample for the same distribution centered at 0 and mu the true rating for all movies.

```
> ## Simple Prediction based on Mean Rating
> mu <- mean(edx$rating)
> mu
[1] 3.512465
```

Now, if all unknown ratings with the mean rating, mu is predicted, we can calculate the naïve RMSE as follows:

```
> naive_rmse <- RMSE(validation$rating, mu)
> naive_rmse
[1] 1.061202
```

 Reference List: Irizarry, A. R. (2015). Introduction to Data Science

We will use the Naïve RMSE in Table 6 to compare our prediction effectiveness. To improve our prediction methodology

```
> # Validate and Save Results in date frame called rmse_ouputs
> rmse_ouputs <- data_frame(method = "Average movie rating model", RMSE = naive_rmse)
Warning message:
`data_frame()` is deprecated, use `tibble()`.
This warning is displayed once per session.
> rmse_ouputs %>% knitr::kable()
```

TABLE 6: Naïve RMSE (simple prediction of the average rating)

```
|method                     |     RMSE|
|:--------------------------|--------:|
|Average movie rating model | 1.061202|
```

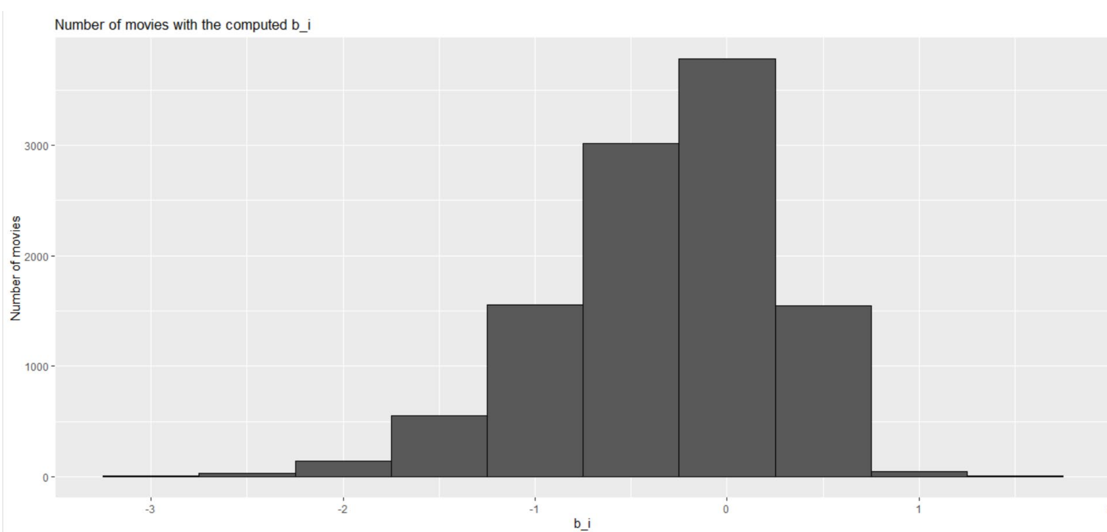
REFINING THE PREDICTION MODEL: Movie Effect

Movie-specific effect is approximated by

> $Y_{u,i} = mu + b_i + e_{u.i}$; where is "b" is bias for each movie "i" that represents the average ranking for movie i

```
# Simple model considerating movie effect b_i
# Subtract the rating minus the mean for each rating the movie received
# Plot number of movies with the computed b_i
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
                     ylab = "Number of movies", main = "Number of movies with the computed b_i")
```

TABLE 7: Number of movies with b_i



Reference List: Irizarry, A. R. (2015). Introduction to Data Science

The penalty incorporated into the movie effect will now be used to improve the prediction model.

```
# Test and save rmse results
predicted_rating_values <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
rmse_model1 <- RMSE(predicted_rating_values, validation$rating)
rmse_ouputs <- bind_rows(rmse_ouputs,
                          data_frame(method="Movie effect model",
                                     RMSE = rmse_model1 ))
```

TABLE 8: Predicted movie rating based on movies rated inconsistently adding b_i to mu

