Move Lens Project

Zenasimlab

1/6/2020

Introduction

In this new era, the trend of using recommender systems to analyze the patterns of users interest in products is becoming more popular. This HarvardX:PH125.9x Data Science Capstone project was motivated by Netlix challenge in October,2006.A challenge was offered by Netflix to the data science community with the aim to improve Netflix's inhouse software by 10% and wins $1M prize. Since the data is not freely available, "MovieLens 10M movie ratings" dataset was used the it was split into 'edx' and 'validation' sets. Challenges encountered in this project are large dataset and presence of different types of baises in the movie reivew.

Goal of the project

The objective of the project is to develop a machine learning algorithm to predict movie ratings in the validation setnand to minimize the root mean square error(RMSE) of the predicted ratings of the 'validation' set. RMSE is one of the frequently used measure to predict accuracy and it is sensitive to outliers.

Data Ingestion

The following code is provided in the edx capson project model [Create Test and Validation Sets] https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+2T2018/courseware/dd9a048b16ca477a8f0aaf1d888f0734/e8800e37aa444297a3a2f35bf84ce452/?child=first (https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+2T2018/courseware/dd9a048b16ca477a8f0aaf1d888f0734/e8800e37aa444297a3a2f35bf84ce452/?child=first)

```
#################################
# Create edx set, validation 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")


dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(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")
# 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)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

The chunk of code provided a partition of the dataset for training and testing our model.With that we are able to remove the unneccesary files from the working directory. Algorithm development is to be carried out on the “edx” subset only, as “validation” subset will be used to test the final algorithm.

```
# Validation dataset can be further modified by removing rating column
validation_CM <- validation
validation <- validation %>% select(-rating)
```

```
#Extra libraries that might be useful
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

Methods and Analysis

Data Analysis

The first 6 rows of “edx” subset contain the six variables “userID”, “movieID”, “rating”, “timestamp”, “title”, and “genres”.

```
# The presentation of 1st few rows of edx
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
```

```
# Summary Statistics of edx
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
##
##
##
```

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

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)))
```

The total of unique movies and users in the edx subset is about 70.000 unique users and about 10.700 different movies:

```
# 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
```

Top 10 movies ranked in order of the number of rating

