DATA SCIENCE: CAPSTONE PROJECT

PROJECT REPORT ON MOVIE LENS RATING ANALYSIS

NAME: VISHNUPRASANTH VIJAYA KUMAR

DUE DATE: 06/23/2020 19:59 EDT

SUBMITTED FOR:

EdX Certification Purpose

Introduction:

One of the major applications of machine learning is the ability to make recommendations on certain items to potential users or customers which in this case is analysis of movie ratings provided by Movie Lens.

The open source dataset from movie Lens ‘10M version of the Movie Lens dataset’ has been utilized. The aim of this project is to develop and implement a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set. Several machine learning algorithms have been used and results are compared to get maximum possible accuracy in prediction.

This report contains problem statement, data set provision and initiation, exploratory analysis, modeling and data analysis, results and conclusions and has been written in that manner.

Problem Statement:

This Data Science EdX capstone project on ’Movie recommendation system predicts the movie rating by a user based on users previous’ rating of movies. The dataset used for this purpose can be found in the following links

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

The challenge is not facile given that there are many different types of biases present in the movie reviews. It can be different psychological, social, demographic variations that changes the taste of every single user for a given movie. The problem can still be designed to tackle the biases which can be expressed via mathematical equations. The idea here is to develop a model which can effectively predict movie recommendations for a given user without our judgement being impaired due to bias differences. In the algorithm, the RMSE calculations are computed which as well is part of the project requirements.

Data Set Provision and Initiation:

The code is provided in the edx capstone project module [Create Test and Validation Sets]

https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+1T2020/courseware/dd9a048b16ca477a8f0aaf1d888f0734/e8800e37aa444297a3a2f35bf84ce452/?child=last

Code from R Studio:

```
#Create test and validation sets
> # Create edx set, validation set, and submission file

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

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

# MovieLens 10M dataset:
> # https://grouplens.org/datasets/movielens/10m/
> # http://files.grouplens.org/datasets/movielens/ml-10m.zip
> dl <- tempfile ()


ratings <- read. table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100
K/ratings.dat"))),
+                       col. names = c("userId", "movieId", "rating", "timest
amp"))
> 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(movi
eId))[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)
> 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)
>
> # Validation dataset can be further modified by removing rating column

> validation_CM <- validation
> validation <- validation %>% select(-rating)
```

```
>
> # extra libraries that might be useful in analysis and visualizations
> library(ggplot2)
> library(lubridate)
```

```
R Output:
```

Attaching package: ‘lubridate’

The following objects are masked from ‘package:dplyr’:

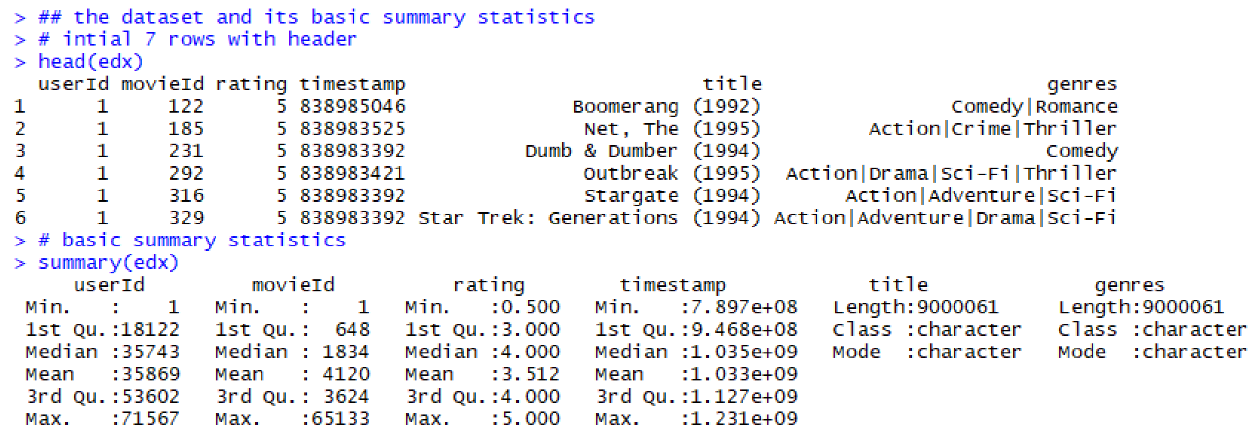
    intersect, setdiff, union

The following objects are masked from ‘package:base’:

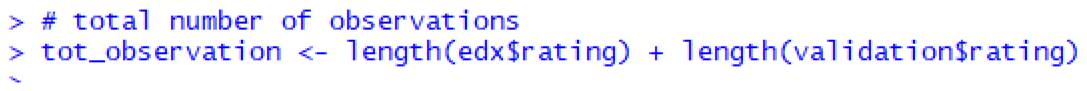
    date, intersect, setdiff, union

The above set of coding gives a partition of dataset for training and validation. It also eliminates needless files from the R working directory which is always the best practice and a pro-active approach.

After the dataset is available, the dataset features and basic statistics must be summarized:



```
> ## the dataset and its basic summary statistics
> # intial 7 rows with header
> head(edx)
  userId movieId rating timestamp                         title                        genres
1      1     122      5 838985046              Boomerang (1992)                Comedy|Romance
2      1     185      5 838983525               Net, The (1995)         Action|Crime|Thriller
3      1     231      5 838983392          Dumb & Dumber (1994)                        Comedy
4      1     292      5 838983421               Outbreak (1995)  Action|Drama|Sci-Fi|Thriller
5      1     316      5 838983392               Stargate (1994)       Action|Adventure|Sci-Fi
6      1     329      5 838983392 Star Trek: Generations (1994) Action|Adventure|Drama|Sci-Fi
> # basic summary statistics
> summary(edx)
     userId         movieId          rating        timestamp             title              genres         
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08   Length:9000061     Length:9000061    
 1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08   Class :character   Class :character  
 Median :35743   Median : 1834   Median :4.000   Median :1.035e+09   Mode  :character   Mode  :character  
 Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09                                        
 3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09                                        
 Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09                                        
```



```
> # total number of observations
> tot_observation <- length(edx$rating) + length(validation$rating)
>
```

The above dataset is in the tidy format, and it is all set for exploratory analysis.

