MovieLensProject_RMSERatingPrediction

Netaverner

27/11/2020

Project Overview

This is the project for HarvardX PH125.9x Data Science: Capstone submission. The project makes use of the MovieLens 10M dataset supplied by HarvardX which is downloaded from "http://files.grouplens.org/datasets/movielens/ml-10m.zip". The project initially starts with a brief overview of the goals of the project followed by a setup and a preperation of the MovieLens 10M dataset. An data analysis will be displayed and carried out on the dataset, as to create a machine learning algorithm that can predict movie ratings. This algorithm will proceed in a step by step manner until as satasfactory RMSE value is reached. The results will be analysed and explain. Finally the project will conclude with a brief study the projects achievements and thoughts of where future improvemts could be made.

Introduction

The creation of a movie recommedation system using the 10M version of MovieLens dataset will be the main focus of this project. The success of the movie recommedation system will be measured by the Root Mean Square Error [RMSE] value scored by the method/algorithm. A RMSE value of less than 0.86549 will be seen as a success.

Recommendation systems make use of ratings provided by users when rating items under specific recommendation critera. Many companies, such as Amazon make use of recommendation systems by collecting masses of data from their users. By collecting the users ratings of Amazon's items, Amazon is able to use this data to predict how users will rate or feel about certain items. This way Amazon is able to display items to their users that they know that their users will like, or rate highly.

Similarly to how Amazon can predict what items their users will like so can be done for other cases. This included with the inspiration aquired from the The Netflix prize. (The Netflix Price was an open competition put out to the Data Science comunity to create a filtering algorithm to predict user ratings for Netflix films, based on previous user ratings.) This project aims to similarly create a method to predict a movies rating from the 10M version of MovieLens dataset.

Thus this project will focus on creating a movie recommendation system for the 10M version of MovieLens dataset.

Aim of Project

The project aims to train a moive predictiting algorithm (machine learning algorithm), that can accurately predict a users rating (between 0.5 to 5 stars) of a movie. The algorithm will be trained using the provided edx dataset which is a subset of the 10M version of MovieLens dataset. The algorithm's predicting ability will be assesed by testing its ablility to predict movie rating in the provided validation set.

The performance of the algorithm will be evaluated by using the RMSE of algorithm. RMSE is a commonly used measurement of the differences between predicted values and observed values. RMSE is a measurement of the accuracy of an algorithm. Accuracy of a model/algorithm is measured by comparing the forecasing erroes of a model for a particular dataset. A lower RMSE value is better than a high value as lower RMSE are indicating the models predictions a more accurate. Large errors have a signifficantly greater impact on RMSE, this is due to the effect of each error on RMSE being proportional to the size of the error squared. Thus also makig RMSE sensitive to outliers.

Within the aim of the project, mutiple models will be created until an acceptable RMSE value for a models is found.

The function that computes the RMSE for vectors of movie ratings and their corresponding predictors is as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

Dataset Initialisation

The code below is provided in the HarvardX PH125.9x Data Science: Capstone project module [Create Train and Validation Sets]: https://learning.edx.org/course/course-v1:HarvardX+PH125.9x+2T2020/block-v1:HarvardX+PH125.9x+2T2020+type@sequential+block@e8800e37aa444297a3a2f35bf84ce452/block-v1:HarvardX+PH125.9x+2T2020+type@vertical+block@e9abcdd945b1416098a15fc95807b5db

```
##########################################################
# Create edx set, validation set (final hold-out test 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-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

#Selecting Librarys to use
library(tidyverse)
library(caret)
library(data.table)
library(dplyr)
library(dslabs)
library(tidyverse)
library(ggplot2)
library(lubridate)

# ##Downloading dataset and setting up training and testing sets according to EDX given instructions

# # 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 <- 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)
```

```r
colnames(movies) <- c("movieId", "title", "genres")

# if using R 3.6 or earlier:
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],title = as.c


# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


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

The MovieLens dataset is split into 2 subsets that will be the "edx", which will be the training subset, and "validation" a subset to test the movie ratings.

Algorithm design and development must be only carried out on the "edx" subset, as the "validation" subset will be used for testing this algorithm. This is done so that one is not testing what is already known as this is bad practices and will not give a real look at how the algorithm will perform with unknown data.

```r
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `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)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Data Analysis

Once the edx subset has been cleaned, it is good practice to view the subset features and calculate basic summary statistics.

```r
# intial 7 rows with header
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1:                Comedy|Romance
```

```
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```r
#basic summary
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
##
##
##
```

From these we can see the subset is in a tidy formate and is therefore ready for exploration and analysis

Quiz: MovieLens Dataset

Q1

```r
#number rows & number cols
dim(edx)
```

```
## [1] 9000055       6
```

Q2

```r
#num zeros
print("Number of Zeros")
```

```
## [1] "Number of Zeros"
```

```r
sum(edx$rating == 0)
```

```
## [1] 0
```

```r
print("Number of Threes")
```

```
## [1] "Number of Threes"
```

```r
#num threes
sum(edx$rating == 3)
```

```
## [1] 2121240
```

Q3

```r
#number movies

numberMovies <- edx %>% group_by(movieId) %>% summarise(numberRatings = n())
nrow(numberMovies)
```

```
## [1] 10677
```

Q4

```r
#number users

numberUsers <- edx %>% group_by(userId) %>% summarise(numberRatings = n())
nrow(numberUsers)
```

```
## [1] 69878
```

Q5

```r
# need to split up genres in edx first (problems with " |  ")
#call
edxSplitGenre <-  edx  %>% separate_rows(genres, sep = "\\|")

#number ratings per genre -- > use edx spited by genre

rateGenre <- edxSplitGenre %>% group_by(genres) %>% summarise(count  = n())%>%
  arrange(desc(count))
rateGenre
```