```
edx%>%
  group_by(movieId,title)%>%
  summarize(count=n())%>%
  arrange(desc(count))
```

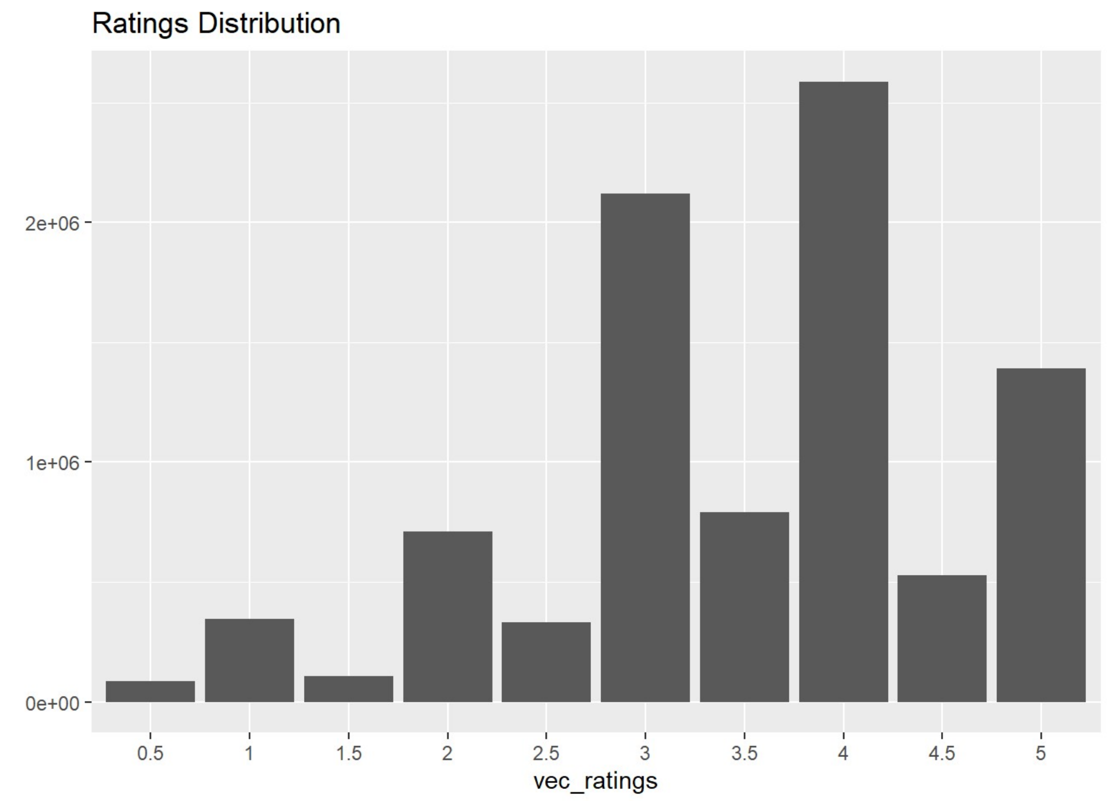
```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                        count
##      <dbl> <chr>                                                        <int>
##  1     296 Pulp Fiction (1994)                                          31362
##  2     356 Forrest Gump (1994)                                          31079
##  3     593 Silence of the Lambs, The (1991)                             30382
##  4     480 Jurassic Park (1993)                                         29360
##  5     318 Shawshank Redemption, The (1994)                             28015
##  6     110 Braveheart (1995)                                            26212
##  7     457 Fugitive, The (1993)                                         25998
##  8     589 Terminator 2: Judgment Day (1991)                            25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (19~ 25672
## 10     150 Apollo 13 (1995)                                             24284
## # ... with 10,667 more rows
```

Ratings distribution

