MovieLens Rating Prediction

Carole Mrad

January 18, 2019

1. Introduction

Statistical and knowledge discovery techniques are applied to the problem of producing product recommendations or ratings through recommmender systems and on the basis of previously recorded data. In the present report, the products are the movies.

The present report covers the 10M version of the movieLens dataset available here https://grouplens.org/datasets/movielens/10m/. The main objective for using this dataset is to build a movie recommendation system that predicts user movie ratings.

The Netflix prize (i.e. challenge to improve the predictions of Netfix's movie recommender system by above 10% in terms of the root mean square error) reflects the importance and economic impact of research in the recommendation systems field.

Used Dataset

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

Data Loading

```r
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")
```

Used Libraries

The following libraries were used in this report:

```r
library(ggplot2)
library(lubridate)
library(caret)
library(tidyverse)
```

Aim & Objectives

The main objective in this report is to train a machine learning algorithm using the inputs of a provided subset (edx dataset) to predict movie ratings in a provided validation set.

Additionally, gglpot2 is used in the data exploration section to reveal some interesting trends in the dataset and the factors that affects the users' ratings. The assessments of the 4 models that will be developed is based on their resulting RMSE. Lastly, the optimal model is used to predict the movie ratings.

2. Methodology & Analysis

Data Pre-processing

Evaluation of Predicted Ratings using RMSE

Computing the deviation of the prediction from the true value is referred to as the Mean Average Error(MAE) and represents a typical way to evaluate a prediction. Another popular measure is the Root Mean Square Error (RMSE). In comparison to MAE, RMSE penalizes larger errors stronger and is hence suitable for situations where minor prediction errors are not very important.In this report, the RMSE value is used to evaluate each model.

```
# function to calculate the RMSE values
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2,na.rm = T))
}
```

Split Raw Data: Train and Test Sets

The prediction of users' ratings for movies that they haven't seen yet will be achieved by building an algorithm on which the movielens dataset is partitioned into 2 sets: edx dataset used for building the algorithm and the validation set used for testing. The validation set represents 10% of the 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,]

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

# Remove unneeded files to free some RAM
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Modifying the Year & Genre

In order to use dependencies between the release year and rating, the release year will be included in a separte column instead of being hidden in the title column between parantheses. The same applies to the genres, hence it is necessary to split the multiple genres for each movie into separate rows.

```
# 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)))

# Modify the genres variable in the edx & validation datasets

split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
split_valid <- validation  %>% mutate(year = as.numeric(str_sub(validation$title,-5,-2))) %>% separate_r
split_valid_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(validation_CM$title,-5,-2))) %>% se
```

Data Exploration & Visualization

Summary Statistics/General Data Information

```
# 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     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
```

```
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     292      5 838983421  Outbreak (1995)   Action 1995
```

```
# edx Summary Statistics
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
```

```
# 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
```

Total Movie Ratings per Genre

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

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
## 20 (no genres listed)       7
```

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) (1~ 25672
## 10    150 Apollo 13 (1995)                                             24284
## # ... with 10,667 more rows
```

The Top 5 Most Given Ratings

```
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5) %>%
  arrange(desc(count))
```

```
## Selecting by count
```

