Movies Recommendation Project

Moaz M. Salaheldeen

2023-11-19

Introduction

Data Set

This dataset lists movies ratings from MovieLens. Dataset is available to download in **grouplens** site for public.

The aim of this project is to explore this dataset and understand the affect of Users, Movies and other factors on the ratings.Then we will work on building a model that participate users' ratings for movies they didnt rate before.

To do so, we will do these steps:

- Data Importing, cleansing and Preparation
- Data Exploring and Visualization
- Modeling
- Results and Conclusions

Step 1: Data Importing, Cleansing and Preparation

The data set we are using contains about 10 Millions ratings. These ratings were made by 69878 user for 10677 movies. If every user rated every movie, this would give more than 746 Million records, so we have less than 10% of the possible records that can be generated by these numbers of users and movies.

Every record consists of 6 columns: UserID as integer, MovieID as integer, rating (from .5 to 5) as numeric, Timestamp as integer, Title as character, and Genres as character. Genres can have more than one value separated by "|" separator.

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

Data Preparation

10% of the dataset will be kept in *final_holdout_test* for the Evaluation of the final model. 20% of the remaining is considered to be the test set and the other 80% is the training set. While creating a testing set, we made sure there is no record in the test set unless it has reference in the training set by movie and user. This is due to a limitation that will be elaborated later that the models we are hoping to develop work only for existing users and movies.

Evaluation Criteria

The Evaluation function used to compare models is the standard RMSE: $RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(d_i - f_i)^2}$

Reference Model

As a reference model, we will consider the average ratings for all movies,

The Average in that case would be:

```
## [1] 3.512478
```

We will create a table to keep the RMSE for every model we will create, so we can compare them. Now we have only one model, which is the base:

```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average  1.06
```

Step 2: Data Visualization (& Insights)

We will study some factors in more details:

Users

Let's look at the number of movies rated by a single user. We will find the numbers ranges from 10 movies per user to 6616:

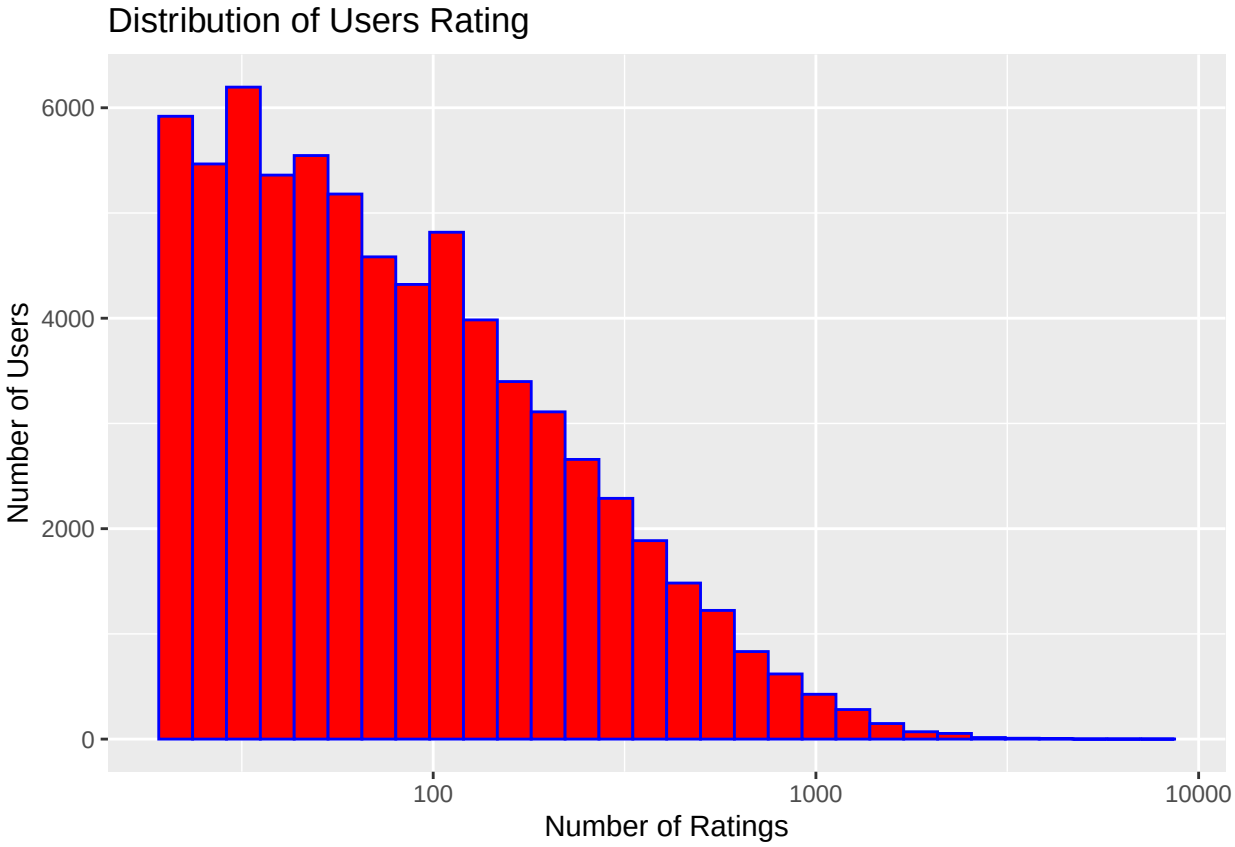
```
## # A tibble: 6 x 2
##   userId     n
##    <int> <int>
## 1      2    20
## 2     59    20
## 3     71    20
## 4     74    20
## 5    132    20
## 6    134    20
```

```
## # A tibble: 6 x 2
##   userId     n
##    <int> <int>
```

```
## 1  59269  7359
## 2  67385  7047
## 3  14463  5169
## 4  68259  4483
## 5  27468  4449
## 6   3817  4165
```

