CAPSTON PROJECT MOVIE RATING ANALYSIS

BY:

TAMIL PAVAI. P

510421103019

3rd YEAR

CIVIL ENGINEERING

ARUNAI ENGINEERING COLLEGE

title: "Capstone Project: MovieLens" author: "Reza Hashemi" date: "8/11/2019" output: pdf_document: default html_document: default —

Introduction

One of the major families of applications of machine learning in the information technology sector is the ability to make recommendations of items to potential users or customers. In year 2006, Netflix has offered a challenge to data science community. The challenge was to improve Netflix???s in house software by 10% and win $1M prize.

This capstone project is based on the winner???s team algorithm and is a part of the course HarvardX:??PH125.9x Data Science: Capstone project. The Netflix data is not freely available so an open source dataset from movieLens '10M version of the MovieLens dataset'??has been used. The aim of this project is to develop a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set. Several machine learning algorithm has been used and results have been compared to get maximum possible accuracy in prediction.

This report contains problem definition, data ingestion, exploratory analysis, modeling and data analysis, results and concluding remarks and have been written in that order.

Problem Defnition

This capstone project on 'Movie recommendation system??? predicts the movie rating by a user based on users past 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 so easy given that there are many different type of biases present in the movie reviews. It can be different social, psychological, demographic variations that changes the taste of every single users for a given particular movie. However the problem can still be designed to tackle major biases which can be expressed via mathematical equations relatively easily. 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 different biases. In the algorithm, the prevalences can be suppressed using some clever mathematical tricks. This will become clear as we follow this document.

Data Ingestion

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+2T2018/courseware/dd9a048b16ca477a8f0aaf1d888f0734/e8800e37aa444297a3a2f35bf84ce452/?child=first

```r
#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-project.org")
# 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 <- 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)
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)
```

```r
#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-project.org")
# 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 <- 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)
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 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').

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

```r
# extra libraries that might be usefull in analysis and visulizations
library(ggplot2)
library(lubridate)
```

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

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

Once a clean dataset is available, one must inquire the dataset features and calculate the basic summary statistics

```r
## the dataset and its basic summary statistics
# 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     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                           genres
## 1                 Comedy|Romance
## 2          Action|Crime|Thriller
## 3                         Comedy
## 4  Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
```

```
# basic summary statistics
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  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
##     title              genres
##  Length:9000061     Length:9000061
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

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

We can see the dataset is in the tidy format and ready for exploration and analysis.

Data pre-processing and 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 = "\\|")
split_valid <- validation   %>% separate_rows(genres, sep = "\\|")
split_valid_CM <- validation_CM  %>% separate_rows(genres, sep = "\\|")
```

##Data Exploration and general statistics

```
# The 1st rows of the edx & split_edx datasets are presented below:
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                          genres year
## 1                Comedy|Romance 1992
## 2         Action|Crime|Thriller 1995
## 3                        Comedy 1994
## 4  Action|Drama|Sci-Fi|Thriller 1995
## 5       Action|Adventure|Sci-Fi 1994
## 6 Action|Adventure|Drama|Sci-Fi 1994
```

```
head(split_edx)
```

```
##   userId movieId rating timestamp                title   genres year
## 1      1     122      5 838985046     Boomerang (1992)   Comedy 1992
## 2      1     122      5 838985046     Boomerang (1992)  Romance 1992
## 3      1     185      5 838983525      Net, The (1995)   Action 1995
## 4      1     185      5 838983525      Net, The (1995)    Crime 1995
## 5      1     185      5 838983525      Net, The (1995) Thriller 1995
## 6      1     231      5 838983392 Dumb & Dumber (1994)   Comedy 1994
```

```
# edx Summary Statistics
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  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
##     title              genres               year
##  Length:9000061     Length:9000061     Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1994
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

