Rating Prediction for MovieLens

José Alejandro Luna Matos

September 6, 2021

MovieLens Project

The goal of the project is to build a recommendation system for movies using machine learning in R . In order to get that , the Data will be explored , transform and exploited.

The function RMSE that it will used to observe effectiveness

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

```
RMSE <- function(real_ratings = NULL, predicted_ratings = NULL) {
    sqrt(mean((real_ratings - predicted_ratings)^2))
}
```

The name of columns,type and description of columns in both datasets are six:

- **userId** `<integer>` Unique identification number for each user.
- **movieId** `<numeric>` Unique identification number for each movie.
- **rating** `<numeric>` Rating by one user of one movie . From 0.5 to 5 with 0.5 of increments.
- **timestamp** `<integer>` Timestamp for one specific rating provided by one user in seconds from 1970-01-01.
- **title** `<character>` title of each movie including the year of the release.
- **genres** `<character>` Genre of each movie separated by pipes.

The original dataset has been divided into 2 subsets "edx" and "validation" .

Dimension of movilens data sets "edx" 90 % of original dataset

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

Dimension of movilens data sets "validation" 10 % of original dataset

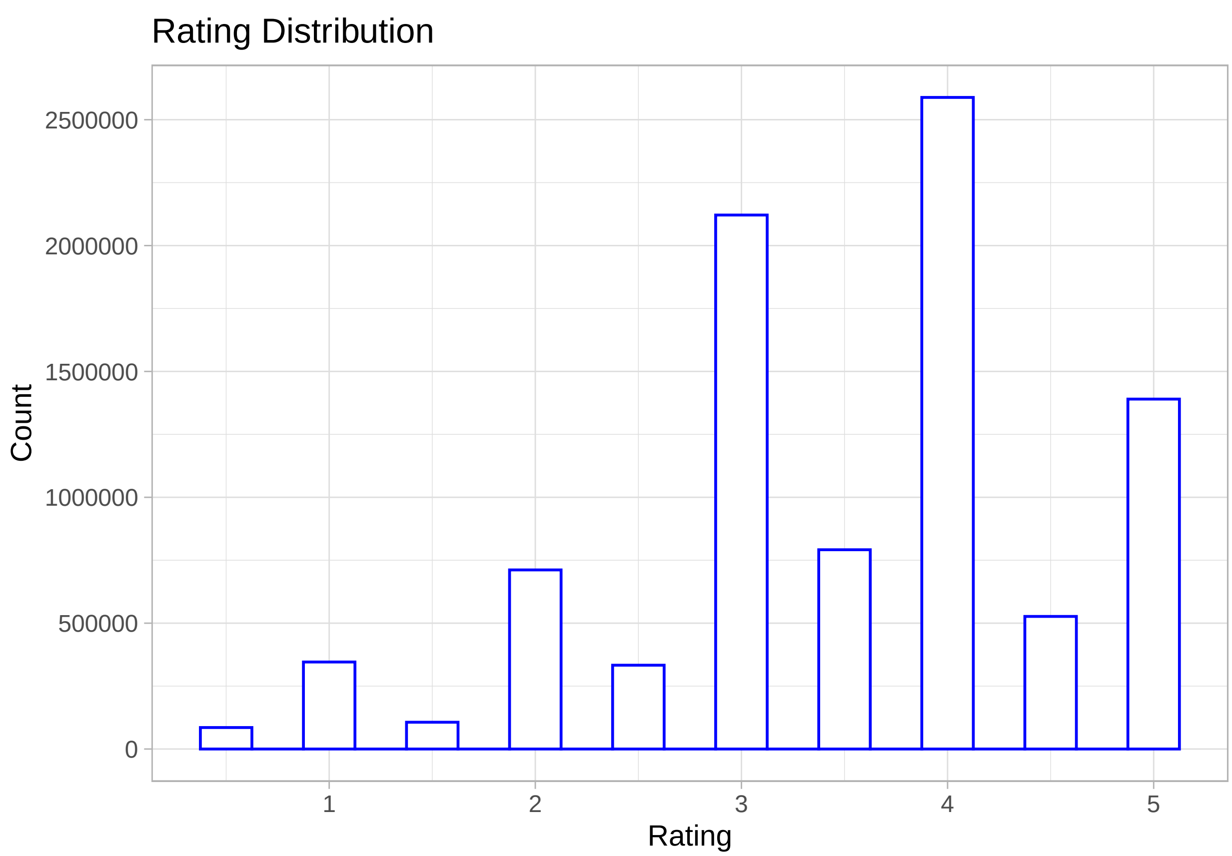
```
## [1] 999999      6
```

A view of ten first rows of set 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)
##  7:      1     356      5 838983653                Forrest Gump (1994)
##  8:      1     362      5 838984885            Jungle Book, The (1994)
##  9:      1     364      5 838983707              Lion King, The (1994)
## 10:      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult (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
##  7:                   Comedy|Drama|Romance|War
##  8:                 Adventure|Children|Romance
##  9: Adventure|Animation|Children|Drama|Musical
## 10:                              Action|Comedy
```

Histogram of Rating distribution



How many users and movies in edx subset ?

```
##   Users Movies
## 1 69878  10677
```

Summary : count,median , mean, max,1rd an 3rd quartiele by field The summary of the subsets "edx" and "validation" shows that there are no missing values. Summary of **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
##
##
##
```

Summary of **validation**

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   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:999999      Length:999999
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

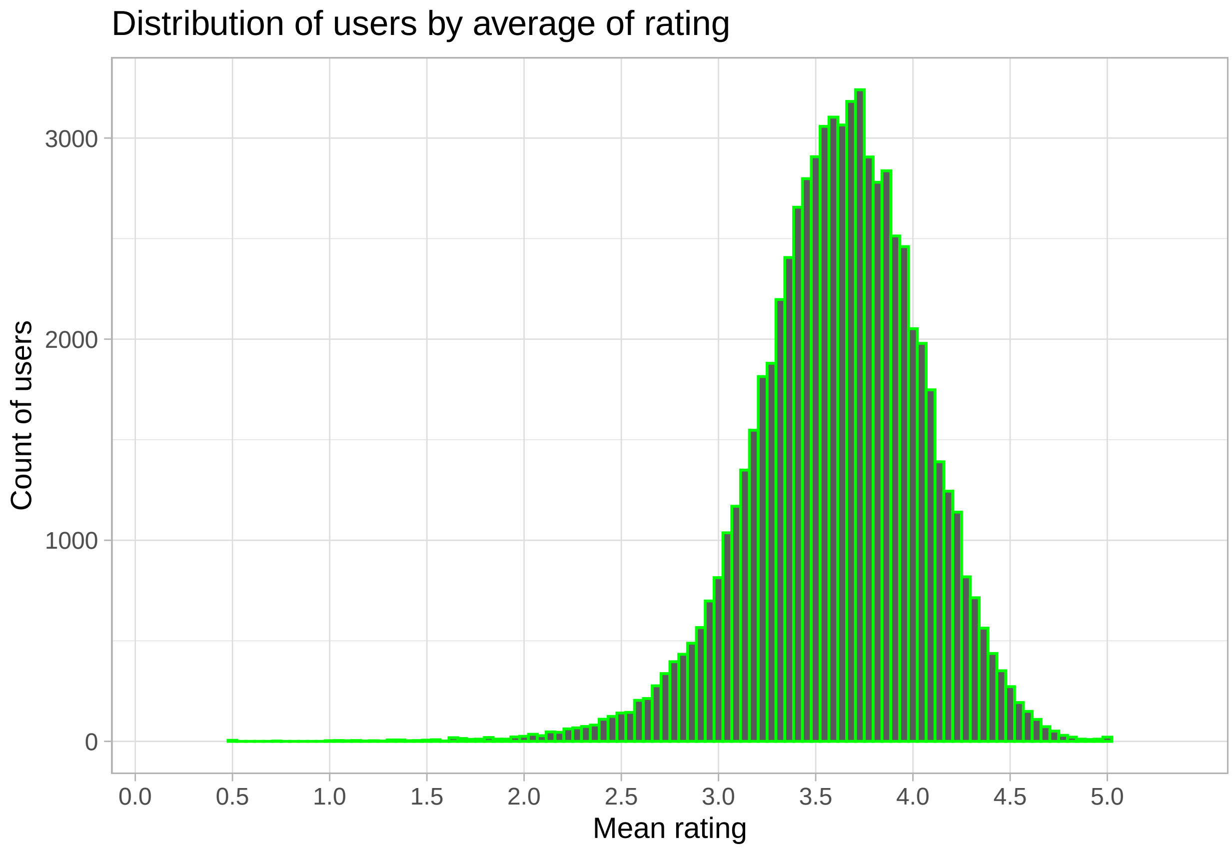
another way … Does exists missing values in any column? No,it does not **edx**

```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

validation

```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

Users are not equally critical of their ratings, some users tend to give ratings much lower or higher than average. The next praphics includes only users who have rated at least 10 movies



it looks like normal distribution with center close to 3.6

Dataset Pre-Processing

The pre-processing phase is composed by this steps (always in both datasets):

1. Convert column timestamp to datetime format.
2. Extract the hour , day, month , the year and week day from the date.
3. Extract the year for each movie from the title.
4. Get antiquity.
5. Get number of rating per movie.
6. Separate each genre actually separated by pipe.
7. Get count genre per movie.
8. Convert columns to desidered data type.

Steps Development

1. Convert column timestamp to datetime format.

```
edx$datetime        <- as.POSIXct(edx$timestamp, origin="1970-01-01")
validation$datetime <- as.POSIXct(validation$timestamp, origin="1970-01-01")
```

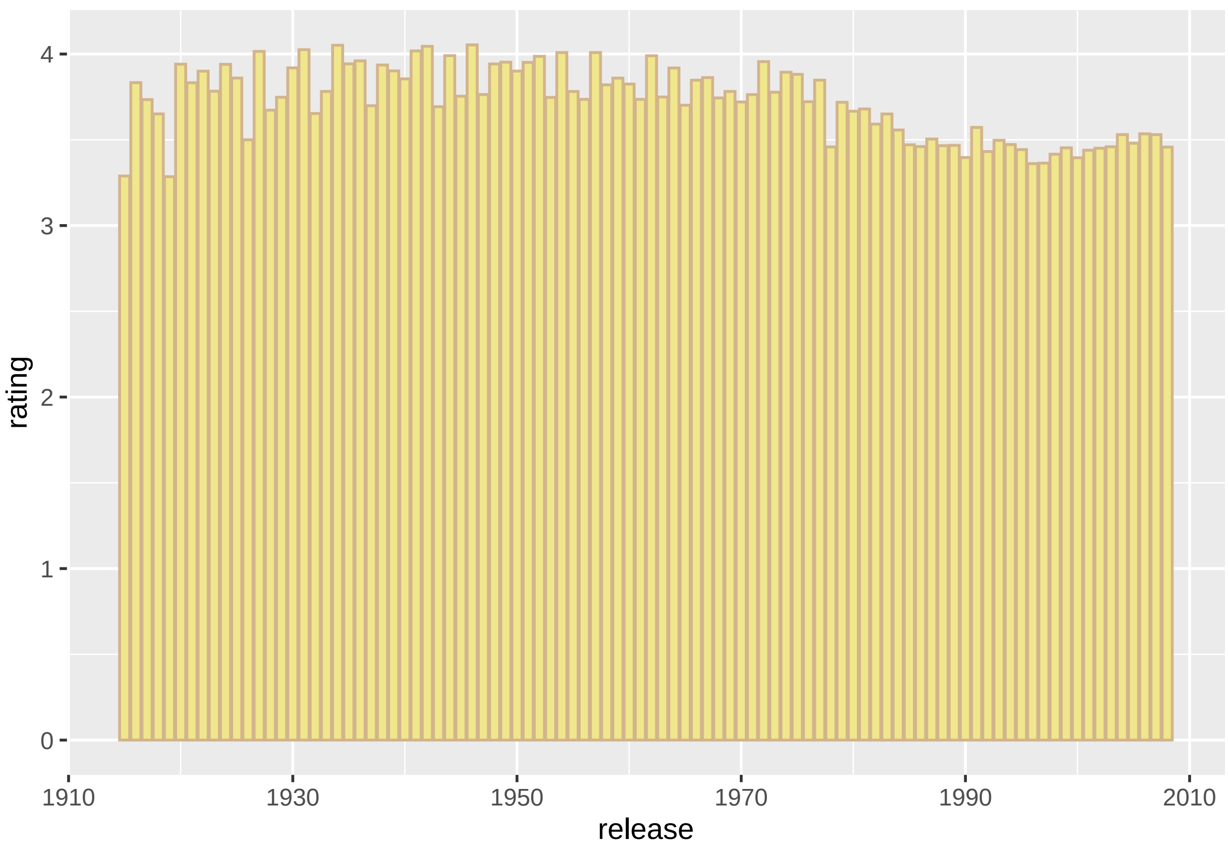
2. Extract the hour , day, month , the year and week day from the date in both datasets

```
edx$weekday_Rate <- weekdays(edx$datetime)
edx$hour_Rate <- format(edx$datetime,"%H")
edx$year_Rate <- format(edx$datetime,"%Y")
edx$month_Rate <- format(edx$datetime,"%m")
validation$weekday_Rate <- weekdays(validation$datetime)
validation$hour_Rate <- format(validation$datetime,"%H")
validation$year_Rate <- format(validation$datetime,"%Y")
validation$month_Rate <- format(validation$datetime,"%m")
```