Exploratory Analysis:

```
#  Since RMSE (root mean squre error) is used frequency so lets define a function
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}
# lets modify the columns to suitable formats that can be further used for analysis
# Modify the year as a column in the edx & validation datasets
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
# Modify the genres variable in the edx & validation dataset (column separated)
split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
```

```
#  Since RMSE (root mean squre error) is used frequency so lets define a function
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}
# lets modify the columns to suitable formats that can be further used for analysis
# Modify the year as a column in the edx & validation datasets
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
# Modify the genres variable in the edx & validation dataset (column separated)
split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
```

```
# The 1st rows of the edx & split_edx datasets are presented below:
head(edx)
userId movieId rating timestamp                         title                         genres year
     1     122      5 838985046              Boomerang (1992)                 Comedy|Romance 1992
     1     185      5 838983525               Net, The (1995)          Action|Crime|Thriller 1995
     1     231      5 838983392          Dumb & Dumber (1994)                         Comedy 1994
     1     292      5 838983421               Outbreak (1995)   Action|Drama|Sci-Fi|Thriller 1995
     1     316      5 838983392               Stargate (1994)        Action|Adventure|Sci-Fi 1994
     1     329      5 838983392 Star Trek: Generations (1994) Action|Adventure|Drama|Sci-Fi 1994
```

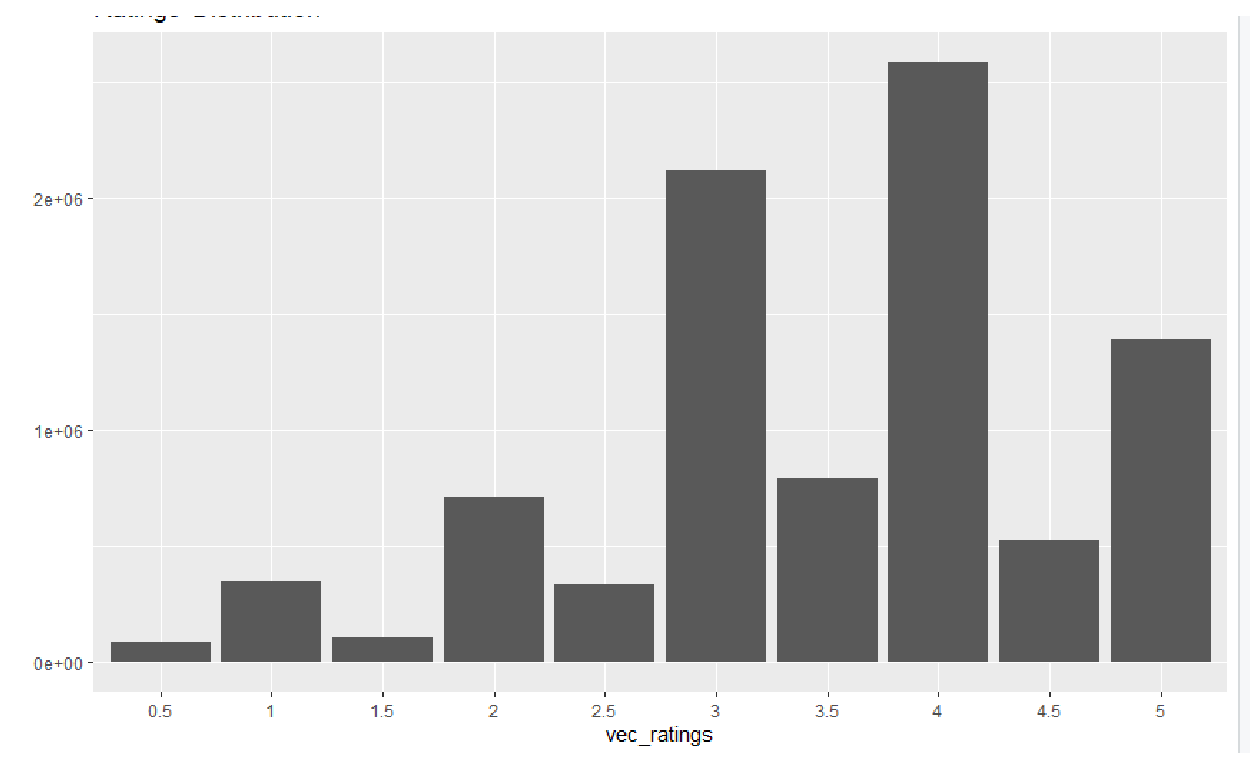
```
> head(split_edx)
Error in head(split_edx) : object 'split_edx' not found
> summary(edx)
     userId         movieId          rating        timestamp            title              genres         
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08   Length:9000061     Length:9000061    
 1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08   Class :character   Class :character  
 Median :35743   Median : 1834   Median :4.000   Median :1.035e+09   Mode  :character   Mode  :character  
 Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09                                        
 3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09                                        
 Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09                                        
      year     
 Min.   :1915  
 1st Qu.:1987  
 Median :1994  
 Mean   :1990  
 3rd Qu.:1998  
 Max.   :2008  
> # Number of unique movies and users in the edx dataset
> edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
  n_users n_movies
1   69878    10677
```

Movie Ratings in total per genre:

```
> genre_rating <- split_edx%>%
+     group_by(genres) %>%
+     summarize(count = n()) %>%
+     arrange(desc(count))
Error in eval(lhs, parent, parent) : object 'split_edx' not found
```

Distribution of Ratings or Ratings' Distribution:

```
> vec_ratings <- as.vector(edx$rating)
> unique(vec_ratings)
 [1] 5.0 3.0 2.0 4.5 3.5 4.0 1.0 1.5 2.5 0.5
> vec_ratings <- vec_ratings[vec_ratings != 0]
> vec_ratings <- factor(vec_ratings)
> qplot(vec_ratings) +
+     ggtitle("Ratings' Distribution")
```