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
```

```
## 20 (no genres listed)       7
```

Q6

```r
#highest rated movie

ratingMovies <- edx %>% group_by(movieId) %>%
  summarize(numRatings = n(), title = first(title)) %>%
  arrange(desc(numRatings)) %>%
  top_n(10, numRatings)

ratingMovies
```

```
## # A tibble: 10 x 3
##    movieId numRatings title
##      <dbl>      <int> <chr>
##  1     296      31362 Pulp Fiction (1994)
##  2     356      31079 Forrest Gump (1994)
##  3     593      30382 Silence of the Lambs, The (1991)
##  4     480      29360 Jurassic Park (1993)
##  5     318      28015 Shawshank Redemption, The (1994)
##  6     110      26212 Braveheart (1995)
##  7     457      25998 Fugitive, The (1993)
##  8     589      25984 Terminator 2: Judgment Day (1991)
##  9     260      25672 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (19~
## 10     150      24284 Apollo 13 (1995)
```

```r
#can see pulp is at the top
```

Q7

```r
#most given rating
givenRating <- edx %>% group_by(rating) %>% summarise(num = n()) %>%
  arrange(desc(num))
givenRating
```

```
## # A tibble: 10 x 2
##    rating     num
##     <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
```

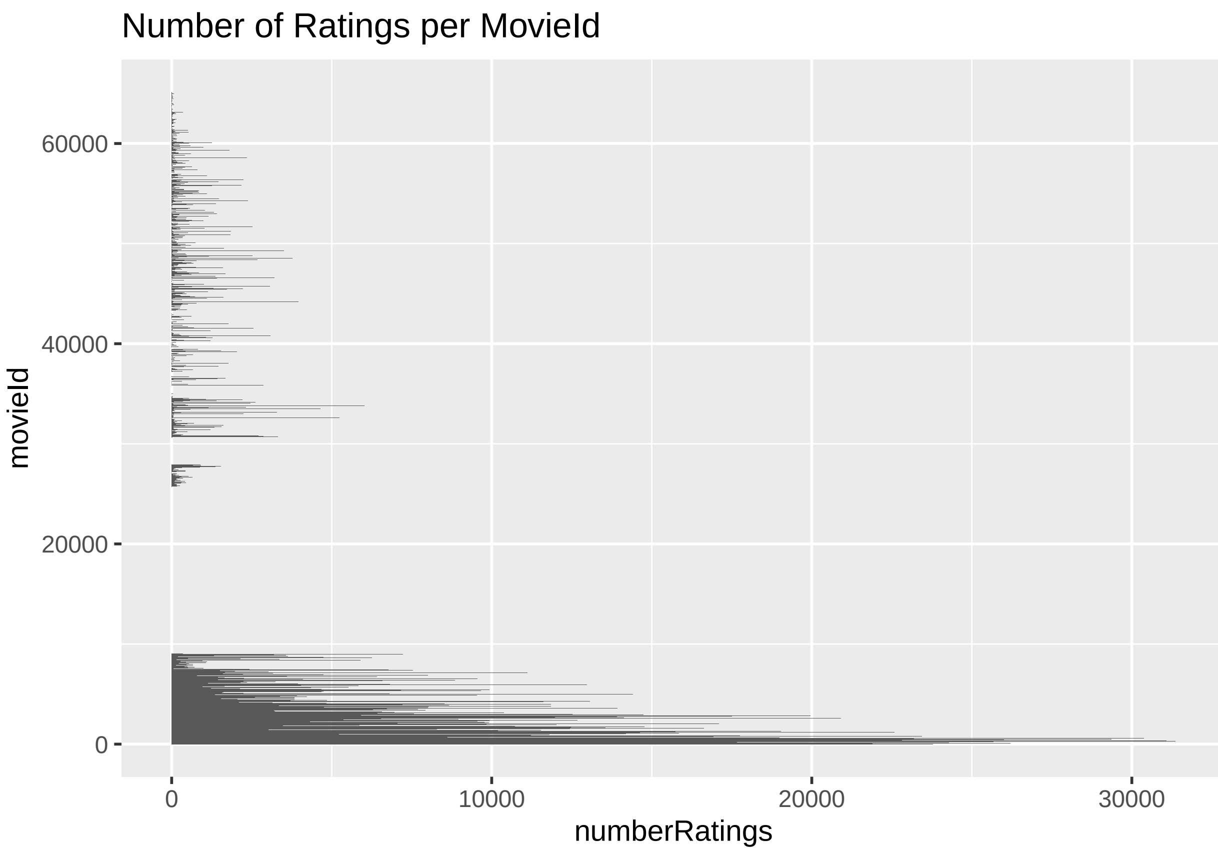
Futher Data Analysis

There is about 70.000 unique users and about 10.700 different movies in the edx subset:

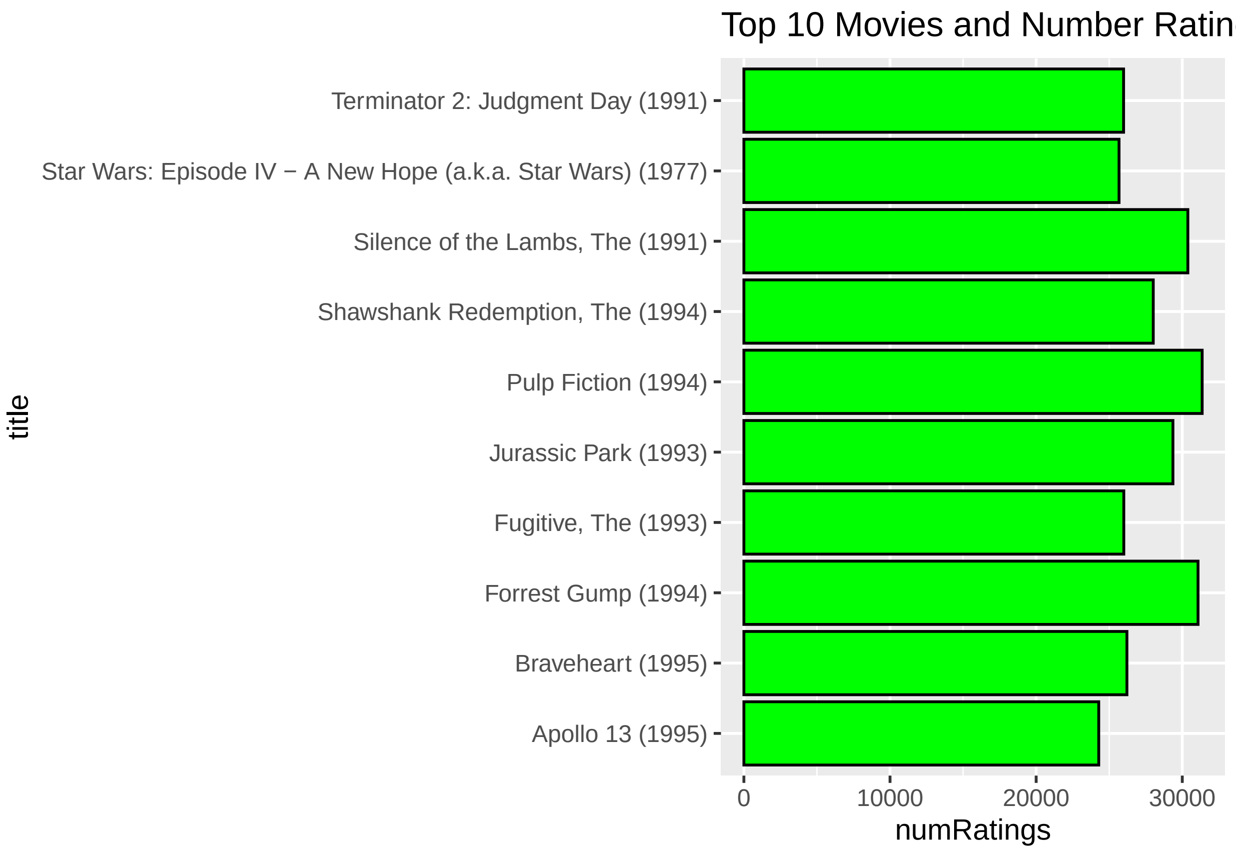
```
##   numberUsers numberMovies
## 1       69878        10677
```

Looking a movies and their ratings

Graph of ratings per moiveID/Title

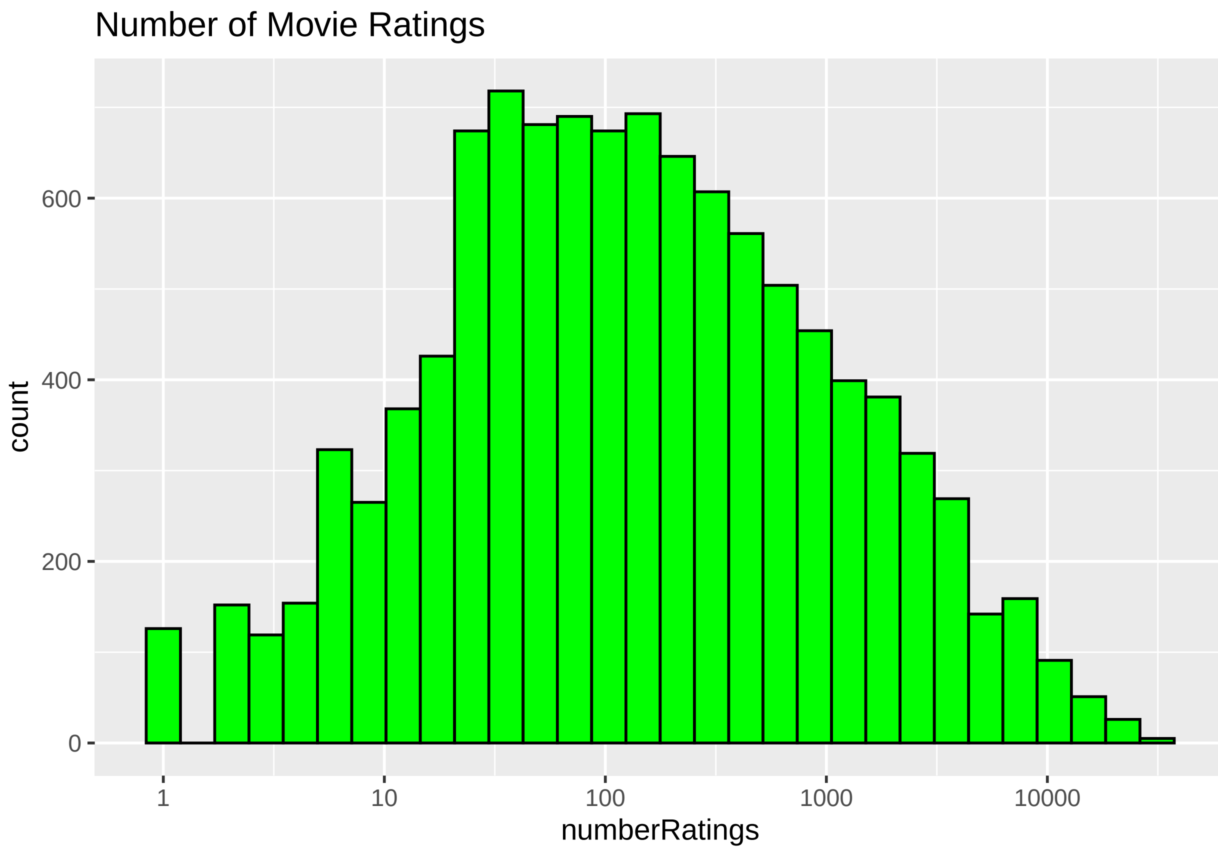


Can be seen that some movies have more ratings than others but this graph is not that usefull otherwise. graph below shows the number of ratings for top 10 movies for interest sake (and viewing summary data)



Graph of movie rating distribution

Information that would be more usefull to understand the edx subset and about the rating of the movies would be about how the number of ratings are distrubuted this can be seen in the graph bellow:



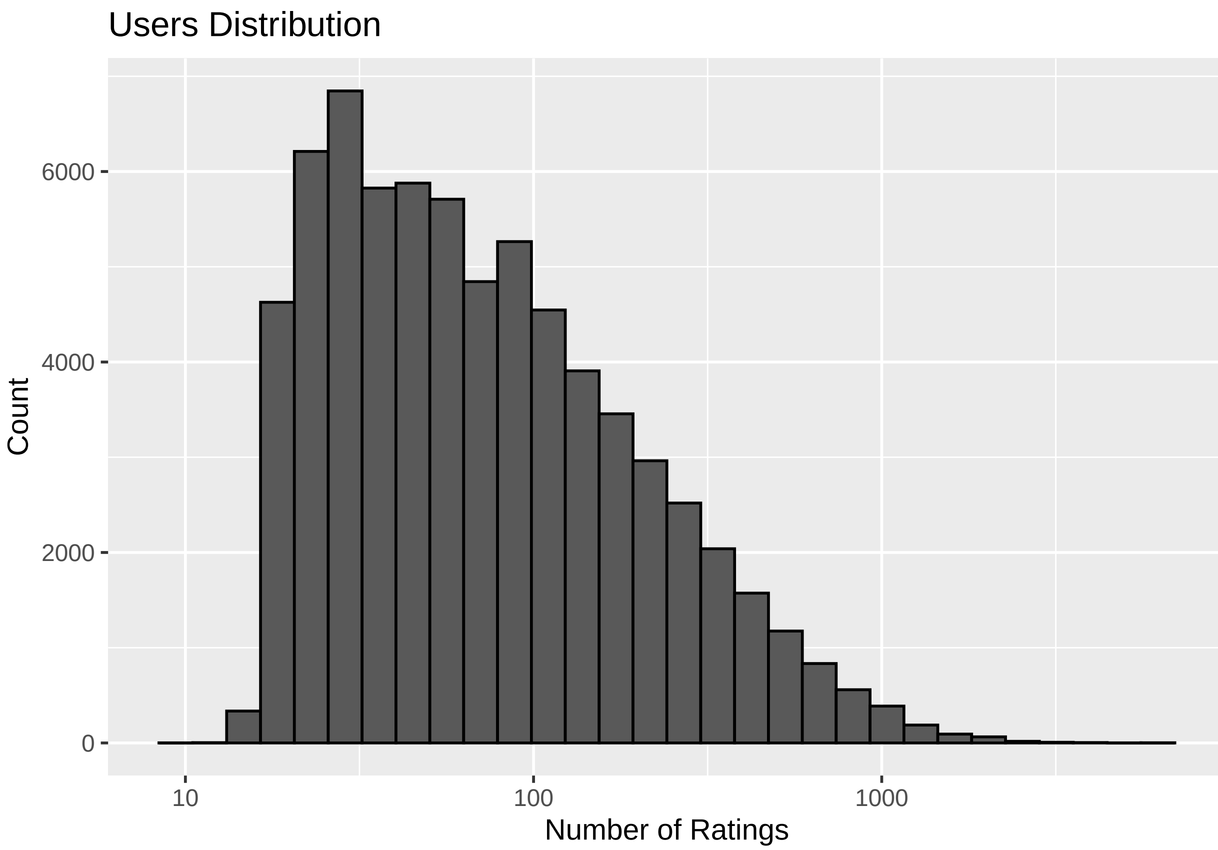
The graph above shows that some movies have been rated more times than others this creates a bais towards these movies. A regularisation and penalty term will need to be added to models as the reduce error caused due to the movies that have rarerly been rated.

From the previous graph it can be seen that there are a number of movies that have only been rated once these movies could, as previously learnt in the course, cause making predictions on ratings inaccurate the following movies displaced below are the singular rated movies - There are 126 of these movies only 10 of these movies are displayed in descending ratings.

title	rating	numberOfRatings
