movielense_project_azm

Muhammad Azeem

1/1/2020

```
## Introduction
> The following data analysis report is prepared as a part of HarvardX Data Science Capstone (Ha
rvardX: PH125.9x) Project, in which the given dataset "movielens" has been analysed by using dif
ferent tools and techniques learned during the program, especially the skills in the use of R-Pr
ogramming and machine learning capabilities. The report includes "calling the movielense data In
to the Project from the given code", "data exploration", and "data modeling to achive project go
al".

## Project Goal
>  The goal is to predict movie ratings, and evaluate the accuracy of the predicted model from t
he given code the dataset called  "edx" was split into the training and validation sets.

## Calling the movielense data Into the Project
> The code is provided in the edx capstone project module:
> The following lines of code will create training and validation sets (provied in the edx capst
one project module)


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

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

```
## -- Attaching packages ---------------------------------------------------------------
--------------------------------- tidyverse 1.3.0 --
```

```
## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------
-------------------------- tidyverse_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
```

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

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

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

```
# 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)
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
```

```
# if using R 3.5 or earlier, use `set.seed(1)` instead
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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)

# The above chunk of code gives a partition of the dataset for training and testing our dataset.
It also removes the unnecessary files from the working directory, which is always a good coding
 practice ('always clean after you cook').
```

Note that the above code is given in the project document, and used as it is to generate the project data sets.

Data Exploration

now we have two data sets "edx" and "validation", lets explore them:

```
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
```

```
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
```

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 8389
84885 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)"
...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "A
ction|Adventure|Sci-Fi" ...
```

```
str(validation)
```

```
## 'data.frame':    999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 113
3571200 844416936 844417070 ...
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy
(1995)" ...
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Dram
a|Romance|War" ...
```

```
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

```
names(validation)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

The output from the above code shows that there are 9000055 cases and 6 variables in edx data set, whereas the valiation data set includes 999999 cases and 6 variables. Notice that the variable “rating” is included in both data sets. The variables in “edx” and “validation” data sets are “userId” “movieId” “rating” “timestamp” “title” and “genres”. Moreover both data sets are data frame. The proportion between the length of two data sets is approximately 1:10

```
nrow(edx)/nrow(validation)
```

```
## [1] 9.000064
```

```
nrow(validation)/nrow(edx)
```

```
## [1] 0.1111103
```

Removing label column from the validation data set

As the project goal is to predict rating, therefore we remove rating column from the validation dataset.

```
validation_rt <- validation  # this is original validation which holds the rating column (just r
enamed).
validation <- validation %>% select(-rating)  # this is updated validation without rating colum
n.
head(validation)
```

```
##   userId movieId timestamp
## 1      1     231 838983392
## 2      1     480 838983653
## 3      1     586 838984068
## 4      2     151 868246450
## 5      2     858 868245645
## 6      2    1544 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
```

Further preprocessing data sets

Notice that in the “title-column” the years are also appearning with the movie name. Similarly in the “genres- column” more than one genre are shown. We need to reformat the “edx” and “valiadation” datasets.

Creating new column as “year” and move year information from the “title” column into the “year” column in the edx datasets

```
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
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 year
## 1                Comedy|Romance 1992
## 2          Action|Crime|Thriller 1995
## 3   Action|Drama|Sci-Fi|Thriller 1995
## 4        Action|Adventure|Sci-Fi 1994
## 5 Action|Adventure|Drama|Sci-Fi 1994
## 6        Children|Comedy|Fantasy 1994
```

Creating new column as “year” and move year information from the “title” column into the “year” column in the validation datasets, and store original “valiadation” with rating column as “validation_rt”

```
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_rt <- validation_rt %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
```

Removing year values form the title column in the edx and validation datasets

```
edx<- edx %>% mutate(title = str_remove_all(edx$title, "[()0123456789]"))
validation<- validation %>% mutate(title = str_remove_all(validation$title, "[()0123456789]"))
validation_rt<- validation_rt %>% mutate(title = str_remove_all(validation_rt$title, "[()0123456
789]"))
```

Genre column issue

```
# Following three lines of code can be used to Seperate the "genres" column in the edx & validat
ion dataset, and add each genre into new separate row. The genre in this project is not seprated
due to machine memory limit issue.
# edx  <- edx  %>% separate_rows(genres, sep = "\\|")
# validation <- validation   %>% separate_rows(genres, sep = "\\|")
# validation_rt <- validation_rt  %>% separate_rows(genres, sep = "\\|")
```