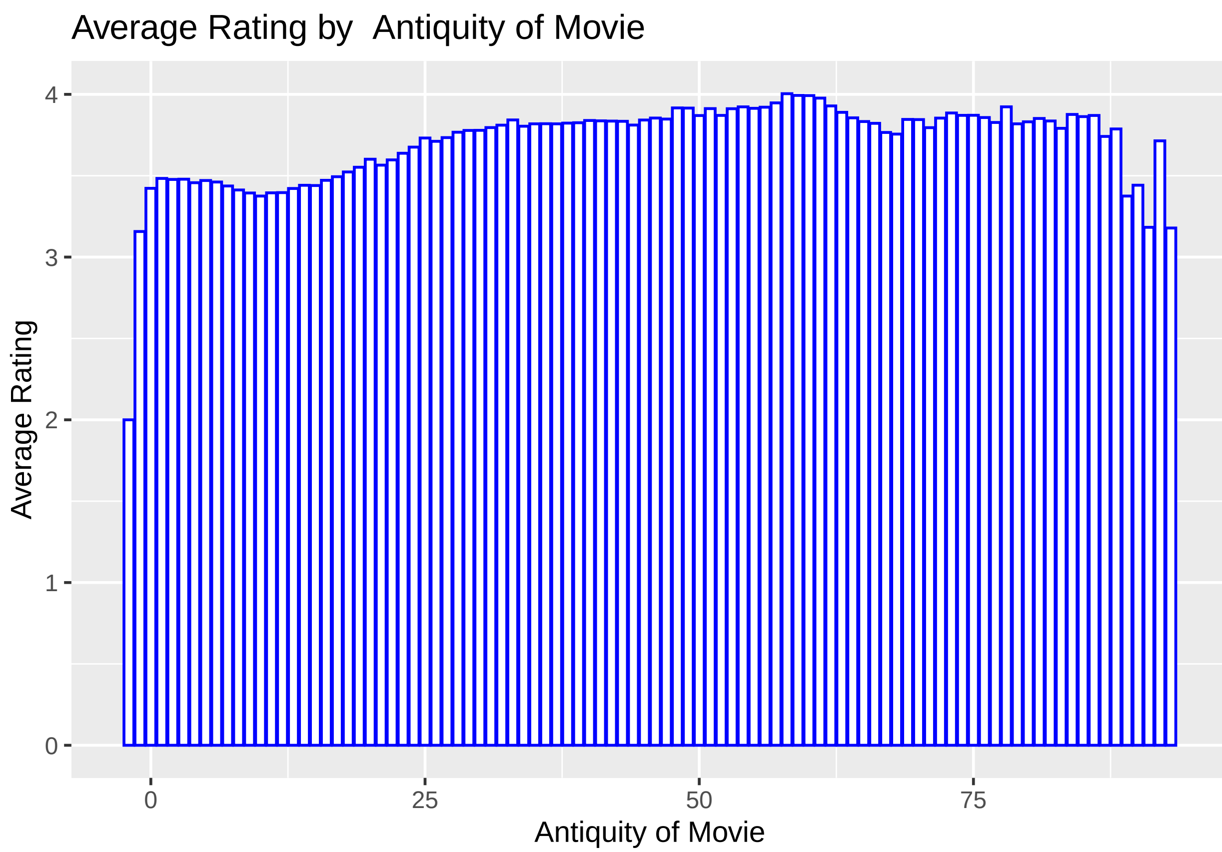
3. Extract the year for each movie from the title.

```
# edx
edx <- edx %>%
   mutate(title = str_trim(title )) %>%
   extract(title,
           c("title_with_out_year", "year_movie"),
           regex = "^(.*) \\(([0-9 \\-]*)\\)$",
           remove = F) %>%
   mutate(release = if_else(str_length(year_movie) >= 5,
                            as.integer(str_split(year_movie, "-",simplify = T)[1]),
                            as.integer(year_movie))
   ) %>%
   mutate(title = if_else(is.na(title_with_out_year),
                          title,
                          title_with_out_year)
         ) %>%
  select(-title_with_out_year)
# validation
validation <- validation %>%
   mutate(title = str_trim(title )) %>%
   extract(title,
           c("title_with_out_year", "year_movie"),
           regex = "^(.*) \\(([0-9 \\-]*)\\)$",
           remove = F) %>%
   mutate(release = if_else(str_length(year_movie) >= 5,
                            as.integer(str_split(year_movie, "-",simplify = T)[1]),
                            as.integer(year_movie))
   ) %>%
   mutate(title = if_else(is.na(title_with_out_year),
                          title,
                          title_with_out_year)
         ) %>%
  select(-title_with_out_year)
```

The graphics below shows the distribution by year of movie


rating
0
1
2
3
4
1910
1930
1950
1970
1990
2010
release

4.Get antiquity
The figure shows that old movies have more rating tan news , except the olders



There are some ratings with a error, the rating year is lower than the year of the movie **edx** dataset 175 rows

```
##      n
## 1: 175
```

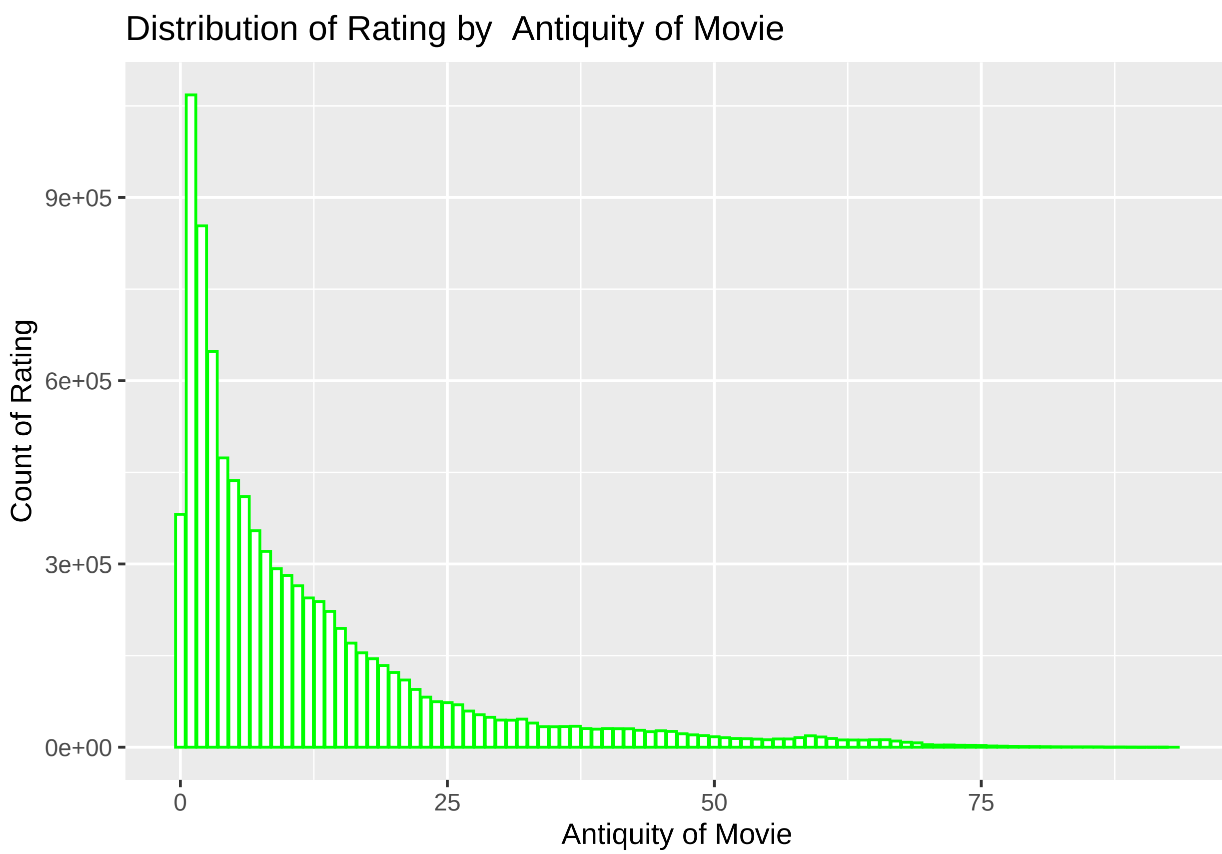
validation dataset 26 rows

```
##     n
## 1: 26
```

Two options : 1 correct the year of rating and antiquity 2 drop the rows

option 2 is taken

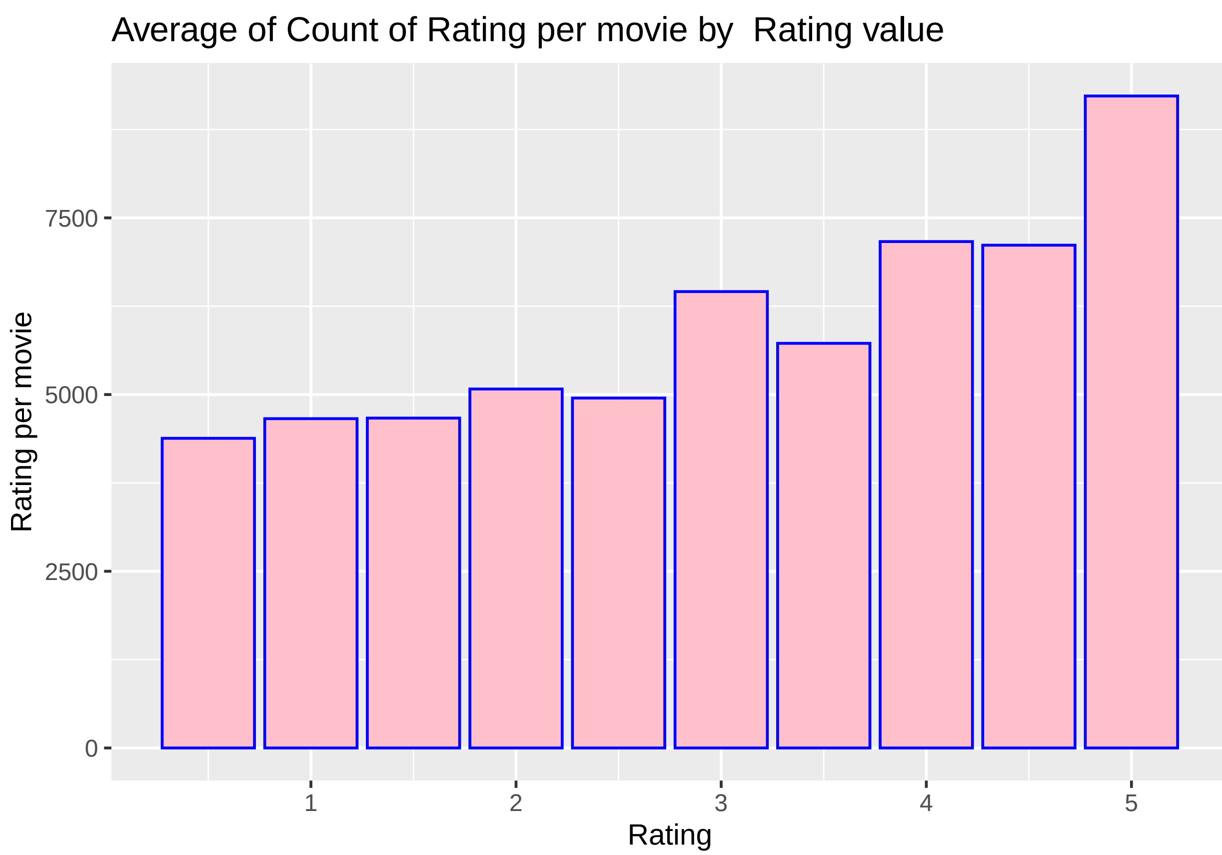
In the next graphic : older movies have less ratings