Blue Light, The (Das Blaue Licht) (1932)	5.0	1
Fighting Elegy (Kenka erejii) (1966)	5.0	1
Hellhounds on My Trail (1999)	5.0	1
Shadows of Forgotten Ancestors (1964)	5.0	1
Sun Alley (Sonnenallee) (1999)	5.0	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Demon Lover Diary (1980)	4.5	1
Kansas City Confidential (1952)	4.5	1
Ladrones (2007)	4.5	1
Man Named Pearl, A (2006)	4.5	1
Mickey (2003)	4.5	1
Please Vote for Me (2007)	4.5	1
Testament of Orpheus, The (Testament d'Orphée) (1960)	4.5	1
Tokyo! (2008)	4.5	1
Valerie and Her Week of Wonders (Valerie a týden divu) (1970)	4.5	1

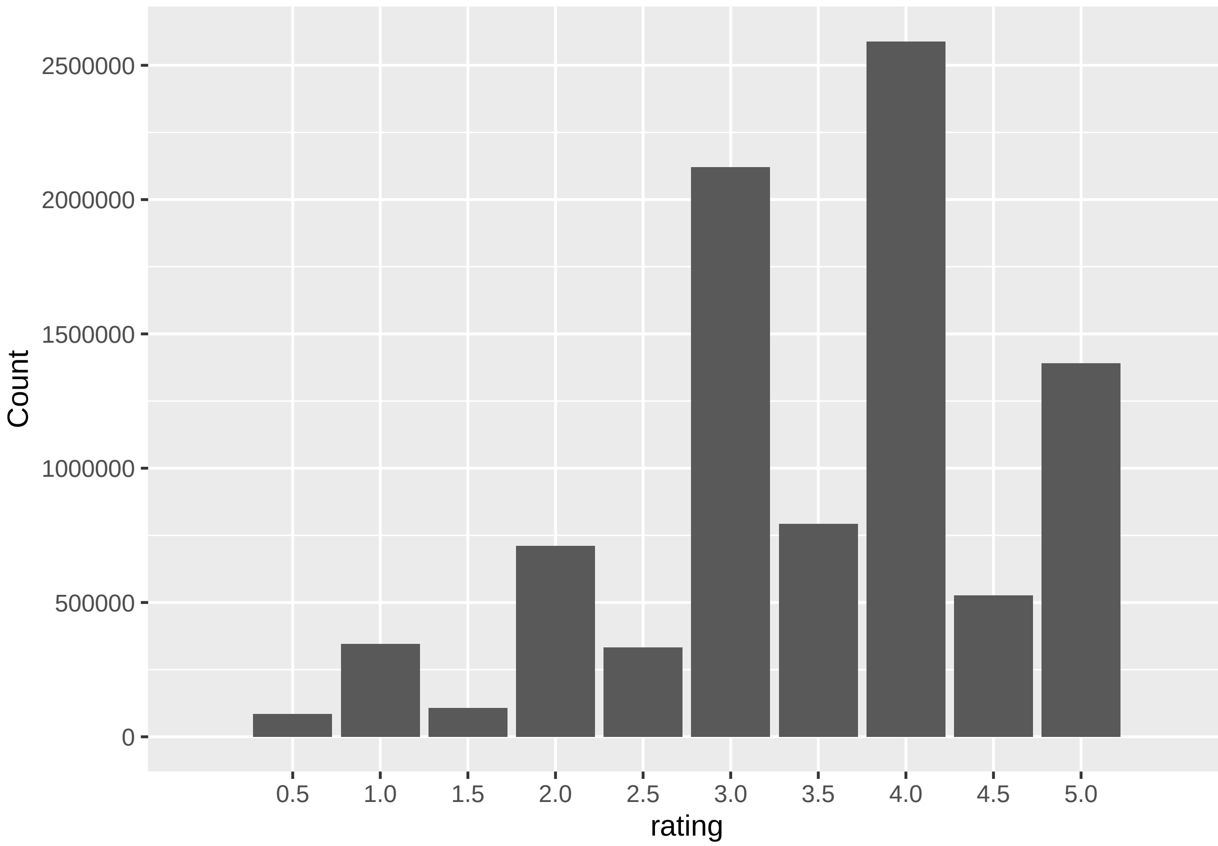
View of Users in the edx subset

Shown below is a graph of the distribution of number of ratings given by users. What can be observed is that the majority of users only rate between 40 and 100 movies. Also evident from the graph is that some users are more active than others. These two observations show that a user bais needs to be taken into account when making predictions.



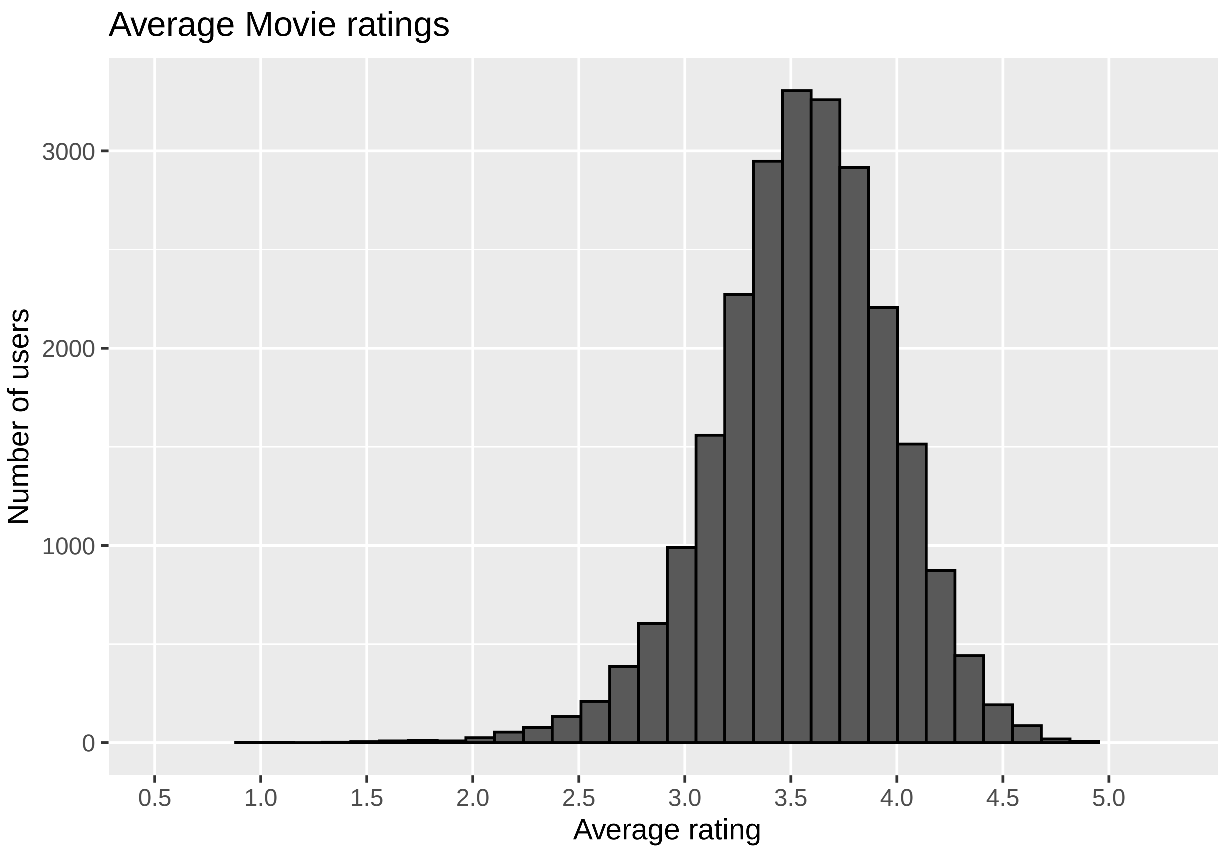
As seen in graph below users tend to rate movies generally higher stars than lower stars, this is evident as the rating of 4 is most common followed by 3 and 5. Also evident in the graph is that users tend to give mover full stars rating compared to half stars as can been seen that .5 ratings are less common and full star ratings.