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

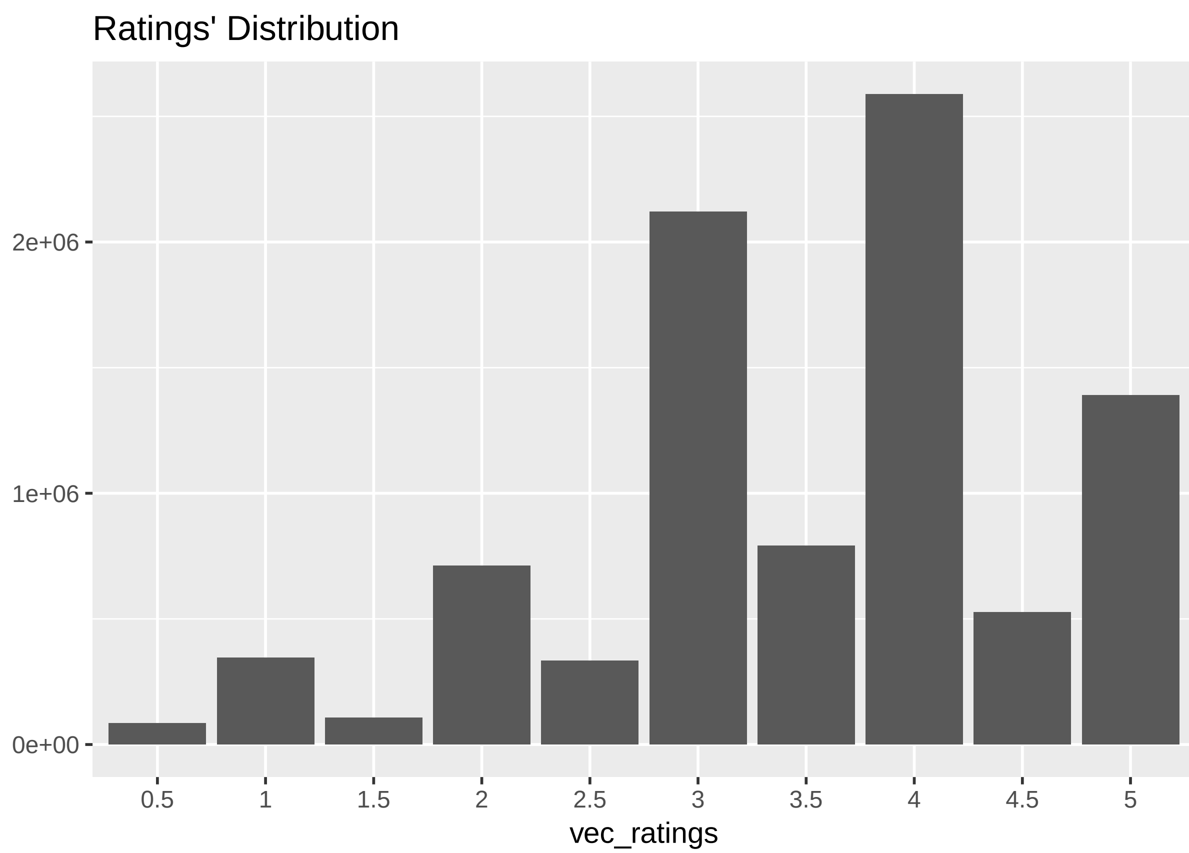
Ratings Distribution

Users have a preference to rate movies rather higher than lower as demonstrated by the distribtion of ratings below. A rating of 3 and 4 represent the most given ratings. In general, half star ratings are less common than whole star ratings (e.g., there are fewer ratings of 3.5 than there are ratings of 3 or 4, etc.).

```
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
```

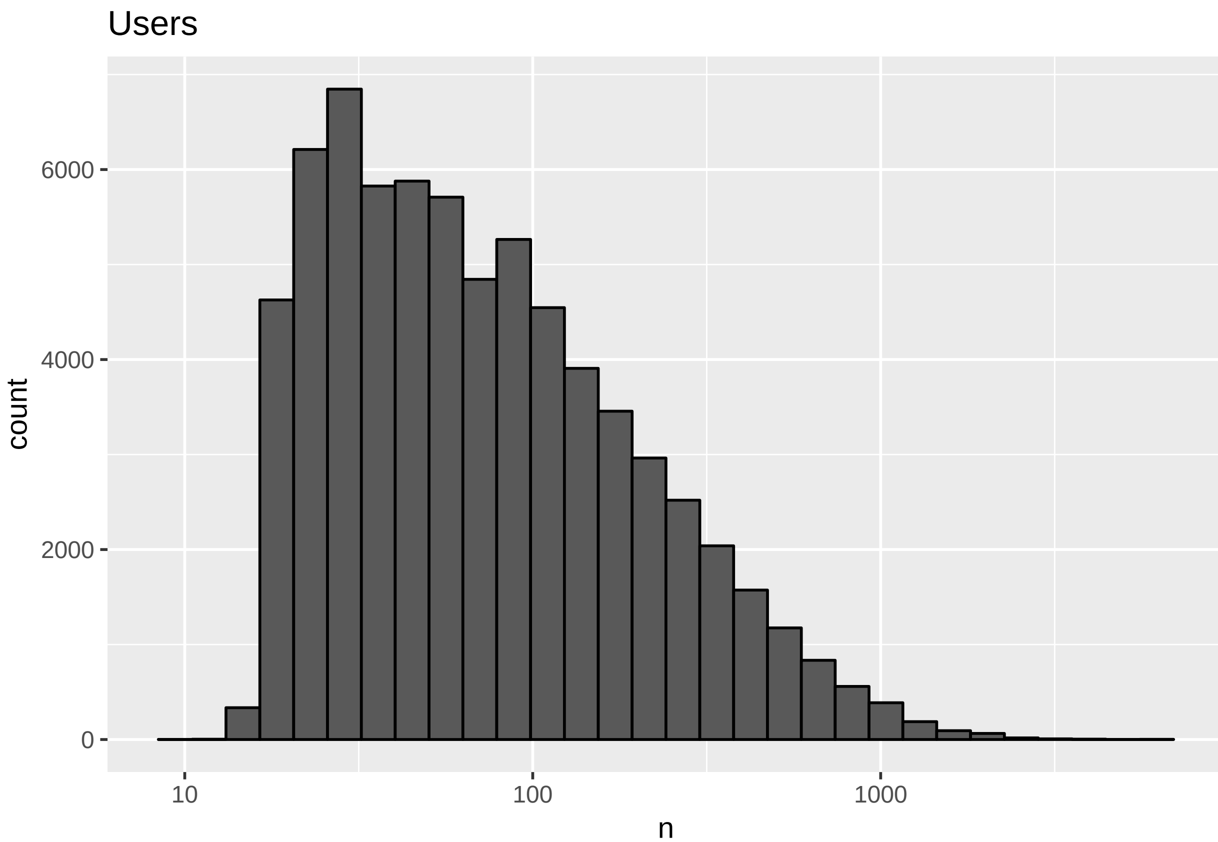
```
vec_ratings <- vec_ratings[vec_ratings != 0]
vec_ratings <- factor(vec_ratings)
qplot(vec_ratings) +
  ggtitle("Ratings' Distribution")
```



The distribution of each user's ratings for movies

The majority of users have rated below 100 movies, but also above 30 movies (a user penalty term will be included in the models).