If we look at the distribution of ratings, we will find that most of the users rate less than 100 movies

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Movies

Let's look at the number of rates movies get from both ends..the movies getting the smallest number of rates and movies getting rated the most

```
## `summarise()` has grouped output by 'movieId'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 6 x 3
## # Groups:   movieId [6]
##   movieId title                                     n
##     <int> <chr>                                 <int>
## 1    3234 Train Ride to Hollywood (1978)            1
## 2    3356 Condo Painting (2000)                     1
```

```
## 3    3561 Stacy's Knights (1982)                                    1
## 4    3583 Black Tights (1-2-3-4 ou Les Collants noirs) (1960)       1
## 5    4071 Dog Run (1996)                                            1
## 6    4075 Monkey's Tale, A (Les Château des singes) (1999)          1
```

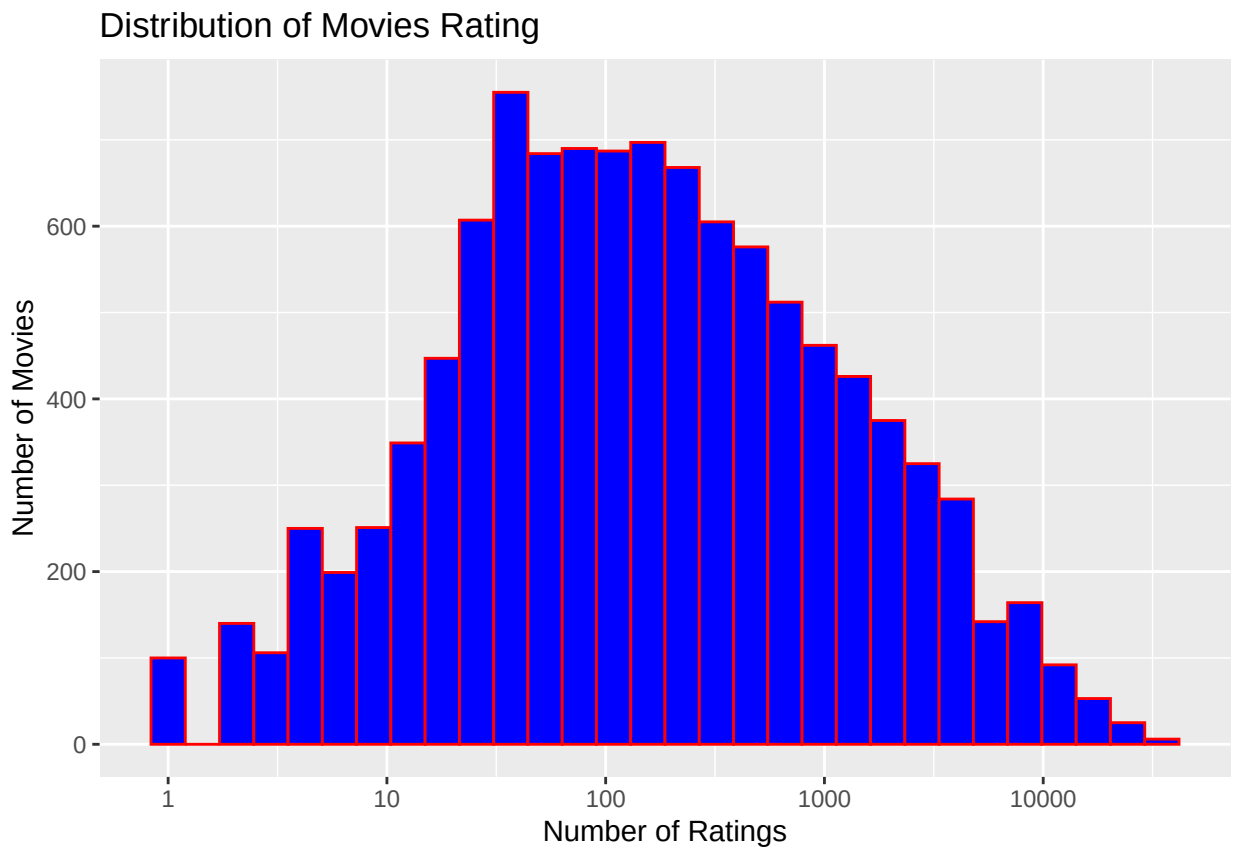
```
## `summarise()` has grouped output by 'movieId'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 6 x 3
## # Groups:   movieId [6]
##   movieId title                                n
##     <int> <chr>                            <int>
## 1     296 Pulp Fiction (1994)              34864
## 2     356 Forrest Gump (1994)              34457
## 3     593 Silence of the Lambs, The (1991) 33668
## 4     480 Jurassic Park (1993)             32631
## 5     318 Shawshank Redemption, The (1994) 31126
## 6     110 Braveheart (1995)                29154
```