```
givenRating %>% ggplot(aes(rating,num)) +
  geom_bar(stat="identity") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ylab("Count")
```



Some users also tend to be more particular with their rating than other users. This can be viewed in the graph below and can be seen that some users give movies a low rating where as others give high ratings. There are also users having rated a hundred or more movies these are used to construct the graph below, this is done as to show there is a trend with the high and low ratings.

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



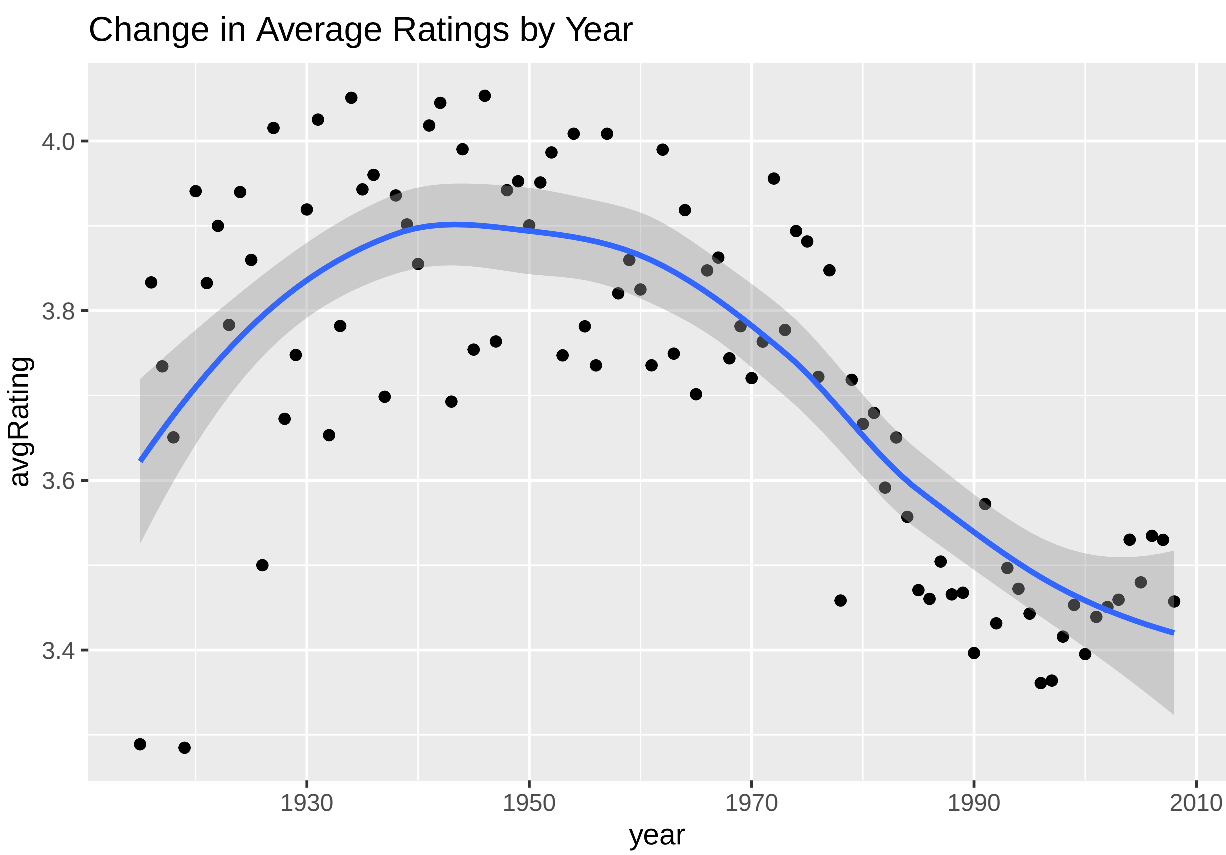
Age of Movies and User rating trend

A brief look at how users ratings change over the years that movies have been released. As observed below there seems to be a trend that indicates that more recent (or younger users, from 1950 till present) tend to rate movies more strictly (lower star rating) than their older counterpart

