# **Movie Recommendation System**

**By Sam Vuong, Raymond (shanhua) Huang, Carmon Ho, Kyle Murphy**

**Abstract:** the goal of this project is to establish a movie recommending application. The success of this project will be determined by the accuracy of which the system recommends movies with similar ratings or themes. This model could be used by movie websites to drive users for further movies.

# **1.0 Introduction and Discussion**

The Internet plays an important role in people's life in the current world. But it comes with problems as people were provided with many choices to select which made them confused and lost. Thus, a recommendation system comes for a rescue to them. Companies such as Youtube provide users with collaborative filtering which filter out items for similar users with relevant interest in video.

The recommender systems could be classified into 2 broad categories:

1. Collaborative filtering approach
2. Content-based filtering approach

## **1.1 Collaborative filtering approach**

Collaborative filtering system presents recommendations according to relevant measures between users or items. This system recommends items that are favored by similar kinds of users. Its advantage lies in the fact that serendipitous recommendations were provided based on users’ similarity rather than item’s similarity. What’s more, the real quality assessment of items is done through explicit ratings made by users. It mainly relies on connection between users.

## **1.2 Content-based filtering approach**

In terms of content-based filtering, the users’ preference and the items’ description contribute to the success of this kind of filtering. In other words, the algorithms used by content-based filtering present items with similar descriptions or similar items that were favored in the past.

## **1.3 Present work**

Many recommendation systems have been introduced in the past few decades. Approaches such as collaborative approach, content-based approach, a utility base approach, hybrid approach etc, have been tested and examined in the past. In this project, we proposed a movie recommendation system which will focus on a combination of content-based and collaborative approach.

## **1.4 Dataset**

The data has been collected by GroupLens Research and made available rating data sets from the MovieLens web site ([http://movielens.org](http://movielens.org/)). The data was generated over different periods of time.

In addition, the data sets were divided into different functions such as for education, development and research.

In this project, we use the latest small dataset with 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. This dataset was last updated in September 2018.

The dataset can be found at: <http://files.grouplens.org/datasets/movielens/ml-latest-small.zip>

The selected data include 4 CSV files:

* movies.csv - this data file contains movies information [movieId, title, genres]
* ratings.csv - this data file contains movie ratings info [userId, movieId, rating, timestamp]
* tags.csv - this data file contains user movie-tags [userId, movieId, tag, timestamp]
* links.csv - this data file contains movie databases links info [movieId, imdbId, tmdbId]

Due to time constraints in this project, our recommendation system only uses the rating information in the “ratings” data file. It does not use the tag information in the tags data file.

## **1.5 Ethical ML Framework**

No movie is similar to the next one being produced as there are many factors that won’t be the same, like cast, storyline, theme and other movie related variables. Will people give similar ratings if they enjoyed the first movie they watched compared to a movie that is similar in genre. The dataset is open source and can only assume that it is transparent without biased opinions from their families, friends or even in today’s world of Social Media. With privacy hidden in all datasets, no analysis can be grouped with respect to people’s age and/or gender with their preference of movie type.

## **1.6 Assumptions**

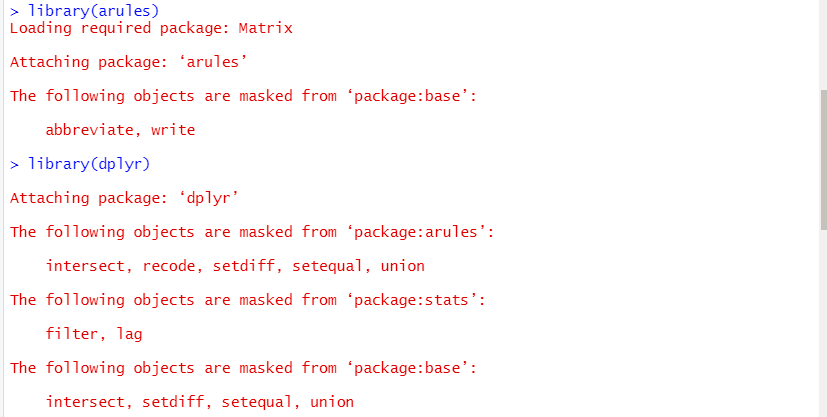
We would assume that the data gathered comes from regions all across the world with various ethinic backgrounds. It does state users were chosen for this data did rate at least 20 movies and hid their demographics.