As can be seen, some movies are rated only one time, while others are rated in ten thousands. *Pulp Fiction* and *Forrest Gump* are the two movies getting most rated, with number of rates 31362 and 31079 respectively.

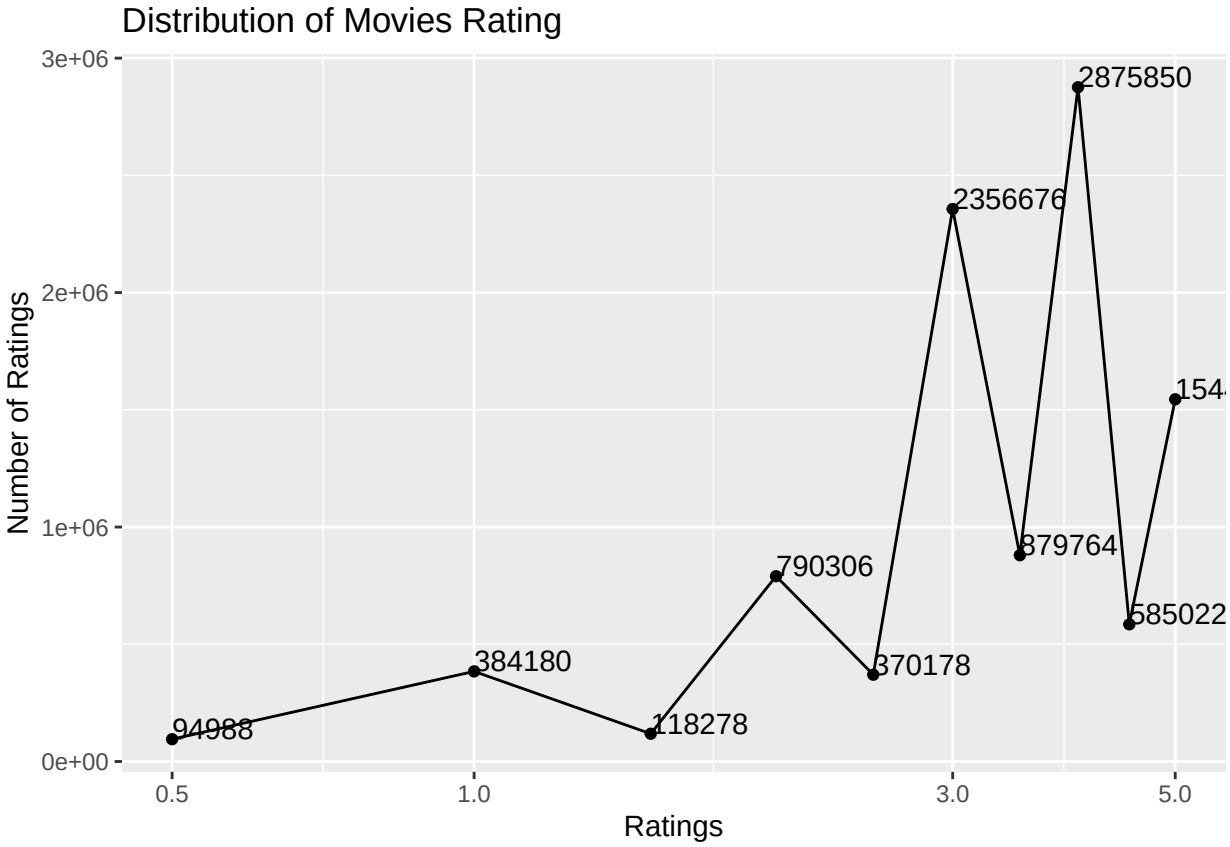
Looking at the distribution in the below diagram, we can see that most movies get rates around 50-500