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

Total movie ratings per genre

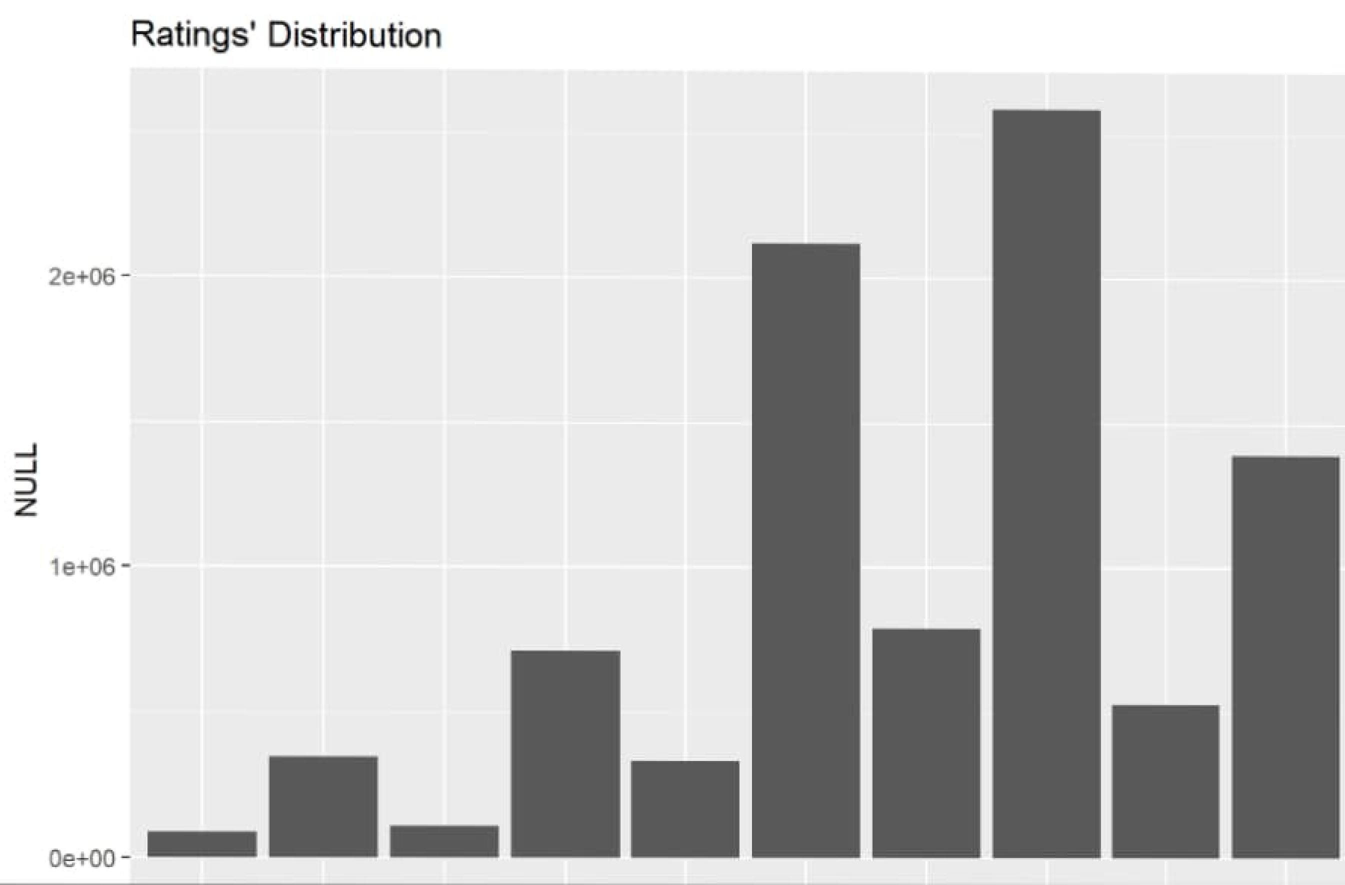
```
genre_rating <- split_edx%>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

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")
```

