# EDX Capstone MovieLens Report

Wanjun Ling

7/12/2021

## Contents

# 1 EXECUTIVE SUMMARY

## 1.1 Introduction

The recommendation system, a subset of information filtering system which predicts users' "preference" or "rating" on an item, is widely used in the field of e-commerce business such as Amazon, JD, Netflix etc. It has become more and more popular in various data science applications since the first Netflix $1M prize challenge in 2006.

## 1.2 Data Set

In this project a 10M movielens data set generated by the GroupLens research lab will be used as source data.

## 1.3 Objective

During this project, the training of the algorithms and modeling are expected. The Root Mean Square Error (short for RMSE) will be used to evaluate the accuracy which is the closeness of multiple predictions to the true values in the validation set. The formula of computing RMSE is sqrt(mean((true_ratings - predicted_ratings)^2)). The objective of this Movielens project is to create a well performed rating recommendation system through machine learning training. The goal of this project is to train an algorithm with RMSE less than 0.86490.

## 1.4 Key Steps

The key steps executed in this project includes:

1. DATA PREPARATION: Download the MovieLens 10M data set through "http://files.grouplens.org/datasets/movielens/ml-10m.zip" and create training data set "edx" and final hold-out test data set "validation".
2. EXPLORATORY ANALYSIS: Collect data set statistics, analyze, visualize and determine potential effects or biases on ratings caused by features such as MovieID, UserID, Time, Genres etc.
3. MODELING: Build multiple models with consideration of rating effects from various combinations of features such as average rating, MovieID, UserID, Time, Genres etc and then evaluate RMSEs of these models.
4. CONCLUSION: Draw a conclusion on the best model as well as the limitation of these models based on modeling results. Then suggest further research directions.

# 2 METHODS

## 2.1 Data Preparation

Download MovieLens 10M data set through "http://files.grouplens.org/datasets/movielens/ml-10m.zip" and create training data set "edx" and final hold-out test data set "validation". After the required data sets are created, remove the temporary files and objects from the working directory.

## 2.2 Data Exploration

After the data is loaded, we start the work by examining the data structure, data classifications and corresponding statistics of data set for better understanding of source data.

First check whether there is any missing value in data sets "edx" and "validation".

```
#check any missing value in training data set "edx"
anyNA(edx)
```

```
## [1] FALSE
```

```
#check any missing value in final hold-out test data set "validation"
anyNA(validation)
```

```
## [1] FALSE
```

```
#list amount of observations and variables in training data set "edx"
dim(edx)
```

```
## [1] 9000055       6
```

```
#list amount of observations and variables in final hold-out test data set "validation"
dim(validation)
```

```
## [1] 999999      6
```

The results of above checking indicate that no invalid or missing value in both data set "edx" and "validation". We observe that there are total 9,000,055 observations with 6 features in "edx" data set and 999,999 observations with 6 features in "validation" data set. Now we need to know what features are they and statistics of them.

Now we further explore the data set "edx" to know more about it.

Here are first 6 observations of data set "edx" for us to have preliminary view.

```
##    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
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

Here is statistics of data set "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
##
##
##
```

Now we know data set "edx" has initially 6 features of "userId", "movieId", "rating", "timestamp", "title" and "genres". Meanwhile, We notice that there is year information in the feature "title" which may need to be split for further analysis. Also, we may need to convert "timestamp" to rated year for further checking the time effect on rating. Moveover we notice that there are several types in feature "genres" for one single observation. We may need to split "genres" for better handling the genres effect on rating in upcoming data processing part. Now we know the average rating is 3.512 and minimum rating is 0.5.

## 2.3 Data Pre-processing

Before we look into more detailed data analysis, we will conduct minor data pre-processing on both "edx" and "validation" data sets. The data pre-processing tasks include converting feature "timestamp" to year rated, extracting year released from feature "title", calculating ages between year rated and year released and adding these 3 new features to original data sets.

```r
# convert timestamp to year rated and add it to edx
edx <- edx %>% mutate(year_rated = year(as_datetime(timestamp)))

# double check any invalid value after convertion of timestamp
unique(edx$year_rated)
```

```
##  [1] 1996 1997 2006 2005 2001 2003 1999 2000 2002 1998 2007 2008 2004 2009 1995
```

```r
# extract the year released from title and add it to edx
edx <- edx %>% mutate(year_released = as.numeric(str_sub(title,-5,-2)))

# double check any invalid value of year released
unique(edx$year_released)
```

```
##  [1] 1992 1995 1994 1993 1991 1937 1970 1977 1990 1996 1971 1983 1989 1974 1967
## [16] 1984 1997 1985 2000 1982 2002 2003 2004 2005 1976 1972 1958 1942 1939 1941
## [31] 1950 1951 1979 1955 1962 1960 1980 1988 1975 1987 1981 1969 1998 1999 1986
## [46] 2001 1952 1959 1946 1940 1954 1949 1973 1948 1933 1931 1963 1922 1943 1944
## [61] 1957 1953 1965 1978 1964 1968 1966 1961 1956 1935 1938 2006 2007 1932 2008
## [76] 1934 1945 1947 1927 1925 1930 1936 1929 1923 1928 1921 1926 1915 1924 1916
## [91] 1917 1918 1920 1919
```

```
# calculate the movie age when movie was rated and add it to edx
edx <- edx %>% mutate(ages = as.numeric(year_rated)-as.numeric(year_released))

# double check any invalid value of ages
unique(edx$ages)
```

```
##  [1]  4  1  2  3  5 59 26 20  7 14 11 10 12 16 31 38 21 13  8  9  6 17 24 15 25
## [26] 39 55 58 56 47 46 18 42 35 37 23 30 29 32  0 51 45 44 62 64 57 63 49 41 54
## [51] 43 19 70 72 36 34 40 81 28 52 60 22 50 53 27 33 66 67 61 69 68 48 65 74 75
## [76] 71 77 73 79 76 83 78 85 80 84 91 82 90 89 88 86 87 -1 93 92 -2
```

We can see the result of data pre-processing is good except that 2 odd values of ages (-1 and -2) are observed. We are not sure whether the movies were rated before they were final released or they were just some data errors. Anyway, we plan to further check how much portion of observations with these odd values.

```
sum(edx$ages == -1)/nrow(edx)
```

```
## [1] 1.911099e-05
```

```
sum(edx$ages == -2)/nrow(edx)
```

```
## [1] 3.333313e-07
```

Now we know the portion of observations with odd values is quite small so it would not have big impact on our modeling and predication. Hence we just leave them as they are.

Now we apply the same data pre-processing on data set "validation" too.

```
# do the same data pre-processing for validation set
validation <- validation %>% mutate(year_rated = year(as_datetime(timestamp)))
validation <- validation %>% mutate(year_released = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(ages = as.numeric(year_rated)-as.numeric(year_released))
```

## 2.4 Data Analysis

After data pre-processing, we start to further analyze each feature for the evaluation of possible effects on our goal "rating".

```
# Number of unique users, movies
edx %>% summarize(unique_users = n_distinct(userId), unique_movies = n_distinct(movieId))
```

```
##   unique_users unique_movies
## 1        69878         10677
```

From the result of above script, we can see there are 69878 unique users and 10677 unique movies. If we simply multiply unique_users 69878 by unique_movies 10677, we'll get over 746 million records while there are only around 9 million observations in edx data set.The ratings in edx data set only count around 1.2% of all possible ratings. This huge gap implies that each user does not rate all movies. If we think in terms of a large matrix, with user on the rows and movies on the columns, the matrix will be a super sparse one. The sparsity of the matrix will be our challenge to our prediction. We will verify it in following analysis part.

First, let us take a look of the feature "rating" and it's distribution since it is our goal.

List all unique values of feature "rating".

```
# list all rating values
unique(edx$rating)
```

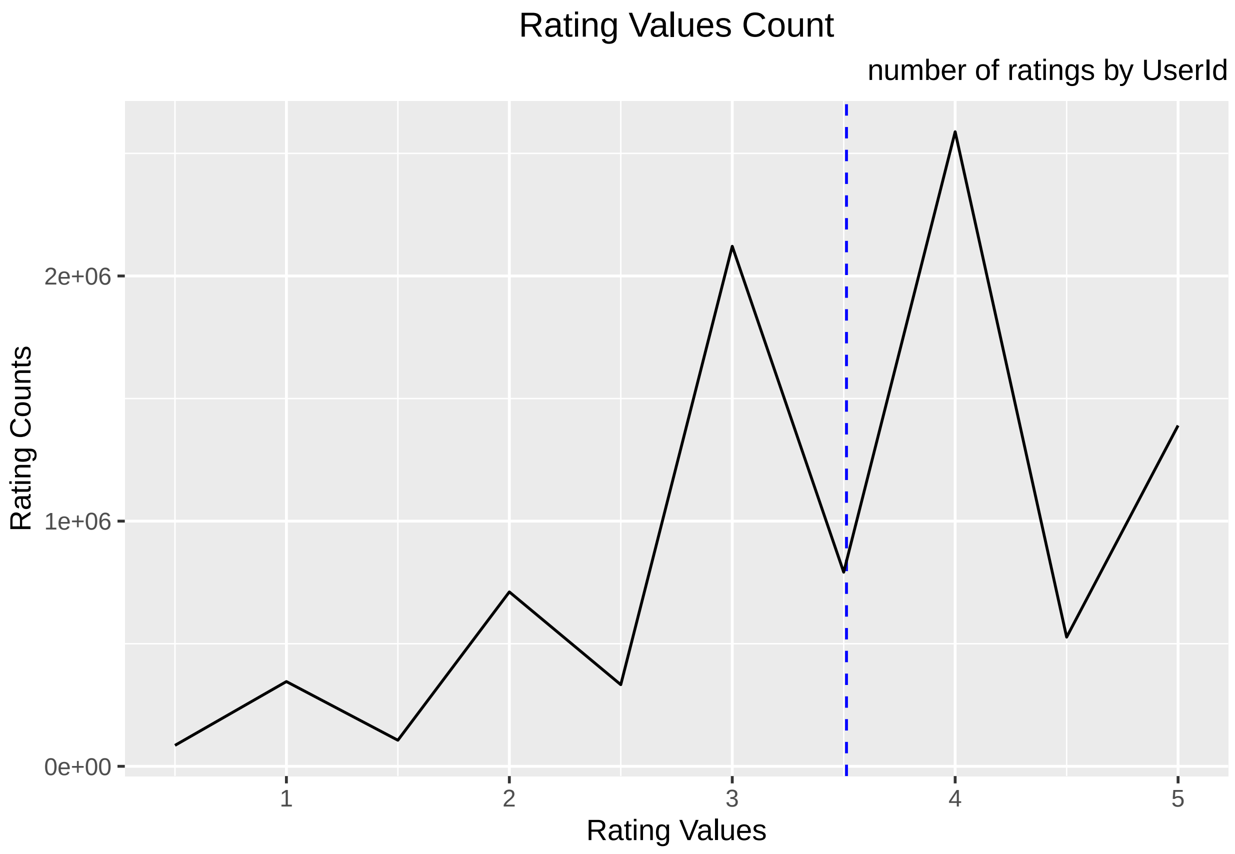
```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

We can see that there are total 10 unique ratings ranging from 0.5 to 5 with interval of 0.5. There is no rating with value 0. Let us further check the rating distributions of these 10 values.

Check which rating values are the top 10 ratings.