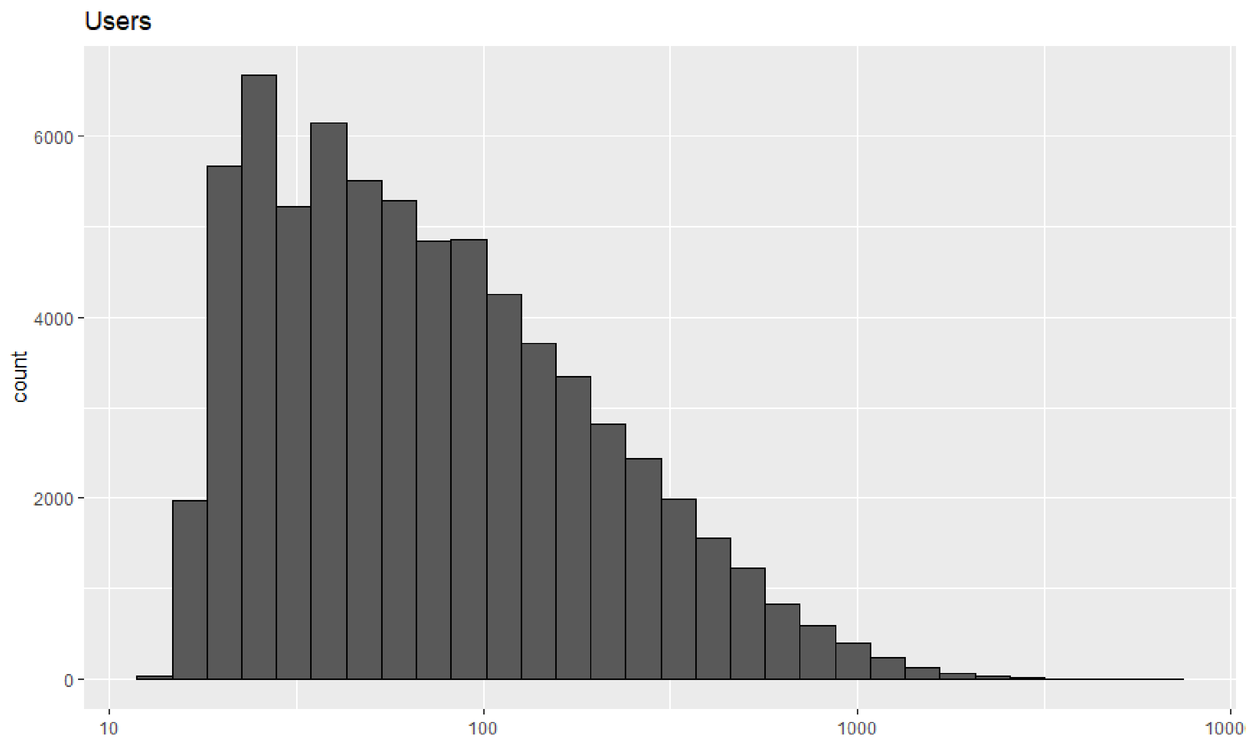
The above distribution of rating shows the user(s) has a general tendency to rank or rate the movies between 3 and 4. Normally this makes it supposedly a very general conclusion. The exploration of effect of different features, should be performed thoroughly to make a good predictive model.

Data Analysis Strategies

- Some movies rated more often than genres and incorporating this in the model to determine movie bias.
- Some users have their opinion of liking or disliking of a movie leading to positive or negative reviews. How to address these characteristics: determining users bias.
- The popularity of the genre solely depends on the present issues. Hence the time dependent analysis should also be explored. How to approach this idea: determine the genre popularity over the years
- The average rating of movies over the years is affected based on user's mindset of rating the movie over the years. Determine using plot rating vs release year.

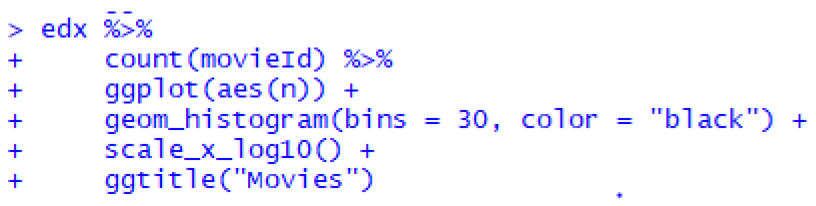
Distribution of user's rating for movies in determination of users' bias:

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

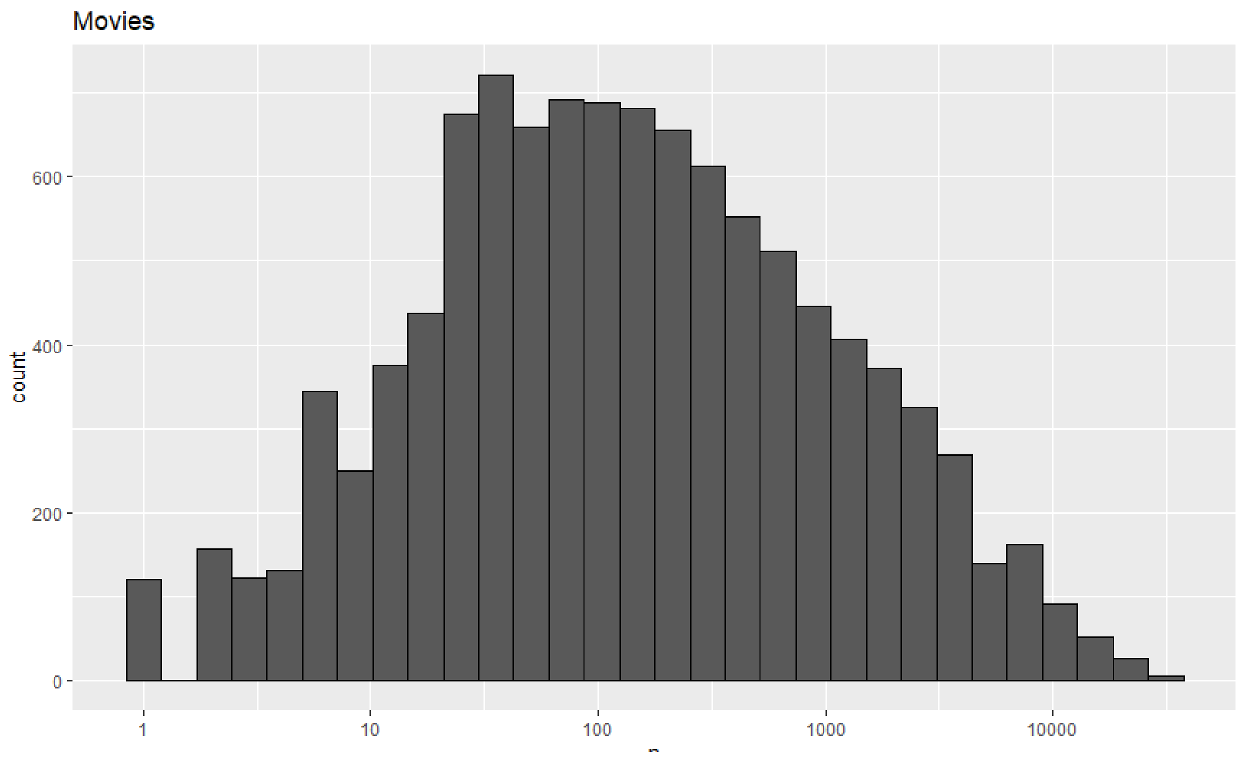


From the above plot not all the users are active. Some or few users have rated very few movies which may bias the prediction results based on their opinions.