```
vec_ratings<-as.vector(edx$rating)
unique(vec_ratings)
```

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

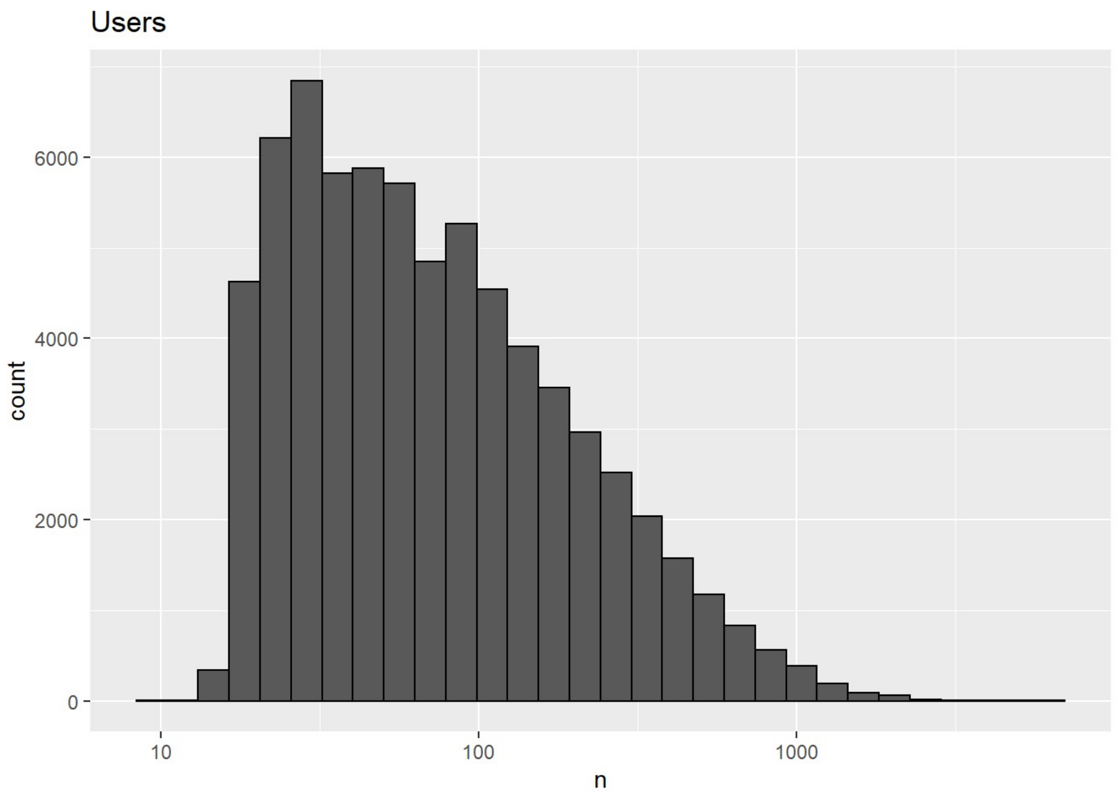
```
vect_ratings<-vec_ratings[vec_ratings!=0]
vec_ratings<-factor(vec_ratings)
qplot(vec_ratings)+ggtitle("Ratings Distribution")
```



The graph illustrated the general tendency of the users to rate the movie between 3 and 4. Additionally the majority of the users try to avoid an in-between ranking (eg: 0.5,1.5,2.5,3.5 and 4.5)

The distribution of each user’s ratings for movie

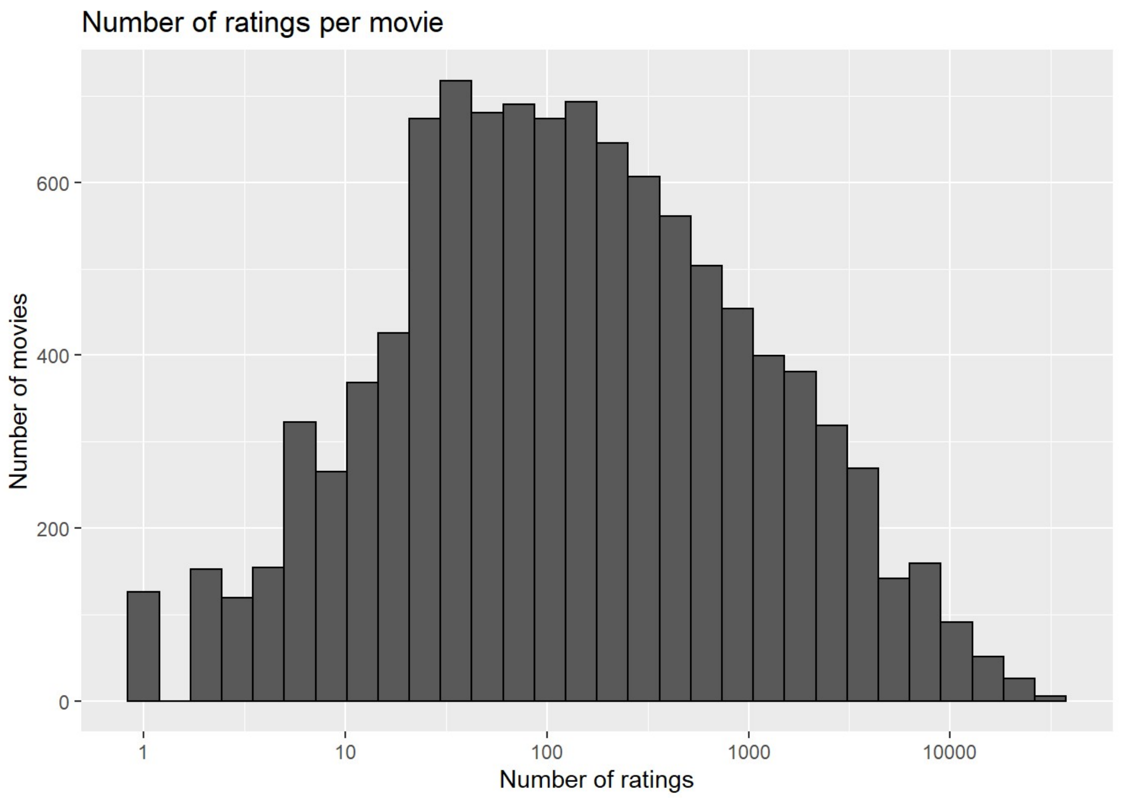
```
edx%>%count(userId)%>%
  ggplot(aes(n))+
  geom_histogram(bins=30,color="black")+
  scale_x_log10()+
  ggtitle("Users")
```



Not all users are actively giving the movie ratings hence the presence of users bias.Therefore we need to include user penalty term in our models later.

Plot number of ratings per movie