```
## # A tibble: 10 x 2
##    rating rating_sum_number
##     <dbl>             <int>
##  1    4             2588430
##  2    3             2121240
##  3    5             1390114
##  4    3.5            791624
##  5    2              711422
##  6    4.5            526736
##  7    1              345679
##  8    2.5            333010
##  9    1.5            106426
## 10    0.5             85374
```

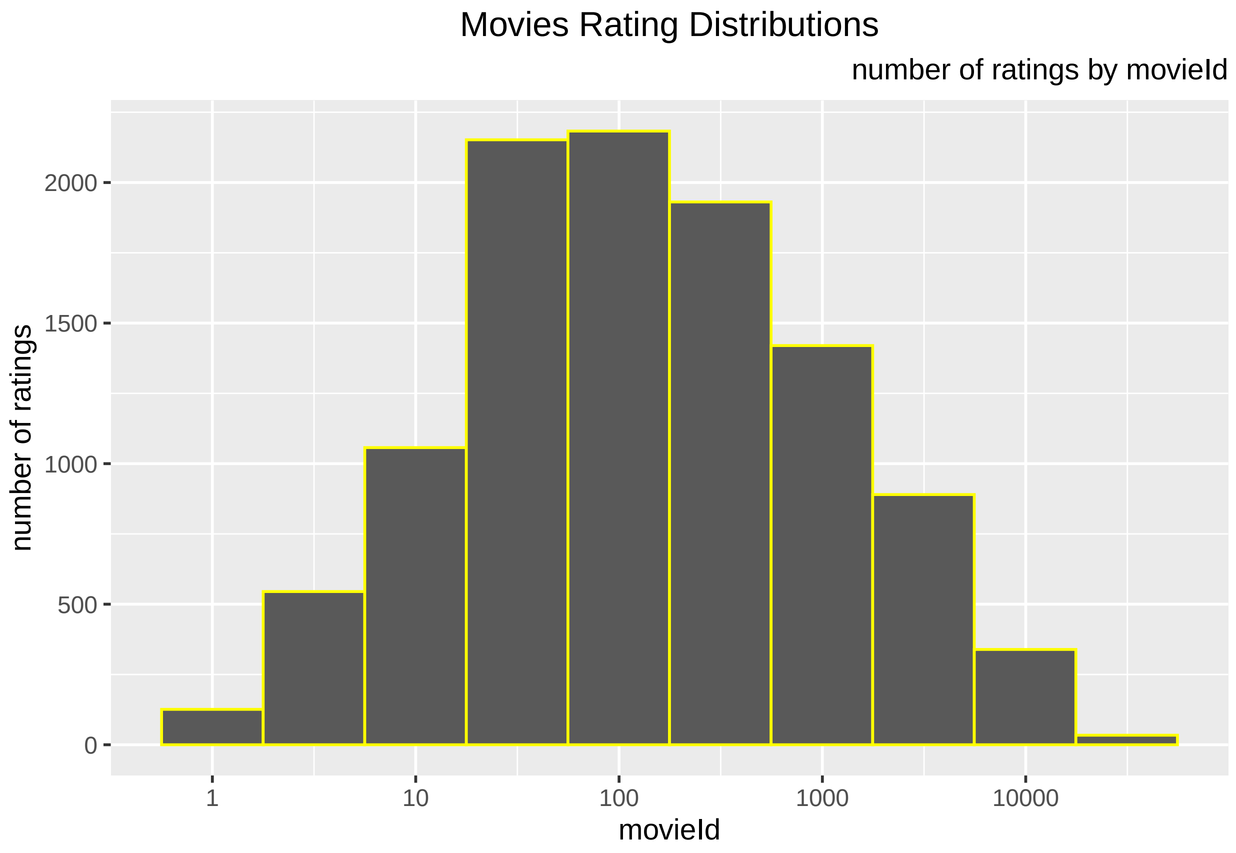
Now plot rating counts for each rating value



From the graph above, it is observed that the most popular rating is 4, followed by rating values 3, 5, 3.5, 2 etc. In general, half star ratings are much less common than whole star ratings. We also show the average rating in the histogram by a blue dashed line. The graph tells that most users tend to give whole star rating rather than half star one.

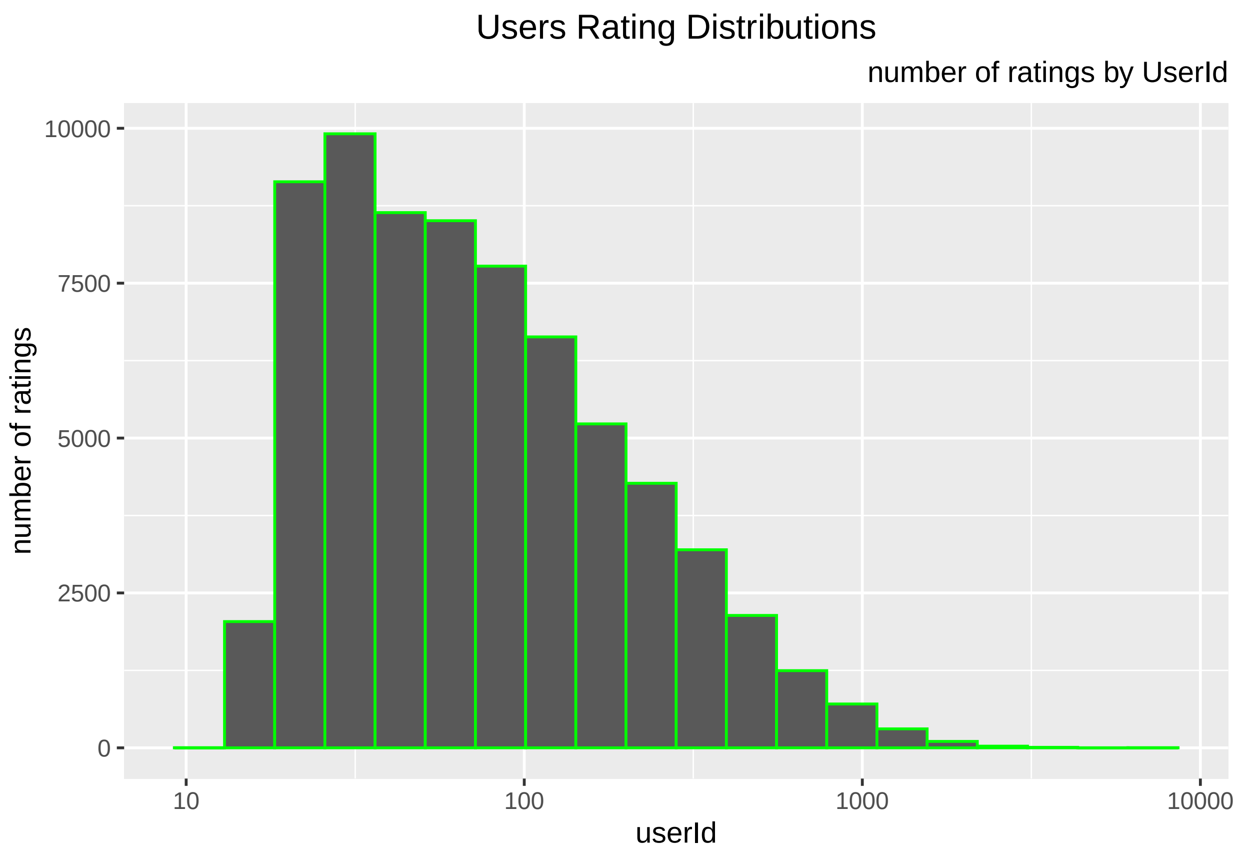
After analyzing feature "rating" itself, we need further check other features in edx data set to identify whether any effect or bias which may exist on movies' rating preference. Now let us explore whether movie bias exists.

Plot histogram on number of ratings by "movieId".



The visualization of the number of ratings by "movieId" indicates that some movies got much more ratings than others. Some movies received over 20,000 ratings while some only got around 150 ratings. Hence movies bias actually exists. We need take movies bias into consideration in training machine learning algorithms.

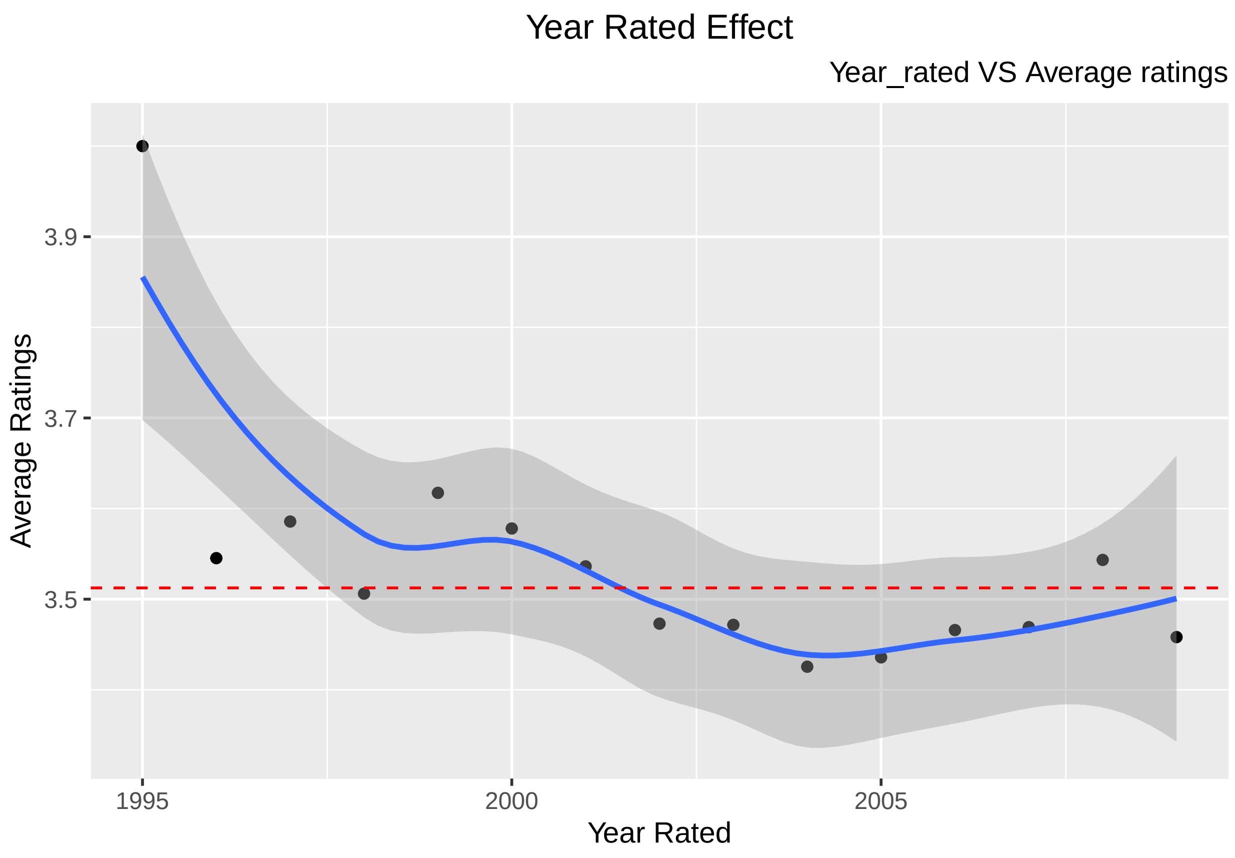
Then we explore whether user bias exists.



The visualization of the number of ratings by "userId" shows that some users are much more active or preferring than others at rating movies. Again users bias actually exists. We need take users bias too into consideration in training machine learning algorithms.

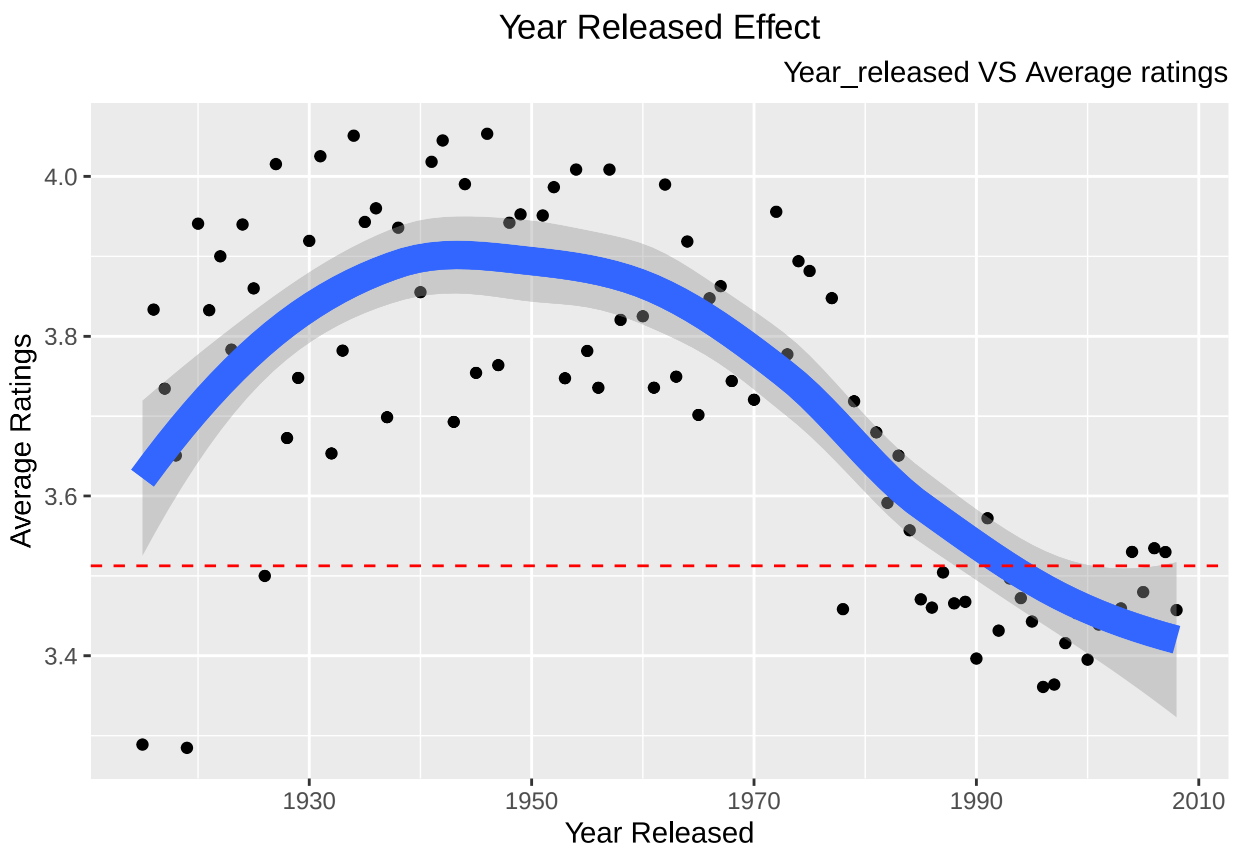
Next we need explore whether time bias exists.

We explore effects of year_rated on rating.



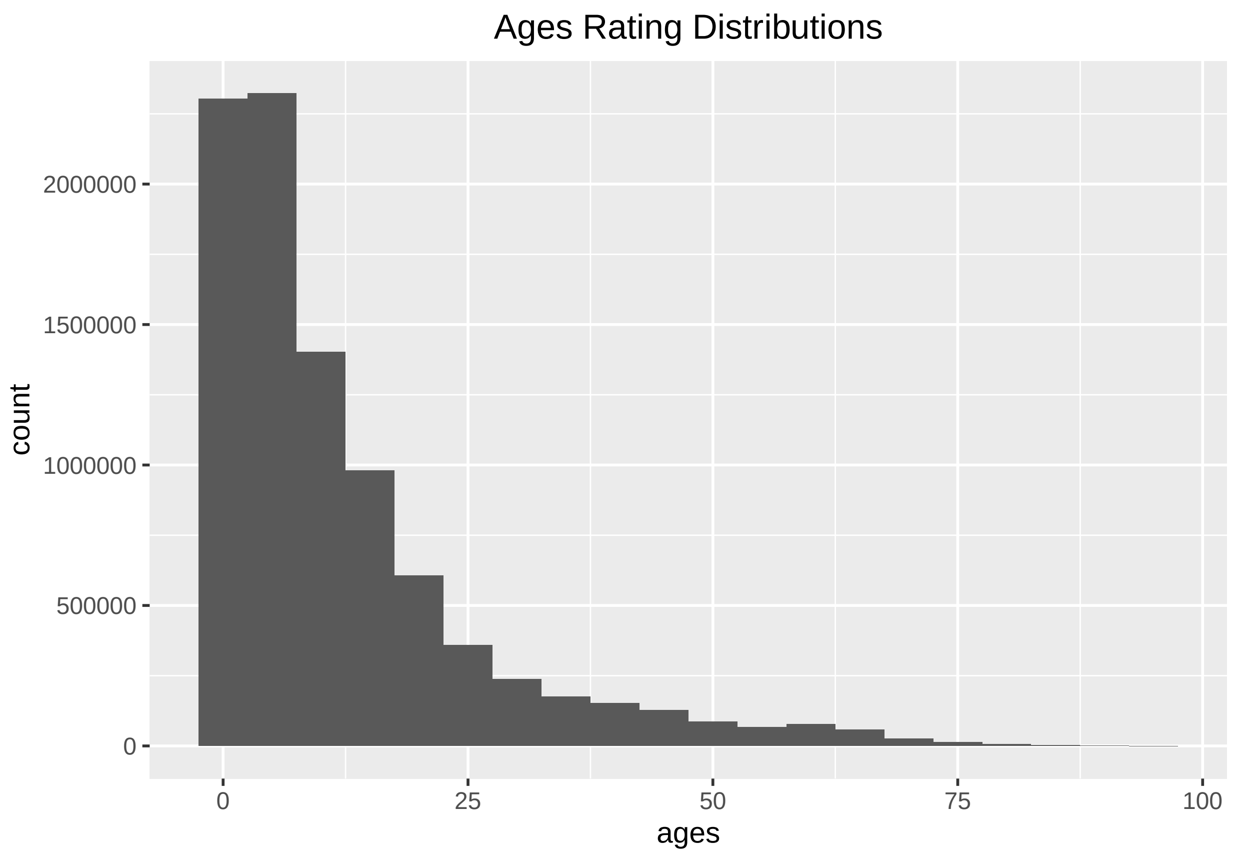
This plot shows that people who rated movies between year 1995 and 2000 usually gave higher rating compared to the ones who rated after year 2000.

Now we explore effects of year_released on rating.

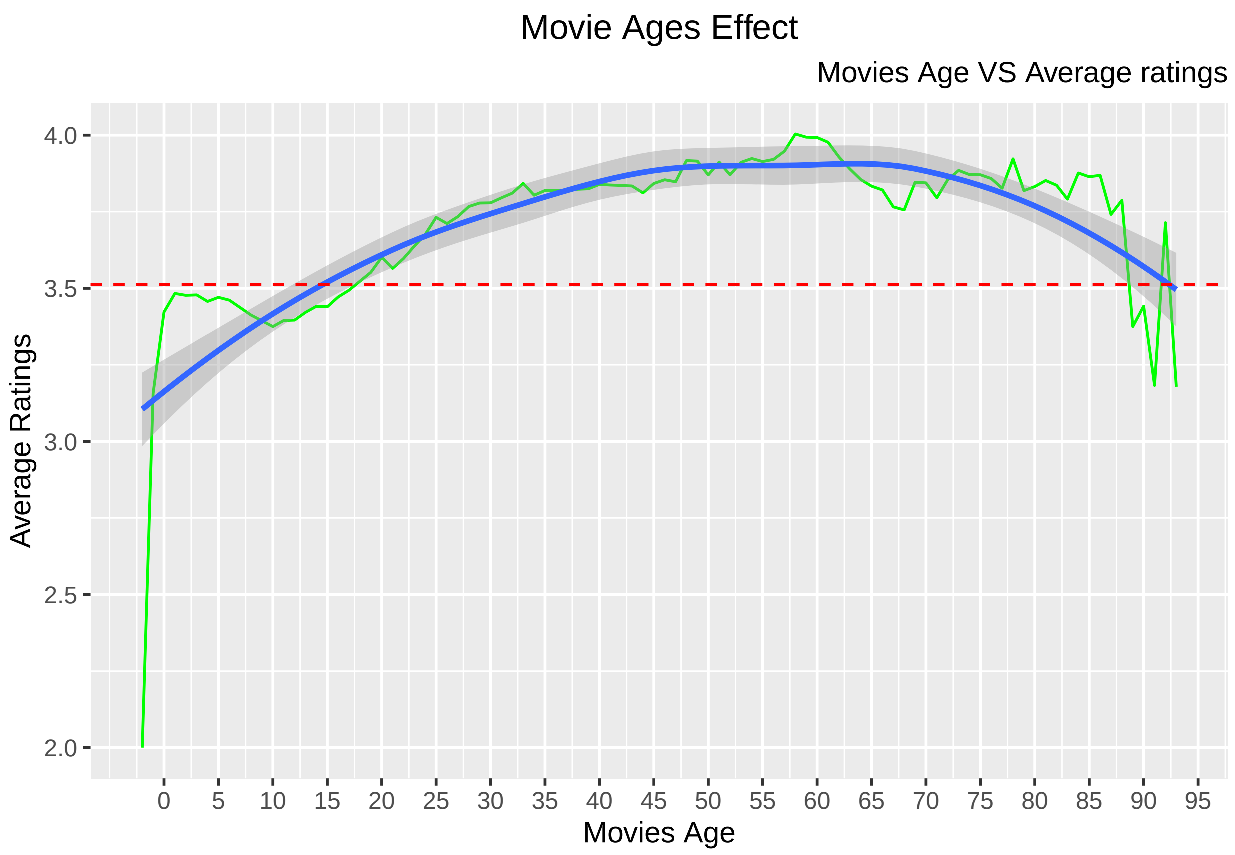


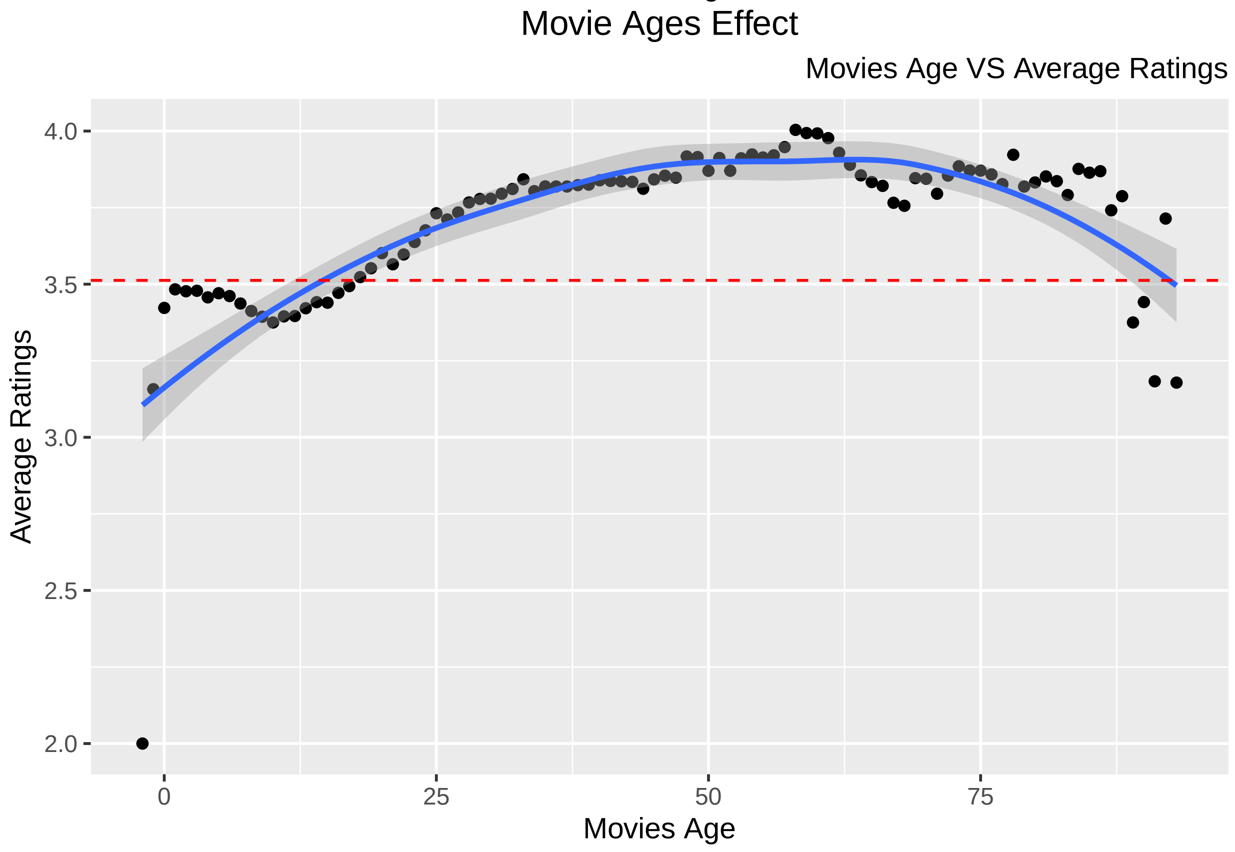
This plot shows that older movies tended to have higher ratings. Movies released before year of 1980 generally were given higher ratings much above the average rating indicated by red dashed line.

We further explore the movies age effect on rating.