Some movies are rated more often than others. Their distribution is shown as below. This explores movie biases:



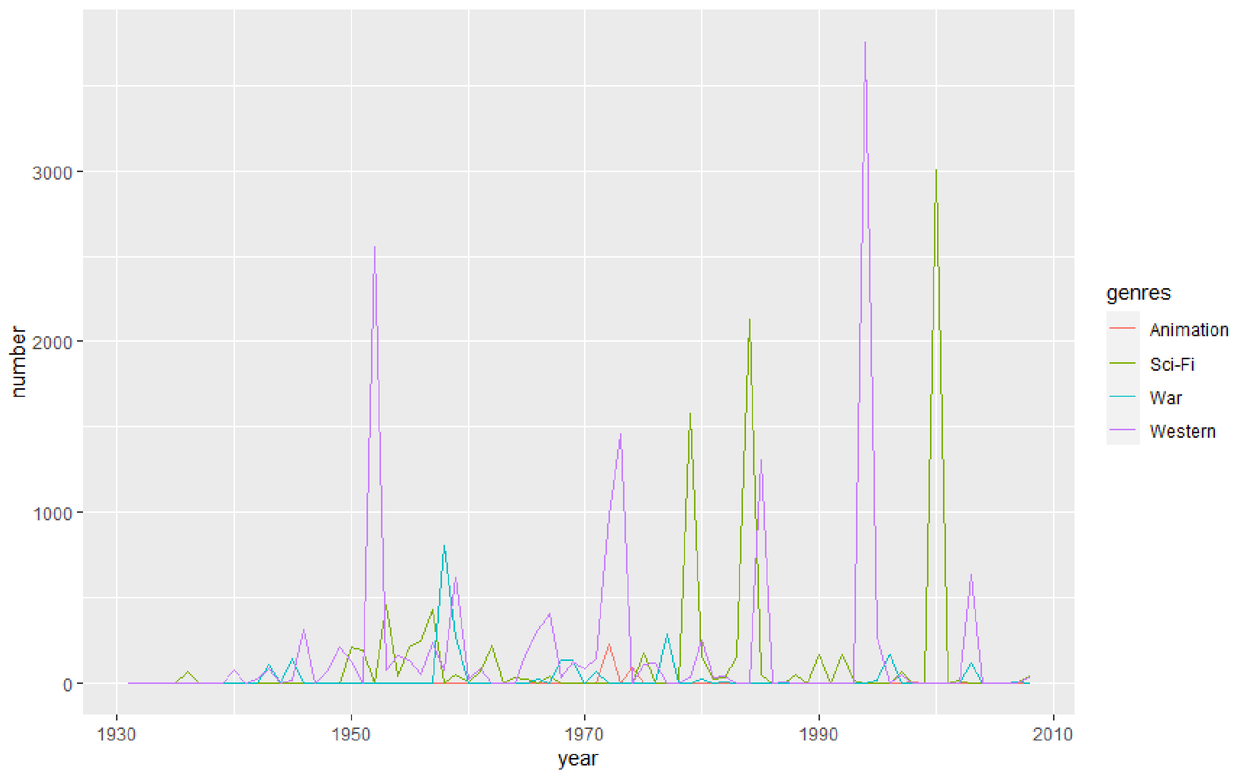
```
> edx %>%
+     count(movieId) %>%
+     ggplot(aes(n)) +
+     geom_histogram(bins = 30, color = "black") +
+     scale_x_log10() +
+     ggtitle("Movies")
```



The above histogram shows where some movies were rated very few numbers of times. In movie rating prediction, they shall be given lower importance.

Genres popularity per year. The issue of evolution of users taste over different popular genres:

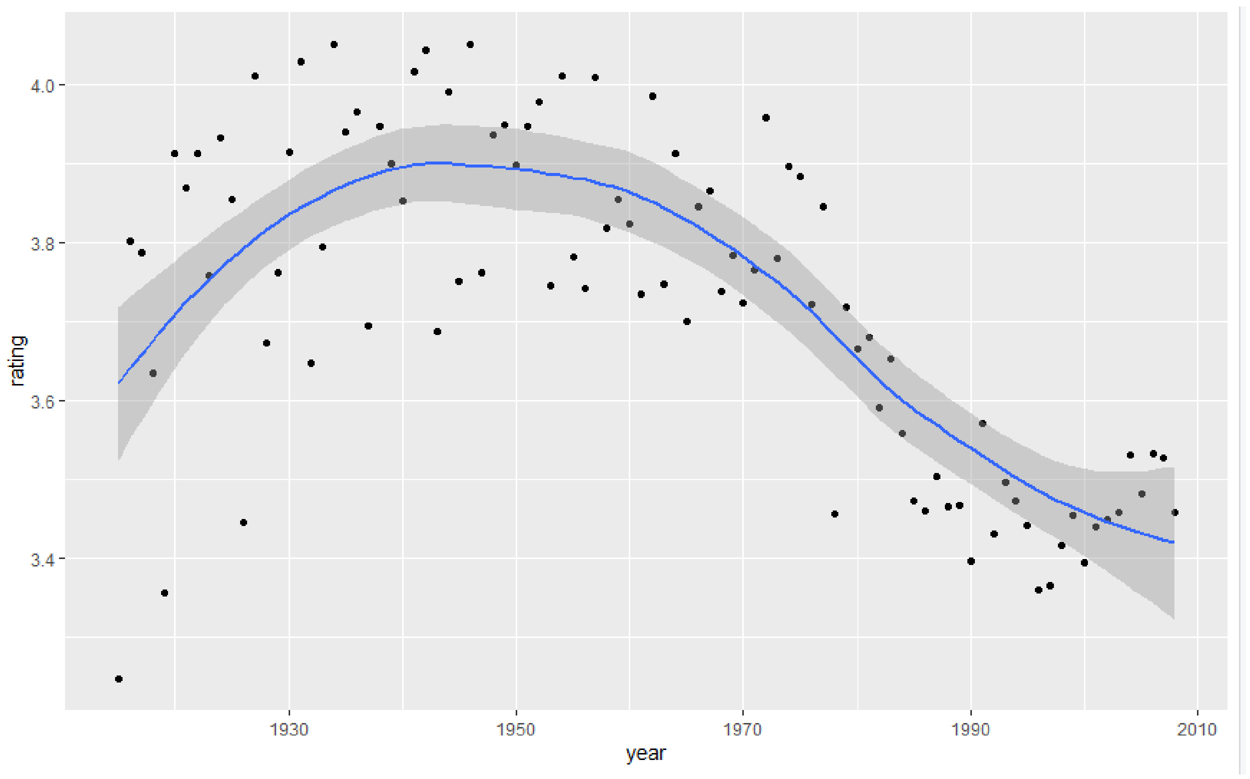
```
> # Genres vs year; 4 genres are chosen for readability: animation, sci-fi, war and western movies.
> genres_popularity %>%
+     filter(year > 1930) %>%
+     filter(genres %in% c("War", "Sci-Fi", "Animation", "Western")) %>%
+     ggplot(aes(x = year, y = number)) +
+     geom_line(aes(color=genres)) +
+     scale_fill_brewer(palette = "Paired")
Error in eval(lhs, parent, parent) : object 'genres_popularity' not found
> genres_popularity <- edx %>%
+     na.omit() %>% # omit missing values
+     select(movieId, year, genres) %>% # select columns we are interested in
+     mutate(genres = as.factor(genres)) %>% # turn genres in factors
+     group_by(year, genres) %>% # group data by year and genre
+     summarise(number = n()) %>% # count
+     complete(year = full_seq(year, 1), genres, fill = list(number = 0)) # add missing years/genres
> # Genres vs year; 4 genres are chosen for readability: animation, sci-fi, war and western movies.
> genres_popularity %>%
+     filter(year > 1930) %>%
+     filter(genres %in% c("War", "Sci-Fi", "Animation", "Western")) %>%
+     ggplot(aes(x = year, y = number)) +
+     geom_line(aes(color=genres)) +
+     scale_fill_brewer(palette = "Paired")
```



The plot conveys the fact that some movie genres have become more popular over others for different periods of time.

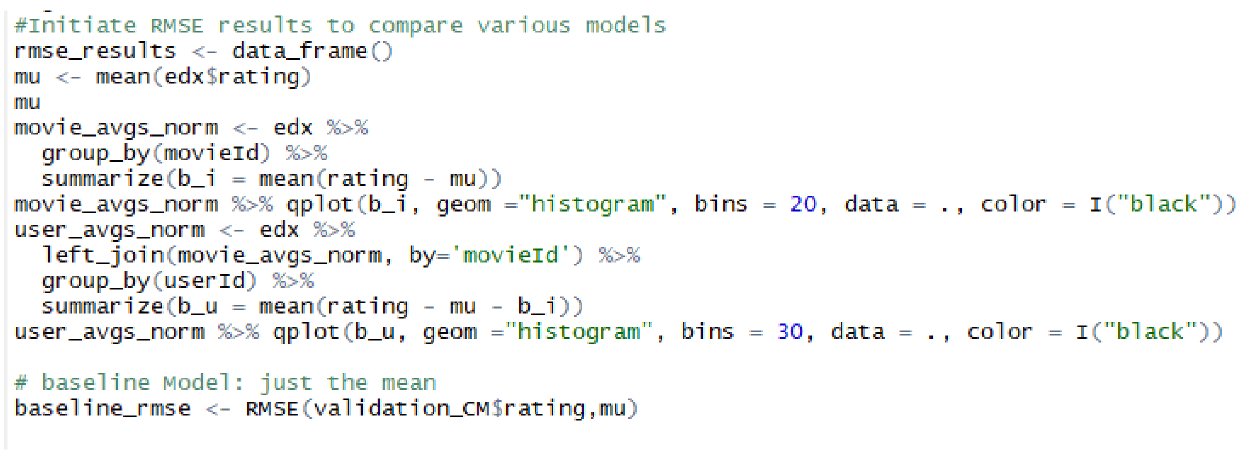
Rating vs release year. The trend of movie viewers and their habit of rating a movie:

```
> edx %>% group_by(year) %>%
+     summarize(rating = mean(rating)) %>%
+     ggplot(aes(year, rating)) +
+     geom_point() +
+     geom_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The above trend confirms today’s users rate the movies relatively lower.

Data Analysis: Model Preparation

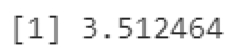


```
#Initiate RMSE results to compare various models
rmse_results <- data_frame()
mu <- mean(edx$rating)
mu
movie_avgs_norm <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs_norm %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("black"))
user_avgs_norm <- edx %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs_norm %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))