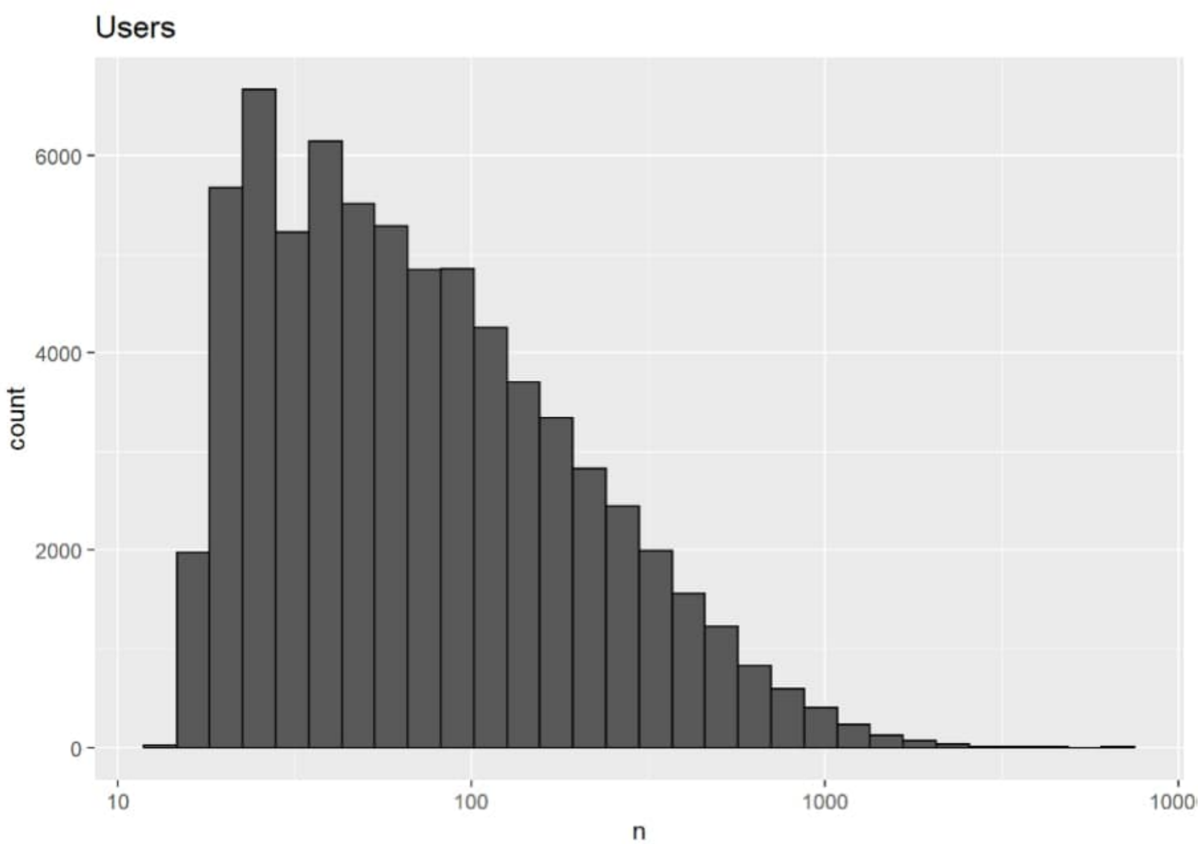


Data Analysis Strategies

- Some movies are rated more often than others (e.g. blockbusters are rated higher). How to incorporate this in our model: find movie bias.
- Some users are positive and some have negative reviews because of their own personal liking/disliking regardless of movie. How to address this characteristics: find users bias.
- The popularity of the movie genre depends strongly on the contemporary issues. So we should also explore the time dependent analysis. How to approach this idea: find the genre popularity over the years
- Do the users mindset also evolve over time? This can also effect the average rating of movies over the years. How do visualize such effect: plot rating vs release year