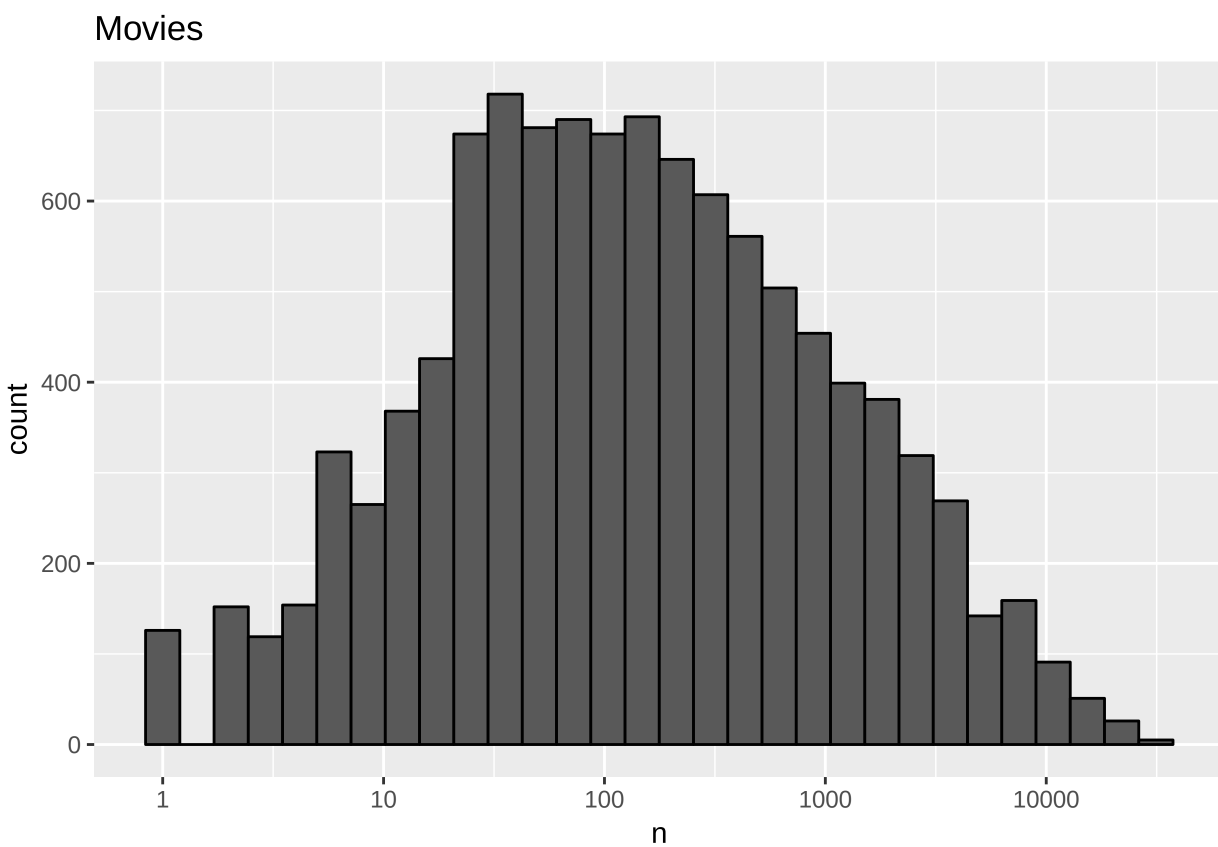
```
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



Some movies are rated more often than others.Below is their distribution:

The histogram below shows that the majority of movies have been reviewed between 50 and 1000 times. A challenge to the ratings prediction is reflected by an interesting finding:around 125 movies have been rated only once. Thus regularization and a penalty term will be applied to the models in this report.

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

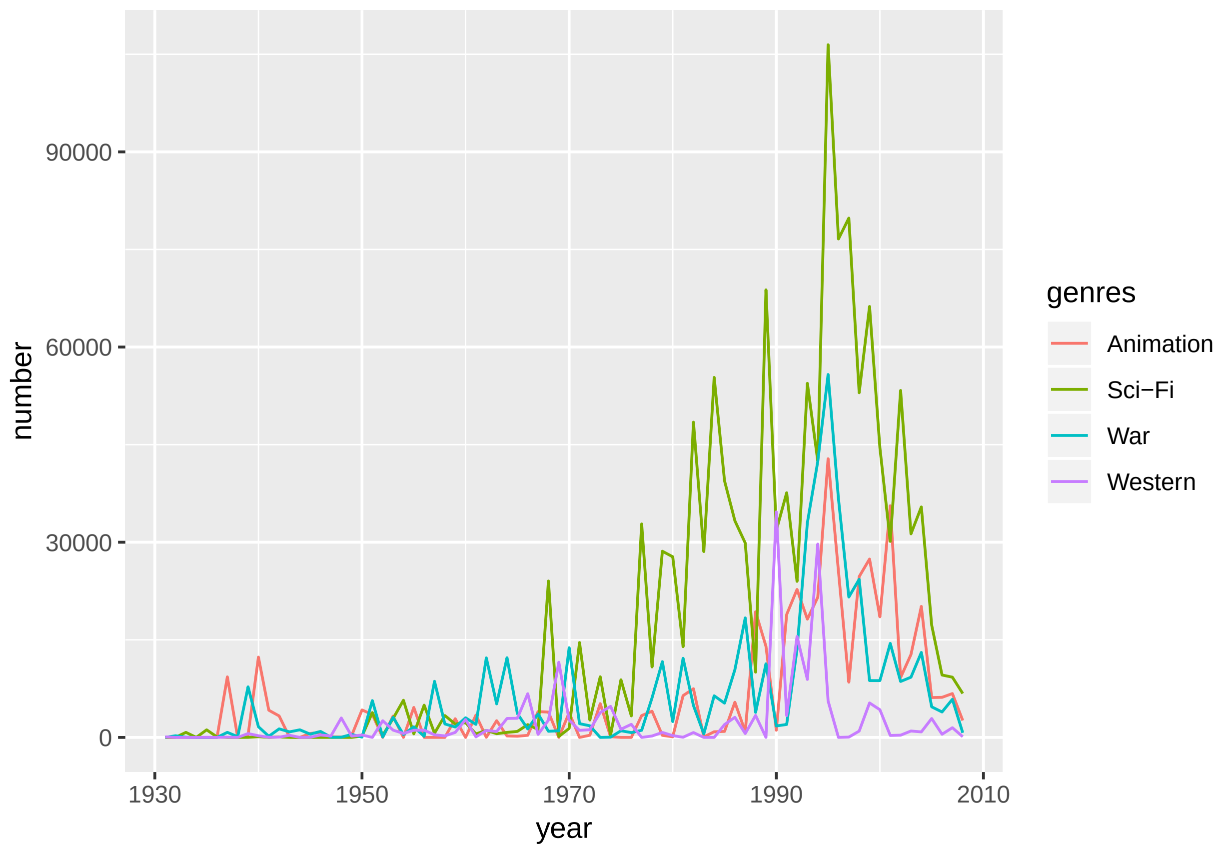


Genres Popularity per Year

