Project 1 June 16 2019

Virginia Howarth

June 16, 2019

1. INTRODUCTION

The objective of the project is to make a movie recommendation system for movies not yet rated by users. The key steps performed are to clean the data, analyse the data to gain some insights and to train a machine learning algorithm on the training set to predict movie recommendations in the validation set. The data set is the 10M version of the Movielens dataset of recommendations of 10681 movies by 69878 users.

```
setwd("c:/Users/Virginia Howarth/Documents/R/Project 1")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-projec
t.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------------------------------
---- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0       v purrr   0.3.0
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.2       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------ t
idyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(dplyr)
```

Create the training and validation sets.

The training set is 90% of the data while the Validation set will be 10% of MovieLens data.

```
edx <- read.csv("edx data.csv")
validation <- read.csv("validation data.csv")
```

[Below is to code used in R code which could not work in Rmarkdown. Error message was Vector of 68Mb could not be allocated``{r datadownload} dl <- tempfile() download.file (“http://files.grouplens.org/datasets/movielens/ml-10m.zip (http://files.grouplens.org/datasets/movielens/ml-10m.zip)”, dl)

ratings <- read.table(text = gsub(“::”, “”, 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”)

Create the test index and test and training sets 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)]

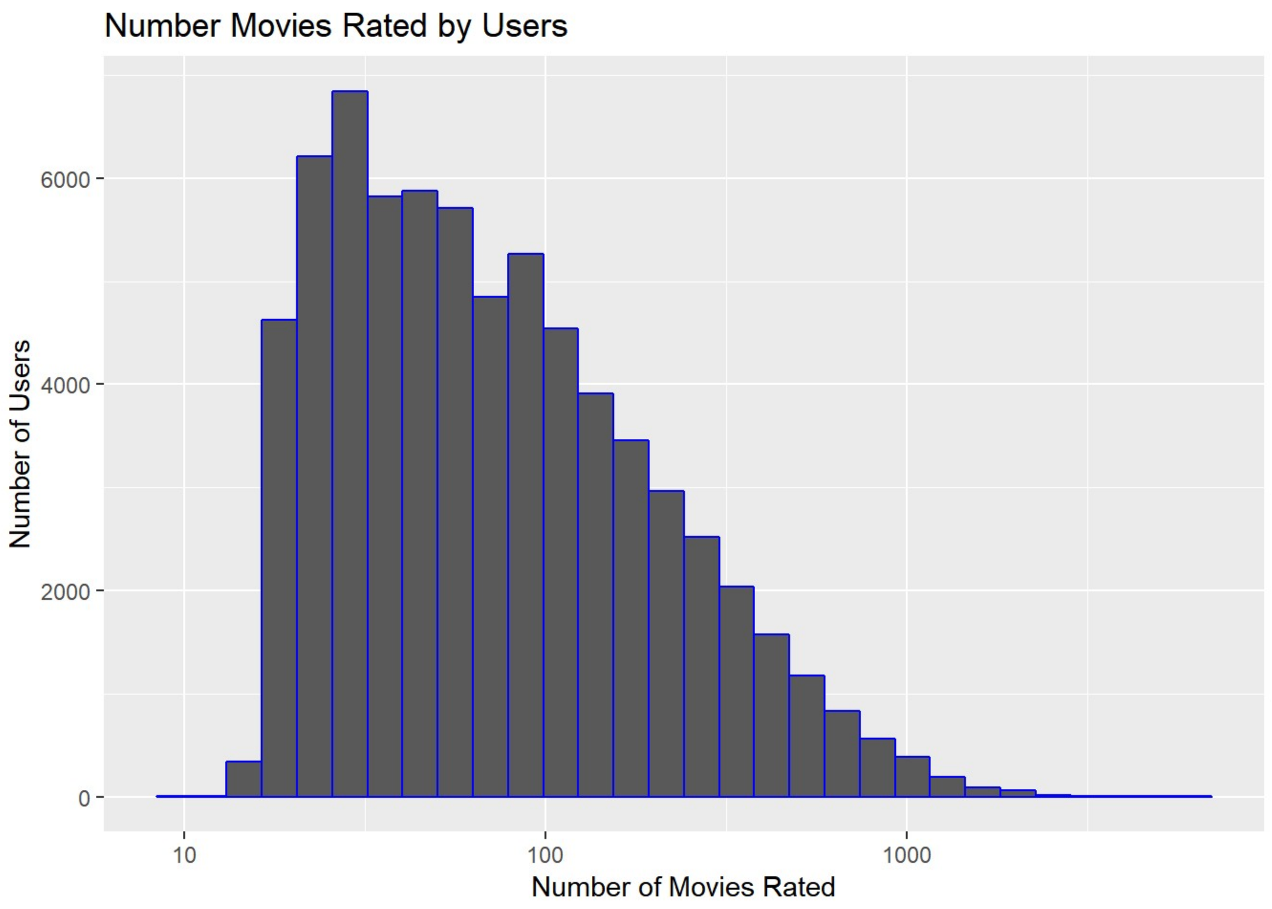
2. ANALYSIS OF THE DATA AND INSIGHTS

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

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

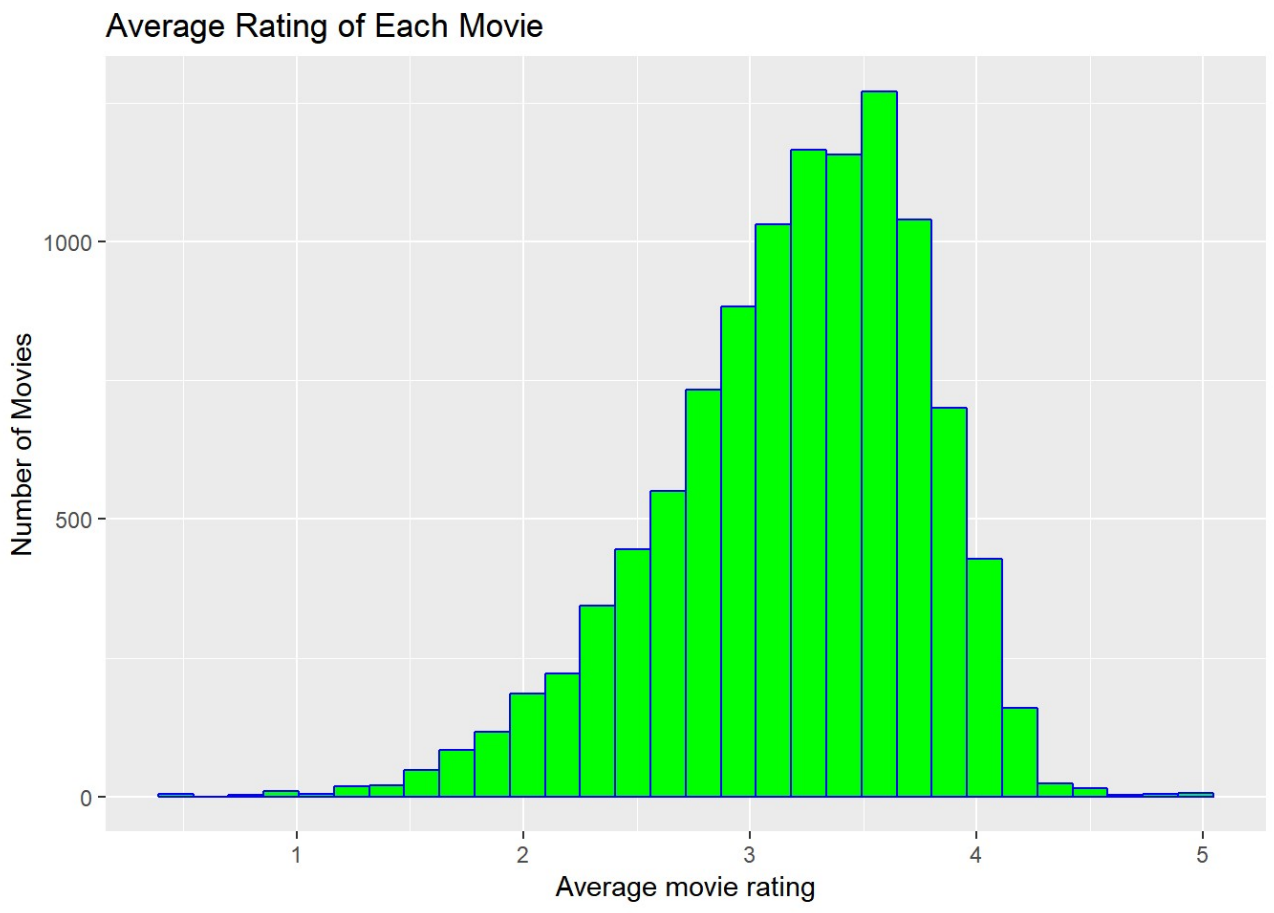
The number of movies in the data set is 10,677 and the number of users is 69,878.

```
edx %>%
     dplyr::count(userId) %>%
     ggplot(aes(n)) +
     geom_histogram(bins = 30, color = "blue") +
     scale_x_log10() +
     labs(x = "Number of Movies Rated", y = "Number of Users") +
     ggtitle("Number Movies Rated by Users")
```