```
edx%>%count(movieId)%>%
  ggplot(aes(n))+
  geom_histogram(bins=30,color="black")+
  scale_x_log10()+
  xlab("Number of ratings")+
  ylab("Number of movies")+
  ggtitle("Number of ratings per movie")
```

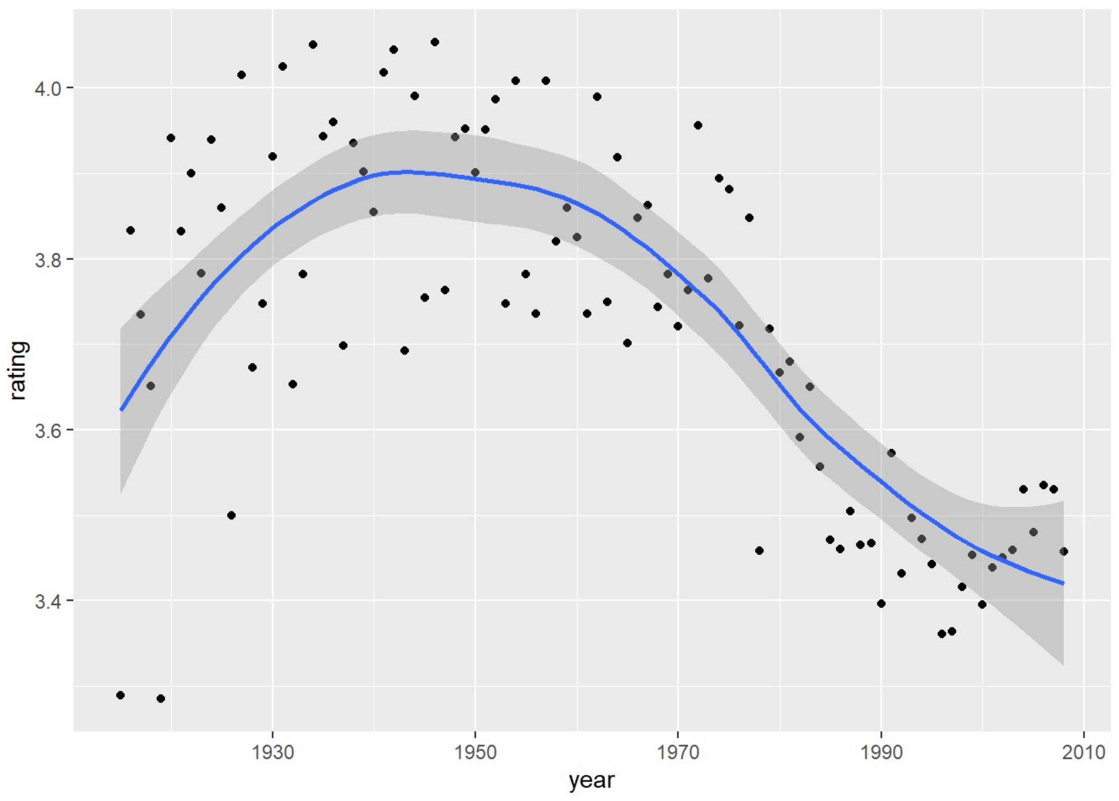


Some movies are rated more often than others leading to movies bias.

Rating vs release year