This Ages Rating Distributions plot shows that most of ratings were given to movies within ages of 15 years. Meanwhile only very few ratings were given for movies with age of over 50 years.





The above plots show that movies with age between 20 to 80 usually were given higher ratings.

Let us summarize our findings based on above time effect analysis. Although new movies with ages less than 15 years usually got the most ratings, older movies were more likely to be given higher ratings which were much above average rating. This could due to new movies got more viewers compared with old movies while old good movies got more reputations with the time flies. This implicit finding suggests that time may have an effect on the average ratings. While the smooth trend of the average ratings versus year dated and ages of movies shows that time effect on the average ratings would be weak. Anyway, we will consider to take time effect into consideration in modeling part to verify time effect.

Here we complete the time effect analysis. Now we will move to analysis of genres effect.

Let us take a look of Top 10 title and genres which contributes most of ratings.

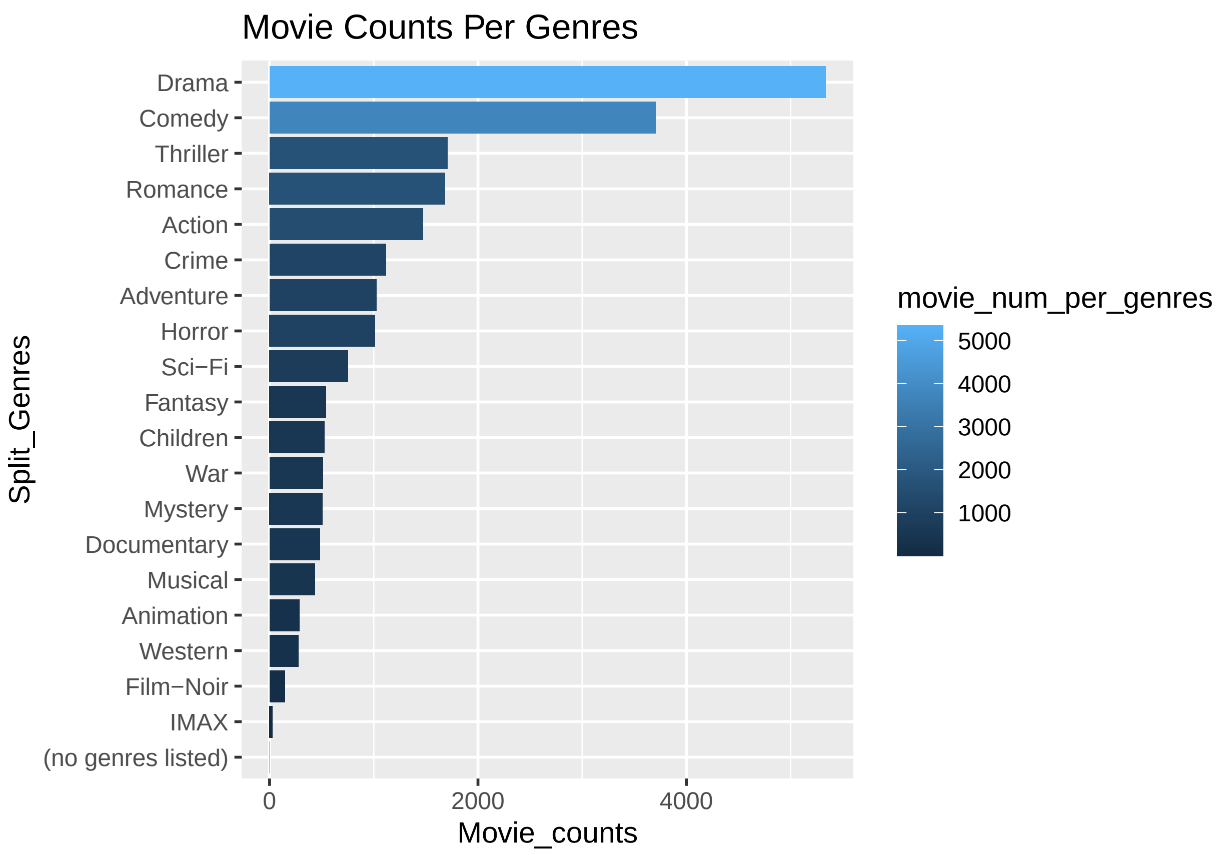
```
## # A tibble: 10,677 x 3
## # Groups:   title [10,676]
##    title                                                 genres                    count
##    <chr>                                                 <chr>                     <int>
##  1 Pulp Fiction (1994)                                   Comedy|Crime|Drama        31362
##  2 Forrest Gump (1994)                                   Comedy|Drama|Romance|War  31079
##  3 Silence of the Lambs, The (1991)                      Crime|Horror|Thriller     30382
##  4 Jurassic Park (1993)                                  Action|Adventure|Sci-Fi~  29360
##  5 Shawshank Redemption, The (1994)                      Drama                     28015
##  6 Braveheart (1995)                                     Action|Drama|War          26212
##  7 Fugitive, The (1993)                                  Thriller                  25998
##  8 Terminator 2: Judgment Day (1991)                     Action|Sci-Fi             25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. St~ Action|Adventure|Sci-Fi  25672
## 10 Apollo 13 (1995)                                      Adventure|Drama           24284
## # ... with 10,667 more rows
```

We observe that one movie may have more than one genres. Are movies with specific genres rated much more than others? To answer this question, we need split the genres before proceeding the analysis.