Examining the new datasets

```
head(edx)
```

```
##   userId movieId rating timestamp                  title
## 1      1     122      5 838985046              Boomerang
## 2      1     185      5 838983525               Net, The
## 3      1     292      5 838983421               Outbreak
## 4      1     316      5 838983392               Stargate
## 5      1     329      5 838983392 Star Trek: Generations
## 6      1     355      5 838984474       Flintstones, The
##                         genres year
## 1                Comedy|Romance 1992
## 2          Action|Crime|Thriller 1995
## 3  Action|Drama|Sci-Fi|Thriller 1995
## 4       Action|Adventure|Sci-Fi 1994
## 5 Action|Adventure|Drama|Sci-Fi 1994
## 6       Children|Comedy|Fantasy 1994
```

```
head(validation)
```

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

```
head(validation_rt)
```

```
##   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
## 2                                Jurassic Park
## 3                                   Home Alone
## 4                                      Rob Roy
## 5                                Godfather, The
## 6 Lost World: Jurassic Park, The Jurassic Park
##                                    genres year
## 1                                  Comedy 1994
## 2        Action|Adventure|Sci-Fi|Thriller 1993
## 3                          Children|Comedy 1990
## 4                Action|Drama|Romance|War 1995
## 5                             Crime|Drama 1972
## 6 Action|Adventure|Horror|Sci-Fi|Thriller 1997
```

We can see that data is now in a better format to proceed further. Now lets explore further.

Exploring the data

Examiningg the distribution of “rating” in the training “edx” data set.

```
table(edx$rating) # it gives the frequency of the distinct values in the variable for a variable (e.g Age)
```

```
##
##     0.5       1     1.5       2     2.5       3     3.5       4     4.5       5
##   85374  345679  106426  711422  333010 2121240  791624 2588430  526736 1390114
```

From above output, we can find the rating range as: 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5

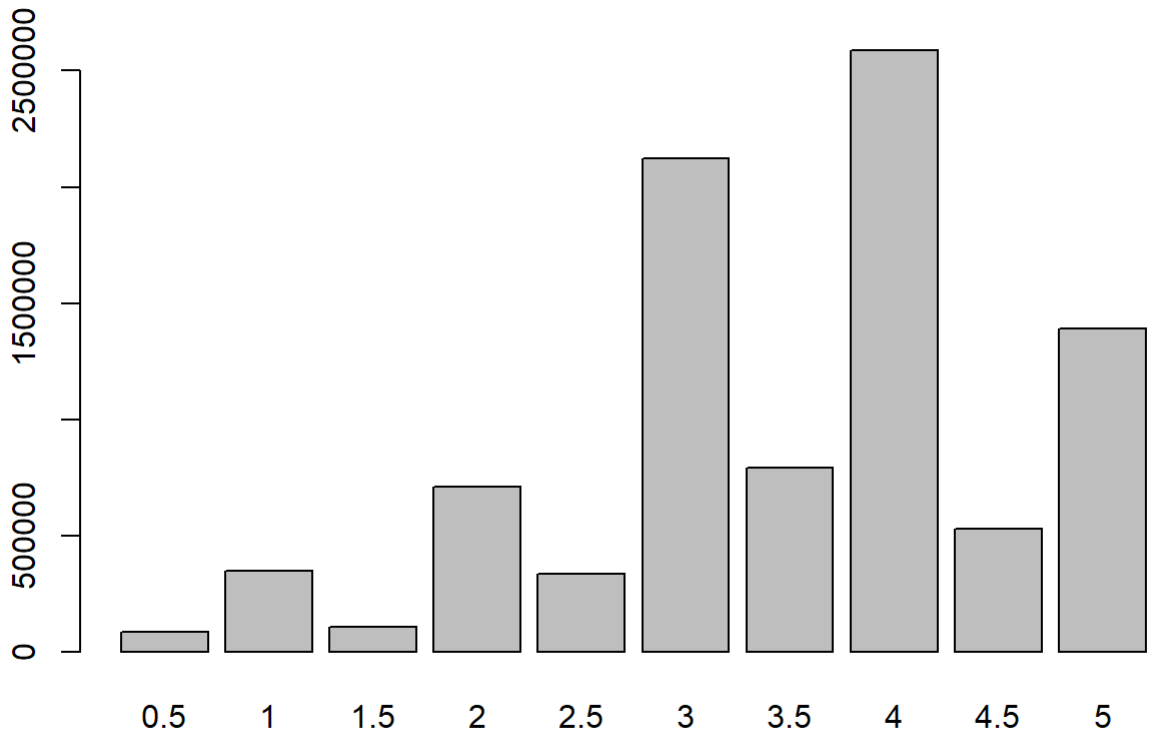
we can also see the proportion of each ratings

```
prop.table(table(edx$rating))
```

```
##
##         0.5           1         1.5           2         2.5           3
## 0.009485942 0.038408543 0.011825039 0.079046406 0.037000885 0.235691893
##         3.5           4         4.5           5
## 0.087957685 0.287601576 0.058525865 0.154456167
```

Examining the liking pattern among cases by plotting the bar chart.

```
barplot(table(edx$rating))
```



The diagram shows that rating level 3 and 4 is very popular with frequency equal to 2121240 and 2588430 respectively.