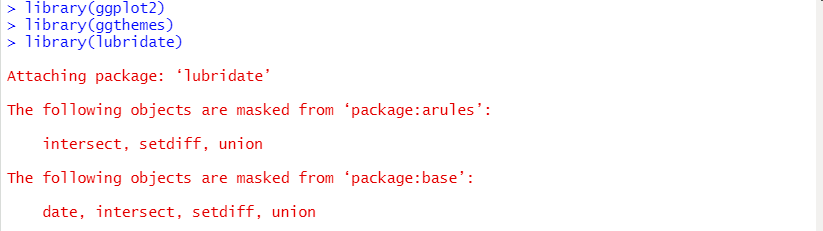
# **2.0 Data Preparation**

## **2.1 Loading R Libraries**

We load the following libraries which are required for our data analysis, visualization and modeling:

* library(arules)
* library(dplyr)
* library(ggplot2)
* library(ggthemes)
* library(lubridate)
* library(reshape2)
* library(Matrix)
* library(recommenderlab)
* library(scales)
* library(stringr)
* library(stringdist)
* library(wordcloud)







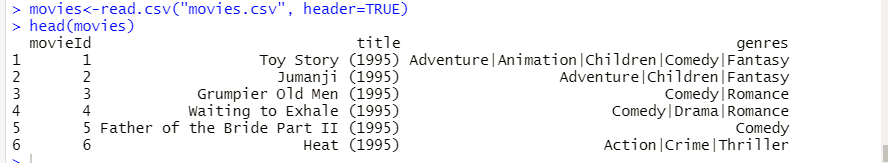
## 

## **2.2 Loading Data**

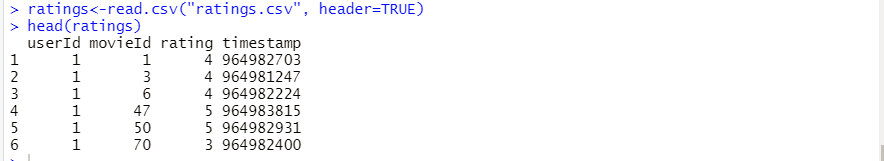
We read input data files and perform basic checks on data.

**Read input CSV data files**

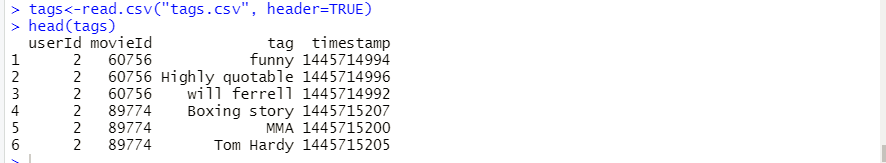
**1. Read movies file** and look at first few lines:



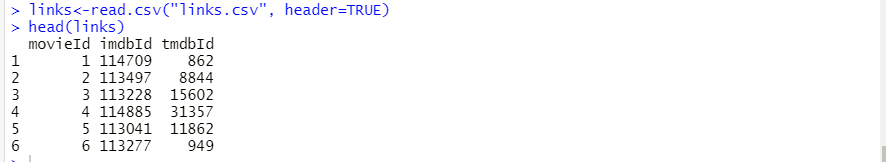
**2. Read ratings file** and look at first few lines:



**3. Read tags file** and look at first few lines:



**4. Read links file** and look at first few lines:

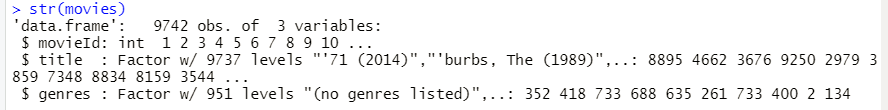


# 

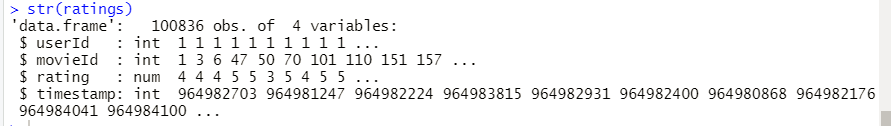
# **3.0 Data Exploration Analysis**

**Checking data types of data**

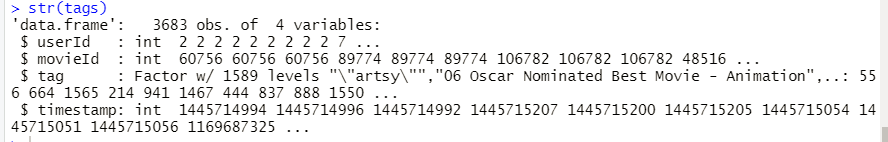
Check movies data:

****

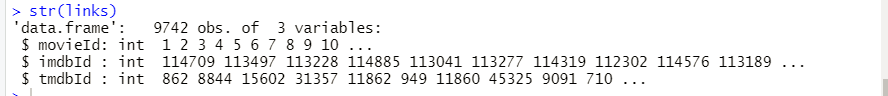
Check ratings data:

****

Check tags data:



Check links data:



**Checking the number of movies and users**

There are 9,724 movies and 610 users.

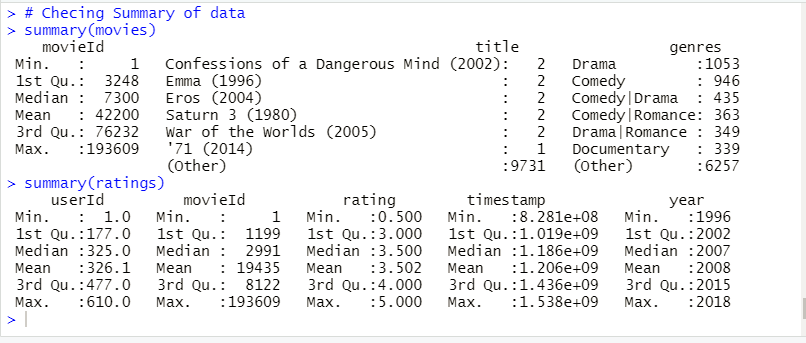
****

**Checking summary of data**

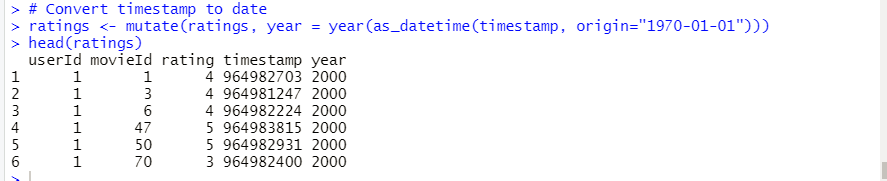
Below is the summary of data for “movies” and “ratings” data.

The summary data shows:

* User IDs range between 1 and 610.
* Rating ranges from 0.5 to 5.0 and has an averaging value of 3.5.
* Rating data were between 1996 and 2018, which about 25 years range.