The distribution of each user's ratings for movies. This shows the users bias

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

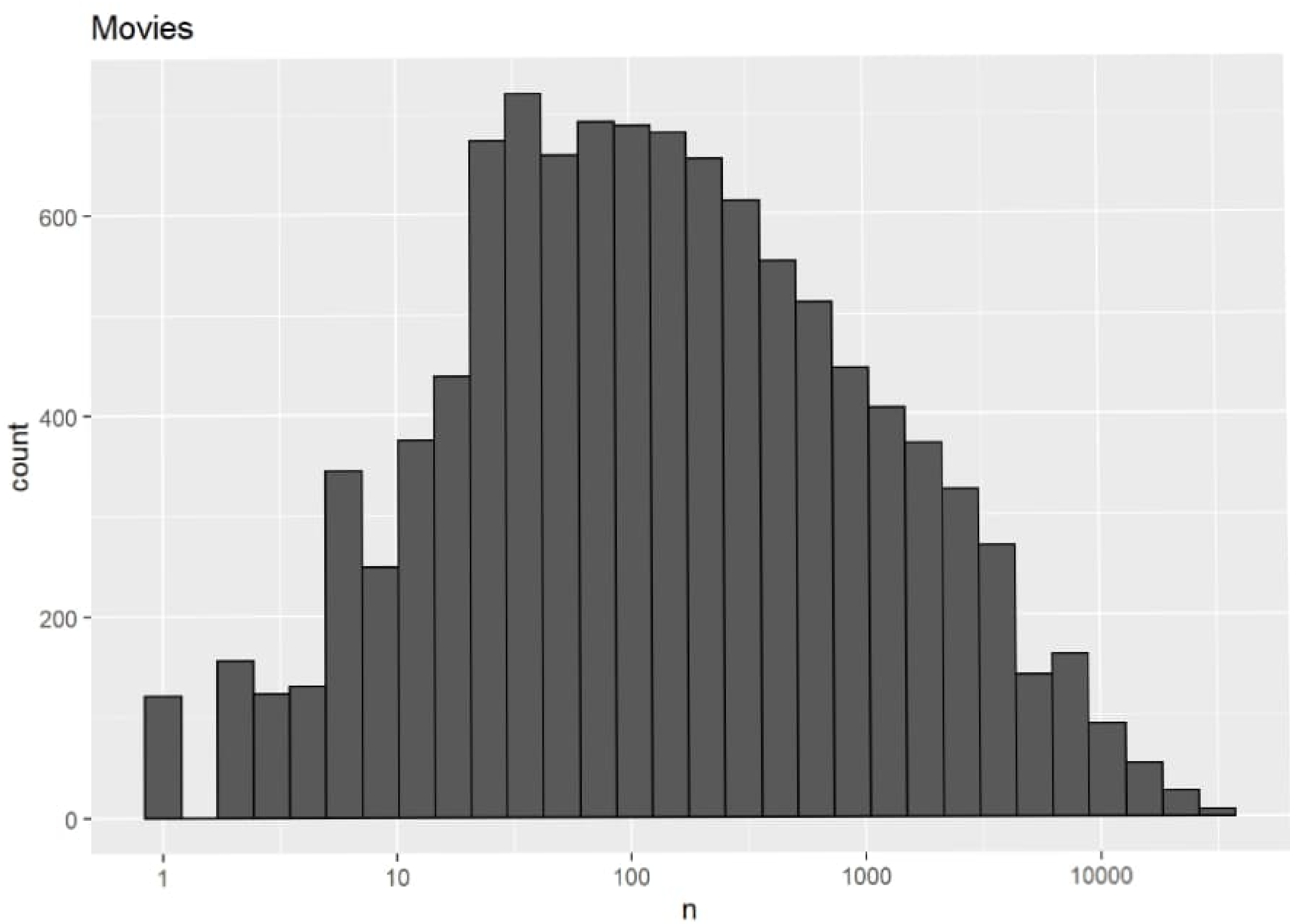


Above plot shows that not every user is equally active. Some users have rated very few movie and their opinion may bias the prediction results.

Some movies are rated more often than others.Below is their distribution. This explores movie biases.

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

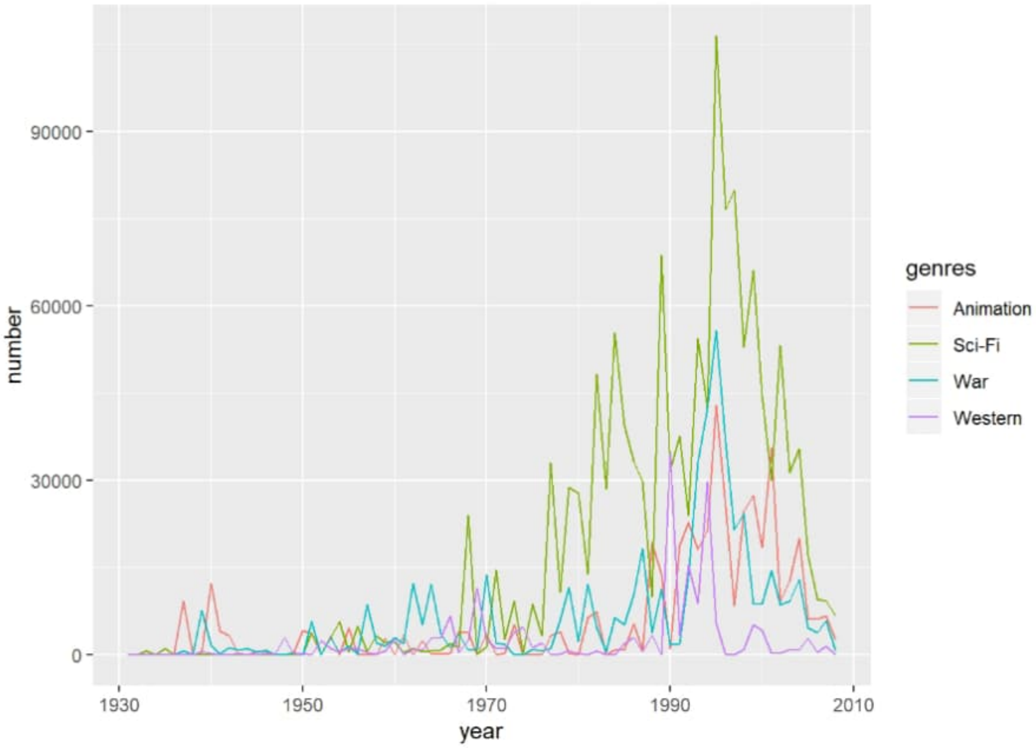
```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



The histogram shows some movies have been rated very few number of times. So they should be given lower importance in movie prediction.

Genres popularity per year. Here we tackle the issue of temporal evolution of users taste over different popular genre.