```
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 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")
```



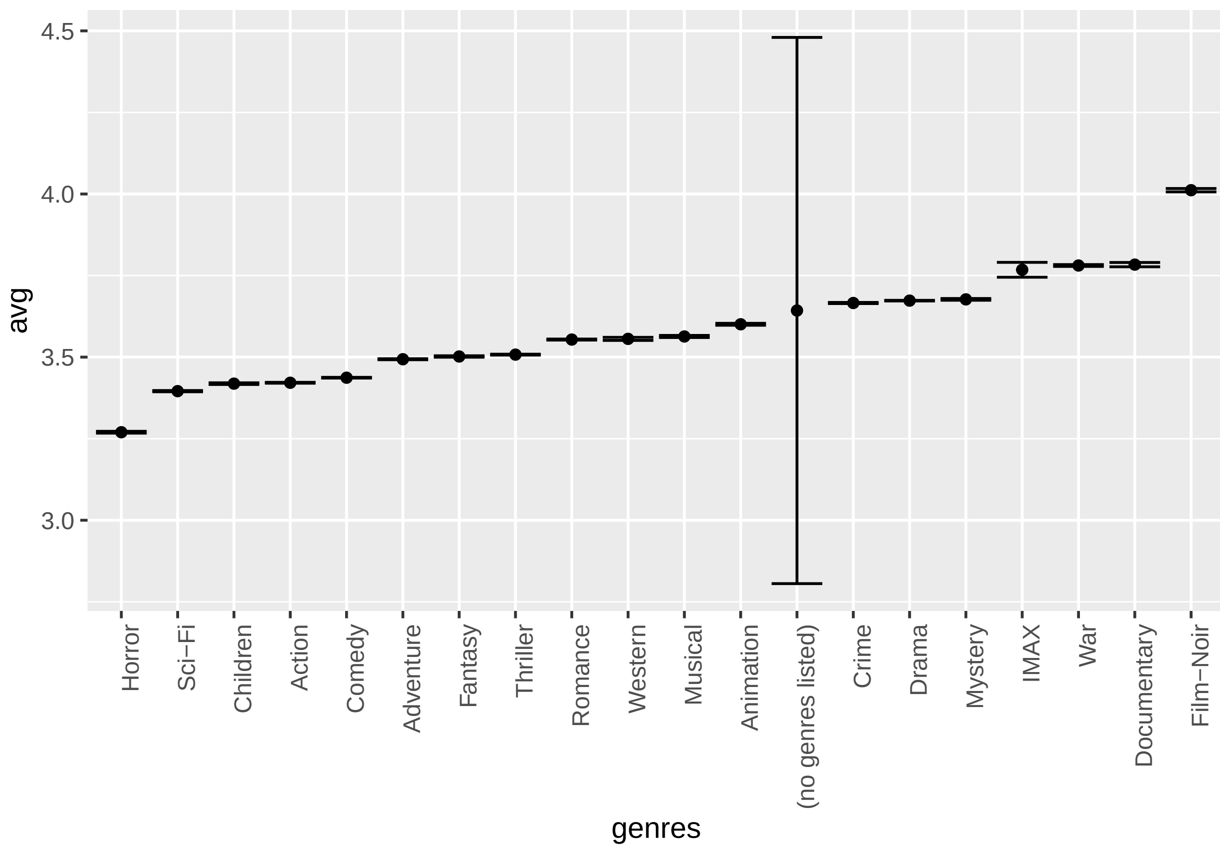
Some interesting trends are observed in the above figure. The period after 1969 which represents the year of the first Moon landing coincides with a growing interest in Sci-fi movies.Additionally,high number of westerns in 1990s is observed reflecting the time when westerns popularity was peaking. As for animated movies, their popularity increased after 1990 probably due to the advancement in computer animation technology which made the production much easier. Lastly, War movies were popular around the time when major military conflicts occured (World War II, Vietnam War, etc.). Hence, the release year and genre affects the user's rating.

The Effects of Release Year and Genres on Ratings

Rating vs Genres

The possible effect of genres on rating was partially explored in the genre popularity section. Users have varied preferences with respect to the movies' genre: Film_noir,IMAX, Documentary, and War represent the primary selection for users, whereas, Horror, Sci-Fi, and Children are less preferable.