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



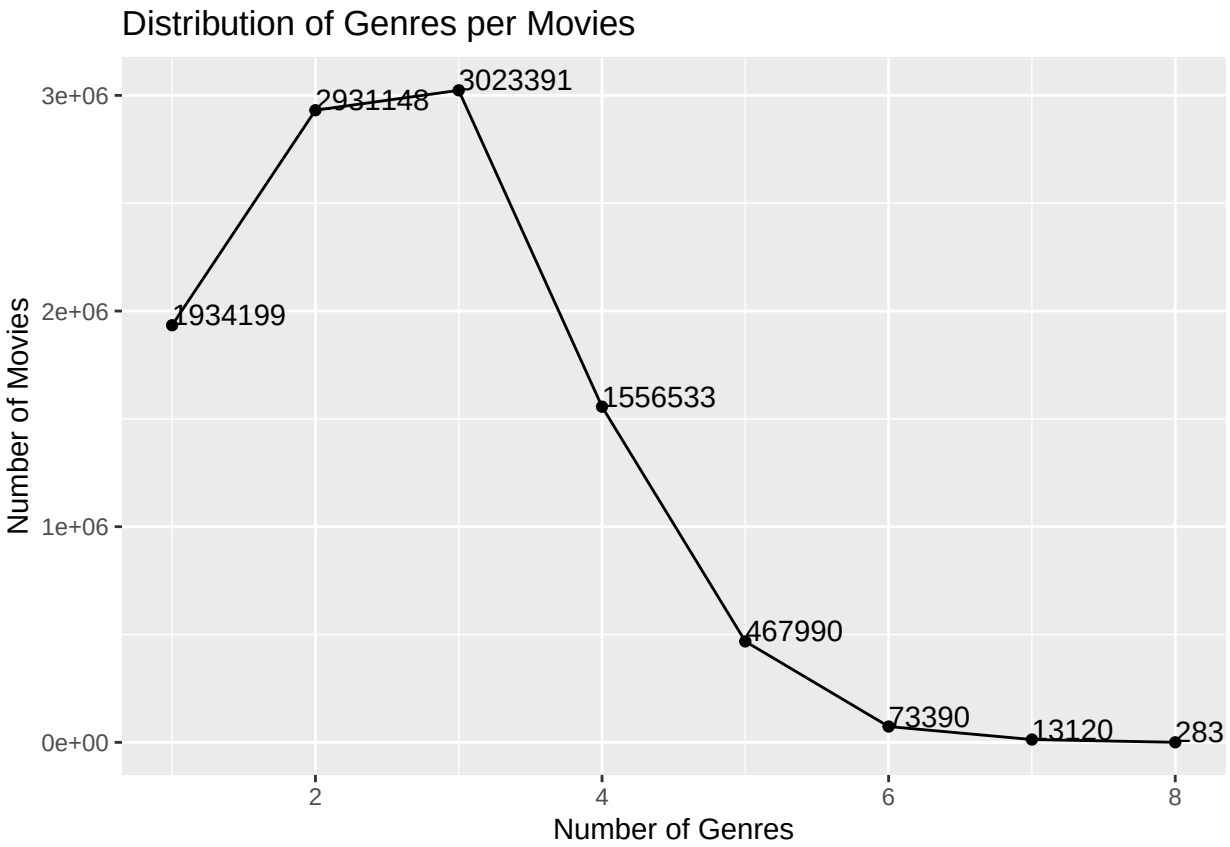
Ratings

First, we check that there is no missing Rating, and we found none. Then, we look at the rating distribution and find that 4 and 3 are the most given ratings by far than other ratings. 0.5 and 1.5 are the least given rates respectively.



Genres

Genres field has multiple values separated by a "|". We need to check how many genres are attached to movies in general. Vast majority of movies get three or less Genres.



Note that what we really count in genres is the separator. Therefore, if the count is 0, that means one Genre.

```
str_count(movielens$genres, fixed("|")
```

That's why we added One to the X-Axis Values.

Step 3: Modeling

First Model: Linear

We will work to build a linear model based on the User biase and Movie biase. The model is $Y_{u,m} = \mu + b_u + b_m + \epsilon_{u,m}$

First, we look at the movie biase:

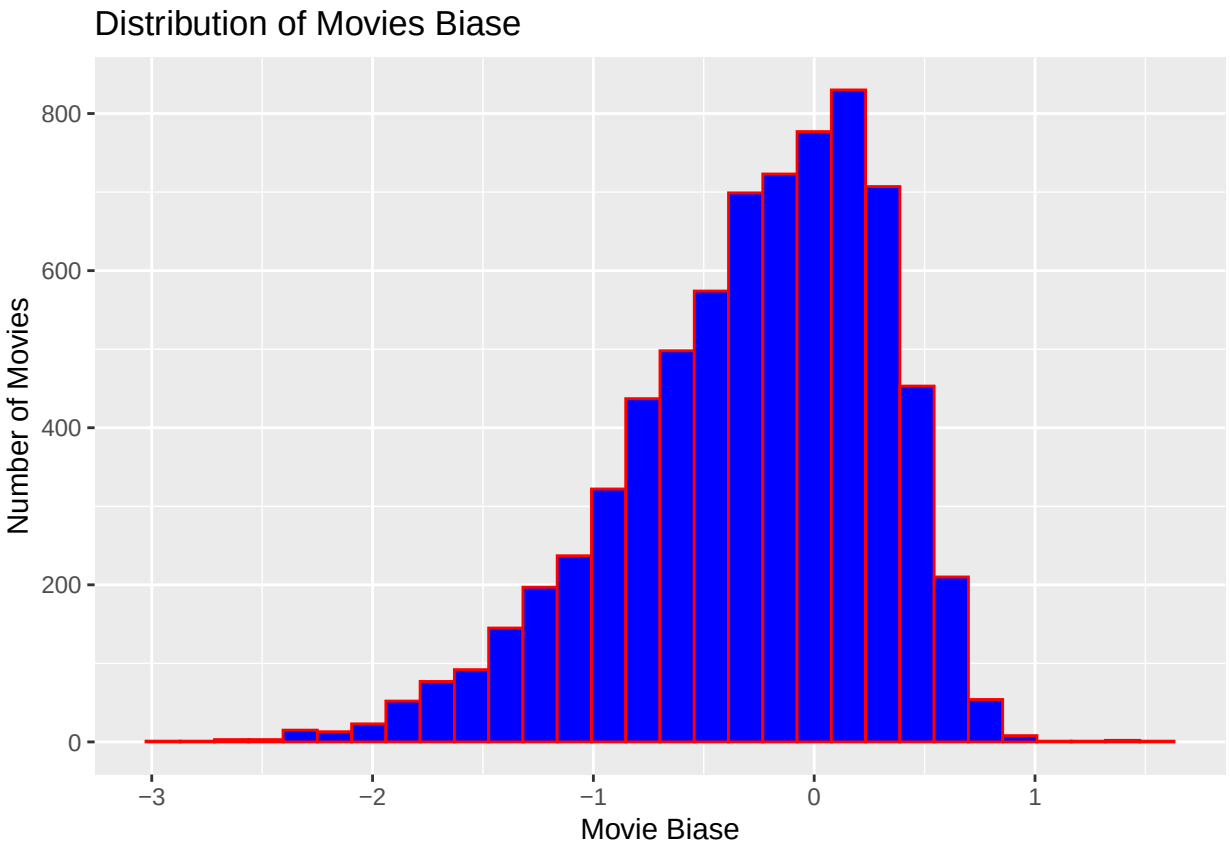
A.1 Add Movie Effect (Biase)

Due to the size of data and the available computing power, we couldn't run the *lm()* function, so we used an alternative method. We converted the Training data into matrix, where columns are MoviesIDs, Rows are UserIDs ,and the Values are the Ratings given by a user for a movie.

So, to get the average biase for a movie, we get the average of the ratings given to that movie (that column) and subtract the general average from it.

From the below diagram, we can see that the highest biases ranges from -0.5 to 0.5, with more tend to negative biasesr

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



By now, we have created this part of the model: $Y_{u,m} = \mu + b_m + \epsilon_{u,m}$ If we compare this to the Ratings in Testing Dataset, and apply the Evaluation function, we can see enhancement over the reference model:

```
## # A tibble: 2 x 2
##   method                   RMSE
##   <chr>                   <dbl>
## 1 Just the average        1.06
## 2 Considering Movie Biase 0.944
```

A.2 Add User Effect (Biase)

We apply the same concept above, to get the average biase for a user; we get the average of the ratings given to that user (row) and subtract the general average AND the movie biase from it.

From the below diagram, we can see that User biase is normal distributed around 0

```
gc()
```