```
genres_popularity <- split_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 movie
s.
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")
```

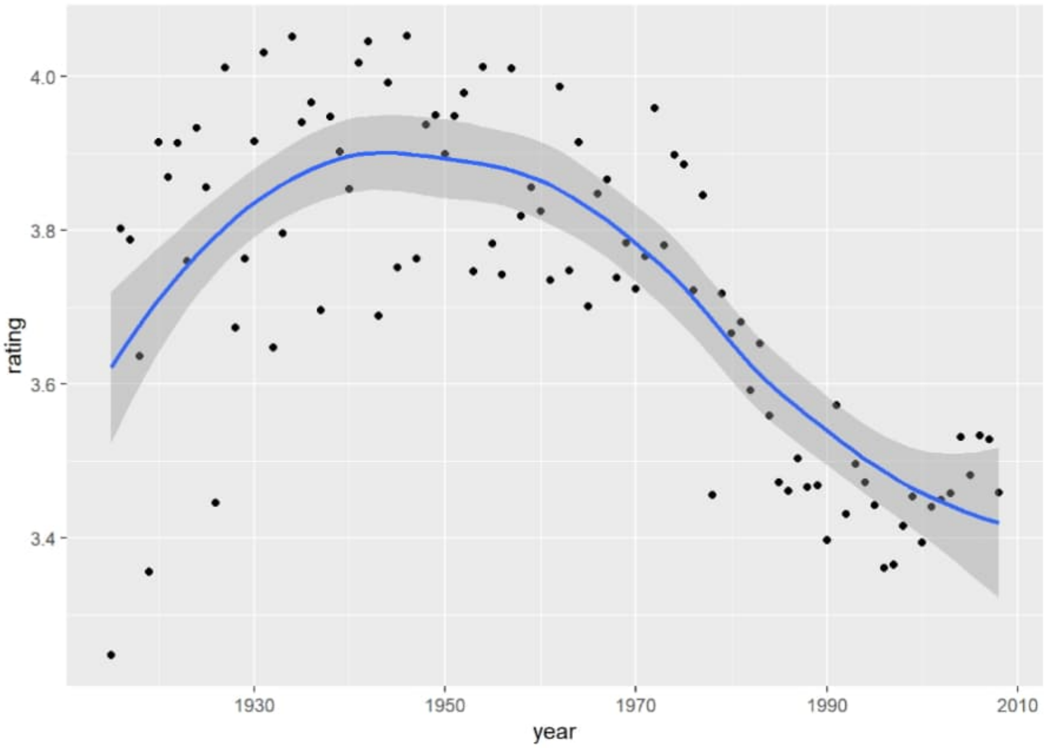


This plots depicts some genre become more popular over others for different period of time.

Rating vs release year. Here, a general trend of movie viewers and their rating habits can be explored.

```
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 general trend shows modern users relatively rate movies lower.

The general trend shows modern users relatively rate movies lower.

Data Analysis: Model Preparation

```
#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.
```

Simplest possible model

Dataset's mean rating is used to predict the same rating for all movies, regardless of the user and movie.

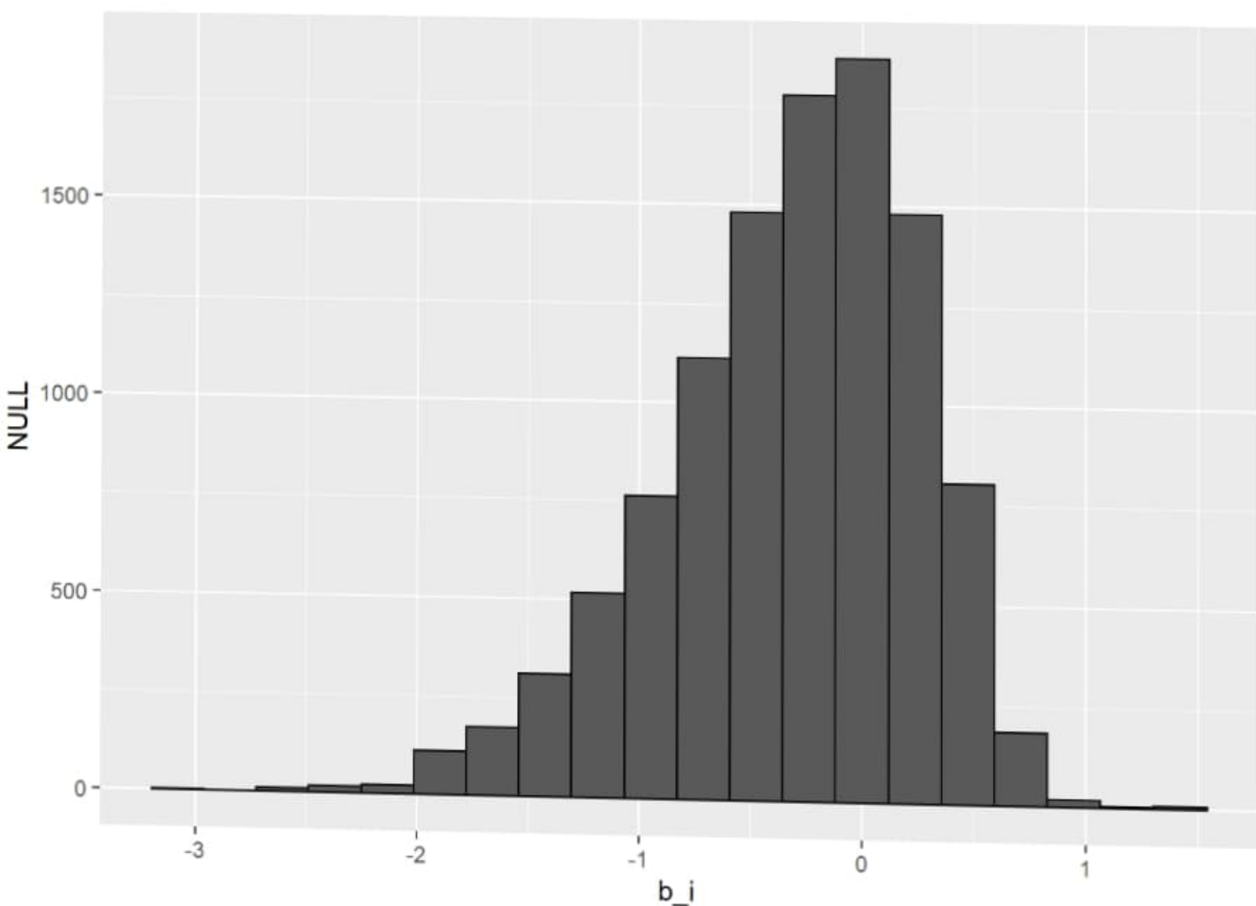
```
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512464
```