```r
# need to change the time stamp into years

edxWithYear <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))

edxWithYear %>% group_by(year) %>%
  summarise(avgRating = mean(rating)) %>%
  ggplot(aes(year,avgRating)) + geom_point() +
  geom_smooth() + ggtitle("Change in Average Ratings by Year")
```



MODELLING APPROACH

Begin with computing the RMSE, which is the loss-function for this model.

```
#Create the RMSE Function as this will be called a lot
RMSE <- function(rating, predRating){
  sqrt(mean((rating - predRating)^2))
}
```

RMSE is viewed as similar to standard deviation (sd) - RMSE is the error that us made when making a prediction of a movie rating. This statement means that a RMSE result larger than 1 is bad. One wants the RMSE to be as close to 0 as possible as this would mean there would be little error when making a prediction

Simplest possible model

This first model uses the edx dataset's rating mean to make predictions. This model predicts the same rating for all movies, regardless of the user. The expected rating of the dataset is between 3 and 4

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

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

Next is to predict a naive RMSE or a baseline model (uses only mean)

```
baselineRMSE <- RMSE(validation$rating,mu)
baselineRMSE
```

```
## [1] 1.061202
```

The results of the RMSE from this simple method can be seen below:

```r
resultsRMSE <- data_frame(method = "Mean Only ", RMSE = baselineRMSE)
```

```
## 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.
```

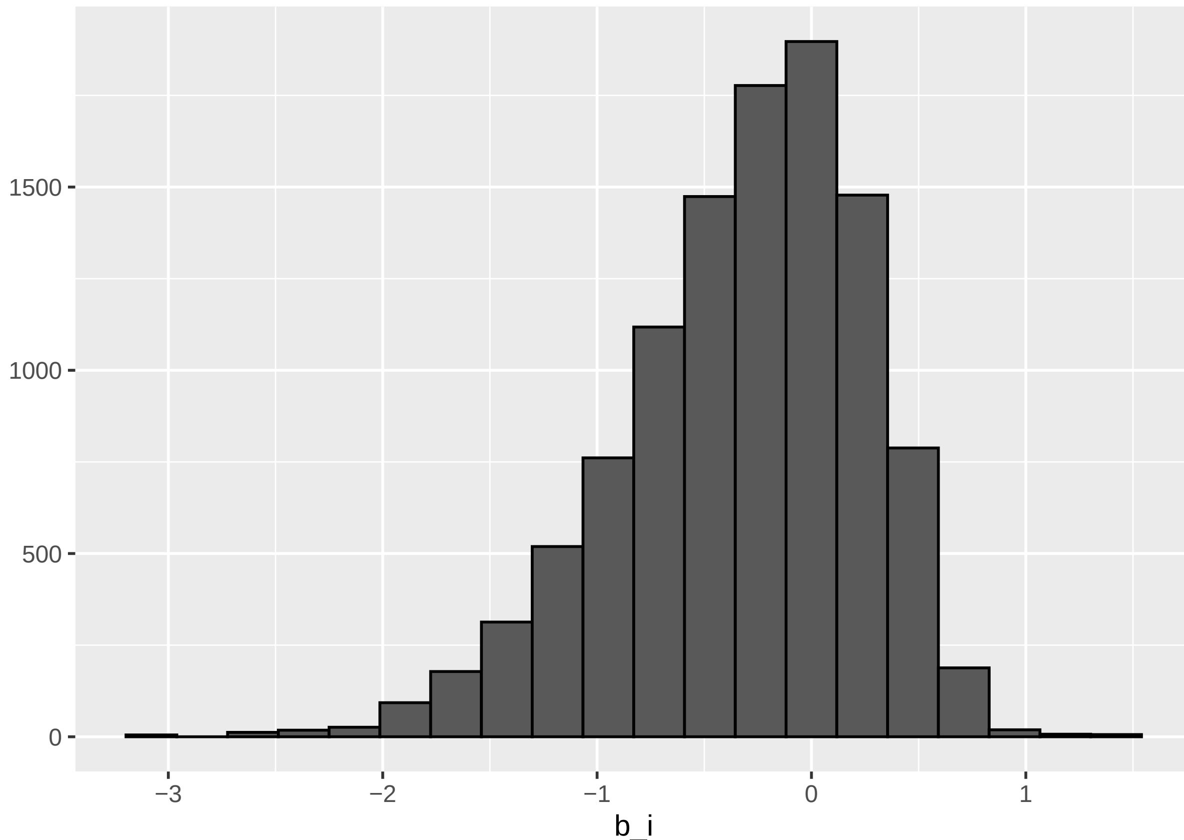
```r
resultsRMSE
```