```
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'
```



This graph depicts the decreasing trend of the user rating habits from the year 1950 onward.The most recent years have in average lower rating than in the earlier years.

Modelling Approach

Residual Mean Square Error(RMSE) measures accuracy and quantify the typical error when predicting the movie ratings.It can be defined as:

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

where; N is the number of users, movie ratings, and the sum incorporating the total combinations.

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

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

Average movie rating model

First we build the simplest possible recommender system by predicting the same rating for all movies regardless of the rating that the users give and the movie effect.

$$Y - u, i = \mu + epsilon_{u,i}$$

where $Y_{u,i}$ is the prediction, $\epsilon_[u, i]$ is the independent error, and μ the expected “true” rating for all movies. It makes the assumption that all differences in movie ratings are explained by random variation alone.

```
# Compute the dataset's mean rating
mu<-mean(edx$rating)
mu
```

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

```
rmse_naive <- RMSE(validation_CM$rating,mu)
rmse_naive
```

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

We populate the results table with the first RMSE:

```
## Save Results in Data Frame
rmse_results<-data_frame(method="Naive Analysis by Mean",RMSE=rmse_naive)
rmse_results%>%knitr::kable()
```

method	RMSE
Naive Analysis by Mean	1.061202

Hence we get our baseline RMSE to compare with next modelling approaches.

Movie Effect Model

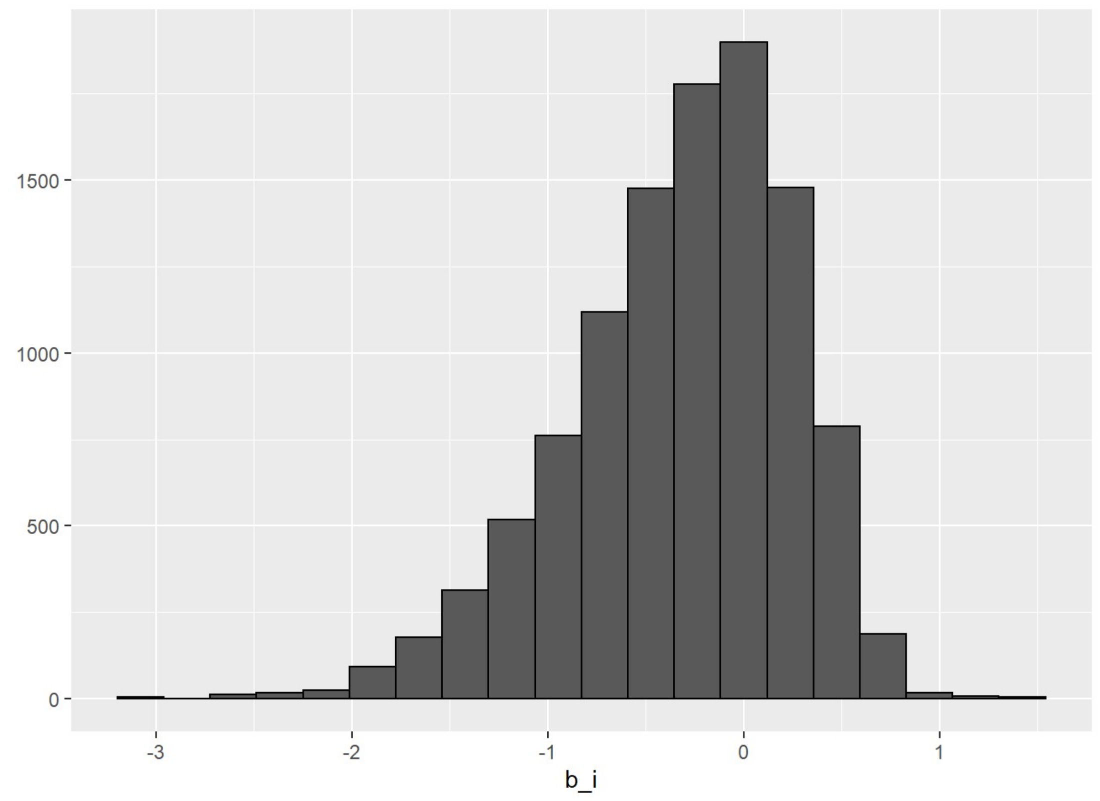
It calculates a bias term for each movie based on the difference between the movies mean rating and the overall mean rating.

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

where $Y_{u,i}$ is the prediction, $\epsilon_{u,i}$ is the independent error, and μ the mean rating for all movies, and b_i is the bias for each movie i

Penalty term of movie effect:

```
#Penalty Term(b_i)-Movie Effect

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"))
```



This histogram is skewed to the left and more movies have negative rating effect.

Using Movie Effect Model for Prediction:

```
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 ))

# save rmse results in a table
rmse_results%>%knitr::kable()
```

method	RMSE
Naive Analysis by Mean	1.0612018
Movie Effect Model	0.9439087

```
rmse_results
```

```
## # A tibble: 2 x 2
##   method                  RMSE
##   <chr>                  <dbl>
## 1 Naive Analysis by Mean 1.06
## 2 Movie Effect Model     0.944
```

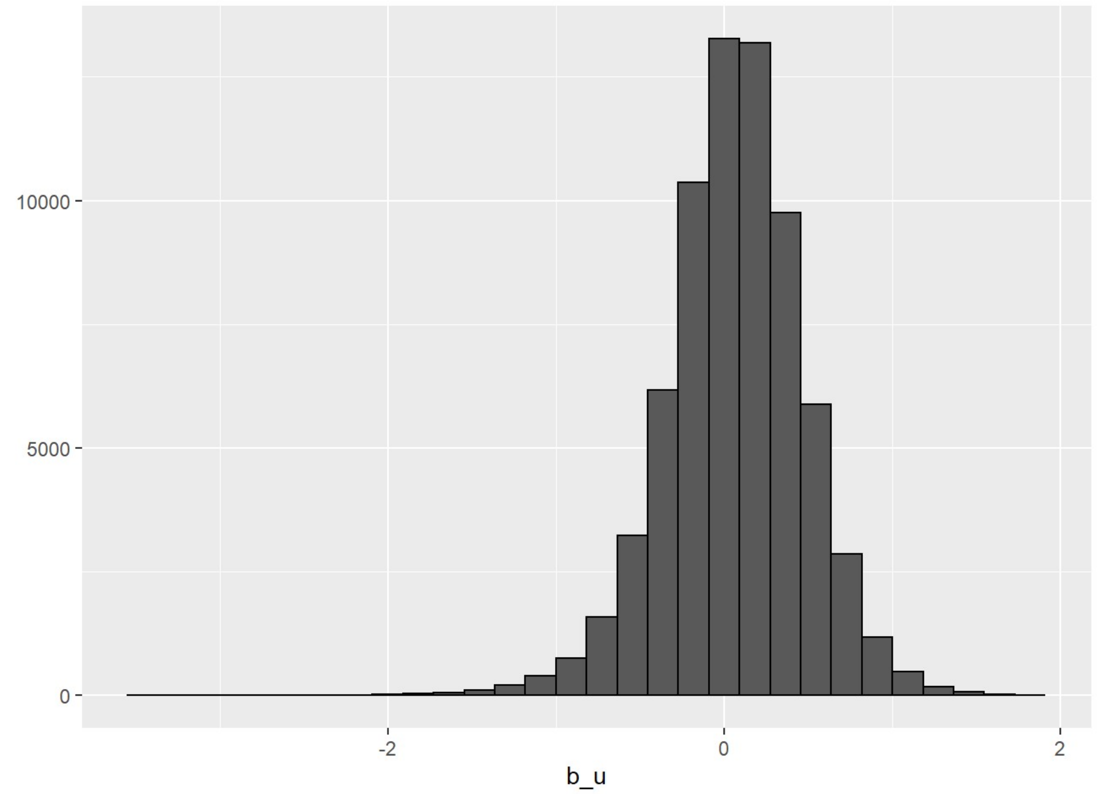
There is improvement of RMSE once we predict using this model,however this model does not consider the individual user rating effect.

Movie And User Effect Model

Next we incorporate the user bias into the model.

Penalty term of user effect:

```
#Penalty Term(b_u)-User Effect:

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"))
```



This histogram is showing significant variability among the users as different users rate the movie with their own preferences.

Now we construct the predictors by taking account of the movei and user effect and there is further improvement of RMSE.

```
# User test set,join movie averages and 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,nu='userId')%>%
  mutate(pred=mu+b_i+b_u)
```

```
## Joining, by = "userId"
```

```r
# 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()
```

method	RMSE
Naive Analysis by Mean	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

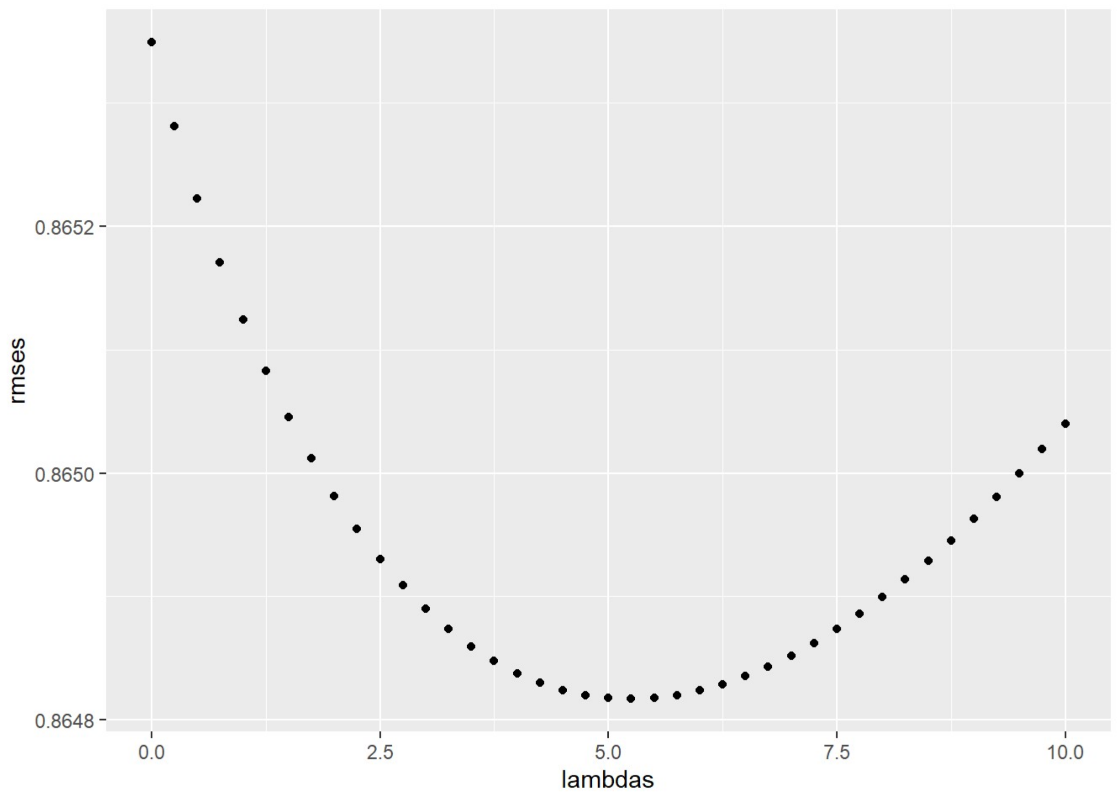
Regularized Movie and User Effect Model

Estimates of b_i and b_u are due to movies with very few ratings and in some users that only rated a very small number of movies and this can strongly influence the prediction.Regularisation allows for reduced errors caused by movies and users with few ratings which can skew the error metric. We should find the value of lambda (that is a tuning parameter) that will minimize the RMSE.