Penalty Term (b_i)- Movie Effect

Different movies are rated differently. As shown in the exploration, the histogram is not symmetric and is skewed towards negative rating effect. The movie effect can be taken into account by taking he difference from mean rating as shown in the following chunk of code.

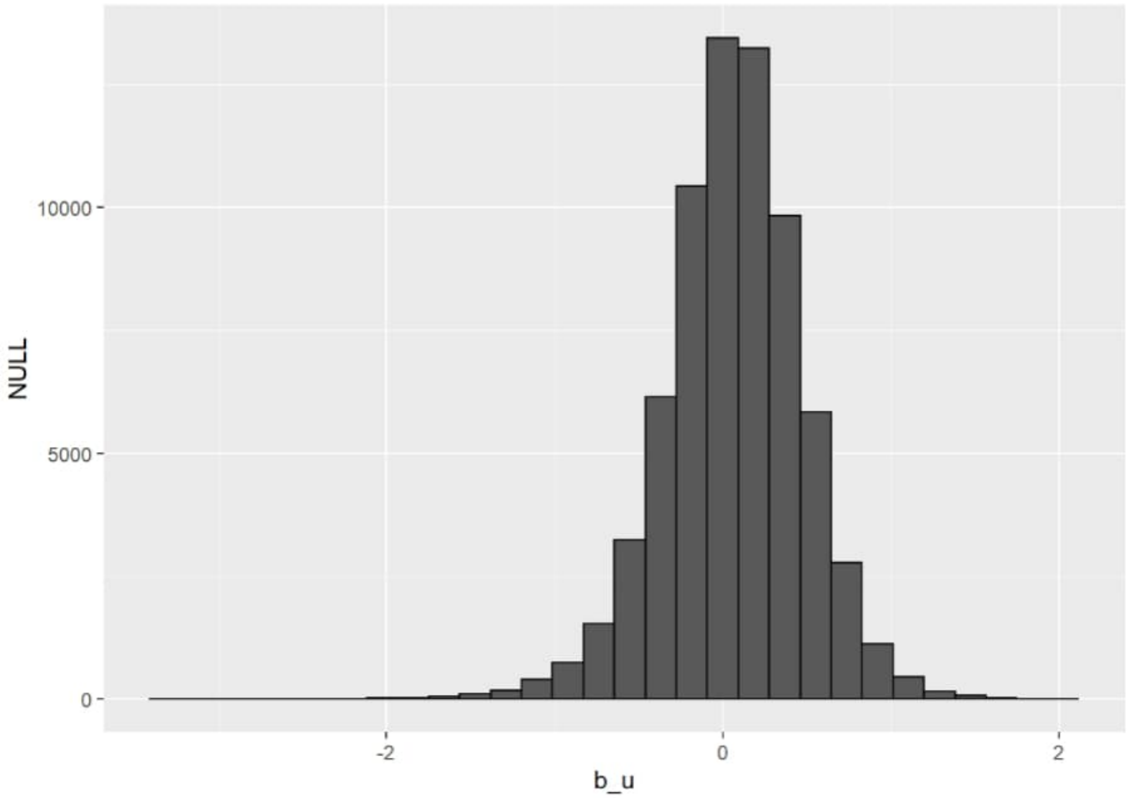
```
movie_avgs_norm <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs_norm %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("black"))
```



Penalty Term (b_u)- User Effect

Different users are different in terms of how they rate movies. Some cranky users may rate a good movie lower or some very generous users just don't care for assessment. We have already seen this pattern in our data exploration plot (user bias). We can calculate it using this code.

```
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"))
```



Model Creation

The quality of the model will be assessed by the RMSE (the lower the better).