Distribution of Rating by Antiquity of Movie
Count of Rating
9e+05
6e+05
3e+05
0e+00
0
25
50
75
Antiquity of Movie

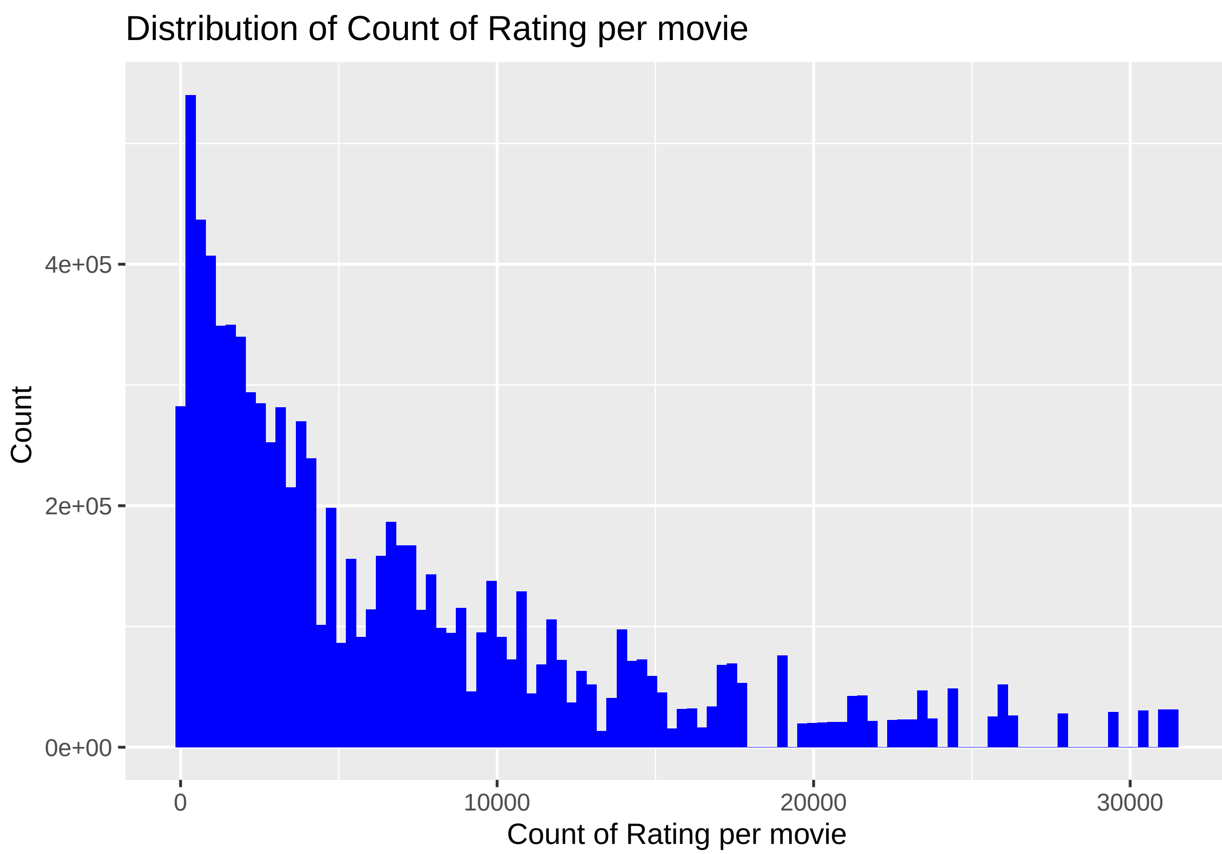
5. Get number of rating per movie
The first 10 movieId and "How many ratings they have?"

```
## # A tibble: 10 x 2
##    movieId cnt_ratings
##      <dbl>       <int>
##  1       1       23790
##  2       2       10779
##  3       3        7028
##  4       4        1577
##  5       5        6400
##  6       6       12346
##  7       7        7259
##  8       8         821
##  9       9        2278
## 10      10       15187
```

More count of ratings mean High ratings per movies

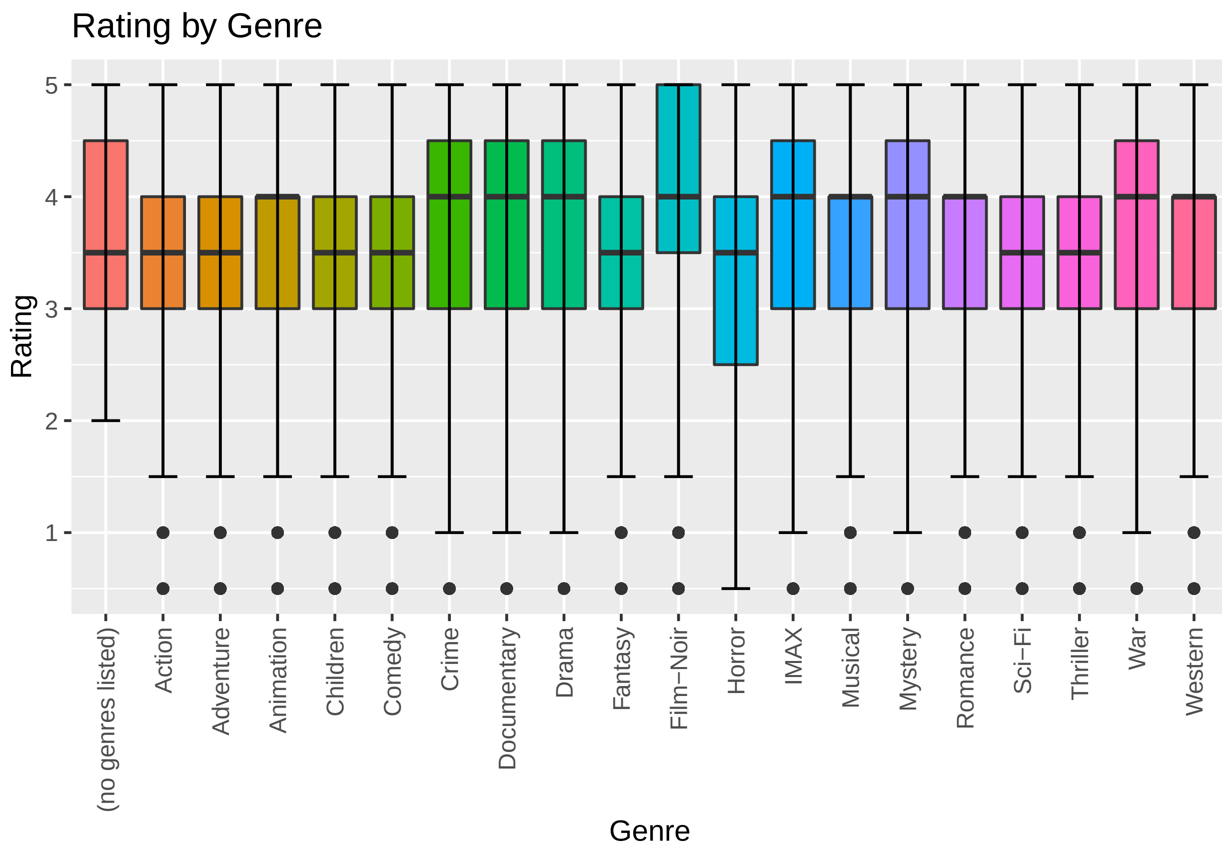


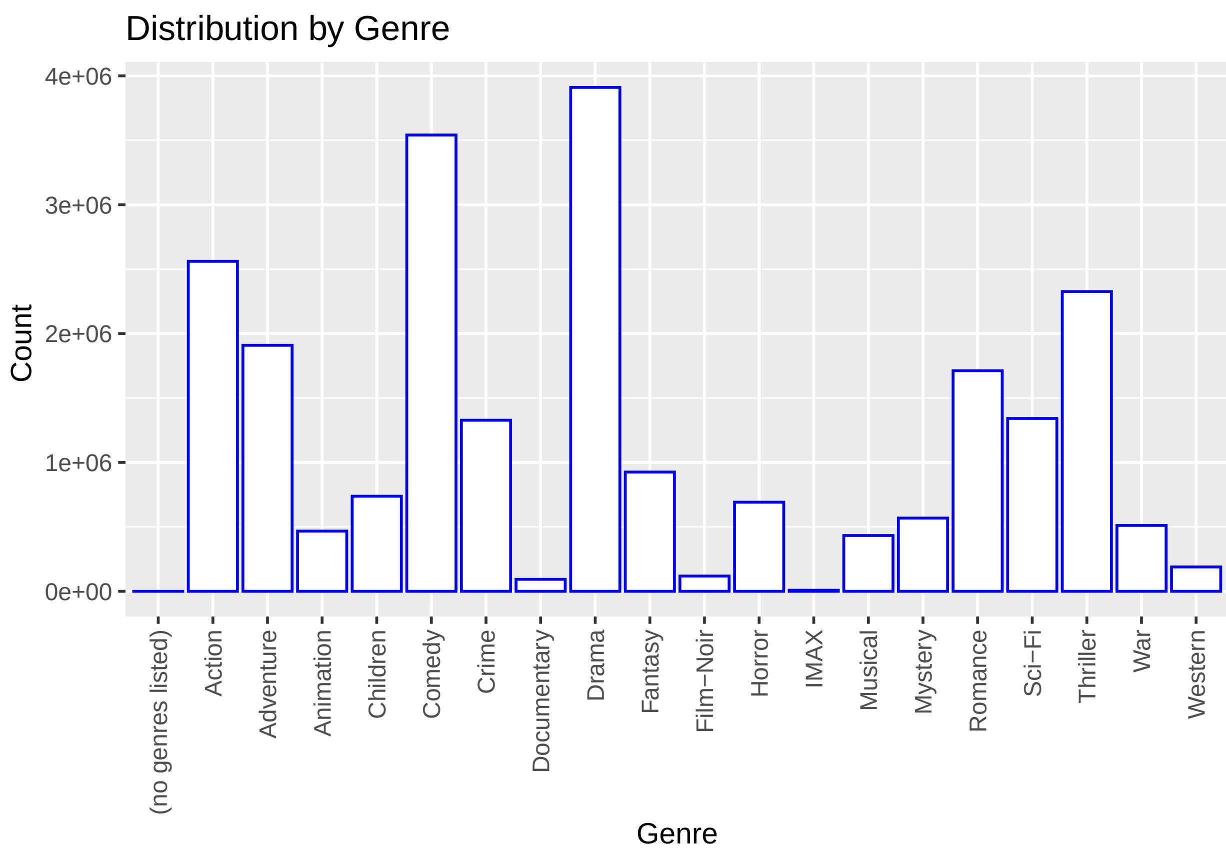
More count of ratings are for less movies


Distribution of Count of Rating per movie
Count
4e+05
2e+05
0e+00
0
10000
20000
30000
Count of Rating per movie