```
split_edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

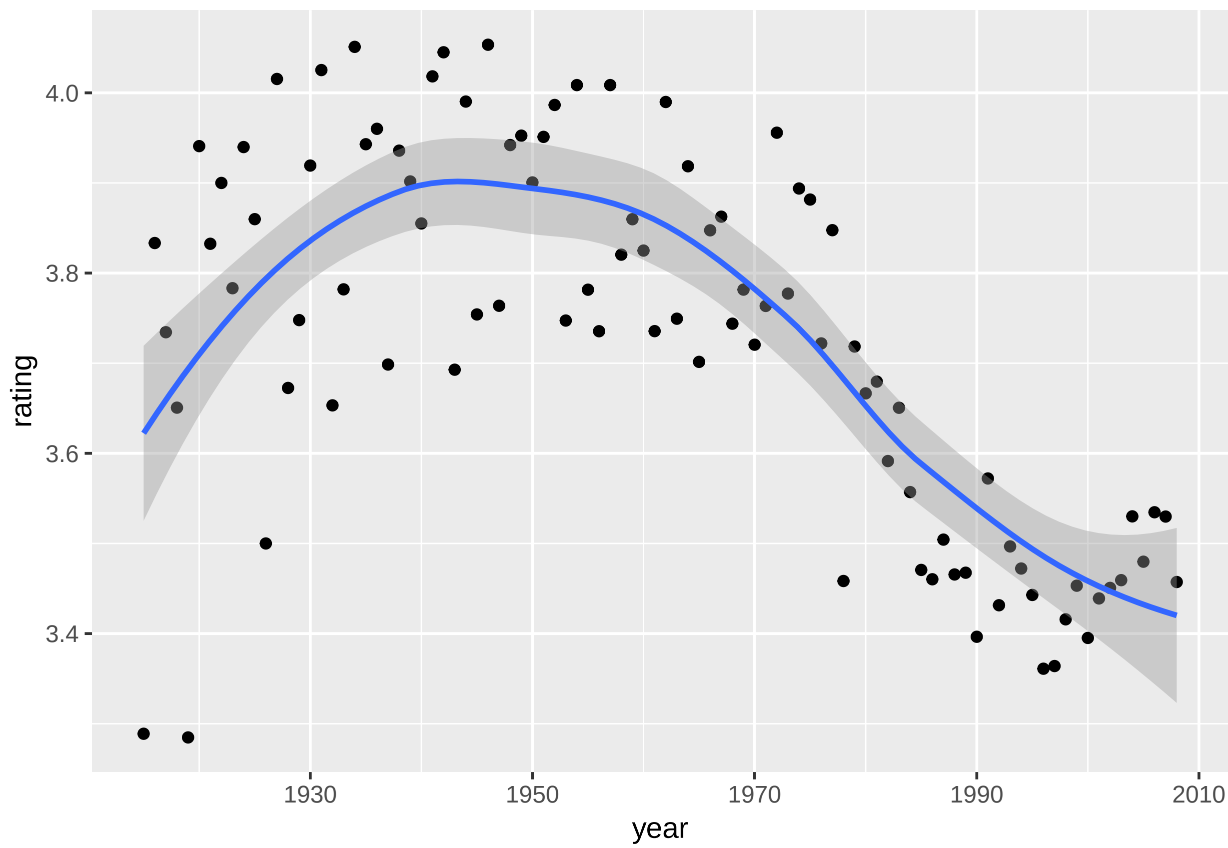


Rating vs Release Year

A clear trend is shown in the below figure: the most recent years have in average lower rating than earlier years.

```
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'
```



Data Analysis/Model Preparation

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

Sample estimate- mean

The initial step is to compute the dataset's mean rating.