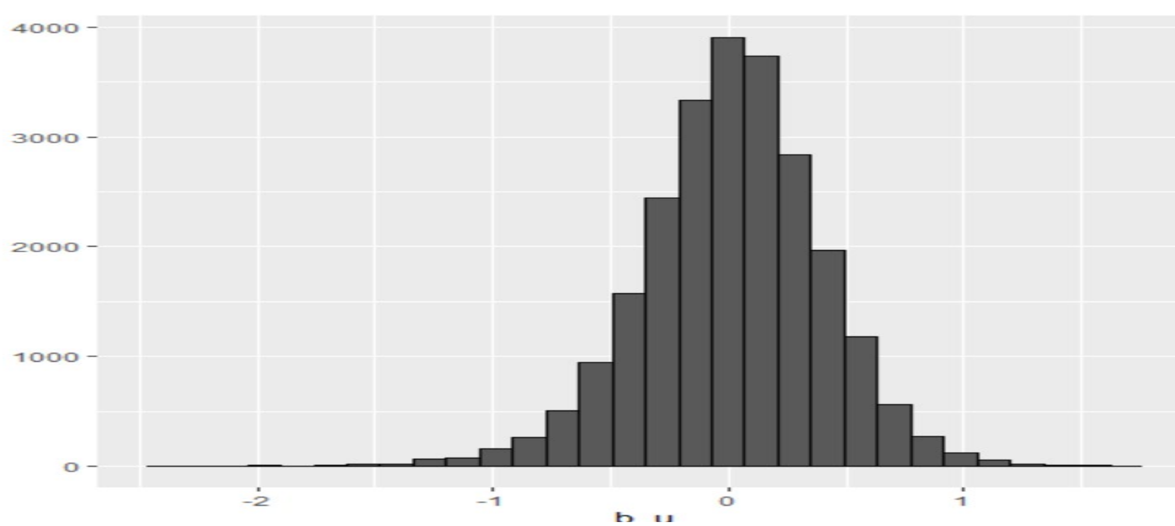
```
|method                    |      RMSE|
|:-------------------------|---------:|
|Average movie rating model | 1.0612018|
|Movie effect model        | 0.9439087|
```

REFINING THE PREDICTION MODEL: Movie Effect & User Effect Model

Now look to calculate the average user rating, mu for those users whom have rated over 100 movies with the incorporation of the penalty term. Looking from Table 9, there is apparent user provided variance across movie ratings.

```
# Plot penalty term user effect #
user_avgs<- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```

TABLE 9: average user ratings (mu) for movies rated over 100+



There is further opportunity to refine the model to mitigate large swings in ratings whereby a user with a negative b_u rates a movie with a positive b_i will result in a negating effect and so will use the following formula below to predict that the user in the aforementioned example gave the movie a 3 score versus a 5.

Calculating Average of b_u by way of $Y_{u,i}$ - mu + b_i + $e_{u.i}$; where a and n approximation is calculated for mu and b_i

From Table 10 below, iterative refinement in prediction models as reduced RMSE value. However, this model is still from perfect as bi estimates (either negative or positive) are likely to increase RMSE values. To enhance the approach, we will need one prediction value not confidence intervals with associated stand error for various levels of uncertainty.

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


# Test and save rmse results
predicted_rating_values <- validation%>%
left_join(movie_avgs, by='movieId') %>%
left_join(user_avgs, by='userId') %>%
mutate(pred = mu + b_i + b_u) %>%
pull(pred)

model_2_rmse <- RMSE(predicted_rating_values, validation$rating)
rmse_ouputs <- bind_rows(rmse_ouputs,
data_frame(method="Movie and user effect model",
RMSE = model_2_rmse))

# Check result
rmse_ouputs %>% knitr::kable()
```