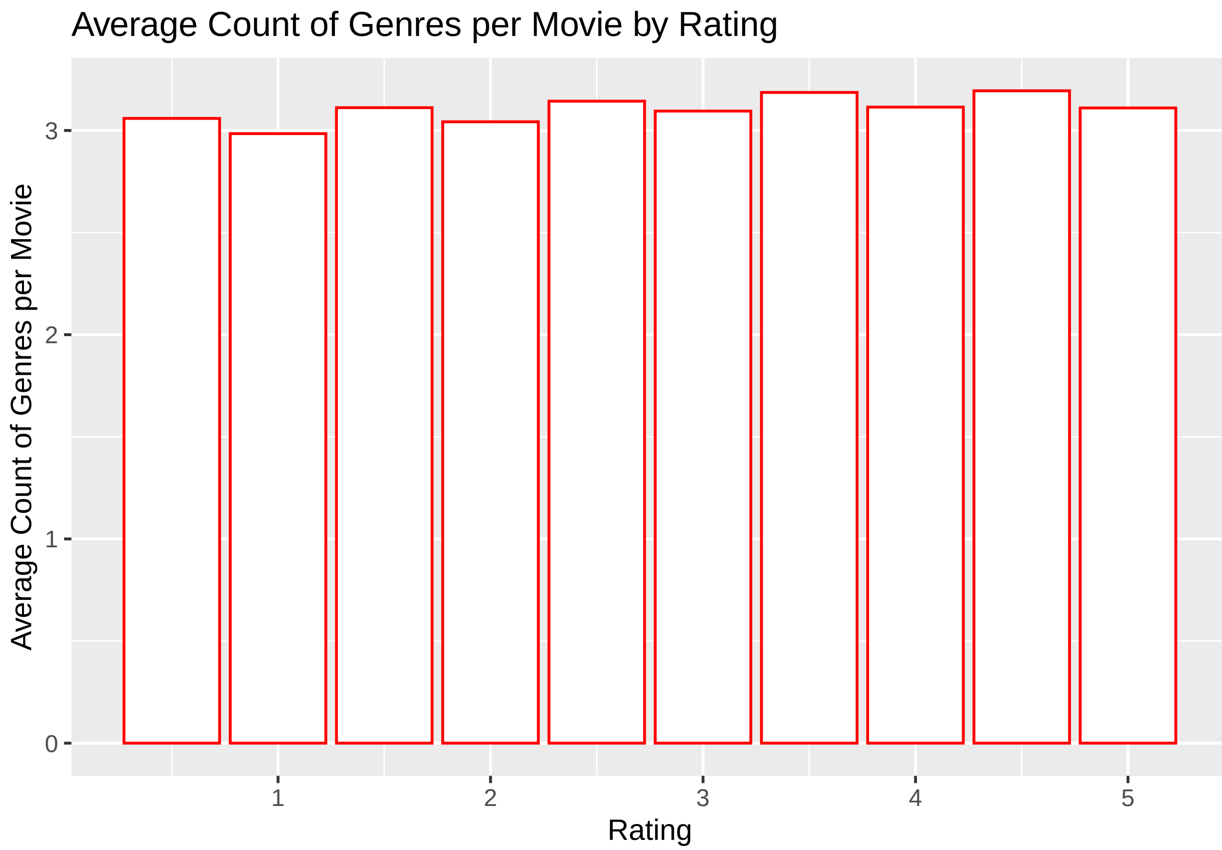
6.Get individual genre
Film noir have best average ratings as genre

```
# Extract the genre in edx datasets
edx <- edx %>%
  mutate(genre = fct_explicit_na(genres,na_level = "(No genres)")) %>%
  separate_rows(genre,sep = "\\|")
# Extract the genre in validation datasets
validation <- validation %>%
  mutate(genre = fct_explicit_na(genres,na_level = "(No genres)")) %>%
  separate_rows(genre,sep = "\\|")
```




Distribution by Genre
Count
4e+06
3e+06
2e+06
1e+06
0e+00
(no genres listed)
Action
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film−Noir
Horror
IMAX
Musical
Mystery
Romance
Sci−Fi
Thriller
War
Western
Genre

7. Get count genre per movie The figure shows a little relation between the rating and the number of genres per movie



8. Convert columns to desidered data type

```
edx$hour_Rate <- as.numeric(edx$hour_Rate)
edx$year_Rate <- as.numeric(edx$year_Rate)
edx$month_Rate <- as.numeric(edx$month_Rate)
edx$release <- as.numeric(edx$release)
edx$year_movie <- as.numeric(edx$year_movie)

validation$hour_Rate <- as.numeric(validation$hour_Rate)
validation$year_Rate <- as.numeric(validation$year_Rate)
validation$month_Rate <- as.numeric(validation$month_Rate)
validation$release <- as.numeric(validation$release)
validation$year_movie <- as.numeric(validation$year_movie)
```

Remove unnecessary columns on edx and validation dataset

```
edx <-edx%>%
   select(userId, movieId, rating, title, genre, release, weekday_Rate,
          hour_Rate, year_Rate, month_Rate,count_genres,antiquity,cnt_ratings)
validation <- validation %>%
   select(userId, movieId, rating, title, genre, release, weekday_Rate,
          hour_Rate, year_Rate, month_Rate,count_genres,antiquity,cnt_ratings)
```

Summary to see is there are NA values count, median,mean ,min,max,1rd an 3rd quartiele

```
##      userId         movieId          rating          title
##  Min.   :    1   Min.   :    1   Min.   :0.500   Length:23371130
##  1st Qu.:18141   1st Qu.:  616   1st Qu.:3.000   Class :character
##  Median :35784   Median : 1748   Median :4.000   Mode  :character
##  Mean   :35886   Mean   : 4277   Mean   :3.527
##  3rd Qu.:53638   3rd Qu.: 3635   3rd Qu.:4.000
##  Max.   :71567   Max.   :65133   Max.   :5.000
##     genre              release      weekday_Rate         hour_Rate
##  Length:23371130    Min.   :1915   Length:23371130    Min.   : 0.00
##  Class :character   1st Qu.:1987   Class :character   1st Qu.: 8.00
##  Mode  :character   Median :1995   Mode  :character   Median :13.00
##                     Mean   :1990                      Mean   :12.59
##                     3rd Qu.:1998                      3rd Qu.:18.00
##                     Max.   :2008                      Max.   :23.00
##    year_Rate      month_Rate     count_genres     antiquity      cnt_ratings
##  Min.   :1995   Min.   : 1.00   Min.   :1.000   Min.   : 0.00   Min.   :    1
##  1st Qu.:2000   1st Qu.: 4.00   1st Qu.:2.000   1st Qu.: 2.00   1st Qu.: 2038
##  Median :2003   Median : 7.00   Median :3.000   Median : 7.00   Median : 5303
##  Mean   :2002   Mean   : 6.79   Mean   :3.111   Mean   :11.84   Mean   : 7514
##  3rd Qu.:2005   3rd Qu.:10.00   3rd Qu.:4.000   3rd Qu.:16.00   3rd Qu.:10757
##  Max.   :2009   Max.   :12.00   Max.   :8.000   Max.   :93.00   Max.   :31362
```

After preprocessing the data, **edx** dataset looks like this:

Processed edx datadaset

```
## # A tibble: 6 x 13
##   userId movieId rating title     genre    release weekday_Rate hour_Rate year_Rate
##    <int>   <dbl>  <dbl> <chr>     <chr>      <dbl> <chr>            <dbl>     <dbl>
## 1      1     122      5 Boomerang Comedy      1992 viernes              6      1996
## 2      1     122      5 Boomerang Romance     1992 viernes              6      1996
## 3      1     185      5 Net, The  Action      1995 viernes              5      1996
## 4      1     185      5 Net, The  Crime       1995 viernes              5      1996
## 5      1     185      5 Net, The  Thriller    1995 viernes              5      1996
## 6      1     292      5 Outbreak  Action      1995 viernes              5      1996
## # ... with 4 more variables: month_Rate <dbl>, count_genres <int>,
## #   antiquity <dbl>, cnt_ratings <int>
```

Model Building and Evaluation

Naive Baseline Model

Naive Model predict the average of all movie ratings is approximately 3.53

```
## [1] "The mean of rating in edx set is: 3.52702434584892"
```

Model 1 : Naive Mean-Baseline

The formula used for this model:

$$Y_{u,i} = \hat{\mu} + \varepsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent sampled errors.

```
mu_hat <- mean(edx$rating)
rmse_model_1 <- RMSE(validation$rating, mu_hat)
c_model<-"Model 1: Naive Mean Baseline  "
results <- data.frame(model=c_model, RMSE=rmse_model_1)
tibble(Method = c_model, RMSE = rmse_model_1)
```

```
## # A tibble: 1 x 2
##   Method                            RMSE
##   <chr>                            <dbl>
## 1 "Model 1: Naive Mean Baseline  "  1.05
```

Model 2 : Movie-Based Model, a Content-based Approach

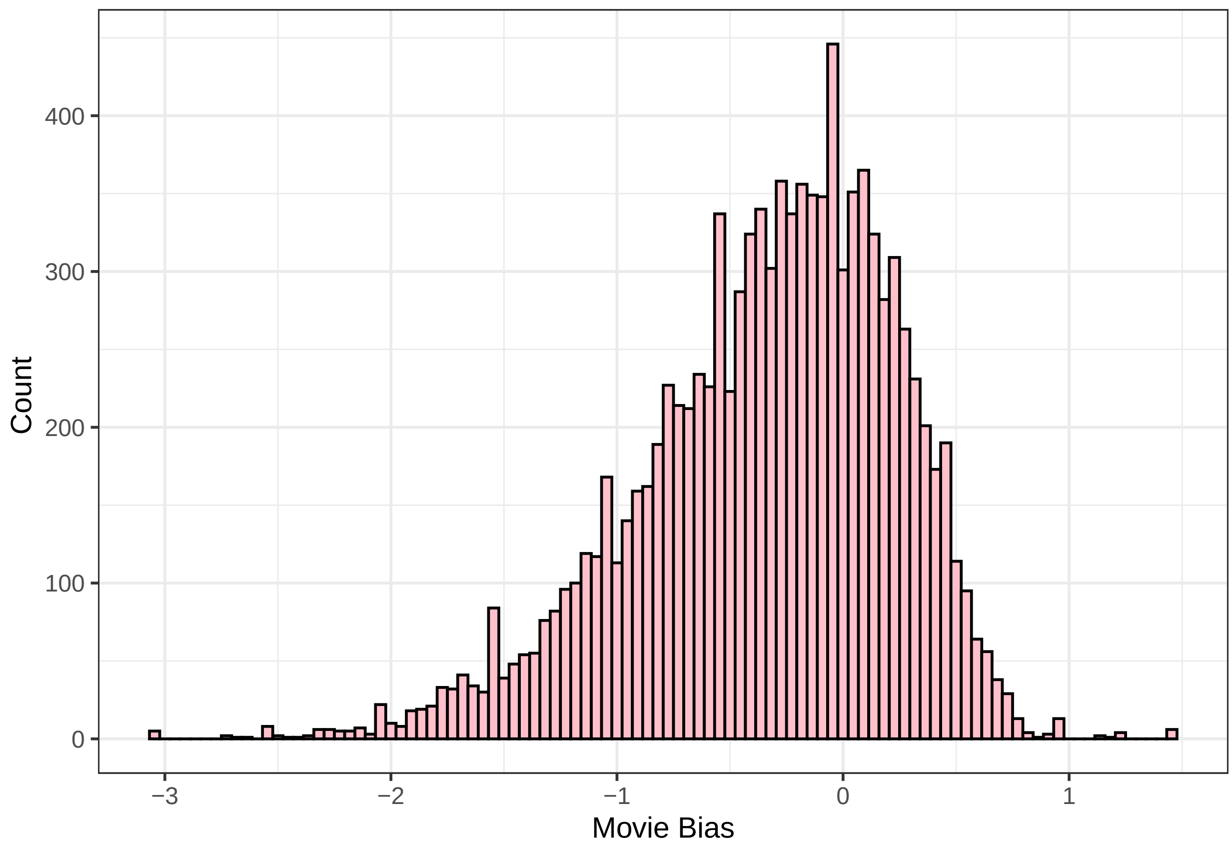
The first Non-Naive Model takes into account the content. In this case the movies that are rated higher or lower resperct to each other.

The formula used is:

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

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i.

```r
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Calculate the average by movie
movie_avgs <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))
# Graph the Movie Bias
movie_avgs %>% ggplot(aes(b_i)) +
  geom_histogram(color = "black", fill = "pink", bins = 100) +
  xlab("Movie Bias") +
  ylab("Count ") +
  theme_bw()
```




```r
# Compute the predicted ratings on validation dataset
rmse_movie_model <- validation %>%
   left_join(movie_avgs, by='movieId') %>%
   mutate(pred = mu_hat + b_i) %>%
   pull(pred)
rmse_model_2 <- RMSE(validation$rating, rmse_movie_model)
# Adding results to the results dataset
c_model<-"Model 2: Movie-Based Model  "
results <- results %>% add_row(model=c_model, RMSE=rmse_model_2)
tibble(Method = c_model, RMSE = rmse_model_2)
```

```
## # A tibble: 1 x 2
##   Method                            RMSE
```

```
##   <chr>                          <dbl>
## 1 "Model 2: Movie-Based Model  " 0.941
```

The RMSE on the **validation** dataset is **0.94107**. It better than the Naive Mean-Baseline Model, but it is also very far from the target RMSE (below 0.87) and that indicates poor performance for the model.