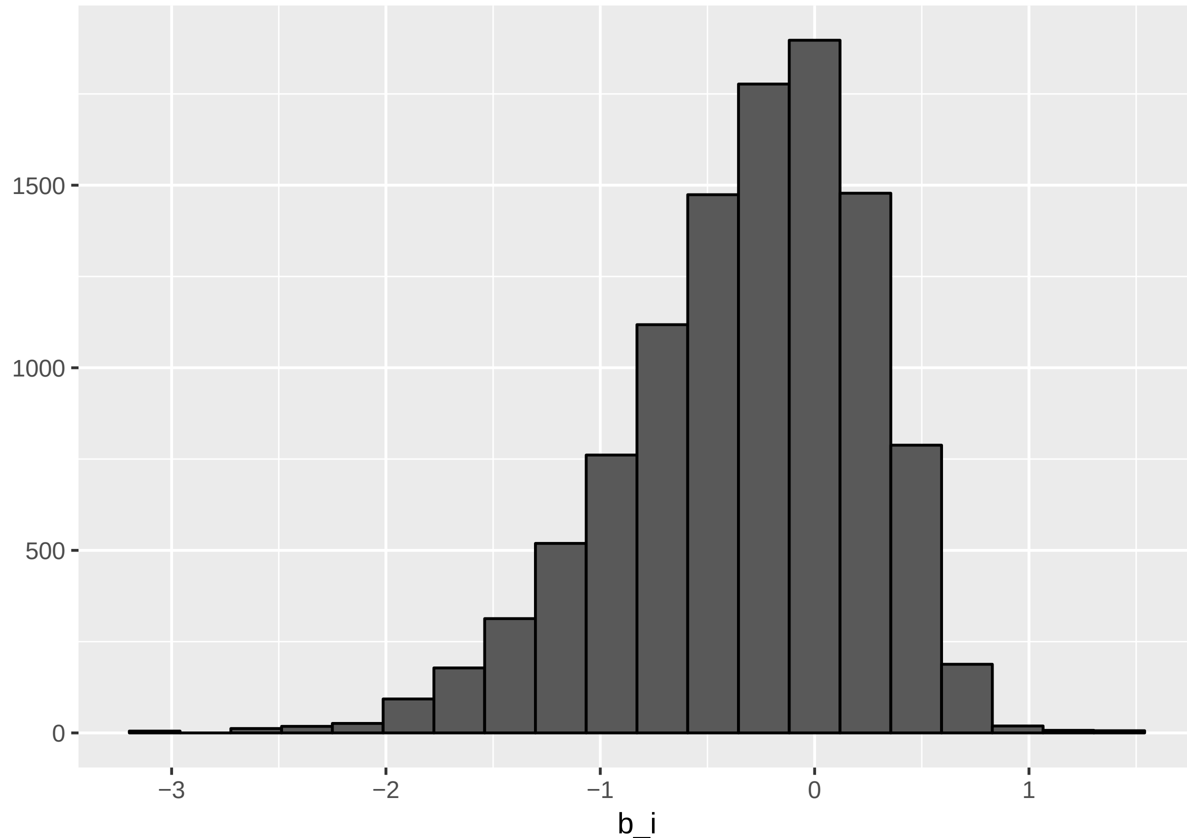
```
mu <- mean(edx$rating)
mu
```

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

Penalty Term- Movie Effect

Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies. The histogram is left skewed, implying that more movies have negative effects.

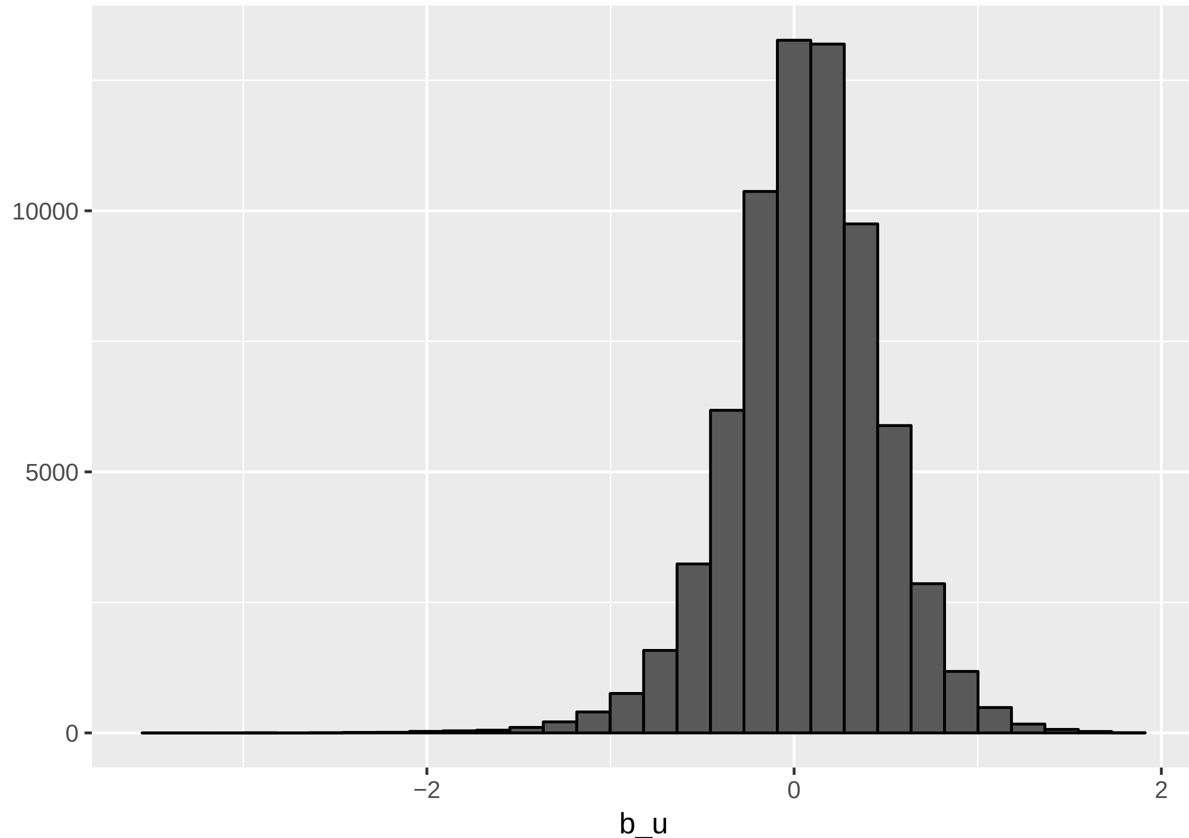
```
movie_avgs_norm <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs_norm %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("black"))
```



Penalty Term- User Effect

Similarly users can also affect the ratings either positivley(by giving higher ratings) or negatively (i.e.lower ratings).

```
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).

Naive Model

Creating a prediction system that solely utilizes the sample mean represents the initial simplest model. This impies that every prediction is the sample average. The resulting RMSE using this approach is quite high.