Descriptive statistics can be obtained through the popular summary() function

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

Finding frequencies of users, movies, genres, and years in the edx dataset

```
edx %>% summarize(users = n_distinct(userId), movies = n_distinct(movieId), genres = n_distinct
(genres), years = n_distinct(year))
```

```
##   users movies genres years
## 1 69878  10677    797    94
```

It is clear from the output that in the edx data set there are 8 users, 373 movies, 18 genres and 63 years in the edx data set.

Frequency distribution of the genres.

```
genre_f <- edx %>% group_by(genres) %>% summarize(count = n()) %>% arrange(desc(count))
genre_f
```

```
## # A tibble: 797 x 2
##    genres                     count
##    <chr>                      <int>
##  1 Drama                     733296
##  2 Comedy                    700889
##  3 Comedy|Romance            365468
##  4 Comedy|Drama              323637
##  5 Comedy|Drama|Romance      261425
##  6 Drama|Romance             259355
##  7 Action|Adventure|Sci-Fi   219938
##  8 Action|Adventure|Thriller 149091
##  9 Drama|Thriller            145373
## 10 Crime|Drama               137387
## # … with 787 more rows
```

Frequency distribution of the movies.

```
movie_f <- edx %>% group_by(title) %>% summarize(count = n()) %>% arrange(desc(count))
movie_f
```

```
## # A tibble: 10,370 x 2
##    title                                                count
##    <chr>                                                <int>
##  1 "Pulp Fiction "                                      31362
##  2 "Forrest Gump "                                      31079
##  3 "Silence of the Lambs, The "                         30382
##  4 "Jurassic Park "                                     29360
##  5 "Shawshank Redemption, The "                         28015
##  6 "Braveheart "                                        26212
##  7 "Fugitive, The "                                     26020
##  8 "Terminator : Judgment Day "                         25984
##  9 "Star Wars: Episode IV - A New Hope a.k.a. Star Wars " 25672
## 10 "Batman "                                            24585
## # … with 10,360 more rows
```

Modeling

Model-1

At the base level, model is developed by using average rating (mean) to train the model and predict the movie rating in the validation model.

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

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

For measuring predictive strength RMSE will be used. Following code will define the RMSE, which is the square root of the mean of the squared difference between observed and estimated ratings.

```
RMSE <- function(observed, estimated){
  sqrt(mean((observed-estimated)^2,na.rm=TRUE))
}
```

```
#Initiate RMSE results to compare various models
rmse_data <- data_frame()
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
m1 <- mean(edx$rating)

rmse1 <- RMSE(validation_rt$rating,m1)
```

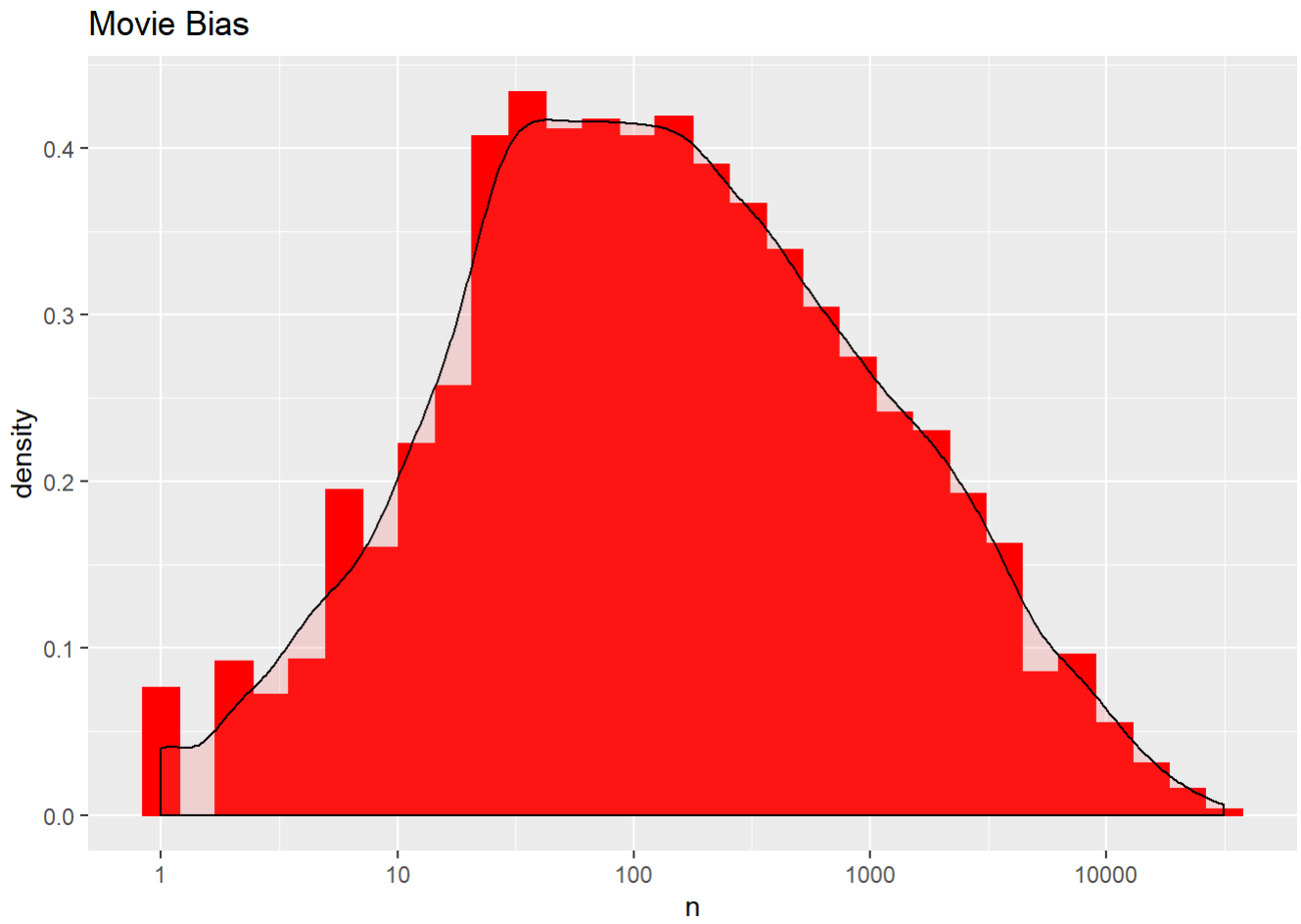
```
rmse_data <- data_frame(method = "m1", RMSE = rmse1)
rmse_data %>% knitr::kable()
```