Model 3: Movie + User Model, a User-based approach

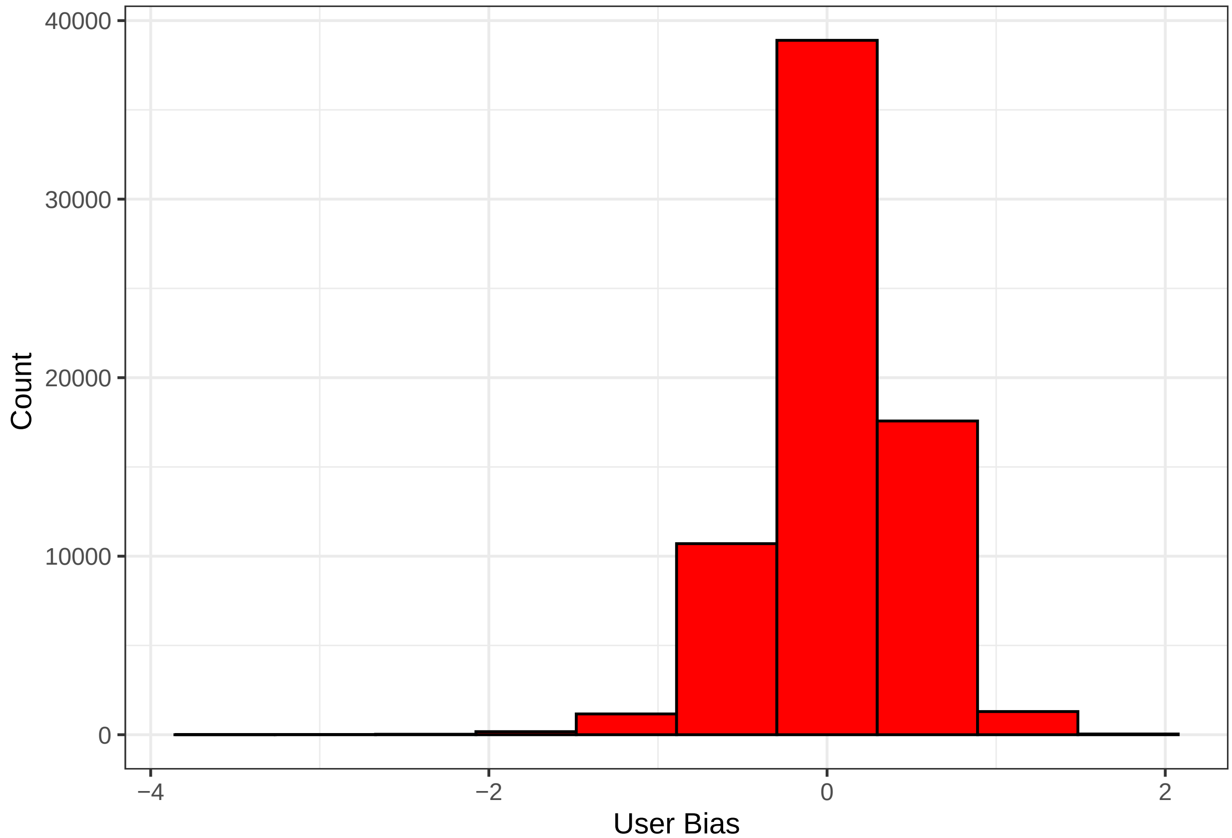
The second Non-Naive Model consider that the users have different tastes and rate differently.

The formula used is:

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

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i. The b_u is a measure for the mildness of user u, i.e. the bias of user u.

```r
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Calculate the average by movie
movie_avgs <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))
# Calculate the average by user
user_avgs <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu_hat - b_i))
# Graph the User Bias
user_avgs %>% ggplot(aes(b_u)) +
  geom_histogram(color = "black", fill = "red", bins = 10) +
  xlab("User Bias") +
  ylab("Count ") +
  theme_bw()
```




```
# Compute the predicted ratings on validation dataset
rmse_movie_user_model <- validation %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   mutate(pred = mu_hat + b_i + b_u) %>%
   pull(pred)
rmse_model_3 <- RMSE(validation$rating, rmse_movie_user_model)
# Adding the results to the results dataset
c_model<-"Model 3: Movie+User Based Model  "
results <- results %>% add_row(model=c_model, RMSE=rmse_model_3)
tibble(Method = c_model, RMSE = rmse_model_3)
```

```
## # A tibble: 1 x 2
##   Method                               RMSE
##   <chr>                               <dbl>
## 1 "Model 3: Movie+User Based Model  " 0.863
```

The RMSE on the **validation** dataset is **0.86337** and this is very good. The Movie+User Based Model reaches the desidered performance but applying the regularization techniques, can improve the performance just a little.

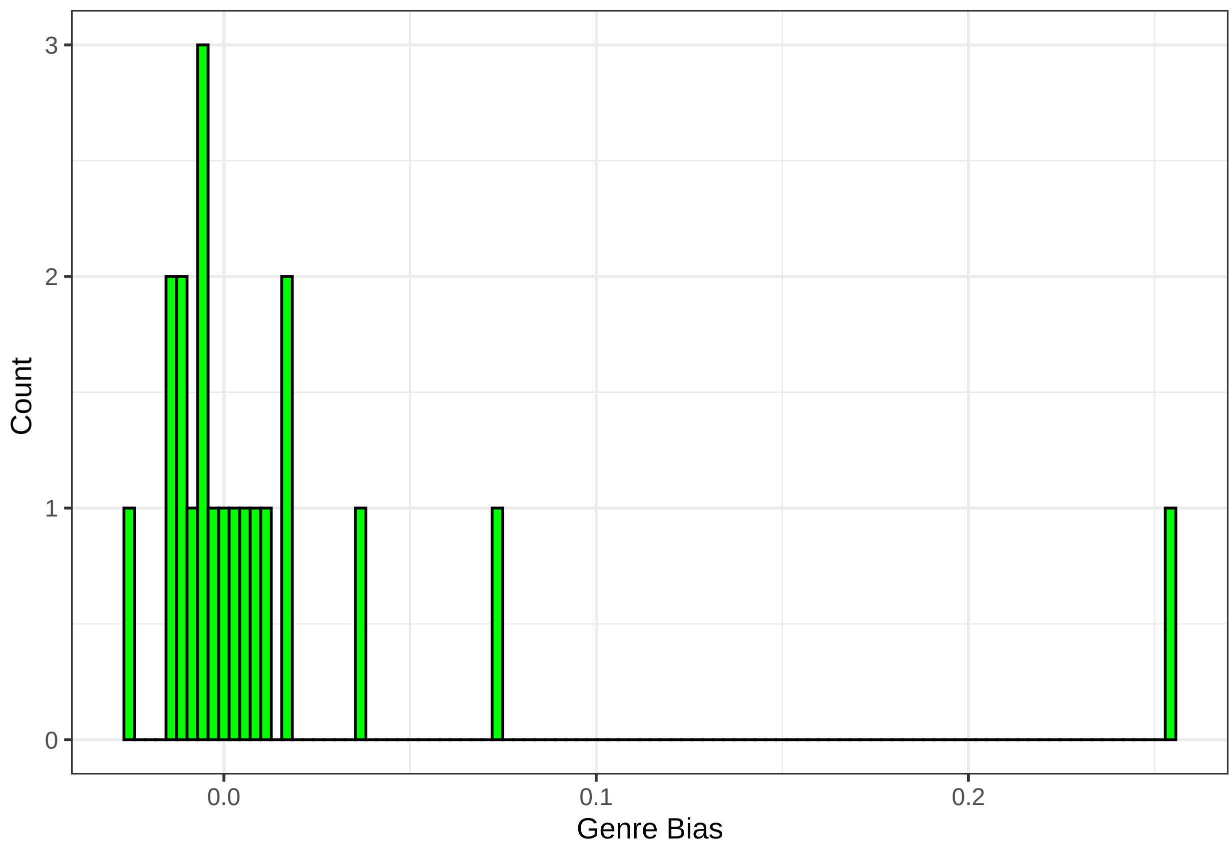
Model 4 : Movie + User + Genre Model, the Genre Popularity

The formula used is:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_{u,g} + \epsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i. The b_u is a measure for the mildness of user u, i.e. the bias of user u. The $b_{u,g}$ is a measure for how much a user u likes the genre g.

```
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Calculate the average by movie
movie_avgs <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))
# Calculate the average by user
user_avgs <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu_hat - b_i))
genre_pop <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   group_by(genre) %>%
   summarize(b_u_g = mean(rating - mu_hat - b_i - b_u))

# Graph the Genre Popularity Bias
genre_pop %>% ggplot(aes(b_u_g)) +
  geom_histogram(color = "black", fill = "Green", bins = 100) +
  xlab("Genre Bias") +
  ylab("Count ") +
  theme_bw()
```



```r
# Compute the predicted ratings on validation dataset
rmse_movie_user_genre_model <- validation %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   left_join(genre_pop, by='genre') %>%
   mutate(pred = mu_hat + b_i + b_u + b_u_g) %>%
   pull(pred)
rmse_model_4 <- RMSE(validation$rating, rmse_movie_user_genre_model)
# Adding the results to the results dataset
c_model<-"Model 4:Movie+User+Genre Based Model  "
results <- results %>% add_row(model=c_model, RMSE=rmse_model_4)
tibble(Method = c_model, RMSE = rmse_model_4)
```

```
## # A tibble: 1 x 2
##   Method                                    RMSE
##   <chr>                                    <dbl>
## 1 "Model 4:Movie+User+Genre Based Model  " 0.863
```

The RMSE on the **validation** dataset is **0.86327** and this is very good. The Movie+User+Genre Based Model reaches the desidered performance but adding the `genre` predictor, doesn't improve significantly the model's performance. Applying the regularization techniques, can improve the performance just a little.

Model 5:Movie + User + Genre + Antiquity Model, the Antiquity Popularity

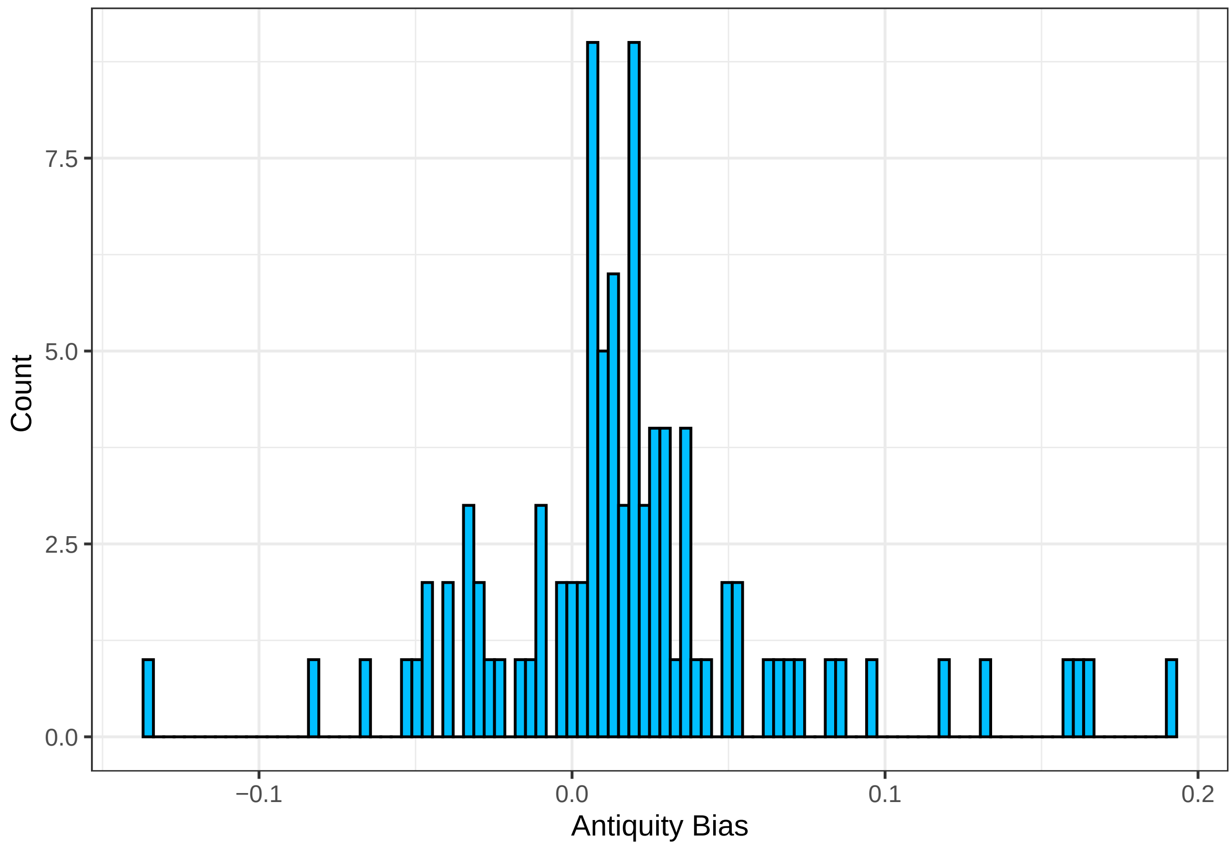
The formula used is:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_{u,g} + b_a + \epsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i. The b_u is a measure for the mildness of user u, i.e. the bias of user u. The $b_{u,g}$ is a measure for how much a user u likes the genre g. The b_a is a measure for the antiquity of movie i, i.e. the bias of antiquity of movie i.

```r
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Calculate the average by movie
movie_avgs <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))
# Calculate the average by user
user_avgs <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu_hat - b_i))

genre_pop <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   group_by(genre) %>%
   summarize(b_u_g = mean(rating - mu_hat - b_i - b_u))

antiquity_avg <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   left_join(genre_pop, by='genre') %>%
```

```
   group_by(antiquity) %>%
   summarize(b_a = mean(rating - mu_hat - b_i - b_u - b_u_g))

# Graph the antiquity Popularity Bias
antiquity_avg %>% ggplot(aes(b_a)) +
  geom_histogram(color = "black", fill = "deepskyblue", bins = 100) +
  xlab("Antiquity Bias") +
  ylab("Count ") +
  theme_bw()
```




```
# Compute the predicted ratings on validation dataset
rmse_movie_user_genre_antiquity_model <- validation %>%
   left_join(movie_avgs,    by='movieId') %>%
   left_join(user_avgs,     by='userId') %>%
   left_join(genre_pop,     by='genre') %>%
   left_join(antiquity_avg, by='antiquity') %>%
   mutate(pred = mu_hat + b_i + b_u + b_u_g + b_a) %>%
   pull(pred)
rmse_model_5 <- RMSE(validation$rating, rmse_movie_user_genre_antiquity_model)
# Adding the results to the results dataset
c_model<-"Movie+User+Genre+Antiquity Based Model  "
results <- results %>% add_row(model=c_model, RMSE=rmse_model_5)
tibble(Method = c_model, RMSE = rmse_model_5)
```

```
## # A tibble: 1 x 2
##   Method                                      RMSE
```

```
##   <chr>                                         <dbl>
## 1 "Movie+User+Genre+Antiquity Based Model  " 0.863
```