```r
# Explore whether "genres" bias exists
# split genres in "edx" and create a new data set "edx_split_genres" for analysis
edx_split_genres <- edx %>% separate_rows(genres, sep ="\\|")
head(edx_split_genres)
```

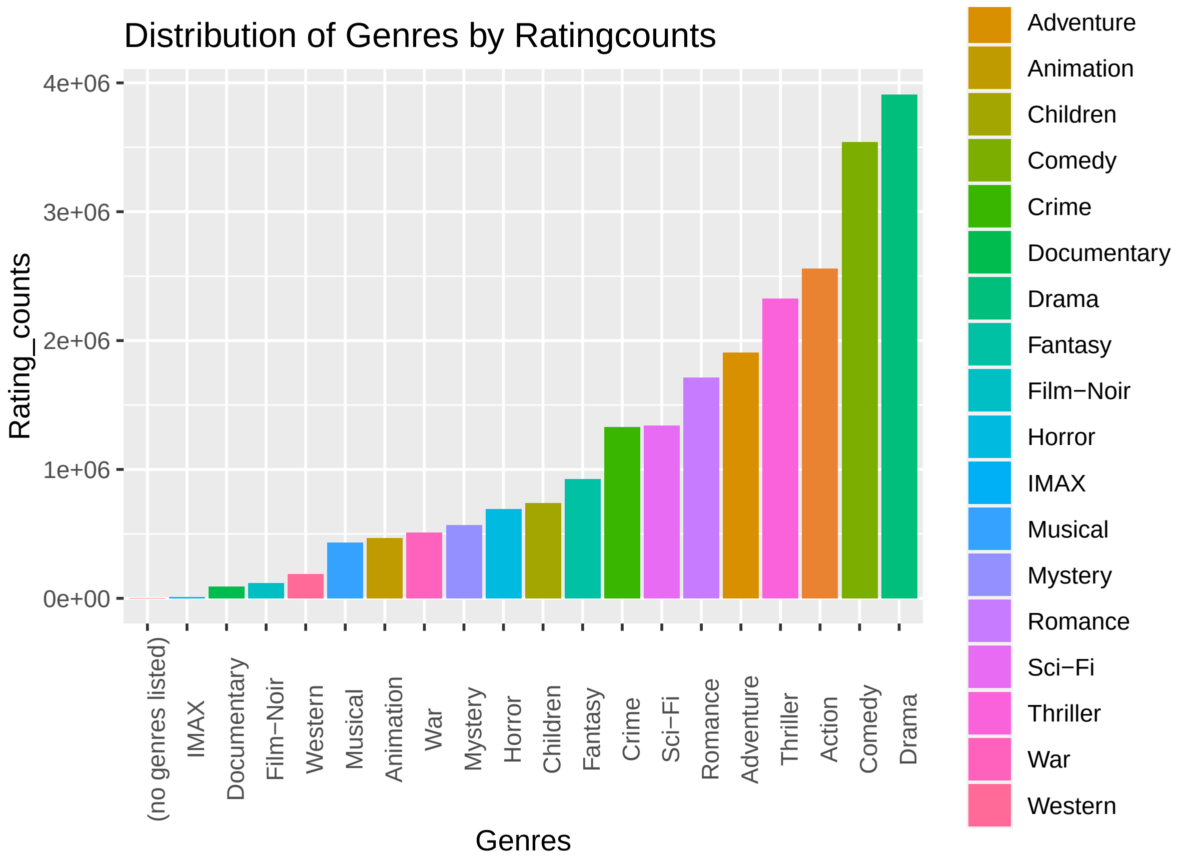
```
## # A tibble: 6 x 9
##   userId movieId rating timestamp title    genres year_rated year_released  ages
##    <int>   <dbl>  <dbl>     <int> <chr>    <chr>       <dbl>         <dbl> <dbl>
## 1      1     122      5 838985046 Boomera~ Comedy       1996          1992     4
## 2      1     122      5 838985046 Boomera~ Roman~       1996          1992     4
## 3      1     185      5 838983525 Net, Th~ Action       1996          1995     1
## 4      1     185      5 838983525 Net, Th~ Crime        1996          1995     1
## 5      1     185      5 838983525 Net, Th~ Thril~       1996          1995     1
## 6      1     292      5 838983421 Outbrea~ Action       1996          1995     1
```

Let us check the numbers of movies of each genres first.



The graph above shows clearly that most of movies were with genre of "Drama" while only few were with genre of "IMAX". The top five genres are Drama, Comedy, Thriller, Romance and Action.

Then we plot rating counts for each genre.



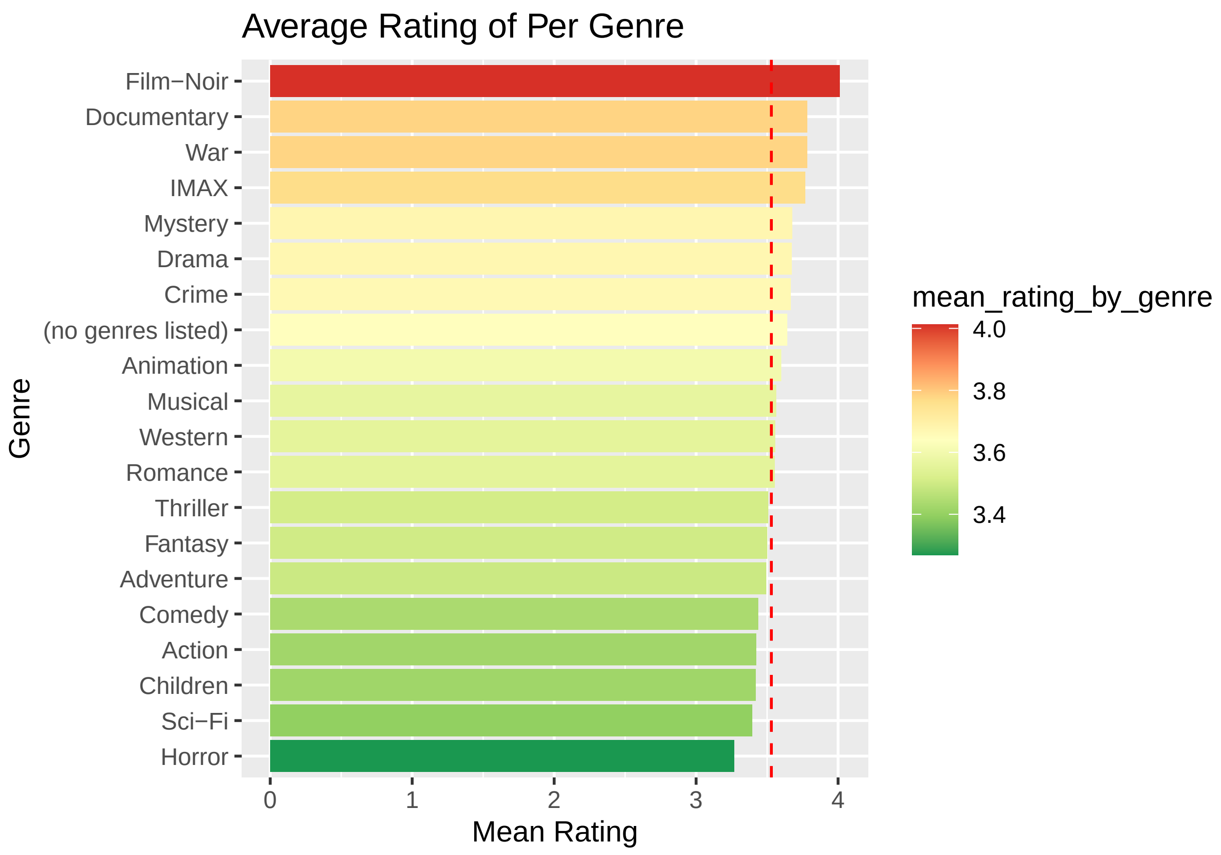
This graph shows the number of ratings in each genres. The movies with genres “Drama” were rated the most while the ones with genres “IMAX” were the least rated. However, we are not able to draw a conclusion that people prefer to rate the Drama movies over other types because this could due to more movies in “Drama” genre as indicated in previous graph.

So in order to identify real genres effect, we need further analyze average rating for each genre.

Plot Mean Rating per Genre compared with general average rating.