# baseline Model: just the mean
baseline_rmse <- RMSE(validation_CM$rating,mu)
```

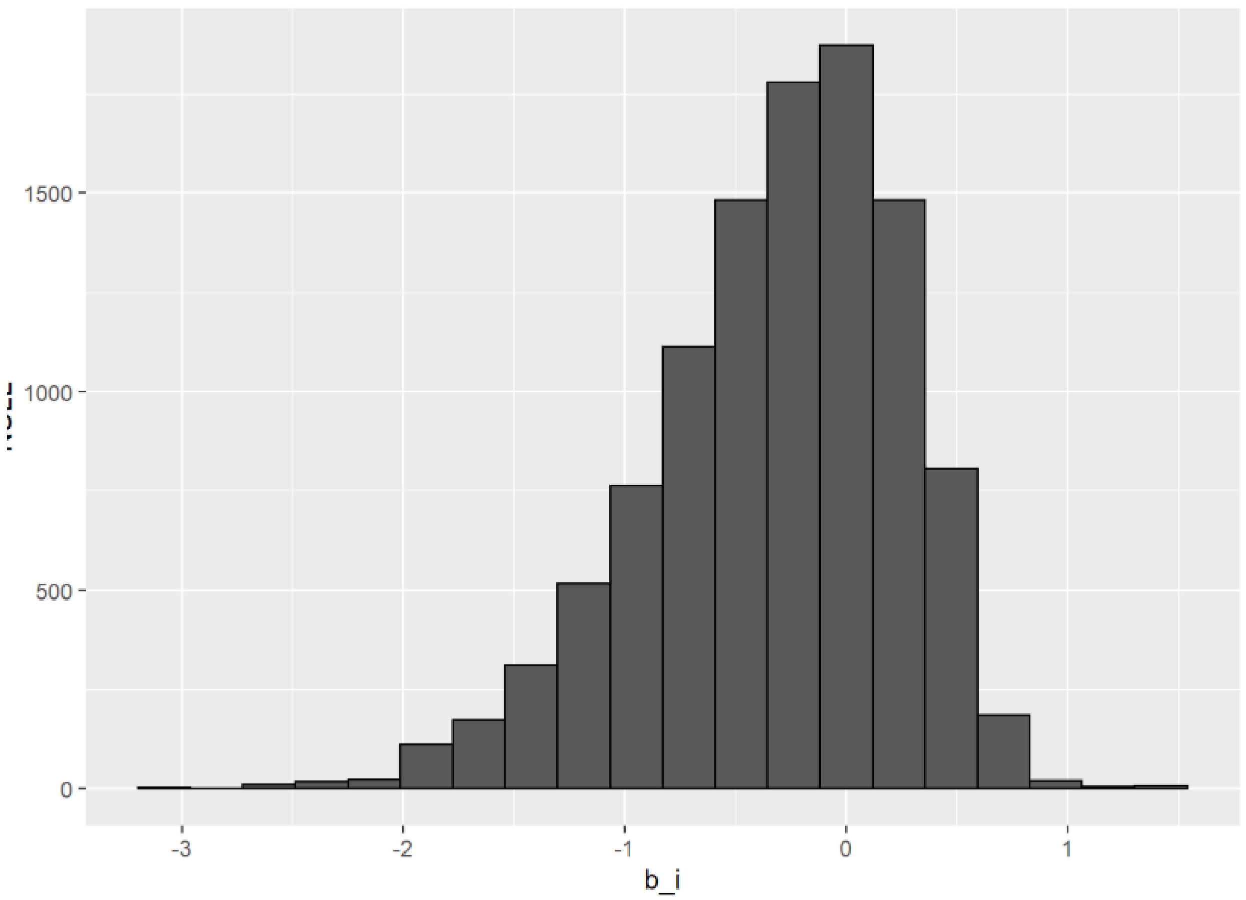
Regardless of the user and movie, the mean rating is used for predicting the rating(same) for all the movies

Movie Effect Rating:

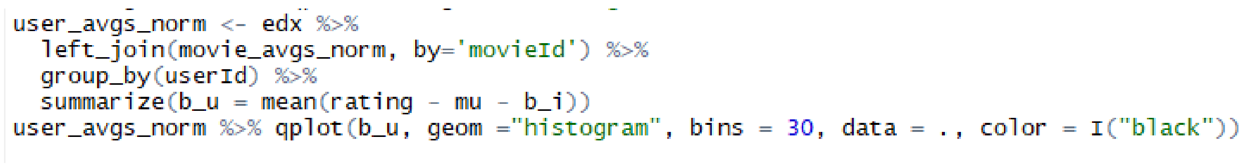
From the below histogram it can be clearly observed, the histogram is negatively skewed, and not symmetric heading towards negative rating. The movie rating could be considered by taking difference from mean rating



```
[1] 3.512464
```

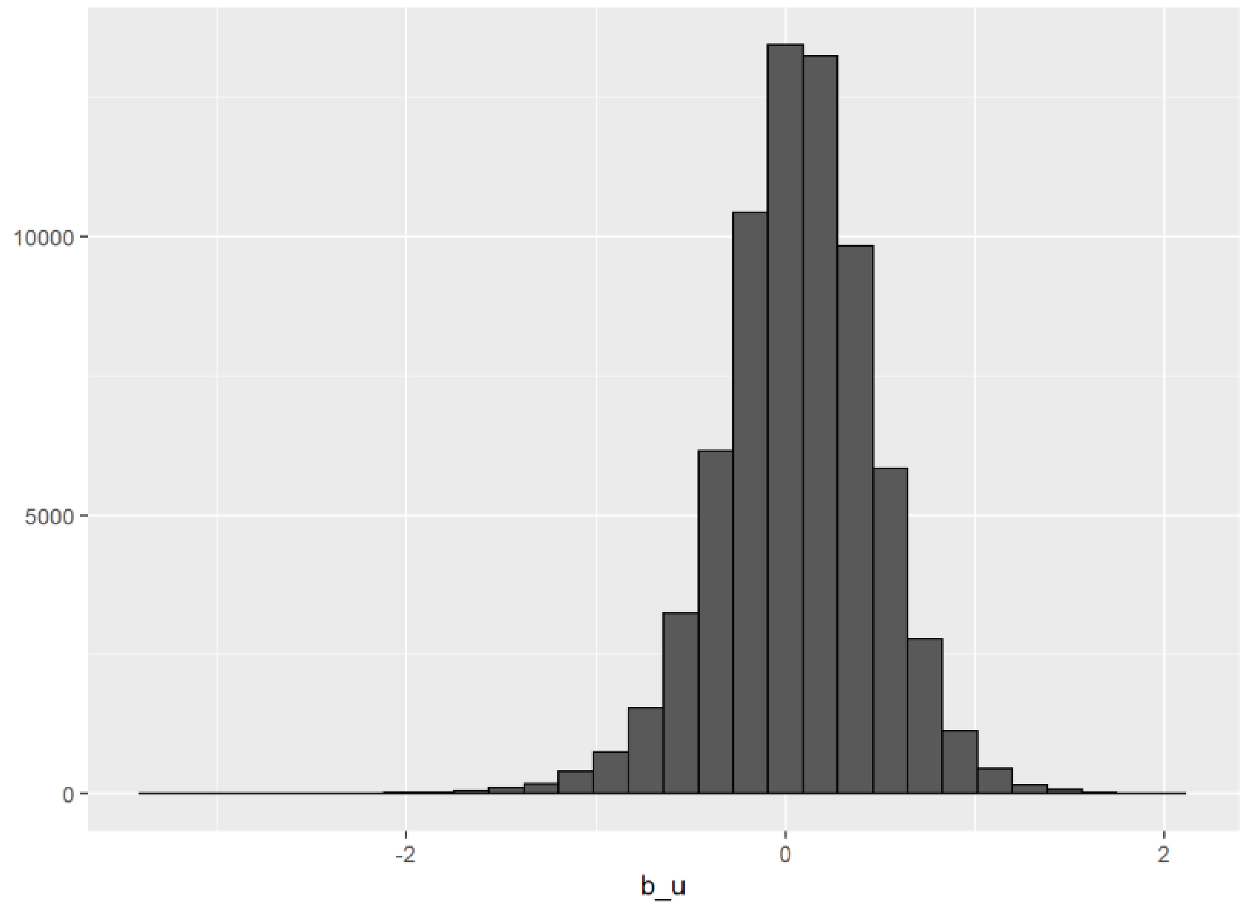


User Effect Rating:



```
user_avgs_norm <- edx %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs_norm %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```

Most of the users have the tendency to rate a good movie in a poor manner, or just give some rating in random. The above R code snippet gives the following output.



Creation of the model:

The model's quality will be assessed based on the RMSE value. The lower the value is, the better the model.

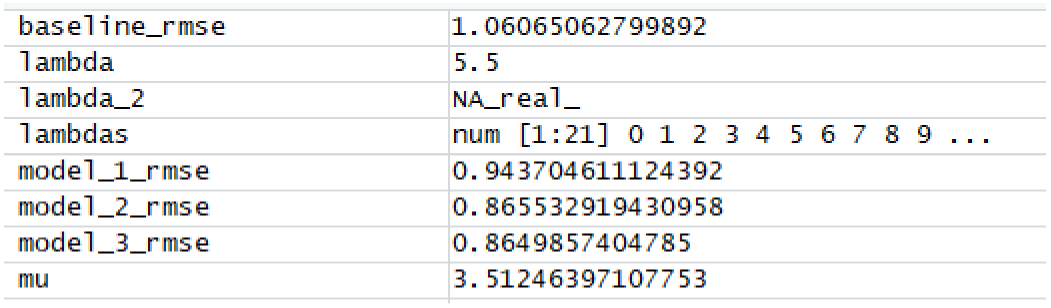
Model- Baseline Version:

The model calculates the mean rating and serves as a baseline version and the RMSE will be improved relative to this current model.

```
# baseline Model: just the mean
baseline_rmse <- RMSE(validation_CM$rating,mu)

## Test results based on simple prediction
baseline_rmse

## Check results
rmse_results <- data_frame(method = "Using mean only", RMSE = baseline_rmse)
rmse_results
```

```
[1] 1.060651
```

```
# A tibble: 1 x 2
  method          RMSE
  <chr>          <dbl>
1 Using mean only  1.06
```

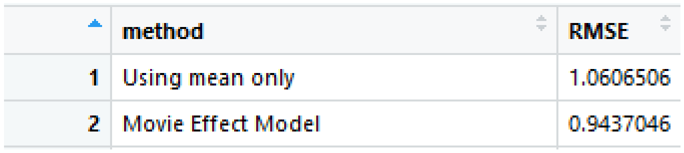


baseline_rmse	1.06065062799892
lambda	5.5
lambda_2	NA_real_
lambdas	num [1:21] 0 1 2 3 4 5 6 7 8 9 ...
model_1_rmse	0.943704611124392
model_2_rmse	0.865532919430958
model_3_rmse	0.8649857404785
mu	3.51246397107753

The Movie Effect Model:

By the addition of movie effect, an improvement in RMSE is achieved.

```
# Movie effects only
predicted_ratings_movie_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  mutate(pred = mu + b_i)
model_1_rmse <- RMSE(validation_CM$rating,predicted_ratings_movie_norm$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```



	method	RMSE
1	Using mean only	1.0606506
2	Movie Effect Model	0.9437046

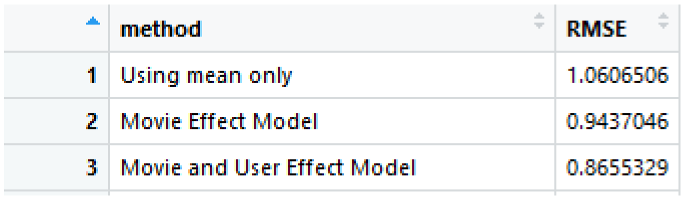
```
# A tibble: 2 x 2
  method              RMSE
  <chr>              <dbl>
1 Using mean only    1.06
2 Movie Effect Model 0.944
```

The error has dropped by around 5% and hence we see an improvement in the model.

Combination of User and Movie Effect Model:

A further improvement in RMSE can be achieved by adding the user since user and movie biases have an uncertain prediction on movie rating

```
# Use test set,join movie averages & user averages
# Prediction equals the mean with user effect b_u & movie effect b_i
predicted_ratings_user_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  left_join(user_avgs_norm, by='userId') %>%
  mutate(pred = mu + b_i + b_u)