Baseline Model

It's simply a model which ignores all the feathers and simply calculates mean rating. This model acts as a baseline model and we will try to improve RMSE relative to this baseline standard model.

```
# baseline Model: just the mean
baseline_rmse <- RMSE(validation_CM$rating,mu)
## Test results based on simple prediction
baseline_rmse
```

```
## [1] 1.060651
```

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

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

Movie Effect Model

An improvement in the RMSE is achieved by adding the movie effect.

```
# 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
Using mean only	1.0606506
Movie Effect Model	0.9437046

```
rmse_results
```

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

The error has drop by 5% and motivates us to move on this path further.

Movie and User Effect Model

Given that movie and users biases both obscure the prediction of movie rating, a further improvement in the RMSE is achieved by adding the user effect.

```
# 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()
```

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

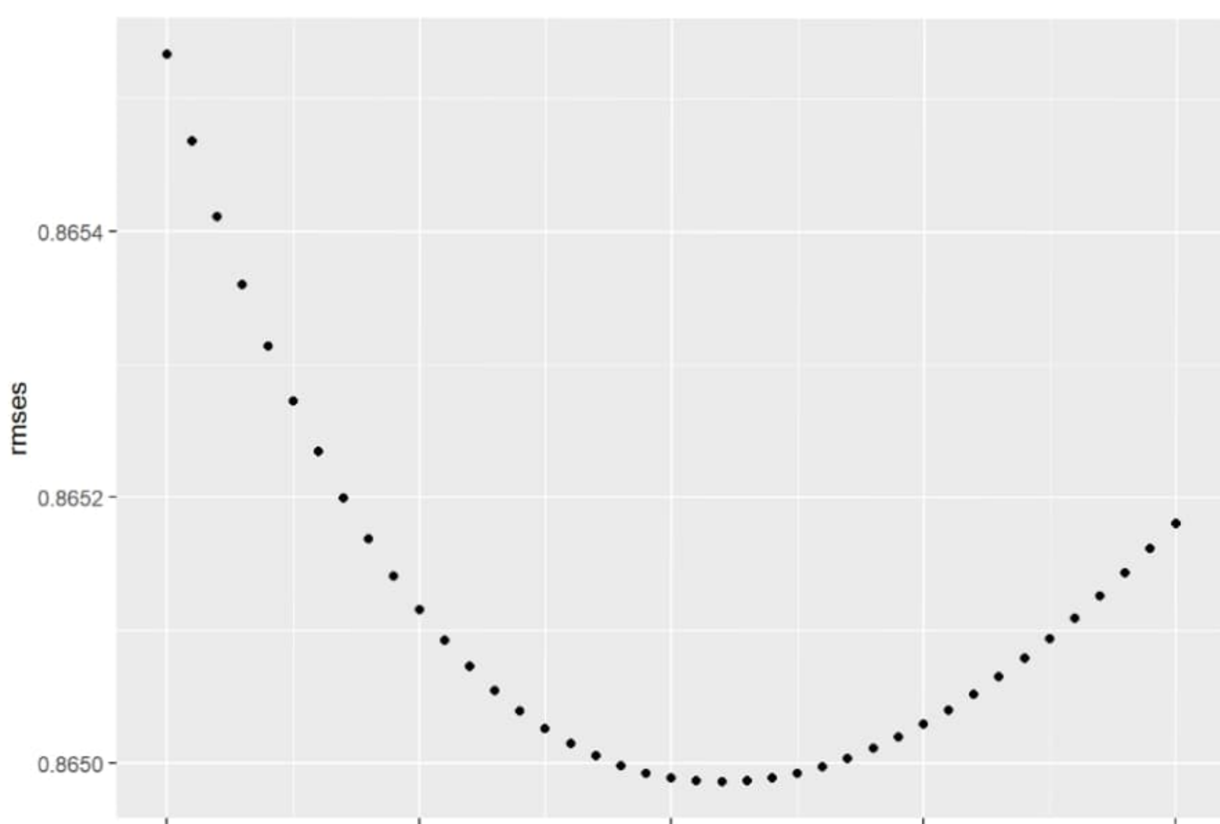
```
rmse_results
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Using mean only            1.06
## 2 Movie Effect Model         0.944
## 3 Movie and User Effect Model 0.866
```

Regularization based approach (motivated by Netflix challenge)

We have noticed in our data exploration, some users are more actively participated in movie reviewing. There are also users who have rated very few movies (less than 30 movies). On the other hand, some movies are rated very few times (say 1 or 2). These are basically noisy estimates that we should not trust. Additionally, RMSE are sensitive to large errors. Large errors can increase our residual mean squared error. So we must put a penalty term to give less importance to such effect.