```
## # A tibble: 20 x 2
##    genres             mean_rating_by_genre
##    <chr>                             <dbl>
##  1 Film-Noir                          4.01
##  2 Documentary                        3.78
##  3 War                                3.78
##  4 IMAX                               3.77
##  5 Mystery                            3.68
##  6 Drama                              3.67
##  7 Crime                              3.67
##  8 (no genres listed)                 3.64
##  9 Animation                          3.60
## 10 Musical                            3.56
## 11 Western                            3.56
## 12 Romance                            3.55
## 13 Thriller                           3.51
## 14 Fantasy                            3.50
## 15 Adventure                          3.49
## 16 Comedy                             3.44
## 17 Action                          3.42
## 18 Children                        3.42
## 19 Sci-Fi                          3.40
## 20 Horror                          3.27
```

```
## # A tibble: 1 x 1
##   `mean(rating)`
##            <dbl>
## 1           3.53
```



The graph shows that some genres such as "Film_Noir" and "Documentary" have higher ratings than the general average ratings while others such as "Comedy" and "Action" have lower ratings than the general average ratings. With consideration of rating counts per genres, the genre effect seems to exist too. Hence the genre effect will also be considered in modeling part.

So far, we have analyzed the potential effects on rating by movie, user, age and genres. Now we will move to modeling stage with the analysis results.

## 3 MODELING AND RESULTS

To measure the model accuracy, first we need define RMSE short for residual mean squared error.