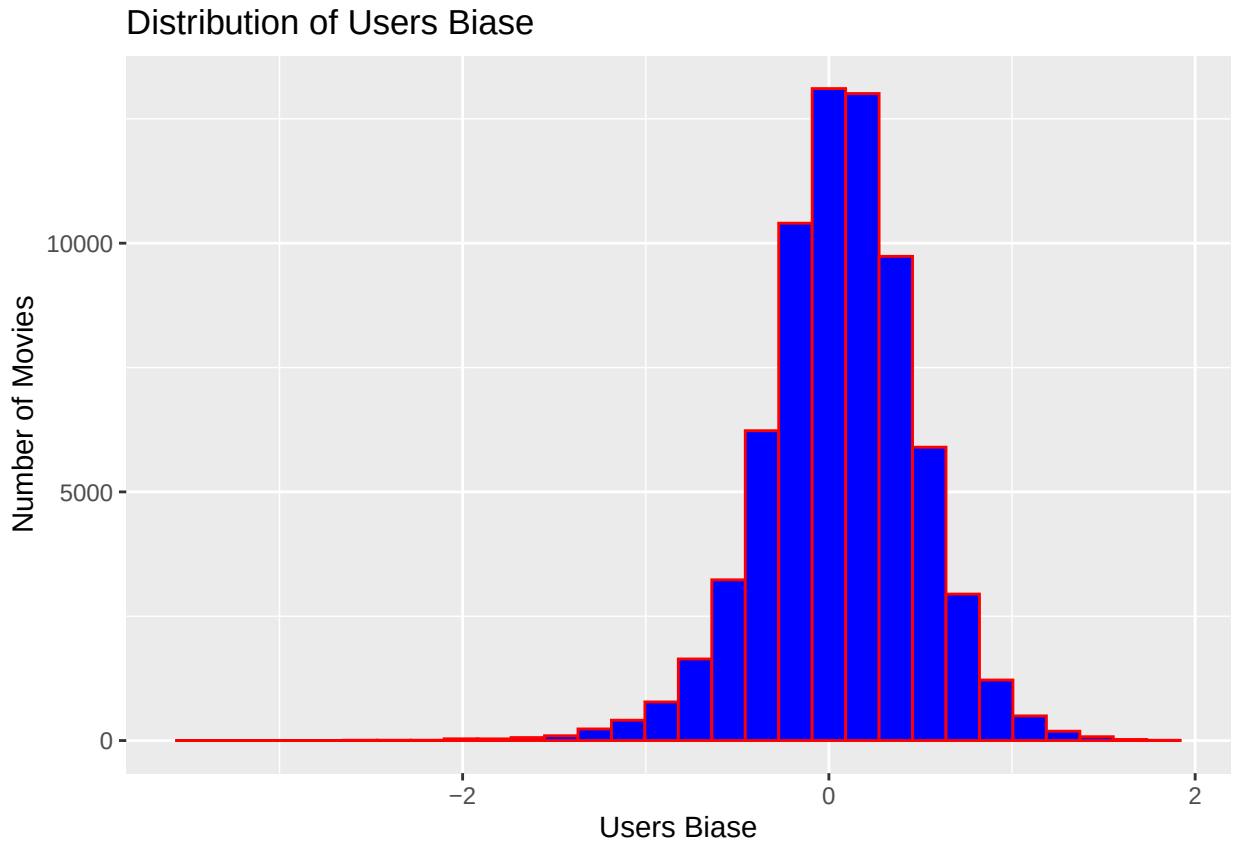
```
##             used   (Mb) gc trigger    (Mb)   max used    (Mb)
## Ncells  11667661  623.2   26431769  1411.7   29583762  1580.0
## Vcells 857671465 6543.6 2329661078 17774.0 1984523871 15140.8
```

```r
#b_u <- rowMeans(sweep(y_mat - mu, 2, b_m), na.rm = TRUE)     #MMS: Remove the Average and the Movie Bi

temp_Work_Data <- work_train_data |> group_by(movieId)  |>  summarize(b_m=mean(rating-mu))
work_train_data_Mod <- work_train_data|> left_join(temp_Work_Data, by='movieId')   |>   group_by(userId
b_u<- work_train_data_Mod$b_u
```

Note: In the above code I commented the statement used in the R file as it always gives *"Error: cannot allocate vector of size 5.6 Gb"*, and replaced with the code below it, that get the same result but consume less memory.

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



By now, we have created this part of the model: $Y_{u,m} = \mu + b_u + b_m + \epsilon_{u,m}$ If we compare this to the Ratings in Testing Dataset, and apply the Evaluation function, we can see enhancement over the reference model:

```
## # A tibble: 3 x 2
##   method                          RMSE
##   <chr>                          <dbl>
## 1 Just the average               1.06
## 2 Considering Movie Biase        0.944
## 3 Considering Movie & User Biase 0.866
```

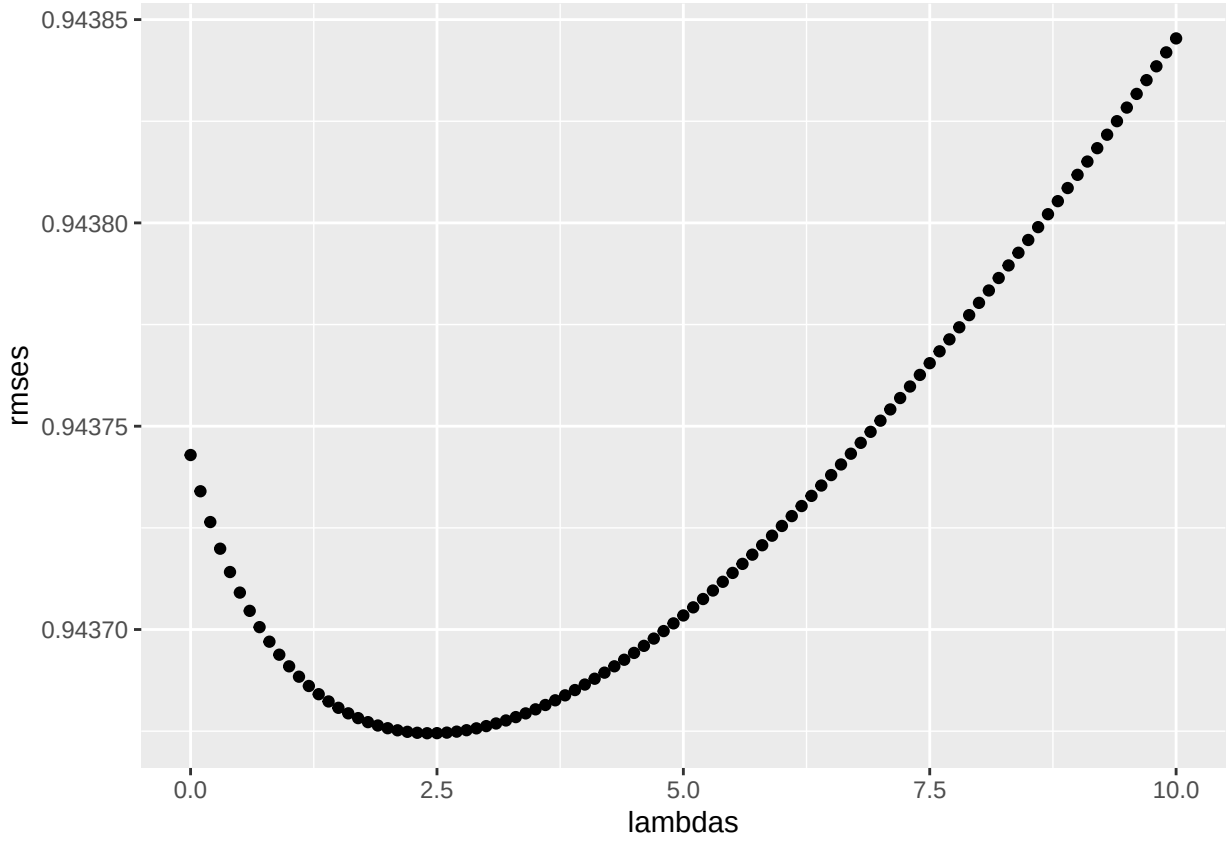
A.3 Enhance Movie Effect by considering Penalization Factor Before we finalize this model, we can do one enhancement to the Movie Biase. That is to consider the number of rates a movie get. So if a movie