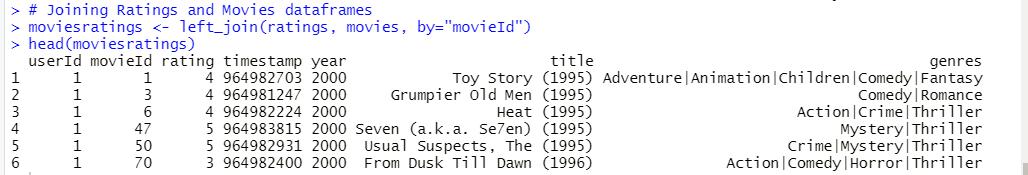


**Convert timestamp to date**



**Joining Ratings and Movies dataframes**

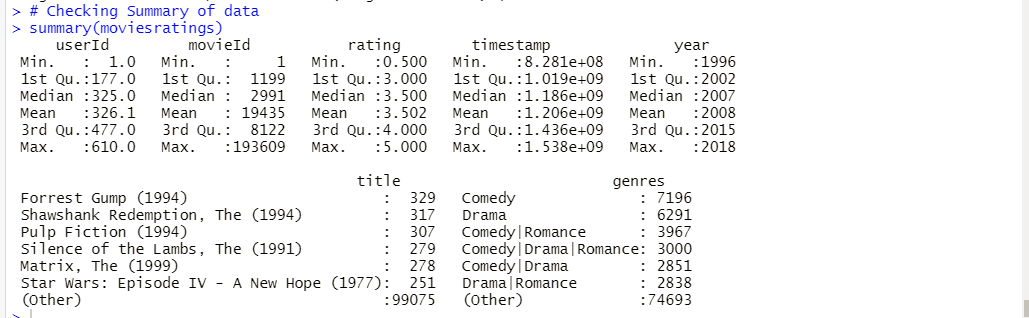
We use left join to join the “ratings” and “movies” dataframes. Left join will return all of the rows from the “ratings” dataframe and the matching rows from the “movies” dataframe. Left join will not return values from the “movies” dataframe which do not exist in the “ratings” dataframe.



We double check the number of movies in the new dataframe. In our case, there are no “movie id’s” that exist in the “ratings” dataframe but not in the “movies” dataframe.

****

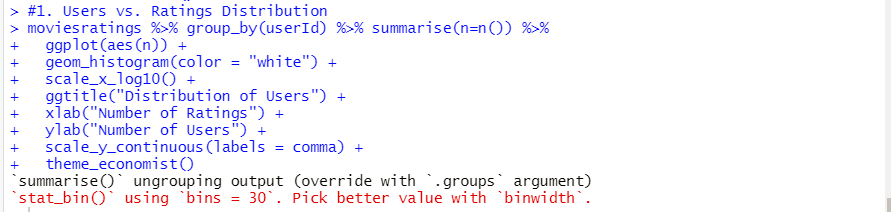
Checking summary of data new dataframe

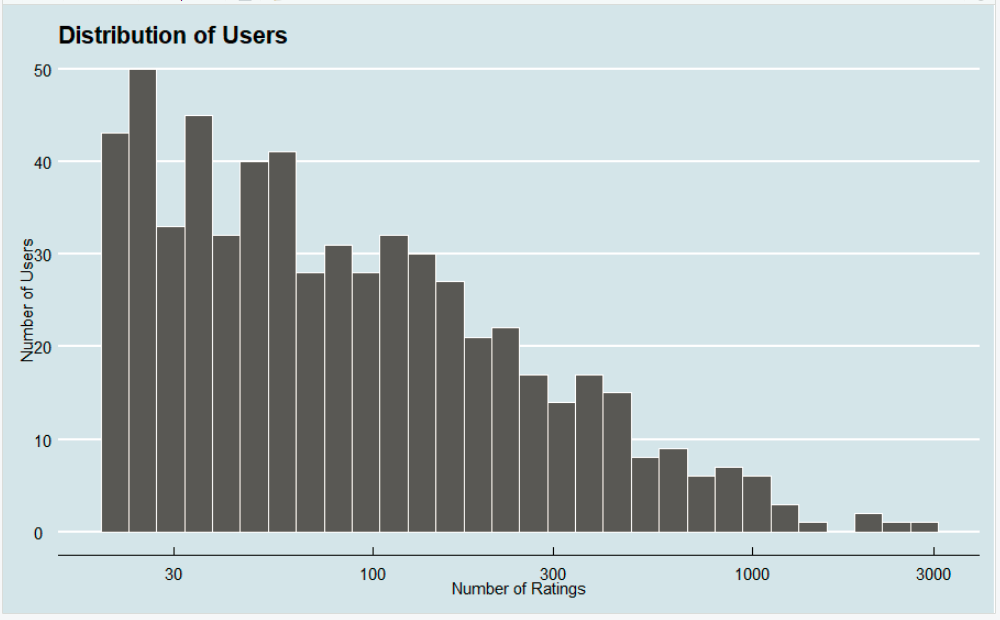
****

# **4.0 Data Visualization**

## **4.1 User vs. Rating Distribution bar graph**

Distribution of user vs. ratings is presented using code the below. Most of the users gave 100 ratings.