```
# Naive Model -- mean only
naive_rmse <- RMSE(validation_CM$rating,mu)
## Test results based on simple prediction
naive_rmse
```

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

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

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

```
## Save prediction in data frame
```

Movie Effect Model

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

```r
# 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	1.0612018
Movie Effect Model	0.9439087

```r
rmse_results
```

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

Movie and User Effect Model

A further improvement in the RMSE is achieved by adding the user effect.

```r
# 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	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

```r
rmse_results
```

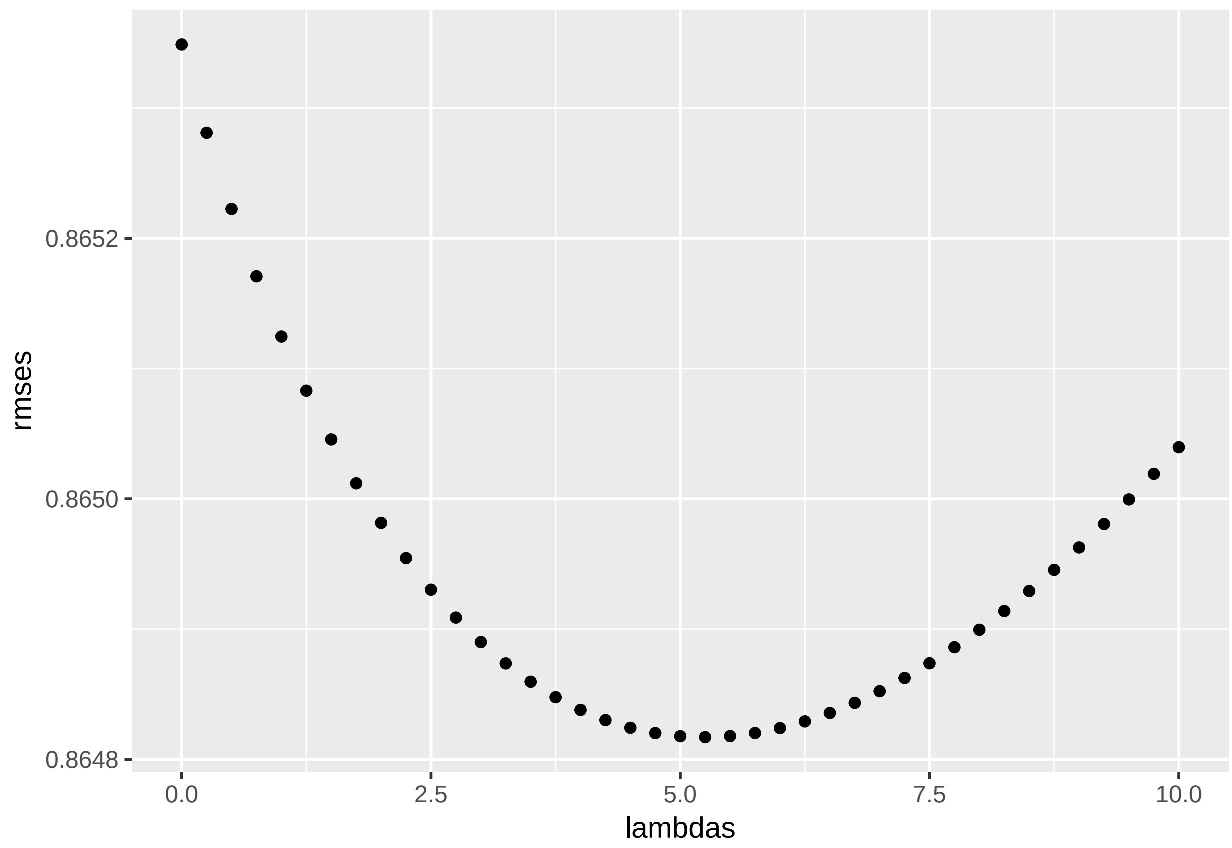
```
## # A tibble: 3 x 2
##   method                       RMSE
```

```
##   <chr>                      <dbl>
## 1 Using mean only            1.06
## 2 Movie Effect Model         0.944
## 3 Movie and User Effect Model 0.865
```

Regularized Movie and User Effect Model

This model implements the concept of regularization to account for the effect of low ratings' numbers for movies and users. The previous sections demonstrated that few movies were rated only once and that some users only rated few movies.Hence this can strongly influence the prediction. Regularization is a method used to reduce the effect of overfitting.