```
## # A tibble: 1 x 2
##   method        RMSE
##   <chr>        <dbl>
## 1 "Mean Only "  1.06
```

Movie Effect Model

This is an attempt to improve on the previous model but incorporating the movie effect into a new model. When making use of the movie effect model, we must take head of the penalty term (b__i) - movie effect. Thus looking at the graph below it can be noted that different movies are rated differently. As seen by the histogram not being symmetric and is skewed toward a negative rating effect. The movie effect can be accounted for by computing the difference from the mean rating.

```r
movieAvg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movieAvg %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("black"))
```



The improvement to the prediction using this model can be viewed below

```
predRating <- validation %>%
  left_join(movieAvg, by="movieId") %>%
  mutate(pred = mu + b_i)
modelMovieEffect <- RMSE(validation$rating,predRating$pred)

resultsRMSE <- bind_rows(resultsRMSE,data_frame(method = "Movie Effect Method", RMSE = modelMovieEffect)

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

method	RMSE
Mean Only	1.0612018
Movie Effect Method	0.9439087

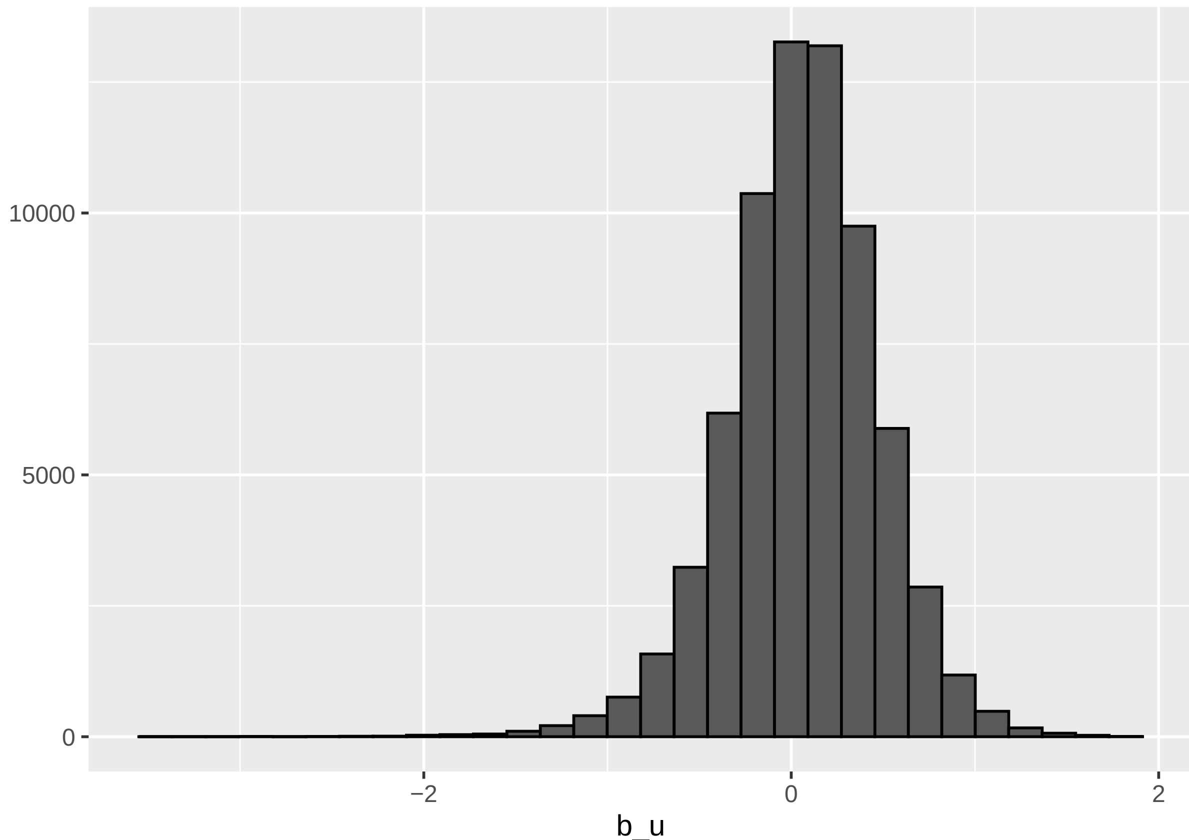
The Error has dropped by 0.1172931 which indicated that the prediction methods are getting better

Movie and User Effect Model

As seen previously different Users rate movies different to others. There are some users that rate critically with low rating, other that rate movies optimistically with high rating and lastly there are users that does care. This behavior is categorized as the penalty term (b_u) User Effect

```
userAvg <- edx %>%
  left_join(movieAvg, by='movieId') %>%
  group_by(userId) %>%
```

```
  summarize(b_u = mean(rating - mu - b_i))
userAvg %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```



As both the movie and user baises obscure the prediction of a movie rating an improvement in RMSE can be obtained by adding the user effect with the movie effect

```
predRatingUM <- validation %>%
  left_join(movieAvg, by = "movieId") %>%
  left_join(userAvg , by = "userId") %>%
  mutate(pred = mu + b_i + b_u)

modelMovieUserEffect <- RMSE(validation$rating,predRatingUM$pred)

resultsRMSE <- bind_rows(resultsRMSE, data_frame(method = "Movie and User Effect Model", RMSE = modelMo

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

method	RMSE
Mean Only	1.0612018
Movie Effect Method	0.9439087
Movie and User Effect Model	0.8653488

The RMSE has decreased further which is good.