```
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
  }
```

We start by building the simplest possible recommendation system: we predict the same rating for all movies regardless of user. We know that the estimate that minimizes the RMSE is the least squares estimate of mu_hat, the average of all ratings.

## 3.1 Model 1: Only by average rating

```
#the simplest possible model: we predict the same rating for all movies regardless of user.
mu_hat <- mean(edx$rating)
mu_hat
```

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

```
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse
```

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

Now we build a data frame rmse_results to store models results

```
rmse_results <- data_frame(Model = "Only by average rating", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

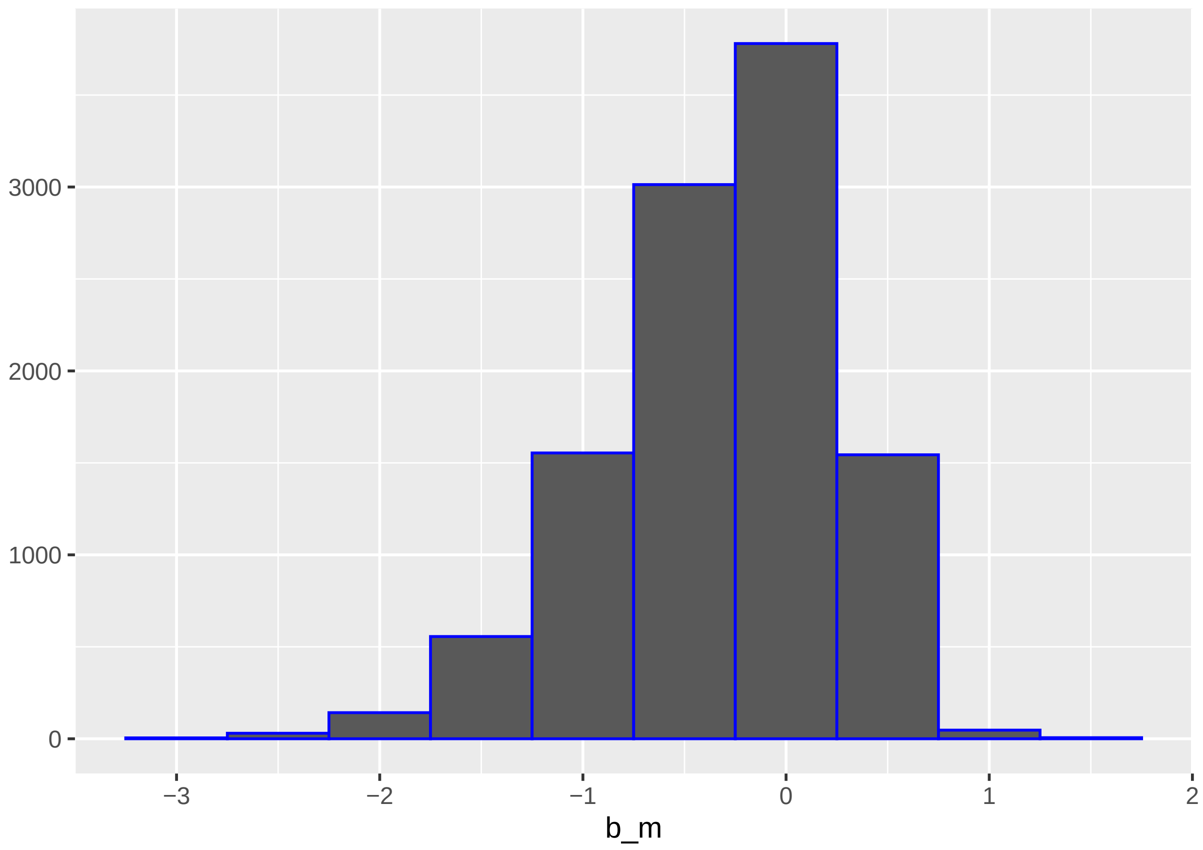
| Model | RMSE |
|---|---|
| Only by average rating | 1.061202 |

Here we get the RMSE 1.0612 of our first model "Only by average rating". We can see RMSE of this model is quite far away from our target 0.8649. Now we add movieId to build the second model.

## 3.2 Model 2: Modeling with movieId effect

Since we are already aware that movieId feature could obviously affect the ratings of a movie in the section of data analysis, we add the bias of movie (b_m) to the model. Now we plot the distribution of the bias and calculate the RMSE of this model.

```
#calculating and plot b_m distribution
movie_effects <- edx %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu_hat))
movie_effects %>% qplot(b_m, geom ="histogram", bins = 10, data = ., color = I("blue"))
```



```r
#calculating predicted ratings
predicted_ratings_m <- mu_hat + validation %>%
  left_join(movie_effects, by='movieId') %>%
  pull(b_m)

#calculating model_m_rmse with consideration of movie effects
model_m_rmse <- RMSE(validation$rating,predicted_ratings_m)
model_m_rmse
```

```
## [1] 0.9439087
```

```r
#insert new result into DF rmse_result
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Movie Effect Model",
                                     RMSE = model_m_rmse))
rmse_results %>% knitr::kable()
```

| Model | RMSE |
|---|---:|
| Only by average rating | 1.0612018 |
| Movie Effect Model | 0.9439087 |

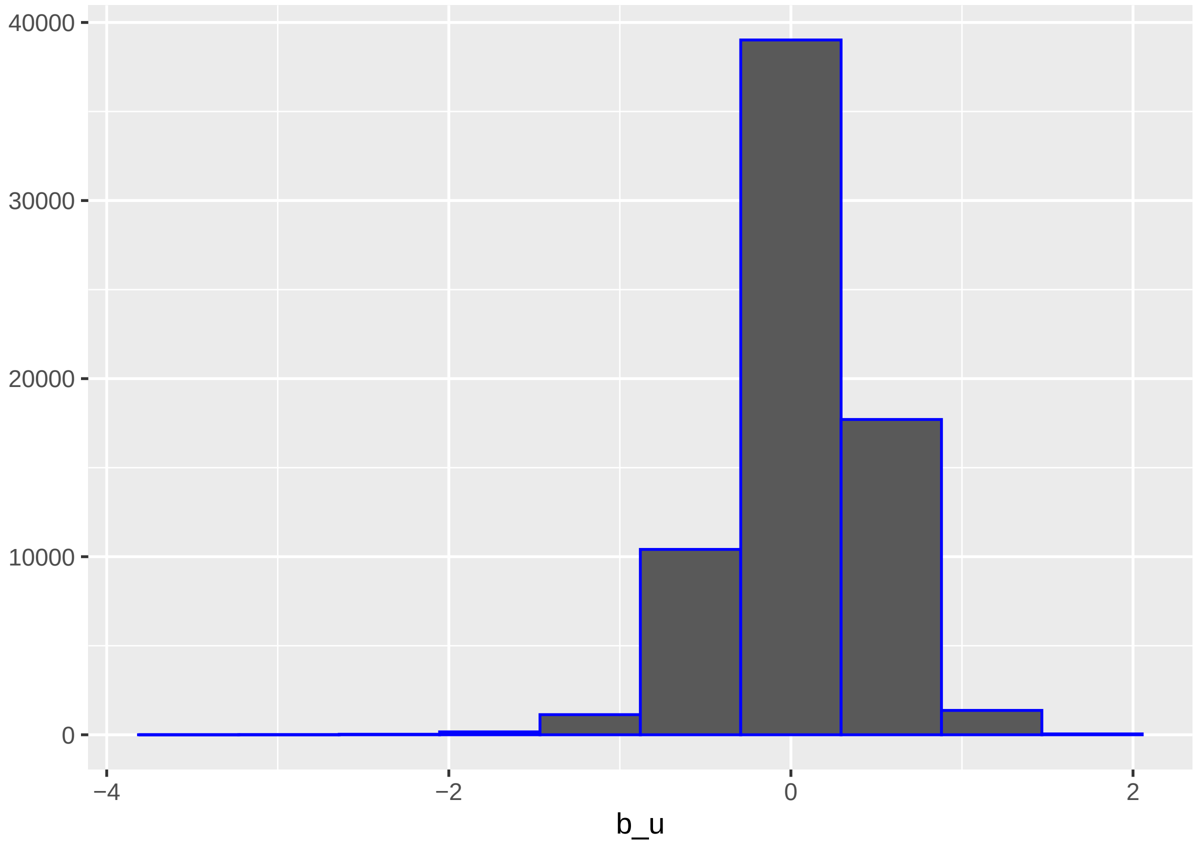
Now the new RMSE by adding movieId drops from 1.0612 to 0.9439. The movie effect model improves predication accuracy around 11% compared with "Only by average rating" model. The result confirms movie effect.