```
# 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)
```




```
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
```

```r
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	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170

```r
rmse_results
```

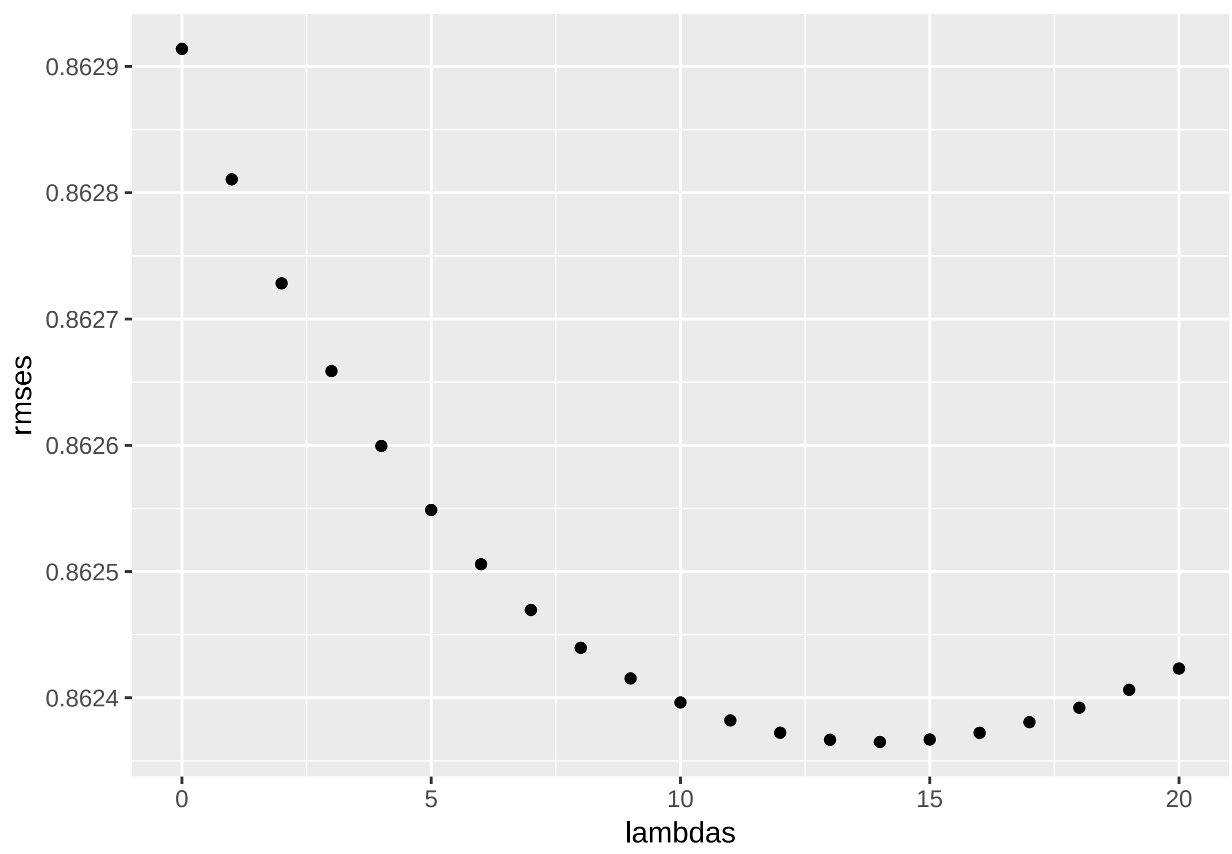
```
## # A tibble: 4 x 2
##   method                                   RMSE
##   <chr>                                   <dbl>
## 1 Using mean only                         1.06
## 2 Movie Effect Model                      0.944
## 3 Movie and User Effect Model             0.865
## 4 Regularized Movie and User Effect Model 0.865
```

Regularized With All Effects Model

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

```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') %>%
```

```r
    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)
```




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

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

```r
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()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170
Reg Movie, User, Year, and Genre Effect Model	0.8623650

3. Results

RMSE overview

The RMSE values for the used models are shown below:

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

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170
Reg Movie, User, Year, and Genre Effect Model	0.8623650

Rating Prediction using Model 4

Model 4 yielded the best rmse result and will hence be the chosing model for the final predictions.

```
lambda_3<-14
# Redo model 4 analysis

movie_reg_avgs_2 <- split_edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_3), 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_3), 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_3), 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_3), n_g = n())

## Adding all effects to the validation set & predicting the ratings
## Group by userId & movieID
## Compute each prediction's mean

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) %>%
  group_by(userId,movieId) %>% summarize(pred_2 = mean(pred))
```

The prediction results of the chosen model is continous and should therefore be modified as only specific ratings are allowed (i.e. 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5).Hence, the final prediction output will be rounded and ratings'values of zero and those above 5 will be replaced.

```
# Round predicted_ratings & confirm that they're between 0.5 & 5
```

```
predicted_ratings <- round(predicted_ratings*2)/2
predicted_ratings$pred_2[which(predicted_ratings$pred_2<1)] <- 0.5
predicted_ratings$pred_2[which(predicted_ratings$pred_2>5)] <- 5
```

4. Conclusion

The regularized model including the effect of movie, user, genre and year is characterized by the lowest RMSE value (0.8623650) and is hence the optimal model to use for the present project.

Further improvement to this model could be achieved by adding the effect of gender and age on the movies' genre preference combined with the user's profession effect on ratings (different professions do not rank things evenly). Additionally exploring different machine learning models (e.g. Neural Networks and Item Based Collaborative Filtering) could also improve the results further, however given the size of this dataset and the computational limitation/ RAM size of a regular laptop , this would be left for future work.