TABLE 10: average user ratings (mu) for movies rated over 100+

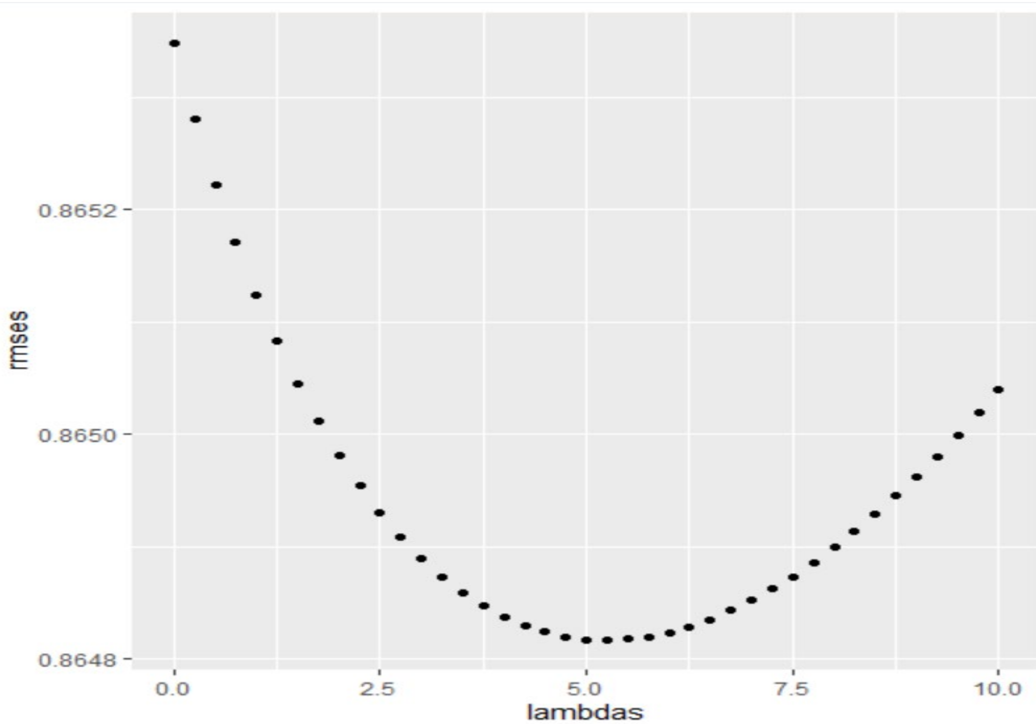
method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

REFINING THE PREDICTION MODEL: Regularized movie effect and user effect model

From Table 11 below, using regularization to penalize the fact that some users only rated a small set of movies as well as the fact that movies may receive few ratings.

```
# For each lambda,find b_i & b_u, followed by rating prediction & testing
# note:the below code could take some time
rmses <- sapply(lambdas, function(l){

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))

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

return(RMSE(predicted_rating_values, validation$rating))
})


# Plot RMSE against Lambdas to find optimal lambda
qplot(lambdas, rmses)
```

TABLE 11: Optimal lambda selection (RMSE compared to lambdas)



The calculated optimal lambda value is 5.25 as denoted below.

```
> # The optimal lambda
> lambda <- lambdas[which.min(rmses)]
> lambda
[1] 5.25
```


Reference List: Irizarry, A. R. (2015). Introduction to Data Science

Table 12: New results factoring optimal lambda value of 5.25

```
> # Test and save results
> rmse_ouputs <- bind_rows(rmse_ouputs,
+ data_frame(method="Regularized movie and user effect model",
+ RMSE = min(rmses)))
>
> # Check result
> rmse_ouputs %>% knitr::kable()


|method                                 |      RMSE|
|:--------------------------------------|---------:|
|Average movie rating model             | 1.0612018|
|Movie effect model                     | 0.9439087|
|Movie and user effect model            | 0.8653488|
|Regularized movie and user effect model | 0.8648170|
```

3. Results

Illustrated below in Table 13, are all of the RMSE values from the iterative prediction models that were utilized throughout this report for comparison:

Table 13: RMSE values associated to respective prediction models

```
> #######################################################################
> # Final Results - MovieLens Dataset
> #######################################################################
> # RMSE Results - Providing the appropriate score given the reported RMSE
> rmse_ouputs %>% knitr::kable()


|method                                 |      RMSE|
|:--------------------------------------|---------:|
|Average movie rating model             | 1.0612018|
|Movie effect model                     | 0.9439087|
|Movie and user effect model            | 0.8653488|
|Regularized movie and user effect model | 0.8648170|
```

It can be concluded that the lowest RMSE value obtained is 0.8648170.

4. Conclusion

It was observed that throughout this project, there was a test and learn approach to iteratively use various predictive models to seek the most accurate method, which in this case is the regularization model; this incorporated the movie and user effect. The objective to achieve a Root Mean Squared Error (RMSE) less than 0.8649 was successfully attained, specifically a RMSE value of 0.8648170 was concluded

Reference List: Irizarry, A. R. (2015). Introduction to Data Science