We will build the third model by adding combination of movieId and userId since we have already known that user bias also exists.

## 3.3 Model 3: Modeling with movieId + userId effects

Similar to the movie effect, we now further add the bias of user effect (b_u) to the movie effect model.

```
#calculating and plot b_u distribution
user_effects <- edx %>%
  left_join(movie_effects, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_m))
user_effects %>% qplot(b_u, geom ="histogram", bins = 10, data = ., color = I("blue"))
```



```
#calculating predicted ratings
predicted_ratings_m_u <- validation %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  mutate(prediction = mu_hat + b_m + b_u) %>%
  pull(prediction)

#calculating model_m_rmse with consideration of movie + user effects
model_m_u_rmse <- RMSE(validation$rating,predicted_ratings_m_u)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Movie + User Effects Model",
                                     RMSE = model_m_u_rmse))
rmse_results %>% knitr::kable()
```

| Model | RMSE |
|---|---|
| Only by average rating | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie + User Effects Model | 0.8653488 |

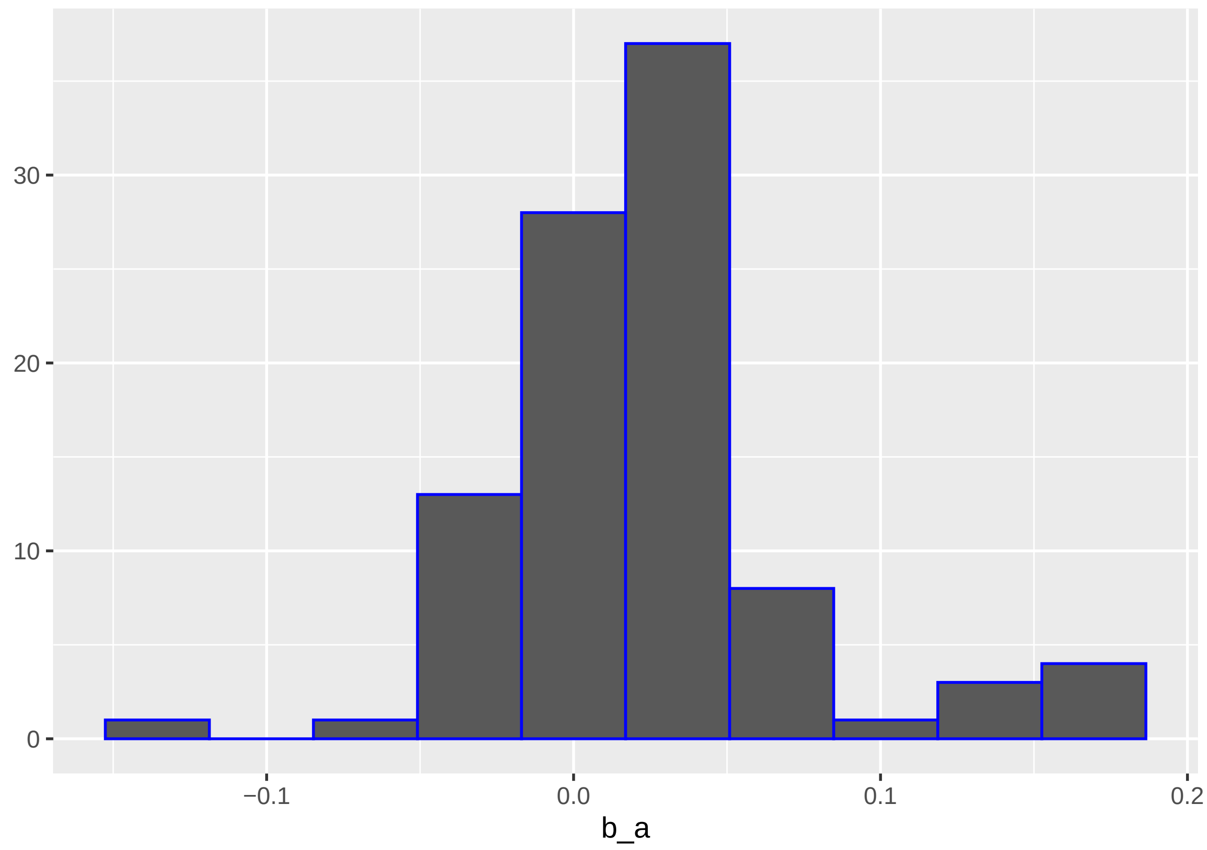
Again, the new RMSE by adding movieId and userId together drops from 0.9439 to 0.8653. The movie and user effect model further improves predication accuracy around 18.5% compared with "Only by average rating" model.

In data analysis part, we estimate the time effect would be weak. What will new RMSE be by adding movie age on top of movie and user effect?

## 3.4 Model 4: Modeling with movieId + userId + movieAges effects

We now further add the bias of movie age effect (b_a) to the movie + user effect model.

```
#calculating and plot b_a distribution
ages_effects <- edx %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  group_by(ages) %>%
  summarize(b_a = mean(rating - mu_hat - b_m - b_u))
ages_effects %>% qplot(b_a, geom ="histogram", bins = 10, data = ., color = I("blue"))
```



```r
#calculating predicted ratings
predicted_ratings_m_u_a <- validation %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  left_join(ages_effects, by='ages') %>%
  mutate(prediction = mu_hat + b_m + b_u + b_a) %>%
  pull(prediction)

#calculating model_m_rmse with consideration of movie + user + age effects
model_m_u_a_rmse <- RMSE(validation$rating,predicted_ratings_m_u_a)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Movie + User + Age Effects Model",
                                     RMSE = model_m_u_a_rmse))
rmse_results %>% knitr::kable()
```

| Model | RMSE |
|---|---|
| Only by average rating | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie + User Effects Model | 0.8653488 |
| Movie + User + Age Effects Model | 0.8649038 |

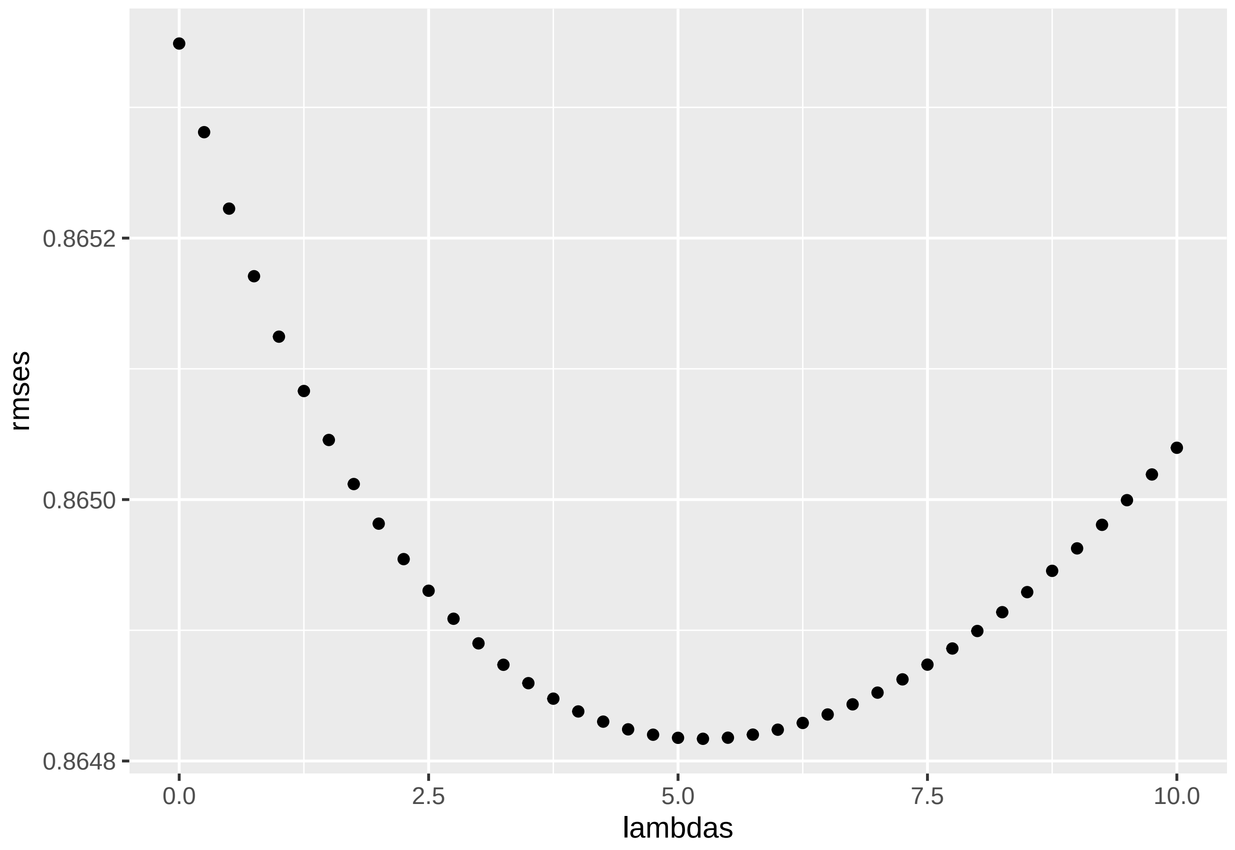
As expected, the difference of RMSE is only 0.00045 with additional consideration of age effect which confirms our judgment of weak effect by movie's age.

Now we will build a new model by regularization of movie and user effects since age effect is weak.