method	RMSE
m1	1.061202

Model-2

In this model the Movies are used to examine the predictive efficiency. Before the development of the model-2, the examination of bias in the data is examined. The presence of inherent bias in dat can effect the quality of the prediction. We can see from the following diagram that some movies are rated more often than others.

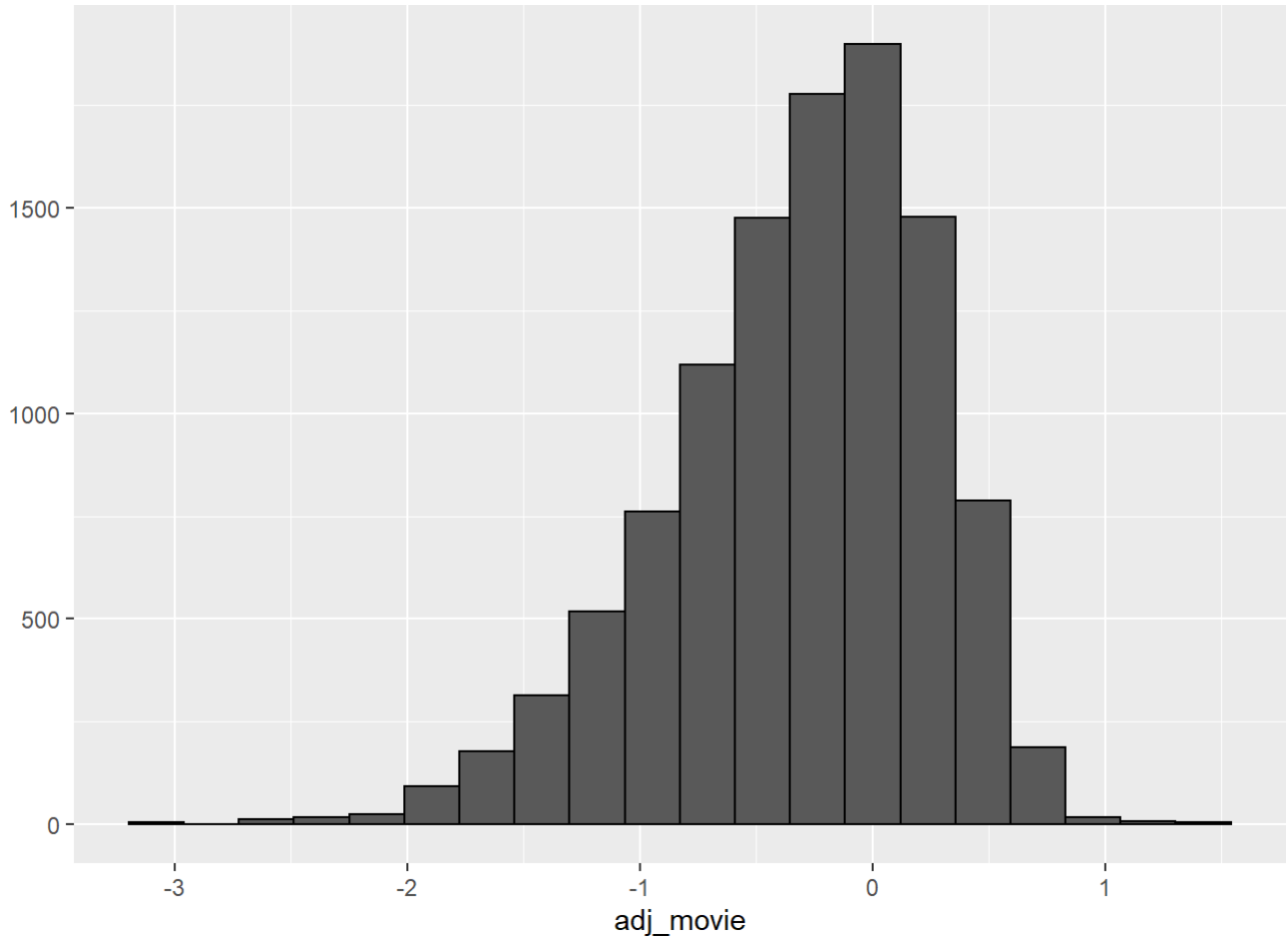
```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, aes(y=..density..), color = "red", fill = "red") +
  geom_density(alpha=.2, fill="#FF6666")+
  scale_x_log10() +
  ggtitle("Movie Bias")
```