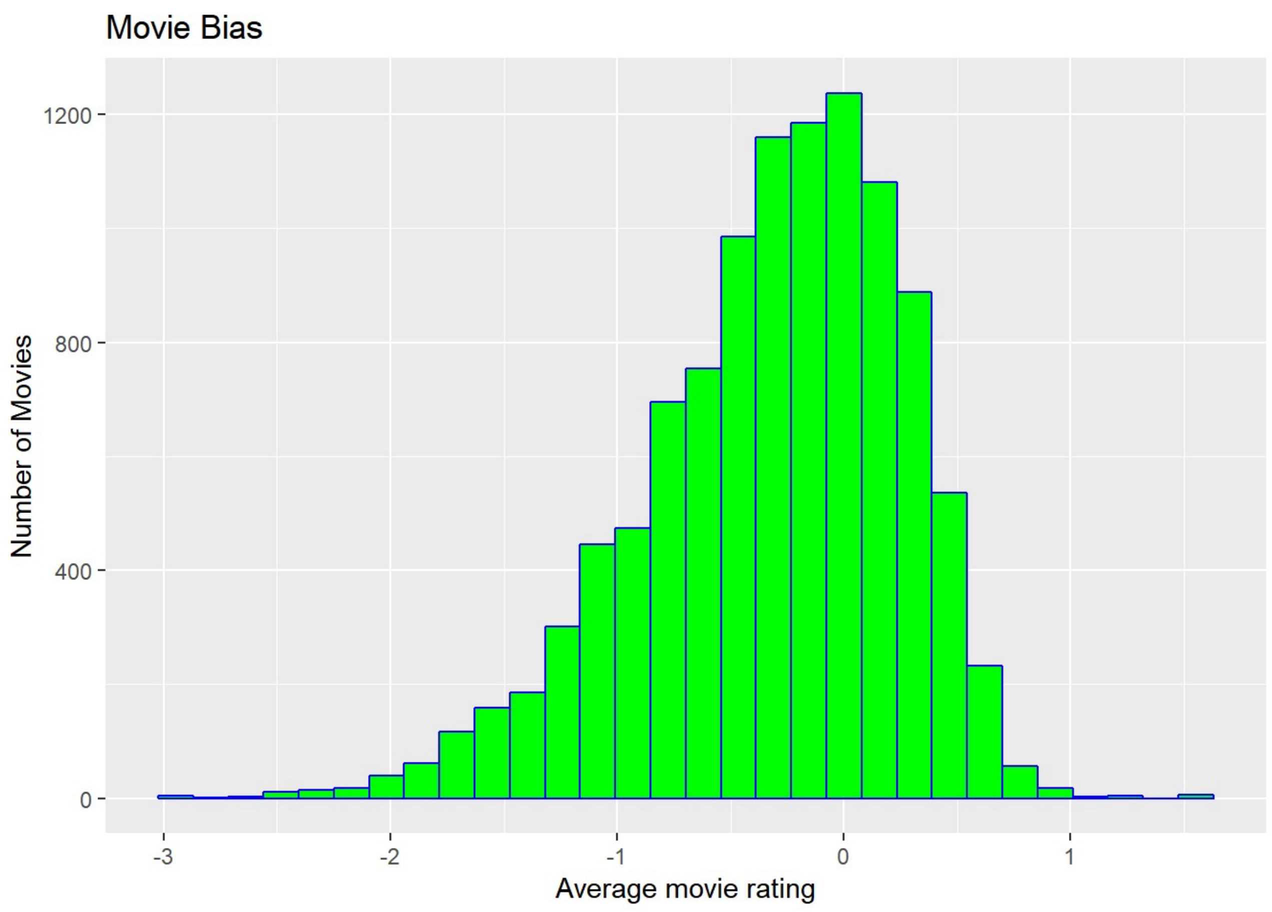
Some movies are rated by more users than others as seen with the number of times a movie is rated on the y axis and the number of movies on the x axis. We can see that there are only a small number of users who rate fewer than 10 movies. There are the highest number of users rating between 10 and 100 movies. There are a diminishing number of users rating between 100 and a 1000 movies while far fewer users rated more than 1000 movies.Here one movie, probably a blockbuster is rated about 3000 times.

```
edx %>%
  group_by(movieId) %>%
  summarize(mu = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(mu)) +
  labs(y = "Number of Movies", x = "Average movie rating",
       title = "Average Rating of Each Movie") +
  geom_histogram(bins = 30, color = "blue", fill = "green")
```