The RMSE on the **validation** dataset is **0.86274** and this is very good. The Movie+User+Genre+Antiquity Based Model reaches the desidered performance but adding the `Antiquity` predictor, doesn't improve significantly the model's performance. Applying the regularization techniques, can improve the performance just a little.

Model 6: Movie + User + Genre + Antiquity +Count Rating Model

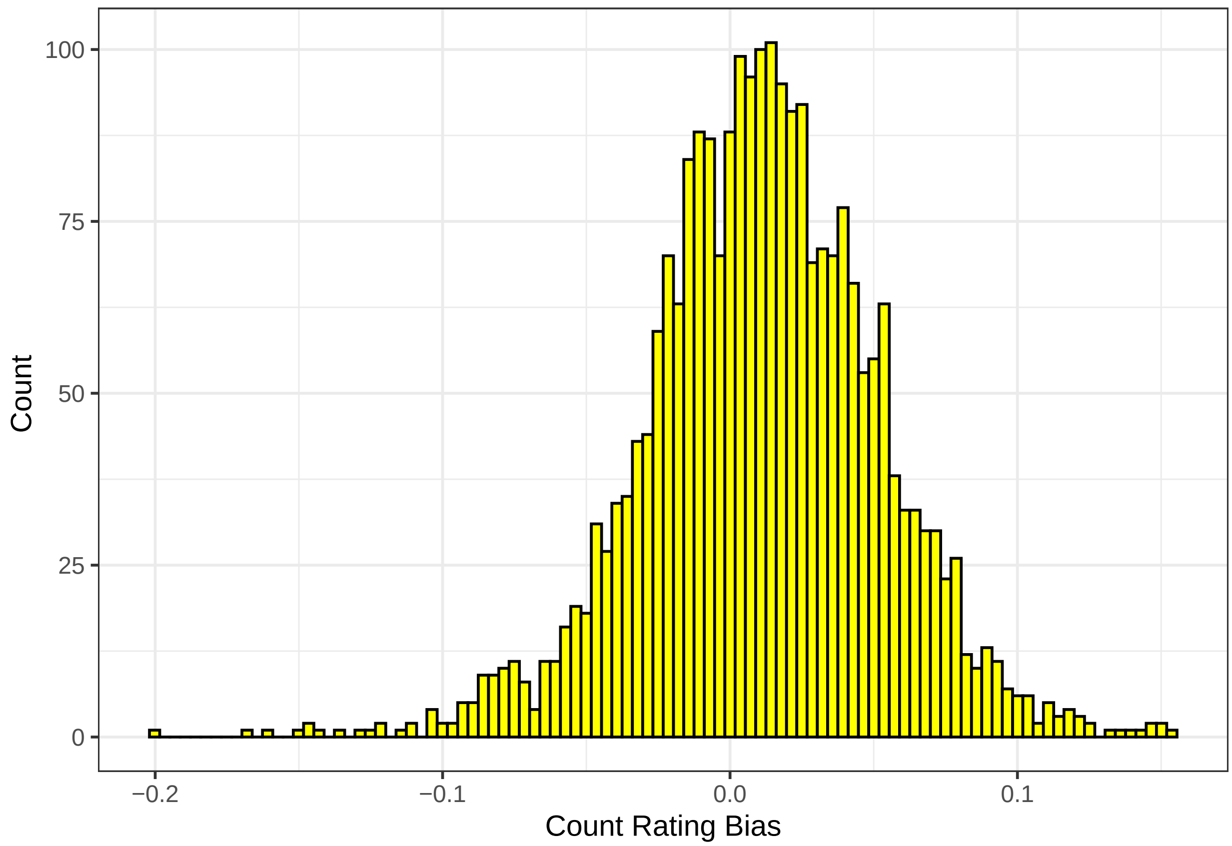
The formula used is:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_{u,g} + b_a + b_c + \epsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i. The b_u is a measure for the mildness of user u, i.e. the bias of user u. The $b_{u,g}$ is a measure for how much a user u likes the genre g. The b_a is a measure for the antiquity of movie i, i.e. the bias of antiquity of movie i. The b_c is a measure for the count of rating of movie i, i.e. the bias of count of rating of movie i.

```r
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Calculate the average by movie
movie_avgs <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))
# Calculate the average by user
user_avgs <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu_hat - b_i))

genre_pop <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   group_by(genre) %>%
   summarize(b_u_g = mean(rating - mu_hat - b_i - b_u))

antiquity_avg <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   left_join(genre_pop, by='genre') %>%
   group_by(antiquity) %>%
   summarize(b_a = mean(rating - mu_hat - b_i - b_u - b_u_g))

cant_rating_avg <- edx %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   left_join(genre_pop, by='genre') %>%
   left_join(antiquity_avg, by='antiquity') %>%
   group_by(cnt_ratings) %>%
   summarize(b_c = mean(rating - mu_hat - b_i - b_u - b_u_g - b_a ))
# Graph the count rating  Bias
cant_rating_avg %>% ggplot(aes(b_c)) +
  geom_histogram(color = "black", fill = "yellow", bins = 100) +
```

```r
  xlab("Count Rating Bias") +
  ylab("Count ") +
  theme_bw()
```




```r
# Compute the predicted ratings on validation dataset
rmse_movie_user_genre_antiquity_cant_rating_model <- validation %>%
   left_join(movie_avgs,      by='movieId') %>%
   left_join(user_avgs,       by='userId') %>%
   left_join(genre_pop,       by='genre') %>%
   left_join(antiquity_avg,   by='antiquity') %>%
   left_join(cant_rating_avg, by='cnt_ratings') %>%
   mutate(pred = mu_hat + b_i + b_u + b_u_g + b_a + b_c) %>%
   pull(pred)
rmse_model_6 <- RMSE(validation$rating, rmse_movie_user_genre_antiquity_cant_rating_model)
# Adding the results to the results dataset
c_model<-"Model 6: Movie + User + Genre + Antiquity +Count Rating Model  "
results <- results %>%
   add_row(model=c_model,RMSE=rmse_model_6)
tibble(Method = c_model, RMSE = rmse_model_6)
```

```
## # A tibble: 1 x 2
##   Method                                                             RMSE
##   <chr>                                                             <dbl>
## 1 "Model 6: Movie + User + Genre + Antiquity +Count Rating Model  " 0.862
```

The RMSE on the **validation** dataset is **0.86187** and this is very good. The Movie+User+Genre+Antiquity Based Model reaches the desidered performance but adding the `Antiquity` predictor, doesn't improve significantly the model's performance. Applying the regularization techniques, can improve the performance just a little.

Regularization

It allows to add a penalty λ (lambda) to penalizes movies with large estimates from a small sample size. In order to optimize b_i, it necessary to use this equation:

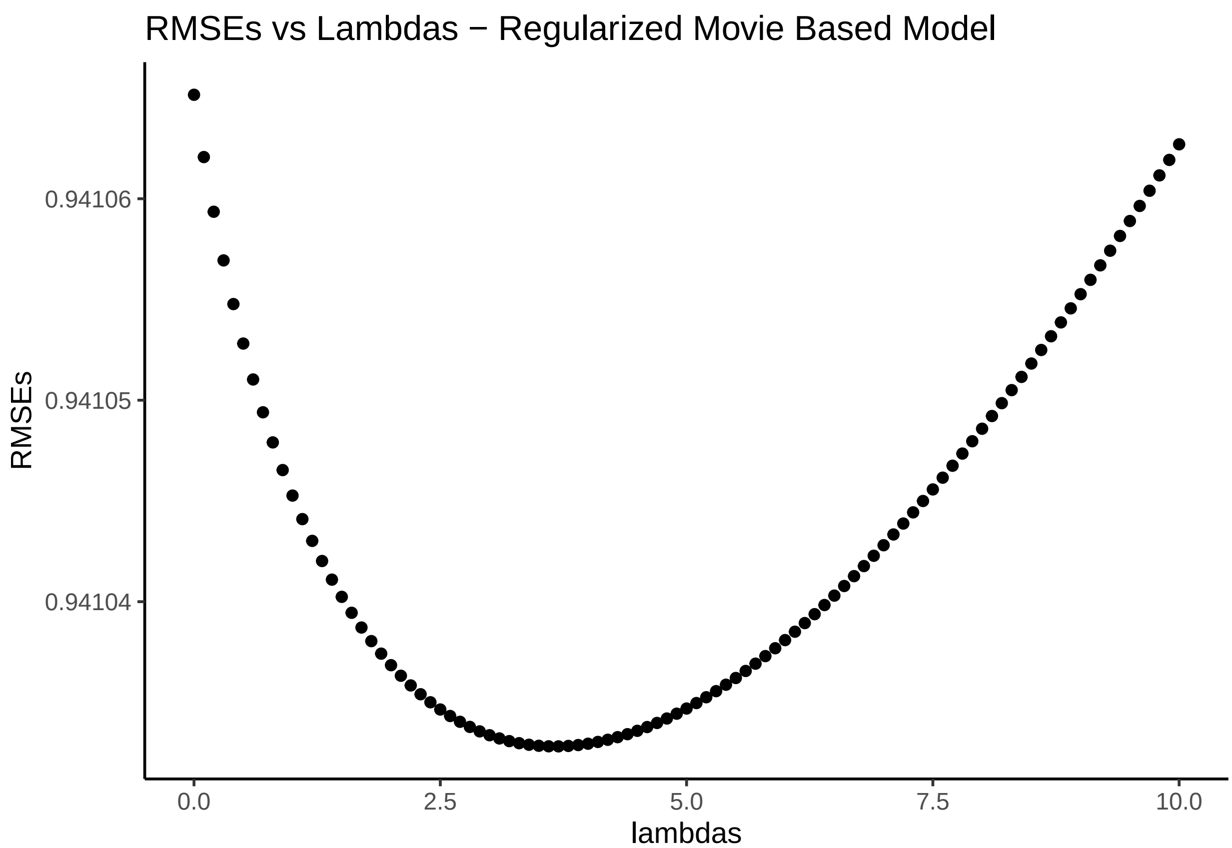
$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i)^2+\lambda\sum_i b_i^2$$

reduced to this equation:

$$\hat{b_i}(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu})$$

Regularized Movie-Based Model

```
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Define a table of lambdas
lambdas <- seq(0, 10, 0.1)
# Compute the predicted ratings on validation dataset using different values of lambda
rmses <- sapply(lambdas, function(lambda) {


  # Calculate the average by user

   b_i <- edx %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

   # Compute the predicted ratings on validation dataset

   predicted_ratings <- validation %>%
      left_join(b_i, by='movieId') %>%
      mutate(pred = mu_hat + b_i) %>%
      pull(pred)

   # Predict the RMSE on the validation set

   return(RMSE(validation$rating, predicted_ratings))
})
# plot the result of lambdas
df <- data.frame(RMSE = rmses, lambdas = lambdas)
ggplot(df, aes(lambdas, rmses)) +
   theme_classic()  +
   geom_point() +
   labs(title = "RMSEs vs Lambdas - Regularized Movie Based Model",
        y = "RMSEs",
        x = "lambdas")
```




```
# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]
# Predict the RMSE on the validation set
rmse_reg_model_2 <- min(rmses)
# Adding the results to the results dataset
c_model<-"Regularized Movie-Based Model"
results <- results %>% add_row(model=c_model, RMSE=rmse_reg_model_2)
tibble(Method = c_model, RMSE = rmse_reg_model_2)
```