```
# lambda is a tuning parameter
# Use cross-validation to choose it.
lambdas <- seq(0, 10, 0.25)
# For each lambda,find b_i & b_u, followed by rating prediction & testing
# note:the below code could take some time
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))
})
# Plot rmses vs lambdas to select the optimal lambda
qplot(lambdas, rmses)
```



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

```
## [1] 5.5
```

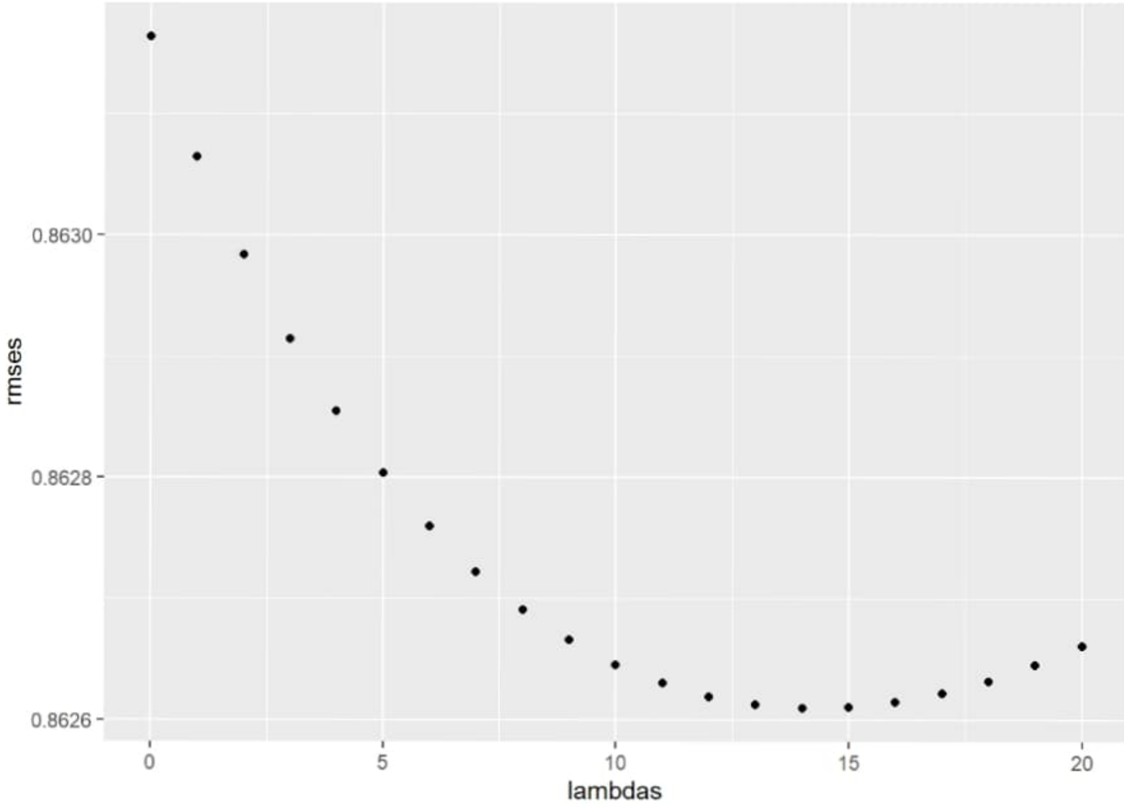
```r
# Compute regularized estimates of b_i using lambda
movie_avgs_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())
# Compute regularized estimates of b_u using lambda
user_avgs_reg <- edx %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  group_by(userId) %>%
  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
Using mean only	1.0606506
Movie Effect Model	0.9437046
Movie and User Effect Model	0.8655329
Regularized Movie and User Effect Model	0.8649857

Regularization using movies, users, years and genres.

The approach utilized in the above model is implemented below with the added genres and release year effects.

```
# 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())
    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)
```




```
lambda_2 <- lambdas[which.min(rmses)]
lambda_2
```

```
## [1] 14
```

```
movie_reg_avgs_2 <- split_edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_2), n_i = n())
user_reg_avgs_2 <- split_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 <- split_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())
genre_reg_avgs <- split_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()
```