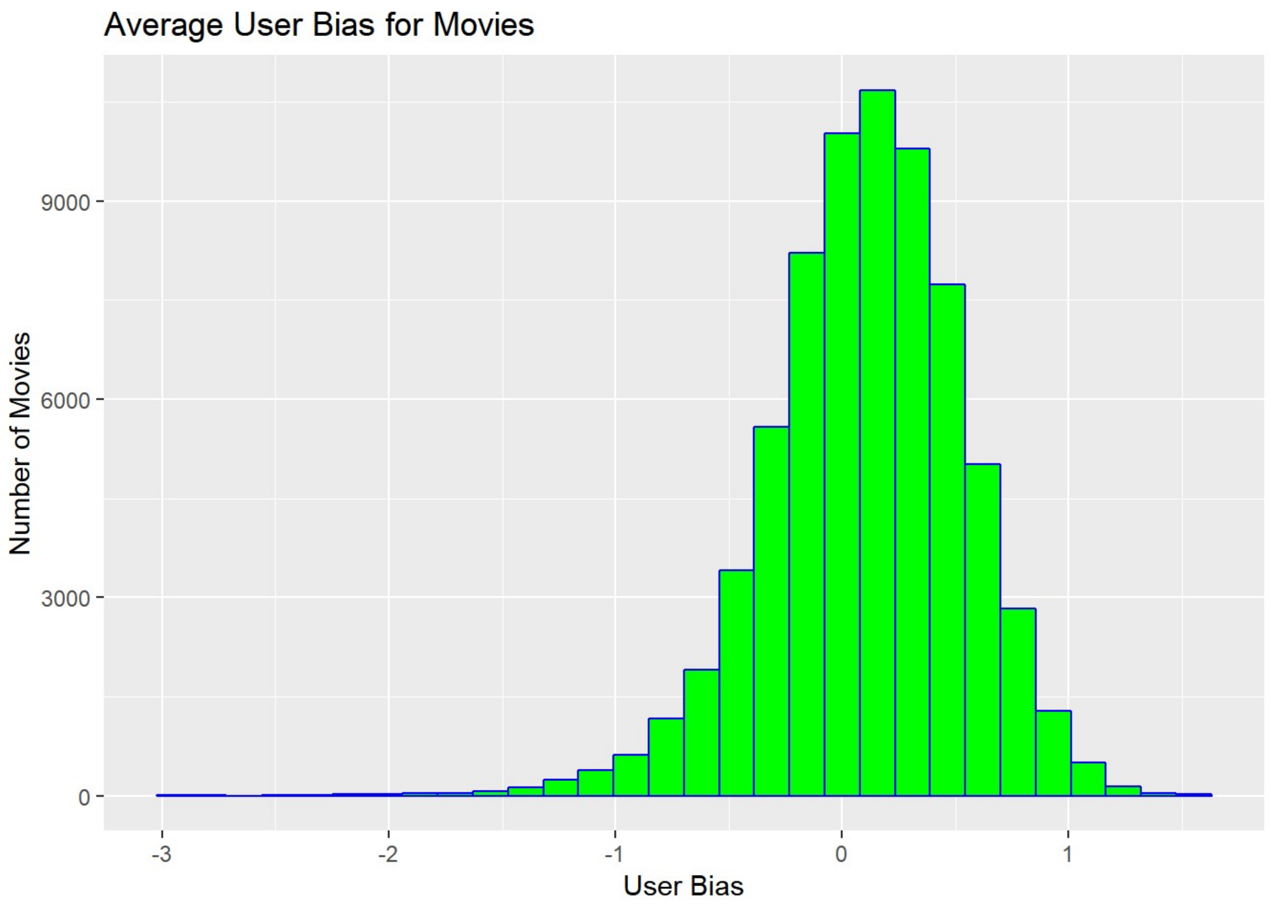
The average rating given to movies is 3.51. A small number of movies achieve a rating of 5 or close to it and similarly for ratings close to zero.

```
mu <- mean(edx$rating)

movie_avgs <- edx %>% group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

movie_avgs %>%
  filter(n()>=100) %>%
  ggplot(aes(b_i)) +
  labs(y = "Number of Movies", x = "Average movie rating",
       title = "Movie Bias") +
  geom_histogram(bins = 30, color = "blue", fill = "green")
```