is rated 4 in average by 50 users for example, we can get more confidence on that rating compared to a movie with the same rate but rated by one user only. Therefor, we will add λ factor that penalize movies based on number of rates they receive, based on this equation: $b(\lambda) = \frac{1}{\lambda + n_m} \sum(Y - \mu)$

To select λ value, we will use Cross Validation

From this diagram, we can get the λ value that minimize the RMSE

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
## [1] 2.4
```

That will reduce RMSE as in this table

```
## # A tibble: 4 x 2
##   method                                    RMSE
##   <chr>                                    <dbl>
## 1 Just the average                         1.06
## 2 Considering Movie Biase                  0.944
## 3 Considering Movie & User Biase           0.866
## 4 Considering Regularized Movie & User Biase 0.866
```

Second Model: Matrix Factorization

We will use "recosystem" package that is R wrapper for recommender system using matrix factorization First, we will need to put data in suitable format. This will be done using *data_memory* function, as we are loading data from the r object used before.

```
library(recosystem)
gc()
```

```
##             used   (Mb) gc trigger    (Mb)   max used    (Mb)
## Ncells  11848901  632.8   26431769  1411.7   29583762  1580.0
## Vcells 858240986 6547.9 2329661078 17774.0 2328799330 17767.4
```

```
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
#MMS: Put data in needed frormat
train_data <-  with(work_train_data, data_memory(user_index = userId,
                                                 item_index = movieId,
                                                 rating     = rating))

reco_obj <-  recosystem::Reco()
```

Then, we will select the best tuning parameters using Cross Validation. We will leave most parameters to their default values, but will change these parameters: *niter* : to reduce it from default value of 20 to 5 *nthread* : to increase it to allow for more parallel processing The main reason to change these parameters is to speed the processing of this function, as it takes a lot of time. On my laptop, it takes around 20 minuets in average (with nthread = 4, niter = 5)

```
opts <- reco_obj$tune(train_data, opts = list(dim = c(10, 20, 30),
                                              #    lrate = c(0.1, 0.2),
                                              #    costp_l2 = c(0.01, 0.1),
                                              #    costq_l2 = c(0.01, 0.1),
                                                  nthread  = 4, niter = 10))
```

After getting our tunned parameters, we will use them to train the model and try it on the Test Dataset, and turn the result back to normal Vector format in order to be able to run the Evaluation function.

The results is:

```
## # A tibble: 5 x 2
##   method                                   RMSE
##   <chr>                                   <dbl>
## 1 Just the average                        1.06
## 2 Considering Movie Biase                 0.944
## 3 Considering Movie & User Biase          0.866
## 4 Considering Regularized Movie & User Biase 0.866
## 5 Matrix Factorization                    0.800
```

Another Variation: Matrix Factorization on Residuals As a trial, I tried applying the same Matrix Factorization technique but on Residuals (After removing all known biases)

```
# compute residuals on train set

work_train_data2 <- work_train_data %>%
  left_join(fit_lm_m, by = "movieId") %>%
  left_join(fit_lm_m_u, by = "userId") %>%
  mutate(res = rating - mu - b_u - b_m_reg)

work_test_data2 <- work_test_data %>%
  left_join(fit_lm_m, by = "movieId") %>%
  left_join(fit_lm_m_u, by = "userId") %>%
  mutate(res = rating - mu - b_u - b_m_reg)
```

And then applied the same tuning and training functions

I added the Average μ to the resulted output before applying the Evaluation Function