The skewed histogram shows that some movies were rated rarely. This is a bias in rating, which is causing imbalence in the data. The histogram is skewed towards negative rating effect. To handle this isuue, deviations from mean is taken. The weighted Movie rating will be obtained with the following code:

```
adjusted_m_rating <- edx %>%
  group_by(movieId) %>%
  summarize(adj_movie = mean(rating - avg))

adjusted_m_rating %>% qplot(adj_movie, geom ="histogram", bins = 20, data = ., color = I("black"
))
```



Model development from the adjusted movie data (Model-2)

```
m2 <- validation %>%
  left_join(adjusted_m_rating, by='movieId') %>%
  mutate(pred = avg + adj_movie)

rmse2 <- RMSE(validation_rt$rating,m2$pred)
```

```
rmse_data <- bind_rows(rmse_data,
                          data_frame(method="m2",
                                     RMSE = rmse2 ))
rmse_data %>% knitr::kable()
```

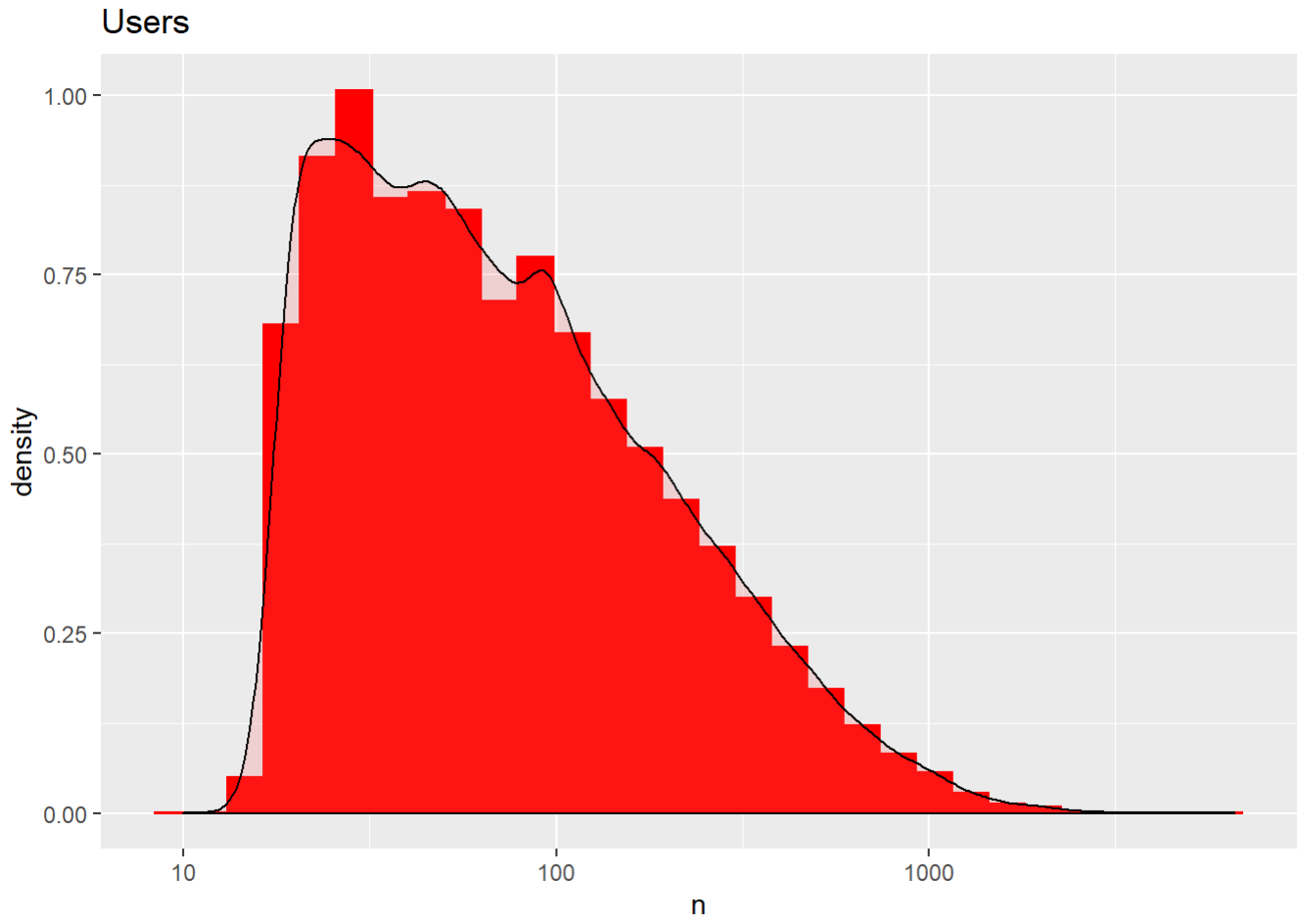
method	RMSE
m1	1.0612018
m2	0.9439087

The error has drop by 5% and motivates us to move on this path further, and an improvement in the RMSE is achieved by adjusting for movie bias.

Model-3

In the model-3, the users data is examined for the biasedness. The diagram below shows that the users distribution of rating is not unform, some users are very active in rating than the others.

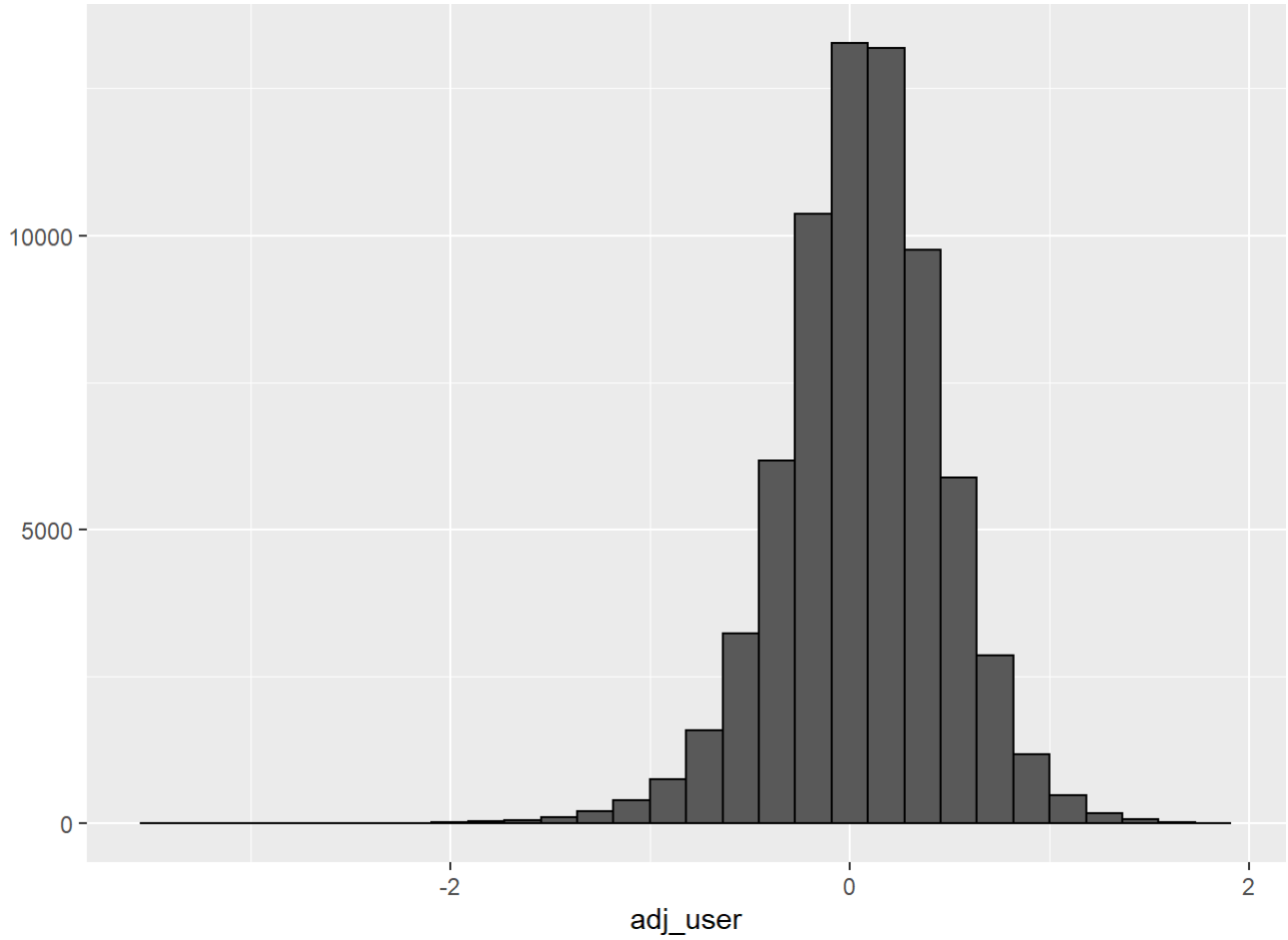
```
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, aes(y=..density..), color = "red", fill = "red") +
  geom_density(alpha=.2, fill="#FF6666")+
  scale_x_log10() +
  ggtitle("Users")
```