The movie bias shows the bias of the rating for each movie relative to the average movie rating. So a bias of 1 or greater would mean that the movie was rated move than 4.5. A movie bias of less than -2 would indicate the movie has a rating of less than 1.5.

```
edx %>%
     group_by(userId) %>%
     summarize(b_u = mean(rating - mu)) %>%
     filter(n()>=100) %>%
     ggplot(aes(b_u)) +
     labs(x = "User Bias",y = "Number of Movies", title = "Average User Bias for Movie
s")+
     geom_histogram(bins = 30, color = "blue", fill = "green")
```



This illustrates that users also have a bias. Some users rate most movies very poorly (eg. around 1 or 2) while other users rate movies, on average, very highly (eg. closer to 4 or even 5).

3. BUILD A MODEL TO PREDICT THE RATING OF EACH USER

Given the insights above that each movie has a rating bias, each user has a user bias and there is a large divergence in the number of movies that each user rates, we can build a model to predict the rating a user will give to a movie who has not yet rated a particular movie. The reason for doing this is to bring movies to the attention of customers which they have not yet seen and which they are expected to rate highly so that the consumers are likely to purchase these movies presented and to increase sales for the company presenting these.

The method is to start with a baseline expectation of the average rating for any movie. Then we will add in the movie bias and the user bias which is a matrix factorization which is optimised in order to minimise the residual mean squared error.

Y = mu + b_i + b_u + e

Finally we will regularise the movie ratings based on the number of movies rated by each user. The purpose of this process is to reduce the ratings from users who have rated very few movies as these are likely to be less robust than ratings by users who rate many movies. After each modification of the model we will review the Residual Mean Squared Error of the Model.

Set up the Recommendation Evaluation Model

```
#Create the recommendation system
library(caret)
#Create the function for the RMSE
RMSE <- function(true_ratings, predicted_ratings){
     sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

The first step is to establish the function to calculate the residual mean squared error.

“Just the Average” model.

```
#"Just the Average model"
mu <- mean(edx$rating)
naive_rmse <- RMSE(validation$rating, mu)
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)
rmse_results
```

```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average  1.06
```

Next we evaluate a baseline model of Just the average movie rating which can also be referred to as the naive approach.

Incorporate the movie bias

```
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,
                          tibble(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

The next step is to adjust the ratings for the movie bias. Adding the movie bias to the model improves the RSME from 1.06 for “Just the Average” 0.94.