```
y_hat_reco_2 <-  reco_obj2$predict(test_data2, out_memory())
y_hat_reco_2 <-  y_hat_reco_2 + mu
```

And this led to this result

```
## # A tibble: 6 x 2
##   method                                   RMSE
##   <chr>                                   <dbl>
## 1 Just the average                        1.06
## 2 Considering Movie Biase                 0.944
## 3 Considering Movie & User Biase          0.866
## 4 Considering Regularized Movie & User Biase 0.866
## 5 Matrix Factorization                    0.800
## 6 Matrix Factorization Residual           1.00
```

Final Model Although it is usually good practice to combine more than one algorithm together, but in that case, I'll work with one model, the best one we got -the one with the smallest RMSE- which is: **Matrix Factorization** model .
This is only to avoid the limited computational power I have which led R to crash in sometimes!

One Final Step to do add to the model is to eliminate any value outside the predefined borders, so any predicted rating with value less than 0.5 will be adjusted to 0.5, and any predicted rating with value more than 5, will be corrected as 5

So, final model is:

```
y_hat_reco <-  reco_obj$predict(test_data, out_memory())
y_hat_reco[y_hat_reco<.5]<-.5
y_hat_reco[y_hat_reco>5]<-5
```

Step 4: Results and Conclusions

Test Model on Final Testing Data Set

We start by transforming the *final_holdout_test* to suitable format and then apply the model directly, followed by the simple data cleansing for out-of-borders points.

Calculating the RMSE:

```
## [1] 0.7989082
```

```
##    userId movieId rating  timestamp                                      title
## 1    7873    1979    5.0  950678331 Friday the 13th Part VI: Jason Lives (1986)
## 2   11760   25963    0.5 1192578815                       Olvidados, Los (1950)
## 3   32463    6524    0.5 1069282152  Never on Sunday (Pote tin Kyriaki) (1960)
## 4   51578     858    0.5 1153766564                      Godfather, The (1972)
## 5   20931      36    0.5 1076727246                   Dead Man Walking (1995)
## 6   27297    2011    0.5 1143034612          Back to the Future Part II (1989)
## 7   68881    1238    5.0  945557994                           Local Hero (1983)
## 8   47110    2140    0.5 1151745377                  Dark Crystal, The (1982)
## 9   46200    2791    0.5 1227241704                        Airplane! (1980)
## 10  18278    8973    0.5 1208468473   Bad Education (La Mala educación) (2004)
##                            genres   predict
## 1                          Horror 0.5000000
## 2                   Crime|Drama 5.0000000
## 3                  Comedy|Drama 5.0000000
## 4                   Crime|Drama 4.8210483
## 5                   Crime|Drama 4.7880487
## 6  Action|Adventure|Comedy|Sci-Fi 4.7473369
## 7                          Comedy 0.7618486
## 8      Adventure|Children|Fantasy 4.7015190
## 9                          Comedy 4.6978946
## 10                Drama|Thriller 4.6833220
```

```
##    userId movieId rating  timestamp
## 1       1     480      5  838983653
## 2      47    2571      5 1162149627
## 3      56     923      5 1162155095
## 4      56    2186      5 1162155400
## 5      56    3435      5 1162155378
## 6      56    7091      5 1162156982
## 7      56    8235      5 1162156036
## 8     104     750      5  945881102
## 9     104     924      5  945879889
## 10    104    1252      5  945878857
##                                                                      title
## 1                                                        Jurassic Park (1993)
## 2                                                          Matrix, The (1999)
## 3                                                         Citizen Kane (1941)
## 4                                                  Strangers on a Train (1951)
## 5                                                      Double Indemnity (1944)
## 6                                                        Horse Feathers (1932)
## 7                                                          Safety Last! (1923)
## 8  Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
## 9                                                  2001: A Space Odyssey (1968)
## 10                                                            Chinatown (1974)
##                              genres predict
## 1  Action|Adventure|Sci-Fi|Thriller       5
## 2            Action|Sci-Fi|Thriller       5
## 3                     Drama|Mystery       5
```

```
## 4    Crime|Drama|Film-Noir|Thriller       5
## 5              Crime|Drama|Film-Noir       5
## 6                             Comedy       5
## 7              Action|Comedy|Romance       5
## 8                         Comedy|War       5
## 9                    Adventure|Sci-Fi       5
## 10 Crime|Film-Noir|Mystery|Thriller       5
```

Conclusion

We tried different models. Linear model based on calculating the biases of Users and Movies. The other one was based on Matrix Factorization, and then tried Matrix Factorization on Residuals.

There are other algorithms that should be tested, like KNN model and Random Forest. Also, we didn't depend on Genres on any of these algorithms, despite the fact that it groups similar movies together and it is expected to really enhance the model. But again the size of the data combined with the limited CPU power limited this work on this phase. One important limitation to note is that this algorithm cant predict a new user or a new movie. I expect if we include Genres (and maybe release date), we can overcome this limitation to some extent.

Also, we shall dig more on the movies that got the largest difference between rating and prediction. We may figure out a pattern for these movies (Like low number of ratings or certain genre)