To handle the issue of user bias, the deviations from mean is taken, and the weighted user rating will be obtained with the following code:

```
adjusted_u_rating <- edx %>%
  left_join(adjusted_m_rating, by='movieId') %>%
  group_by(userId) %>%
  summarize(adj_user = mean(rating - avg - adj_movie))

adjusted_u_rating %>% qplot(adj_user, geom ="histogram", bins = 30, data = ., color = I("black"
))
```



Model development from the adjusted movie and user data (Model-3)

```
m3 <- validation %>%
  left_join(adjusted_m_rating, by='movieId') %>%
  left_join(adjusted_u_rating, by='userId') %>%
  mutate(pred = avg + adj_movie + adj_user)

rmse3 <- RMSE(validation_rt$rating,m3$pred)
```

```
rmse_data <- bind_rows(rmse_data,
                         data_frame(method="m3",
                                    RMSE = rmse3 ))
rmse_data %>% knitr::kable()
```

method	RMSE
m1	1.0612018
m2	0.9439087
m3	0.8653488

There is an improvement from m1 to m3.

Model-4

This model is developed by refining the data to further level. The method of regularization is used. Cross validation is used to select optimum value of lambda (lmb). For every value of lmb, adj_movie and adj_user is determined.

```
lmb <- seq(0, 10, 0.25)

rmse_1 <- sapply(lmb, function(l){

  avg <- mean(edx$rating)

  adj_movie <- edx %>%
    group_by(movieId) %>%
    summarize(adj_movie = sum(rating - avg)/(n()+l))

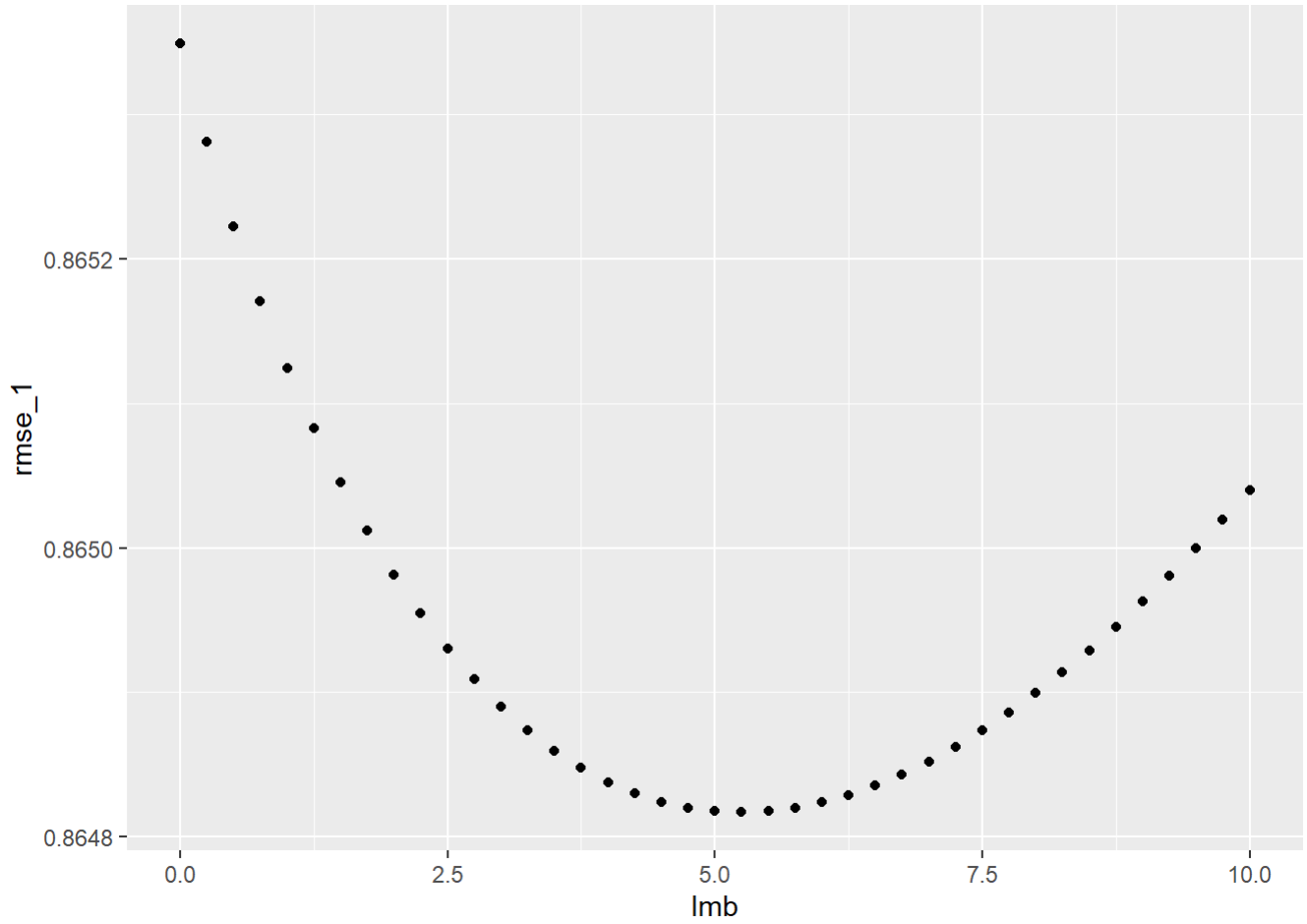
  adj_user <- edx %>%
    left_join(adj_movie, by="movieId") %>%
    group_by(userId) %>%
    summarize(adj_user  = sum(rating - adj_movie - avg)/(n()+l))

  pred_r <- validation %>%
    left_join(adj_movie, by = "movieId") %>%
    left_join(adj_user , by = "userId") %>%
    mutate(pred = avg + adj_movie + adj_user ) %>%
    .$pred

  return(RMSE(validation_rt$rating,pred_r))
})
```

The optimum value of lmb can also be seen with the help of following diagram. This is a plot between rmse1 and lmb.

```
qplot(lmb, rmse_1)
```



To find accurate value of lmb (optimum), following code is used.

```
lmb <- lmb[which.min(rmse_1)]
lmb
```