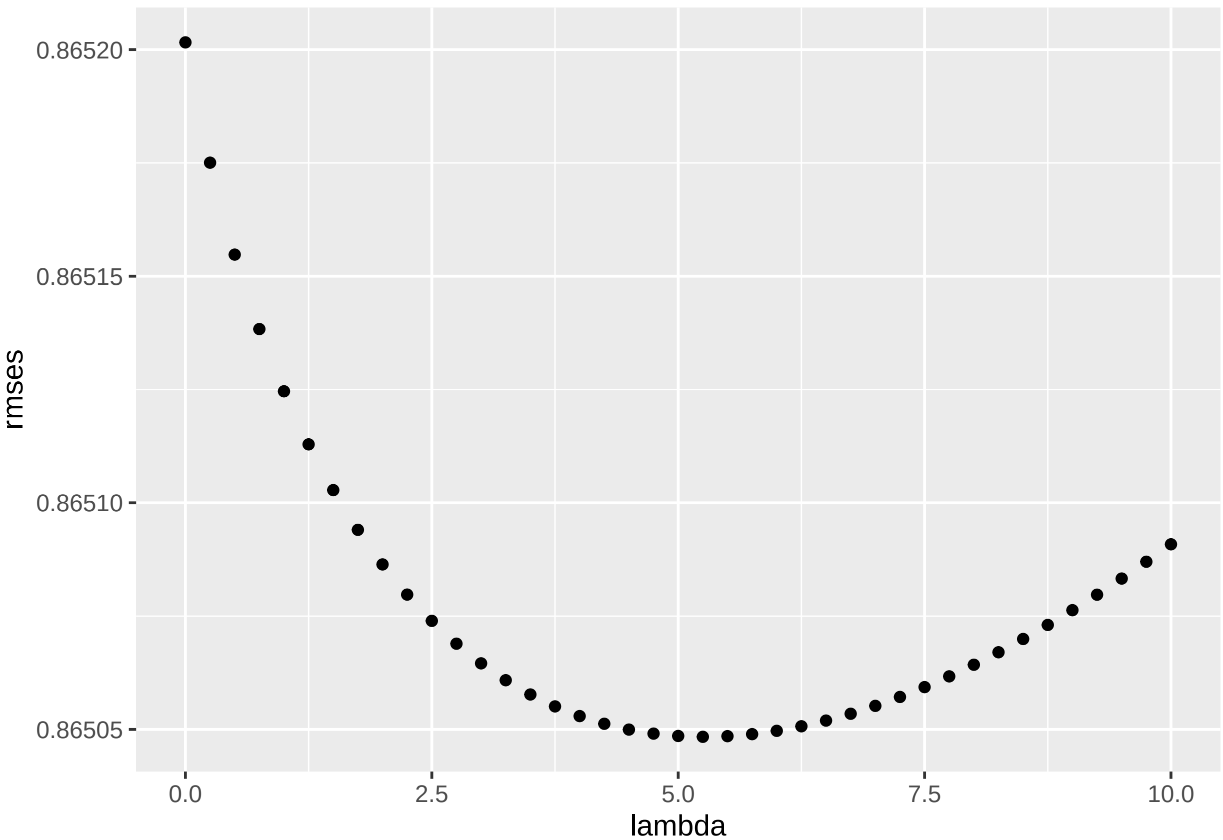
Regularisation of Movie and User Effect Model

As noted in the Visualisation/data exploration section, some users rate far more than other users and other users that rated very few movies. The user effect combined with some movies being rates very few times, such as only 1 time (there are 126 movies with a single user rating), this makes the predictions noisy and untrustworthy. Therefore a regularisation is used to create a penalty term to gives lessens importance of the effect that increases the error, thus reducing RMSE.

A value of lambda that will minimise RMSE must be found.

Find optimal lambda from Graph below:

```
qplot(lambda, rmses)
```




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

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

Use lambda = 5.25 for final model

Final model results are below

```
resultsRMSE <- bind_rows(resultsRMSE, data_frame(method = " Regularisation of Movie and User Effect Mode

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

method	RMSE

method	RMSE
Mean Only	1.0612018
Movie Effect Method	0.9439087
Movie and User Effect Model	0.8653488
Regularisation of Movie and User Effect Model	0.8650484

Results

The results from the models are as follows :

```r
resultsRMSE %>% knitr::kable()
```


method	RMSE
Mean Only	1.0612018
Movie Effect Method	0.9439087
Movie and User Effect Model	0.8653488
Regularisation of Movie and User Effect Model	0.8650484

The lowest RMSE is 0.8650484 and this is achieved by the regularisation of Movie and User Effect Model

CONCLUSION

The RMSE table shows that there was a continued improvement from model to models as new penalty terms where added. The Mean Only calculated a RMSE of greater than 1, indicating a high error in prediction that was over a single star, which is terrible. There was significant improvement with the implementations of the Movie Effect Method and Movie and User Effect Model, these reduced the RMSE to 0.9439087 and 0.8653488 respectively. Finally the Regularisation of Movie and User Effect Model reduces the RMSE to 0.8650484 which is within the acceptable goal for this project and one can somewhat trust the prediction.

Idea's for Future Improvement

It can be noted that future improvement to the RMSE could be achieve by including other effects such as genre, year, and movie age to a model. One could also try other machine learning techniques such as perhaps a neural network to better predict a movie rating.