# test and save rmse results
model_2_rmse <- RMSE(validation_CM$rating,predicted_ratings_user_norm$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and User Effect Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
rmse_results
```



	method	RMSE
1	Using mean only	1.0606506
2	Movie Effect Model	0.9437046
3	Movie and User Effect Model	0.8655329

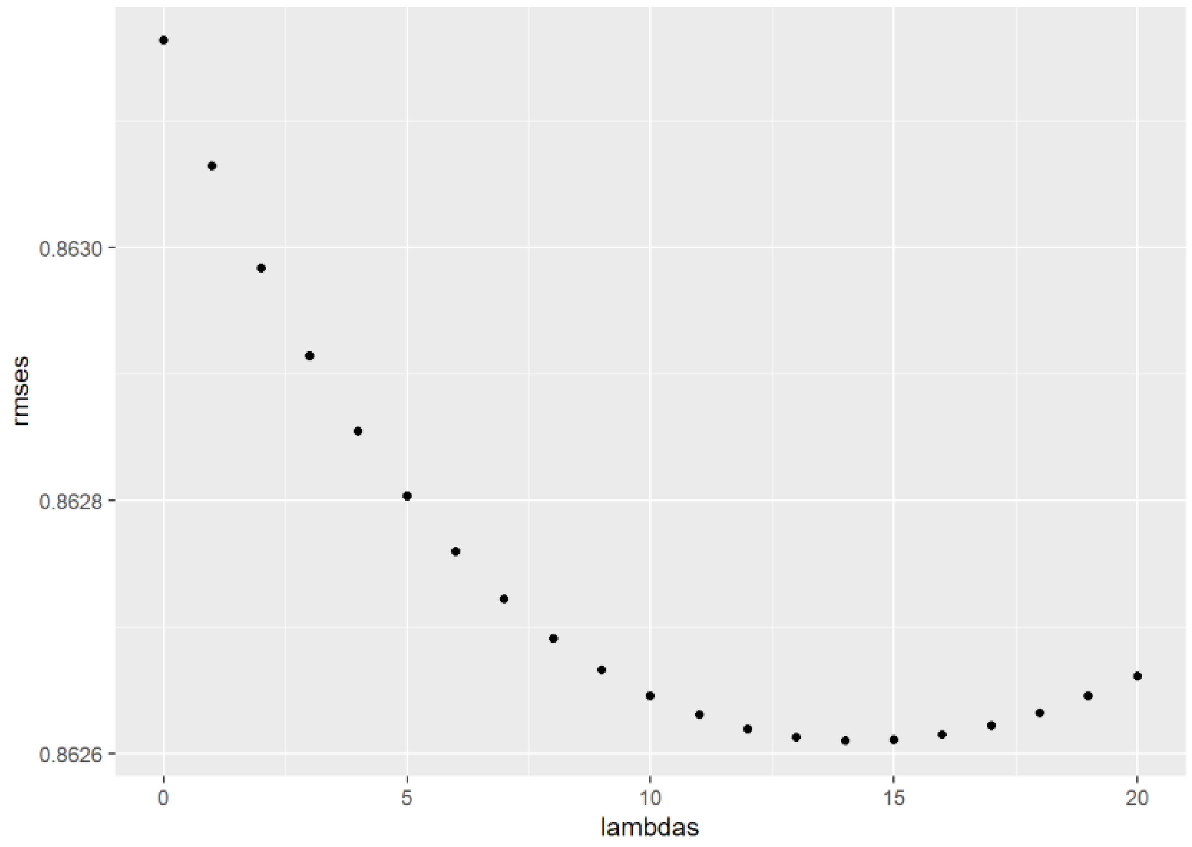
The above table shows a further improvement compared to the previous model.

Regularized approach using user and movies ratings:

In the data exploration that was conducted so far from the above results, only some users were active in participation regarding movie reviewing. In certain cases, users only rated very few movies, or some movies receive very less ratings. As well as RMSE are prone to large errors in terms of sensitivity. Large errors lead to increase in residual MSE. Hence a penalty term must be used, to give less importance to those effect.

```r
# b_y and b_g represent the year & genre effects, respectively
lambdas <- seq(0, 20, 1)

# Note: the below code could take some time
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  b_i <- split_edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- split_edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  b_y <- split_edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+lambda), n_y = n())
  b_g <- split_edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by = 'year') %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u - b_y)/(n()+lambda), n_g = n())
```

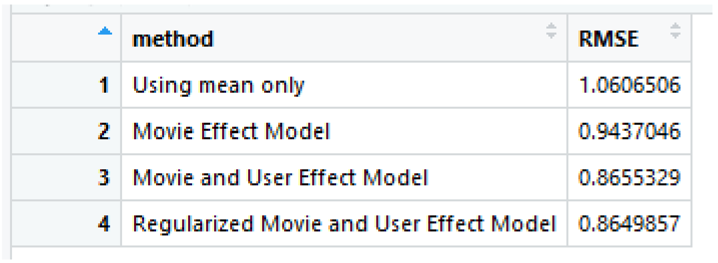
```r
  predicted_ratings <- split_valid %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by = 'year') %>%
    left_join(b_g, by = 'genres') %>%
    mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
    .$pred
  return(RMSE(split_valid_CM$rating,predicted_ratings))
})

# Compute new predictions using the optimal lambda
# Test and save results
qplot(lambdas, rmses)
```



```
# Compute new predictions using the optimal lambda
# Test and save results
qplot(lambdas, rmses)
lambda_2 <- lambdas[which.min(rmses)]
lambda_2
movie_reg_avgs_2 <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_2), n_i = n())
user_reg_avgs_2 <- edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda_2), n_u = n())
year_reg_avgs <- edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  group_by(year) %>%
  summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+lambda_2), n_y = n())
```

```r
genre_reg_avgs <- edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  left_join(year_reg_avgs, by = 'year') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u - b_y)/(n()+lambda_2), n_g = n())
predicted_ratings <- split_valid %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  left_join(year_reg_avgs, by = 'year') %>%
  left_join(genre_reg_avgs, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
  .$pred
model_4_rmse <- RMSE(split_valid_CM$rating,predicted_ratings)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Reg Movie, User, Year, and Genre Effect Model",
                                     RMSE = model_4_rmse ))
rmse_results %>% knitr::kable()
return(RMSE(split_valid_CM$rating,predicted_ratings))
rmse_results %>% knitr::kable()
```

```
[1] 14
```



	method	RMSE
1	Using mean only	1.0606506
2	Movie Effect Model	0.9437046
3	Movie and User Effect Model	0.8655329
4	Regularized Movie and User Effect Model	0.8649857

Reg Movie, User, Year, and Genre Effect Model	0.8626097

Conclusions:

The RMSE table shows an improvement of the model based on previous certain assumptions. The simplest model 'Using mean only' gives RMSE more than 1, which means the rating may be missed by one star .After incorporating 'Movie effect' and 'Movie and user effect' on model there was a significant improvement by 5% and 13.45% respectively. This is drastic improvement given the simplicity of the model. A regularization model was used to penalize the data points to prevent overfitting. The final RMSE is 0.8626 with an improvement over 13.26% with respect to the baseline model. This implies the prediction can be trsuted for movie rating given by the users.