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

Determining the regularized estimates of adj_movie by using lmb.

```
adj_m <- edx %>%
  group_by(movieId) %>%
  summarize(adj_movie = sum(rating - avg)/(n()+lmb), m_n = n())
```

Determining the regularized estimates of adj_user by using lmb.

```
adj_u <- edx %>%
  left_join(adj_m, by='movieId') %>%
  group_by(userId) %>%
  summarize(adj_user  = sum(rating - avg - adj_movie )/(n()+lmb), u_n = n())
```

Predicting rating

```
pred_r_1 <- validation %>%
  left_join(adj_m, by='movieId') %>%
  left_join(adj_u, by='userId') %>%
  mutate(pred = avg + adj_movie + adj_user ) %>%
  .$pred
```

Results

```
rmse4 <- RMSE(validation_rt$rating,pred_r_1)
rmse_data <- bind_rows(rmse_data,
                          data_frame(method="Regularized Movie and User Effect Model",
                                     RMSE = rmse4 ))
rmse_data %>% knitr::kable()
```

method	RMSE
m1	1.0612018
m2	0.9439087
m3	0.8653488
Regularized Movie and User Effect Model	0.8648170

The above table provides the evidence of an improvement from m1 to m4

Concluding Remarks

The results table shows that RMSE has reduced from m1 to m4. The key feature used to predict rating are movie and user. The adjusted model with regularization approach has reduced the error further. For improvement to the further level, the factors like year, and genres can also be used as features in the model.

Limitation

> One serious limitation was the machine memory, which was disabeling the code execution. Serious bottleneck was observed when sperating the various genres from each row to the new rows in th etraining dataset. Therefore the current project was done without seprating the genres into rows. Another limit was the internet download speed which was taking longer time to download the data.

References

> To complete this project, the following sources were studied.

[1]: Material provided in the machine learning course of the edx (HarvardX: PH125.8x). [2]: (https://github.com/cmrad/Updated-MovieLens-Rating-Prediction (https://github.com/cmrad/Updated-MovieLens-Rating-Prediction))