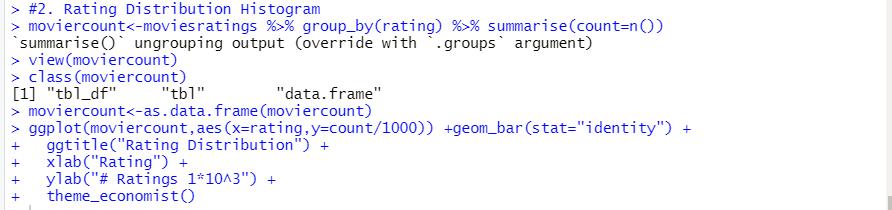


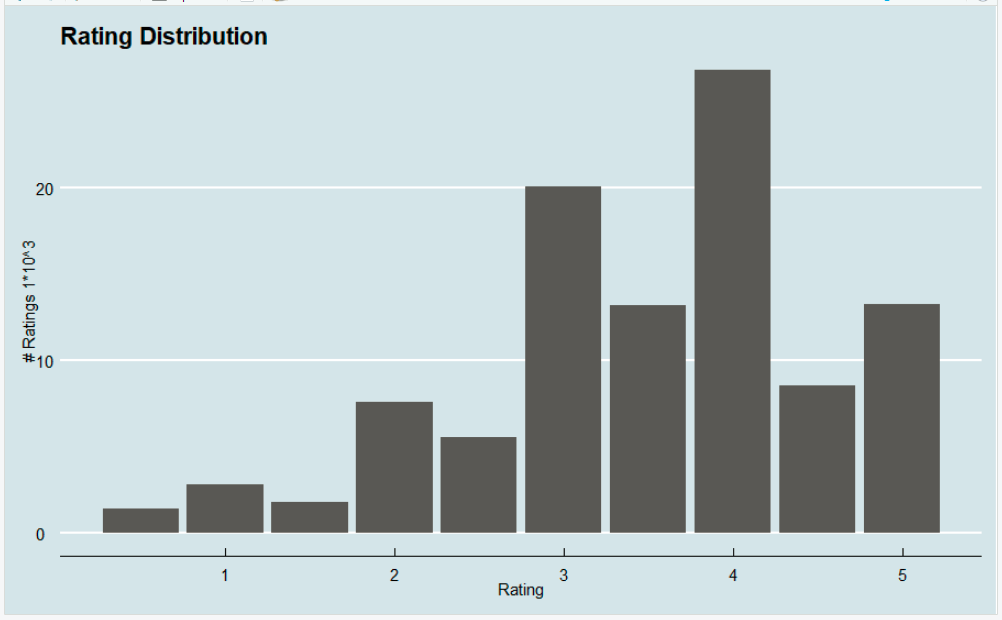


## Based on the visualization above, the data suggests that the majority of users will rate few movies, while very few rate more than a thousand movies. Less than users rated movies between 1000 and 3000 times. The distribution of users and total number of ratings could be considered a right-skewed or positive-skew distribution. The average number ratings per user is located to the right of the distribution peak.

## **4.2 Rating Distribution Histogram chart**

Below is the chart for Rating distribution. It appears that users tend to favor rating movies 3 and 4.