```r
# lambda is a tuning parameter
# User cross-validation to choose it
lambdas<-seq(0,10,0.25)

# For each lambda,find b_i & b_u, followed by rating prediction & testing
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(validation_CM$rating,predicted_ratings))
   })
```

We then plot rmse vs lambdas to select the optimal lambda value.

```r
qplot(lambdas,rmses)
```



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

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

```
# Compute regularized estimates of b_i using lambda
movie_avgs_reg<-edx %>%
  group_by(movieId)%>%
  summarize(b_i=sum(rating-mu)/(n()+lambda),n_i = n())

# Compute regularized estimates of b_u using lambda
user_avgs_reg<-edx %>%
  left_join(movie_avgs_reg,by='movieId')%>%
  group_by(userId)%>%
  summarize(b_u=sum(rating-mu-b_i)/(n()+lambda),n_u = n())

# Predict ratings
predicted_ratings_reg<-validation%>%
  left_join(movie_avgs_reg,by='movieId')%>%
  left_join(user_avgs_reg,by='userId')%>%
  mutate(pred=mu+b_i+b_u) %>%
  .$pred

# Test and save results

model_3_rmse<-RMSE(validation_CM$rating,predicted_ratings_reg)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie and User Effect Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Naive Analysis by Mean	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170

```
rmse_results
```

```
## # A tibble: 4 x 2
##   method                                   RMSE
##   <chr>                                   <dbl>
## 1 Naive Analysis by Mean                  1.06
## 2 Movie Effect Model                      0.944
## 3 Movie and User Effect Model             0.865
## 4 Regularized Movie and User Effect Model 0.865
```

The RMSE following regularization approach of Movie and User Effect Model had led to further drop of RMSE which is much better.

Results

The RMSE values of all the represented models are shown below:

```
# RMSE results overview
rmse_results%>%knitr::kable()
```

method	RMSE
Naive Analysis by Mean	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170

Conclusion

The models from most accurate to least accurate are as follows; Regularized Movie,User,Year and Genre Effect Model and Regularized Movie and User Effects Model.

The final model optimised for the prediction is the following;

$$Y_{u,i} = \mu + b_{i,n,\lambda} + b_{u,n,\lambda} + \epsilon_{u,i}$$

In summary, a machine learning algorithm had been built to predict movie ratings with MovieLens dataset. The regularization approach is the optimal model for prediction with the lowest RMSE value among all methods. The final RMSE achieved by regularization of movie and user effect model is 0.8648 with an improvement over 18.5% compared to the average movie rating model. We could also improve the RMSE result by futher regularization by adding other effects (genre, year and age). However, due the hardware limitation (RAM) it was not run in this project.If I have hardware which is equipped with better setting ,I would like to include regularization of other effects in order to obtain a lesser RMSE value in the future.

My “movieLens Github repository” is **in this link (https://github.com/zenasimlab/Capsone_MovieLens)**

References: 1.https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+2T2019/courseware/dd9a048b16ca477a8f0aaf1d888f0734/e8800e37aa444297a3a2f35bf84ce452/?child=last