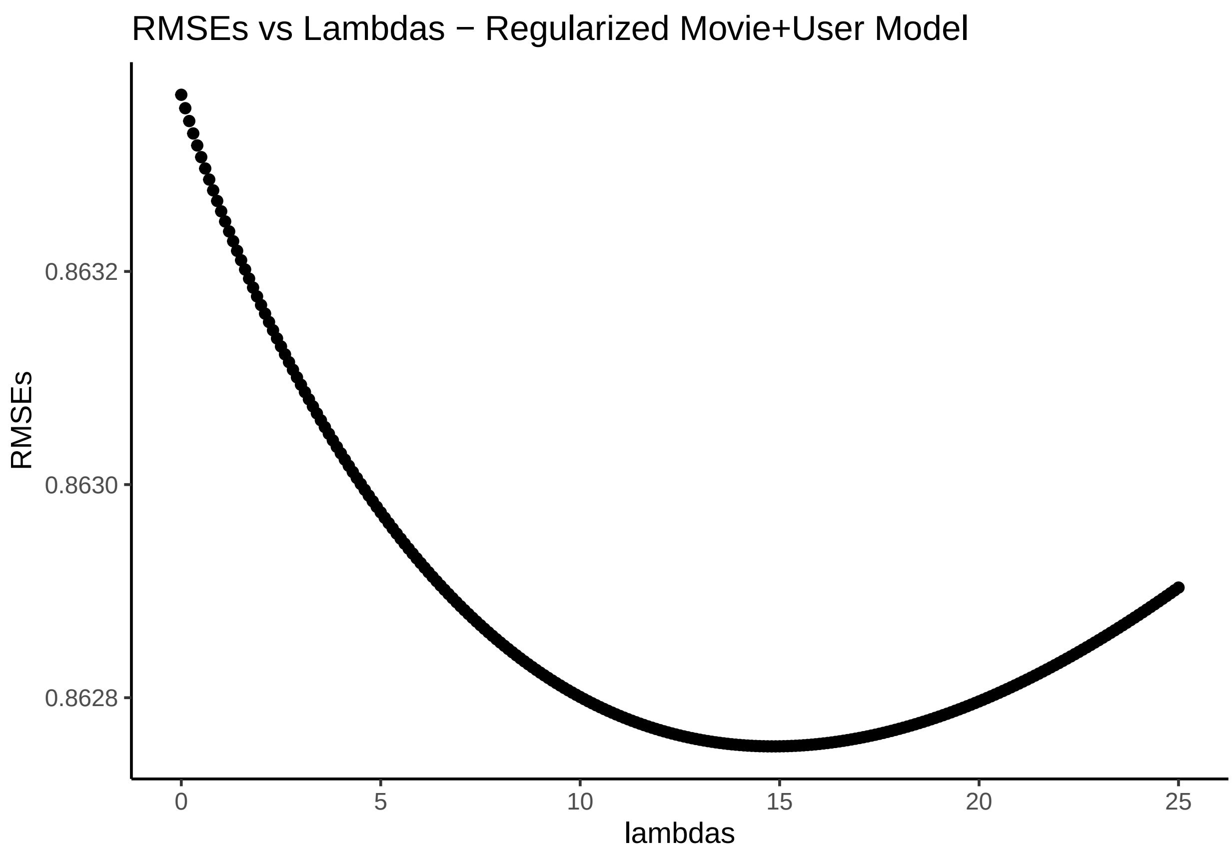
```
## # A tibble: 1 x 2
##   Method                         RMSE
##   <chr>                         <dbl>
## 1 Regularized Movie-Based Model 0.941
```

The RMSE on the **validation** dataset is **0.94103** and this is better than without Regularized. The Movie Based Model is not reaches yet the desired performance but applying the regularization techniques improve it just a little.

Regularized Movie+User Model

```
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Define a table of lambdas
lambdas <- seq(0, 25, 0.1)
# Compute the predicted ratings on validation dataset using different values of lambda
rmses <- sapply(lambdas, function(lambda) {
```

```r
  # Calculate the average by user

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  # Calculate the average by user

  b_u <- edx %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

  # Compute the predicted ratings on validation dataset

  predicted_ratings <- validation %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    mutate(pred = mu_hat + b_i + b_u) %>%
    pull(pred)

  # Predict the RMSE on the validation set

  return(RMSE(validation$rating, predicted_ratings))
})
# plot the result of lambdas
df <- data.frame(RMSE = rmses, lambdas = lambdas)
ggplot(df, aes(lambdas, rmses)) +
  theme_classic()  +
  geom_point() +
  labs(title = "RMSEs vs Lambdas - Regularized Movie+User Model",
       y = "RMSEs",
       x = "lambdas")
```




```r
# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]
# Predict the RMSE on the validation set
rmse_reg_model_3 <- min(rmses)
# Adding the results to the results dataset
c_model<-"Regularized Movie+User Based Model"
results <- results %>% add_row(model=c_model, RMSE=rmse_reg_model_3)
tibble(Method = c_model, RMSE = rmse_reg_model_3)
```

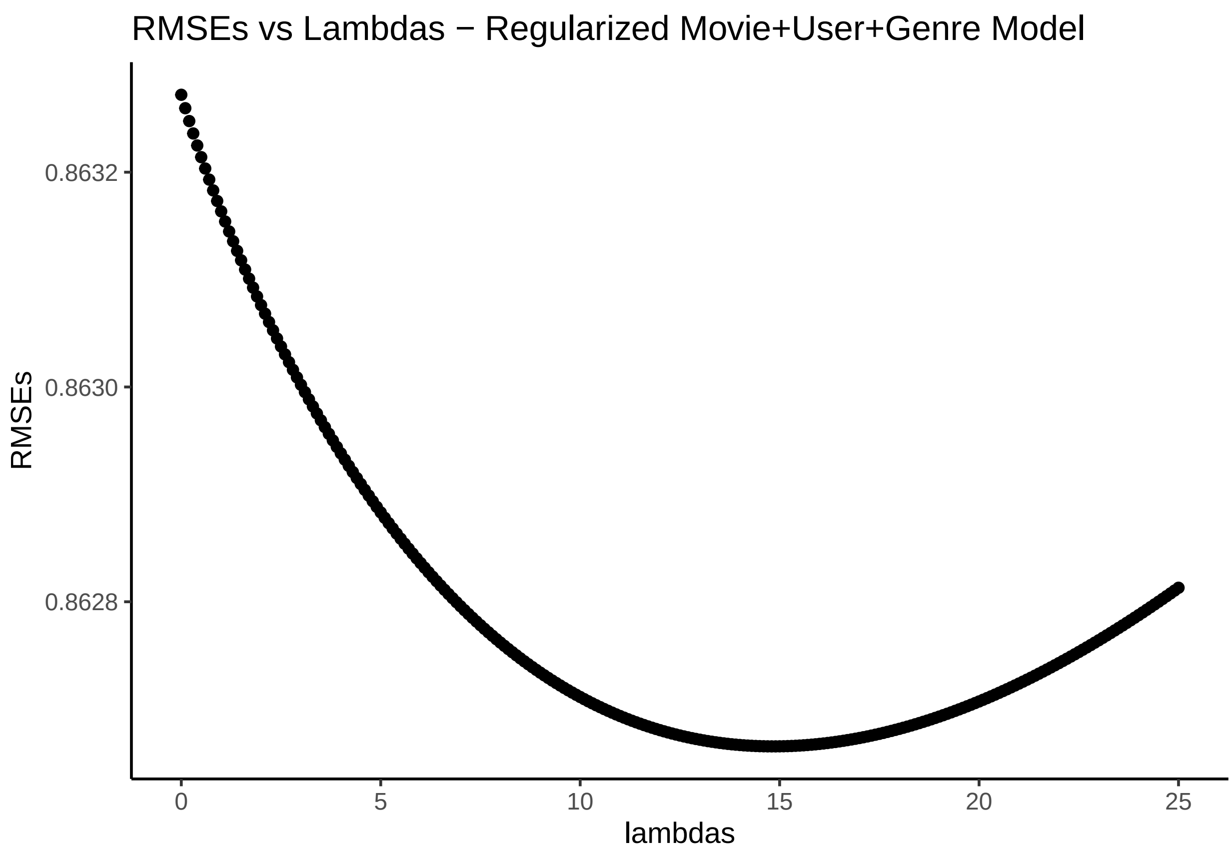
```
## # A tibble: 1 x 2
##   Method                              RMSE
##   <chr>                              <dbl>
## 1 Regularized Movie+User Based Model 0.863
```

The RMSE on the **validation** dataset is **0.86275**. The Regularized Movie+User Based Model improves just a little the result of the Non-Regularized Model.

Regularized Movie+User+Genre Model

```r
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Define a table of lambdas
lambdas <- seq(0, 25, 0.1)
# Compute the predicted ratings on validation dataset using different values of lambda
rmses <- sapply(lambdas, function(lambda) {

  # Calculate the average by user
```

```
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  # Calculate the average by user

  b_u <- edx %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

   b_u_g <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(genre) %>%
    summarize(b_u_g = sum(rating - b_i - mu_hat - b_u) / (n() + lambda))

  # Compute the predicted ratings on validation dataset

  predicted_ratings <- validation %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_u_g, by='genre') %>%
    mutate(pred = mu_hat + b_i + b_u + b_u_g) %>%
    pull(pred)

  # Predict the RMSE on the validation set

  return(RMSE(validation$rating, predicted_ratings))
})
# plot the result of lambdas
df <- data.frame(RMSE = rmses, lambdas = lambdas)
ggplot(df, aes(lambdas, rmses)) +
  theme_classic()  +
  geom_point() +
  labs(title = "RMSEs vs Lambdas - Regularized Movie+User+Genre Model",
       y = "RMSEs",
       x = "lambdas")
```




```r
# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]
# Predict the RMSE on the validation set
rmse_reg_model_4 <- min(rmses)
# Adding the results to the results dataset
c_model<-"Regularized Movie+User+Genre Based Model "
results <- results %>% add_row(model=c_model, RMSE=rmse_reg_model_4)
tibble(Method = c_model, RMSE = rmse_reg_model_4)
```