Adjust for the User bias

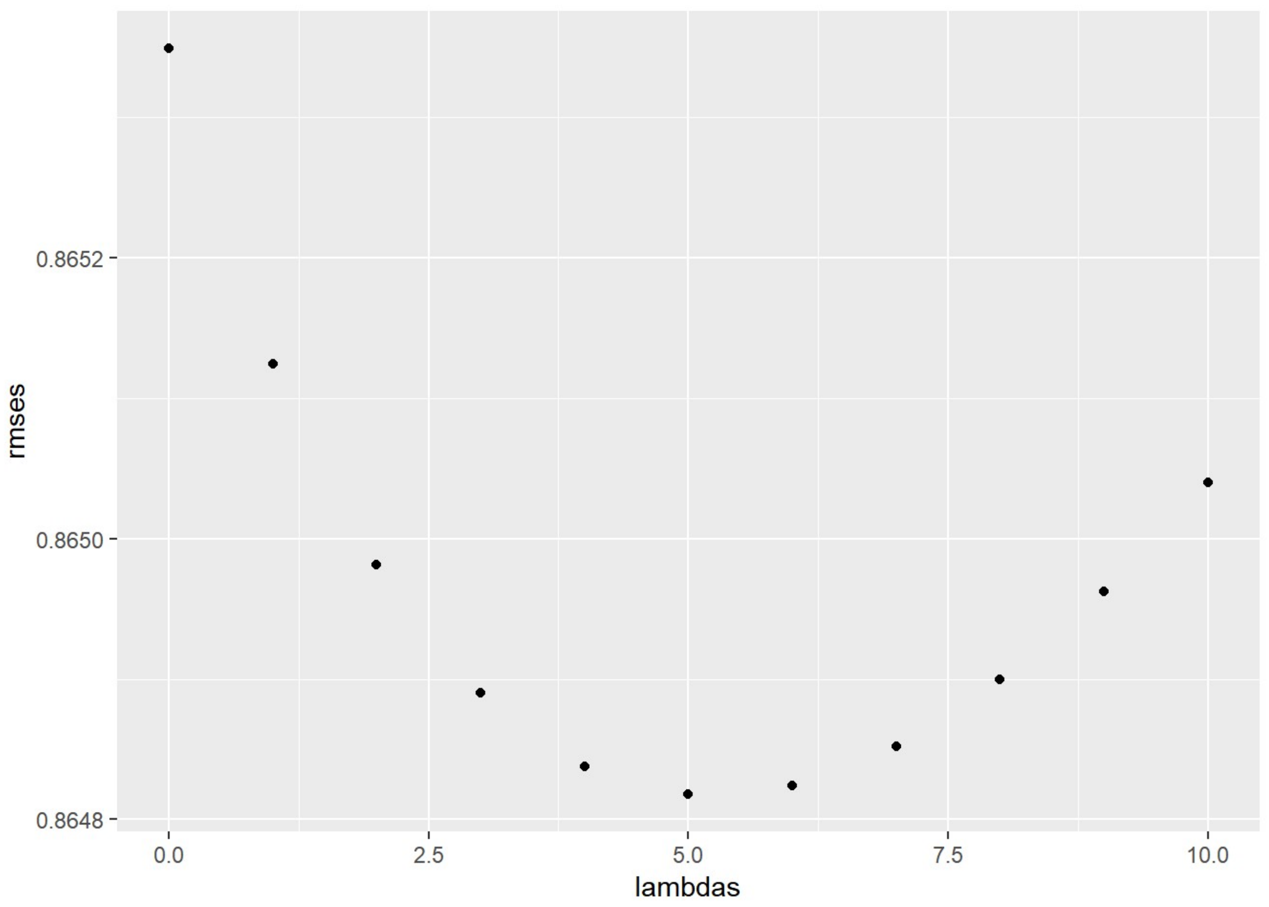
```
user_avgs <- edx %>%
     left_join(movie_avgs, by='movieId') %>%
     group_by(userId) %>%
     summarize(b_u = mean(rating - mu - b_i))

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,
                          tibble(method="Movie + User Effects Model", RMSE = model_2_r
mse))
rmse_results %>% knitr::kable()
```

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

By incorporating the user bias we see that “The Movie + User Effects model”" achieves a further significant improvement in the RMSE to 0.8653.

####Regularise the model for the number of ratings provided by each user.

```r
lambdas <- seq(0, 10, 1)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```



Finally we regularise the model.We have the tuning parameter lambda and we test the RMSE for lambdas between 0 to 10 in increments of 1.

```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5
```

The optimal lambda is 5.

```
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie + User Effect Model",
                                      RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

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

method	RMSE
Regularized Movie + User Effect Model	0.8648177

The addition of the regularisation test achieves a smaller incremental improvement to the RMSE of the model to 0.8648.

RESULTS

Just the Average: just the average achieved a residual mean squared error of 1.06. Model 1: adjusting for the movie bias achieved a reduction in the RMSE to 0.9439. Model 2: added the user bias to the prior model with the movie bias and reduced the RMSE further to 0.8653. Model 3: RMSE of 0.8648 was achieved after when regularising for the number of users.

CONCLUSION

Using matrix factorisation, we can predict the rating of a user of a movie which has not yet been rated by the user. We adjusted for two biases in the rating relative to the average rating which are the movie and the user bias. Then we regularised the ratings for the number of movies rated by each user. This achieved a sigificantly improved score of 0.8648.