## 3.5 Model 5: Regularization of movieId + userId effects

The general idea behind regularization is to constrain the total variability of the effect size. As lambda is a tuning parameter, we can use cross-validation to choose optimized one for model building and performance evaluation.

```r
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
mu_reg <- mean(edx$rating)

#regulation movie effect
b_m_reg <- edx %>%
group_by(movieId) %>%
summarize(b_m_reg = sum(rating - mu_reg)/(n()+l))

#regulation user effect
b_u_reg <- edx %>%
left_join(b_m_reg, by="movieId") %>%
group_by(userId) %>%
summarize(b_u_reg = sum(rating - b_m_reg - mu_reg)/(n()+l))

#calculating predicted ratings
predicted_ratings_b_m_u <-
validation %>%
left_join(b_m_reg, by = "movieId") %>%
left_join(b_u_reg, by = "userId") %>%
mutate(prediction = mu_reg + b_m_reg + b_u_reg) %>%
.$prediction
return(RMSE(validation$rating,predicted_ratings_b_m_u))
})

qplot(lambdas, rmses)
```



```
#For the full model, the optimal lambda is given as
lambda <- lambdas[which.min(rmses)]
lambda
```

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

```
#calculating regularization model RMSE: model_m_u_reg_rmse
model_m_u_reg_rmse <- min(rmses)
model_m_u_reg_rmse
```

```
## [1] 0.864817
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Movie + User Regularization Model",
                                     RMSE = model_m_u_reg_rmse))
rmse_results %>% knitr::kable()
```

| Model | RMSE |
|---|---:|
| Only by average rating | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie + User Effects Model | 0.8653488 |
| Movie + User + Age Effects Model | 0.8649038 |
| Movie + User Regularization Model | 0.8648170 |

Now we finally achieve the goal of RMSE which is less than 0.86490 by model of regularization of movie and user effects. The new RMSE is 0.8648.

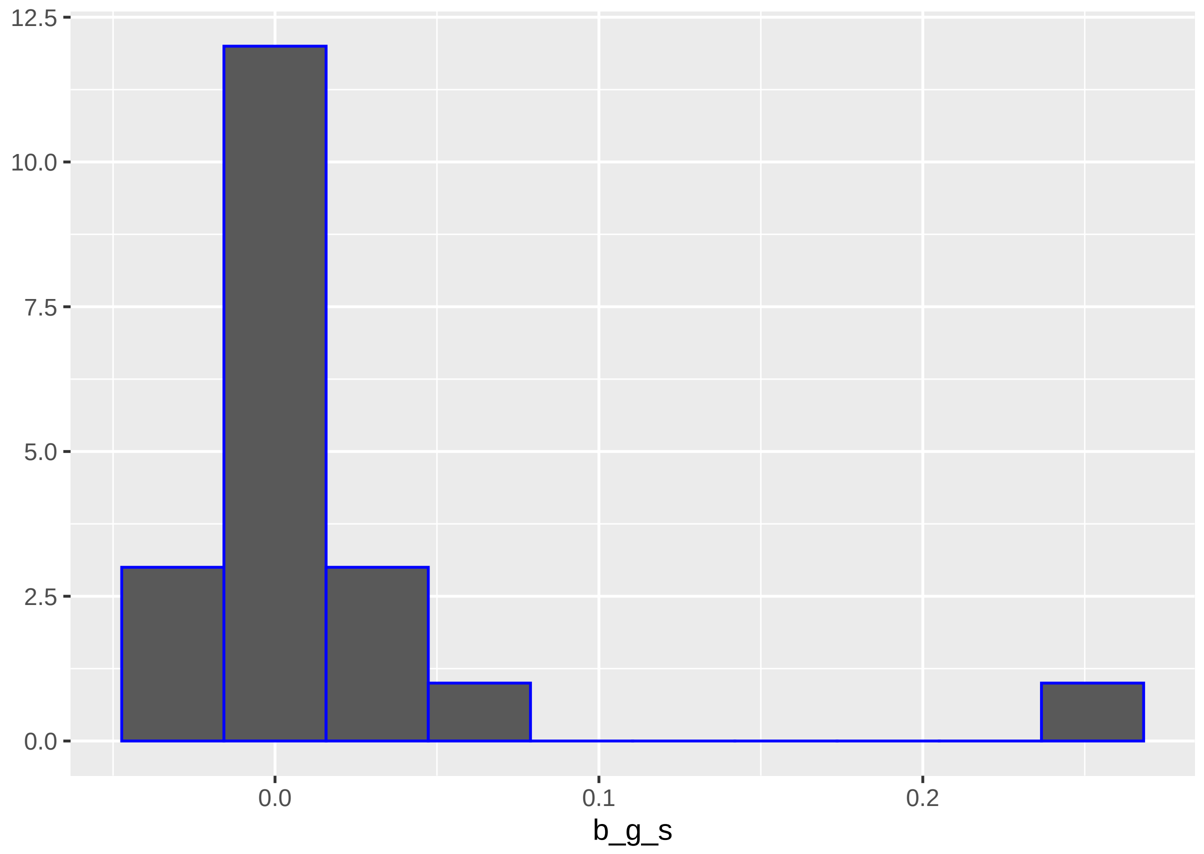
Sa far, we have not tried to add genres effect in our model yet. Since genres also has effect on rating, we will build the sixth model to combine genres effect.

## 3.6 Model 6: Modeling with movieId + userId + movieAges + genres effects

We now further add the bias of genres effect (b_g) to the movie + user + ages effect model.

#“‘{r fig = TRUE, message=FALSE, warning=FALSE}

```
# Create data set validation_split_genres with split genres from data set validation

validation_split_genres <- validation %>% separate_rows(genres, sep ="\\|")

# Chunk options
knitr::opts_chunk$set(
 fig.width = 6,
 fig.asp = 0.8,
 out.width = "80%"
)

mu_hat_s <- mean(edx_split_genres$rating)

movie_effects_s <- edx_split_genres %>%
  group_by(movieId) %>%
  summarize(b_m_s = mean(rating - mu_hat_s))

user_effects_s <- edx_split_genres %>%
  left_join(movie_effects_s, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u_s = mean(rating - mu_hat_s - b_m_s))

ages_effects_s <- edx_split_genres %>%
  left_join(movie_effects_s, by='movieId') %>%
  left_join(user_effects_s, by='userId') %>%
  group_by(ages) %>%
  summarize(b_a_s = mean(rating - mu_hat_s - b_m_s - b_u_s))

#calculating and plot b_g distribution
genres_effects_s <- edx_split_genres %>%
  left_join(movie_effects_s, by='movieId') %>%
  left_join(user_effects_s, by='userId') %>%
  left_join(ages_effects_s, by='ages') %>%
  group_by(genres) %>%
  summarize(b_g_s = mean(rating - mu_hat_s - b_m_s - b_u_s - b_a_s))
genres_effects_s %>% qplot(b_g_s, geom ="histogram", bins = 10, data = ., color = I("blue"))
```



```
#calculating predicted ratings
predicted_ratings_m_u_a_g_s <- validation_split_genres %>%
  left_join(movie_effects_s, by='movieId') %>%
  left_join(user_effects_s, by='userId') %>%
  left_join(ages_effects_s, by='ages') %>%
  left_join(genres_effects_s, by='genres') %>%
  mutate(prediction = mu_hat_s + b_m_s + b_u_s + b_a_s + b_g_s) %>%
  pull(prediction)

#calculating model_m_rmse with consideration of movie + user + age effects
model_m_u_a_g_s_rmse <- RMSE(validation_split_genres$rating,predicted_ratings_m_u_a_g_s)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Movie + User + Age + Genres Effects Model",
                                     RMSE = model_m_u_a_g_s_rmse))
rmse_results %>% knitr::kable()
```

| Model | RMSE |
|---|---|
| Only by average rating | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie + User Effects Model | 0.8653488 |
| Movie + User + Age Effects Model | 0.8649038 |
| Movie + User Regularization Model | 0.8648170 |
| Movie + User + Age + Genres Effects Model | 0.8627469 |

Now we further improve the predication accuracy with RMSE 0.8627469 by modeling with movieId + userId + movieAges + genres effects. Just like age effect, the genres effect is weak too. So far, compared with the first model “Only by average rating”, RMSE of this model drops around 19%.

Comparing model 3 (Movie + User Effects Model) and model 5(Movie + User Regularization Model), we find that there is not much improvement with regularization. The RMSE by regularization only drops 0.0005 which is quite limited. We suspect it may be caused by sparsity issue mentioned in the section of 2.4 Data Analysis.

# 4 CONCLUSION

## 4.1 Conclusion

This MovieLens data science project enhance students’ understanding of all 8 courses greatly. To achieve the project goal of generating RMSE less than 0.86490, total six models and machine learning algorithms have been successfully developed and trained with results as follows.

| Model | RMSE |
|---|---:|
| Only by average rating | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie + User Effects Model | 0.8653488 |
| Movie + User + Age Effects Model | 0.8649038 |
| Movie + User Regularization Model | 0.8648170 |
| Movie + User + Age + Genres Effects Model | 0.8627469 |

When looking at the first model “Only by average rating”, we are not surprising to the bad prediction with RMSE 1.06120 as our assumption of same rating for all movies does not reflect the real world situation. With consideration of movie effects on rating, we improve our accuracy around 11% with RMSE 0.94390. Further by adding user effects, we improve our accuracy around 18% compared with “Only by average rating” model. Then, by using the penalized least squares approach, regulation of movie and user effects, we successfully improve our accuracy with RMSE 0.86481 which is less than 0.86490. Last, by considering all effects caused by four features movieId, userId, ages and genres together, we successfully improve our accuracy with RMSE 0.86274.

However, we notice that there is very limited improvement (from 0.8654 to 0.8648) by introducing regularization model. The reason behind is the sparsity challenge of MovieLens data set. If we think in terms of a large matrix, with user on the rows and movies on the columns, we’ll get a large sparse matrix containing many empty cells. Obviously, all 6 models and machine learning algorithms developed till now in this project have big limitations on sparse data matrix.

## 4.2 Future Work

We need more powerful machine learning algorithms and techniques to design a recommendation system such as Matrix factorization which is very much related to factor analysis, singular value decomposition (SVD), and principal component analysis (PCA).

Another method to be considered for future work could be Ensembles which can usually greatly improve the final results by combining the results of different algorithms.