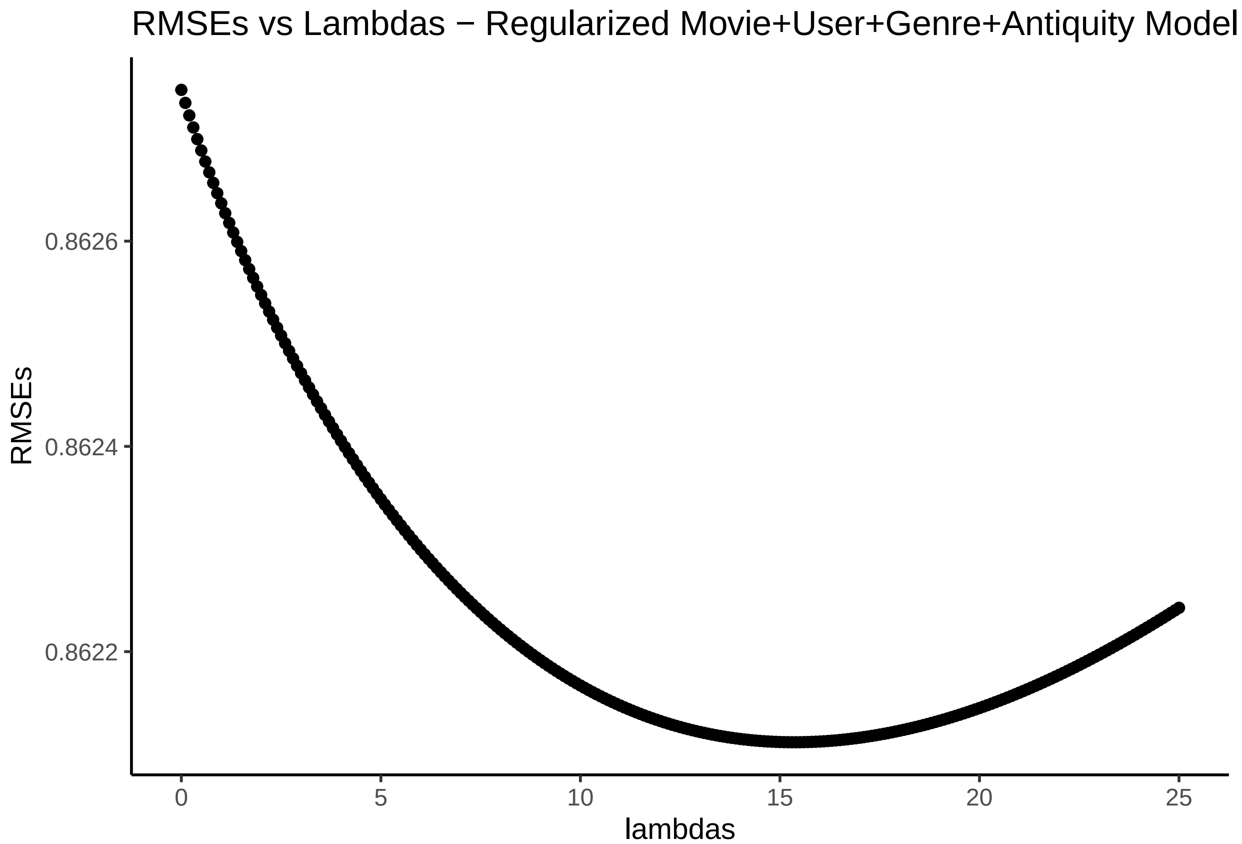
```
## # A tibble: 1 x 2
##   Method                                        RMSE
##   <chr>                                        <dbl>
## 1 "Regularized Movie+User+Genre Based Model " 0.863
```

The RMSE on the **validation** dataset is **0.86267** . The Regularized Movie+User+Genre Based Model improves just a little the result of the Non-Regularized Model. As the Non-Regularized Model, the `genre` predictor doesn't improve significantly the model's performance.

Regularized Movie+User+Genre+Antiquity Model

```r
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Define a table of lambdas
lambdas <- seq(0, 25, 0.1)
# Compute the predicted ratings on validation dataset using different values of lambda
rmses <- sapply(lambdas, function(lambda) {
   # Calculate the average by user
```

```r
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  # Calculate the average by user

  b_u <- edx %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

  b_u_g <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(genre) %>%
    summarize(b_u_g = sum(rating - b_i - mu_hat - b_u) / (n() + lambda))

  b_a <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_u_g, by='genre') %>%
    group_by(antiquity) %>%
    summarize(b_a = sum(rating - b_i - mu_hat - b_u - b_u_g ) / (n() + lambda))

  # Compute the predicted ratings on validation dataset

  predicted_ratings <- validation %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_u_g, by='genre') %>%
    left_join(b_a, by='antiquity') %>%
    mutate(pred = mu_hat + b_i + b_u + b_u_g + b_a) %>%
    pull(pred)

  # Predict the RMSE on the validation set

  return(RMSE(validation$rating, predicted_ratings))
})
# plot the result of lambdas
title_gg<-"RMSEs vs Lambdas - Regularized Movie+User+Genre+Antiquity Model"
df <- data.frame(RMSE = rmses, lambdas = lambdas)
ggplot(df, aes(lambdas, rmses)) +
  theme_classic()  +
  geom_point() +
  labs(title = title_gg,
       y = "RMSEs",
       x = "lambdas")
```




```
# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]
# Predict the RMSE on the validation set
rmse_reg_model_5 <- min(rmses)
# Adding the results to the results dataset
c_model<-"Regularized Movie+User+Genre+Antiquity Based Model"
results <- results %>%
  add_row(model=c_model,RMSE=rmse_reg_model_5)
tibble(Method = c_model, RMSE = rmse_reg_model_5)
```

```
## # A tibble: 1 x 2
##   Method                                              RMSE
##   <chr>                                              <dbl>
## 1 Regularized Movie+User+Genre+Antiquity Based Model 0.862
```

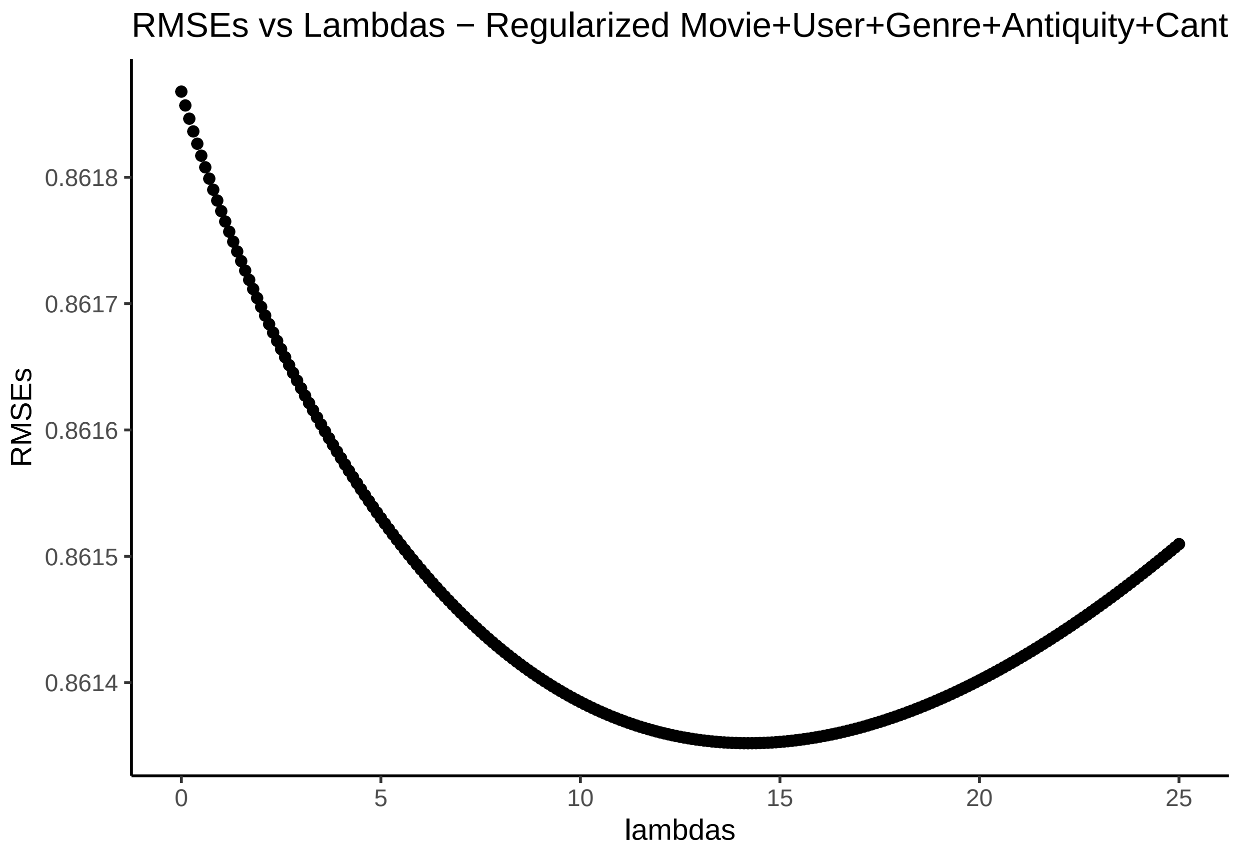
The RMSE on the **validation** dataset is **0.86211**. The Regularized Movie+User+Genre+Antiquity Base Model improves just a little the result of the Non-Regularized Model. As the Non-Regularized Model, the `Antiquity` predictor doesn't improve significantly the model's performance.

Regularized Movie+User+Genre+Antiquity+count Rating Model

```
# Calculate the average of all movies
mu_hat <- mean(edx$rating)
# Define a table of lambdas
lambdas <- seq(0, 25, 0.1)
# Compute the predicted ratings on validation dataset using different values of lambda
rmses <- sapply(lambdas, function(lambda) {
```

```r
  # Calculate the average by user

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  # Calculate the average by user

  b_u <- edx %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

  b_u_g <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(genre) %>%
    summarize(b_u_g = sum(rating - b_i - mu_hat - b_u) / (n() + lambda))

  b_a <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_u_g, by='genre') %>%
    group_by(antiquity) %>%
    summarize(b_a = sum(rating - b_i - mu_hat - b_u - b_u_g ) / (n() + lambda))

  b_c <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_u_g, by='genre') %>%
    left_join(b_a, by='antiquity') %>%
    group_by(cnt_ratings) %>%
    summarize(b_c = sum(rating - b_i - mu_hat - b_u - b_u_g - b_a) / (n() + lambda))

    # Compute the predicted ratings on validation dataset

  predicted_ratings <- validation %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_u_g, by='genre') %>%
    left_join(b_a, by='antiquity') %>%
    left_join(b_c, by='cnt_ratings') %>%
    mutate(pred = mu_hat + b_i + b_u + b_u_g + b_a + b_c) %>%
    pull(pred)

  # Predict the RMSE on the validation set

  return(RMSE(validation$rating, predicted_ratings))
})
# plot the result of lambdas
title_gg<-"RMSEs vs Lambdas - Regularized Movie+User+Genre+Antiquity+Cant Rating Model"
df <- data.frame(RMSE = rmses, lambdas = lambdas)
ggplot(df, aes(lambdas, rmses)) +
```

```
  theme_classic()  +
  geom_point() +
  labs(title = title_gg,
       y = "RMSEs",
       x = "lambdas")
```




```
# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]
# Predict the RMSE on the validation set
rmse_reg_model_6 <- min(rmses)
# Adding the results to the results dataset
c_model<-"Regularized Movie+User+Genre+Antiquity+Count Rating Based Model"
results <- results %>%
  add_row(model=c_model, RMSE=rmse_reg_model_6)
tibble(Method = c_model, RMSE = rmse_reg_model_6)
```

```
## # A tibble: 1 x 2
##   Method                                                          RMSE
##   <chr>                                                          <dbl>
## 1 Regularized Movie+User+Genre+Antiquity+Count Rating Based Model 0.861
```

The RMSE on the **validation** dataset is **0.86135** and this is the best result of the builted models. The Regularized Movie+User+Genre+Antiquity+count Rating Model improves just a little the result of the Non-Regularized Model. As in the Non-Regularized Model, the **count Rating** predictor doesn't improve significantly the model's performance.

Additional notes

The kable function was used at the begginig, but when pdf was generated it does work (using MikTex in Windows 10). weekday_Rate was not use in any model because that was indifferent. hour_Rate was not use in any model because that was indifferent. year_Rate was not use in any model because that was indifferent except 1995 when the avg was 4, but for future It will does not make sense.

Results

This is the summary results for all the model builted, trained on **edx** dataset and validated on the **validation** dataset.

```r
# Shows the results
results
```

```
##                                                          model      RMSE
## 1                                 Model 1: Naive Mean Baseline   1.0525533
## 2                                   Model 2: Movie-Based Model   0.9410652
## 3                              Model 3: Movie+User Based Model   0.8633658
## 4                         Model 4:Movie+User+Genre Based Model   0.8632721
## 5                       Movie+User+Genre+Antiquity Based Model   0.8627474
## 6  Model 6: Movie + User + Genre + Antiquity +Count Rating Model   0.8618678
## 7                                Regularized Movie-Based Model 0.9410328
## 8                            Regularized Movie+User Based Model 0.8627544
## 9                     Regularized Movie+User+Genre Based Model  0.8626654
## 10          Regularized Movie+User+Genre+Antiquity Based Model 0.8621117
## 11 Regularized Movie+User+Genre+Antiquity+Count Rating Based Model 0.8613521
```