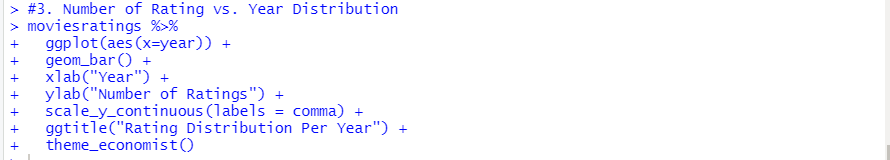


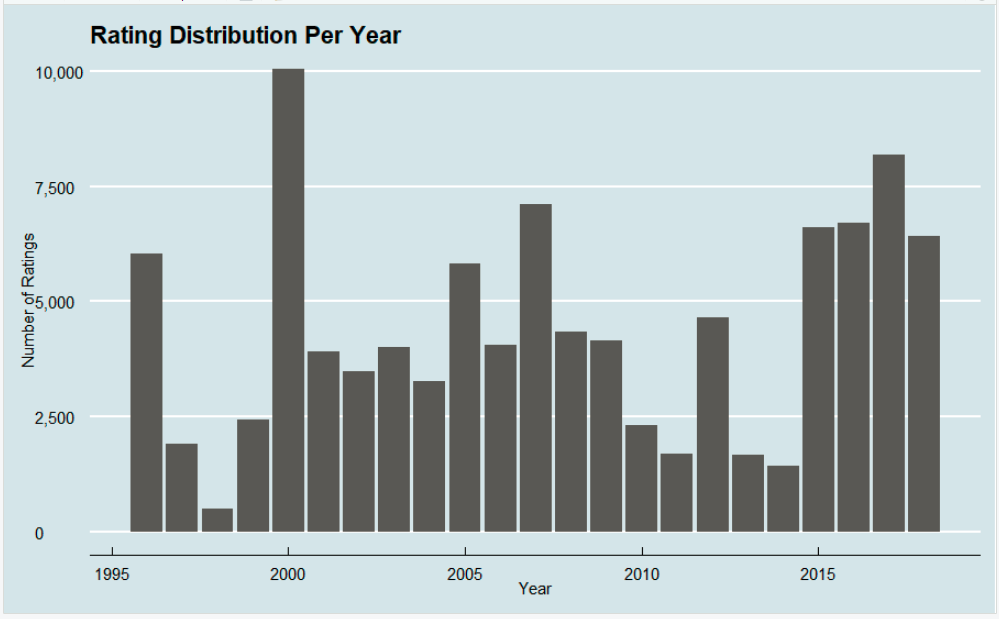


Based on the visualization above, the data suggests that users tend to favor ratings ‘3’ and ‘4’ when rating movies. The average rating per movie will appear to be higher in this dataset as users tend to choose medium to high ratings for movies.

## **4.3 Rating vs. Year Distribution bar graph**

Distribution of rating vs. Year is presented using the code below. In general, more users were found to provide ratings in the last few years in comparison to the past.

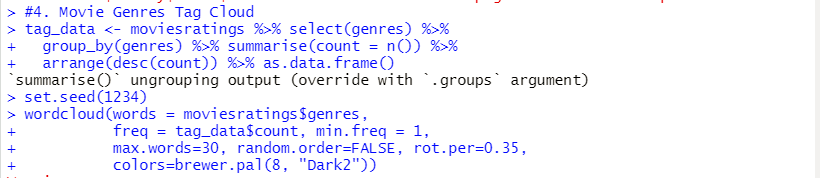




## Based on the visualization above, the data suggests that users rated more movies in 1996, 2000, 2005, 2007, and 2017. These years could have been years where more major blockbusters were released.

## **4.4 Movie Genres Tag Cloud**

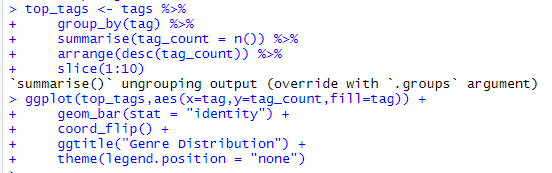
The Tag Cloud below shows the top 30 watched movie genres. The Tag Cloud shows most watched movie genres are Action|Crime|Thriller, Action|Drama|War, Action|Drama|Romance, Comedy|War and Comedy.

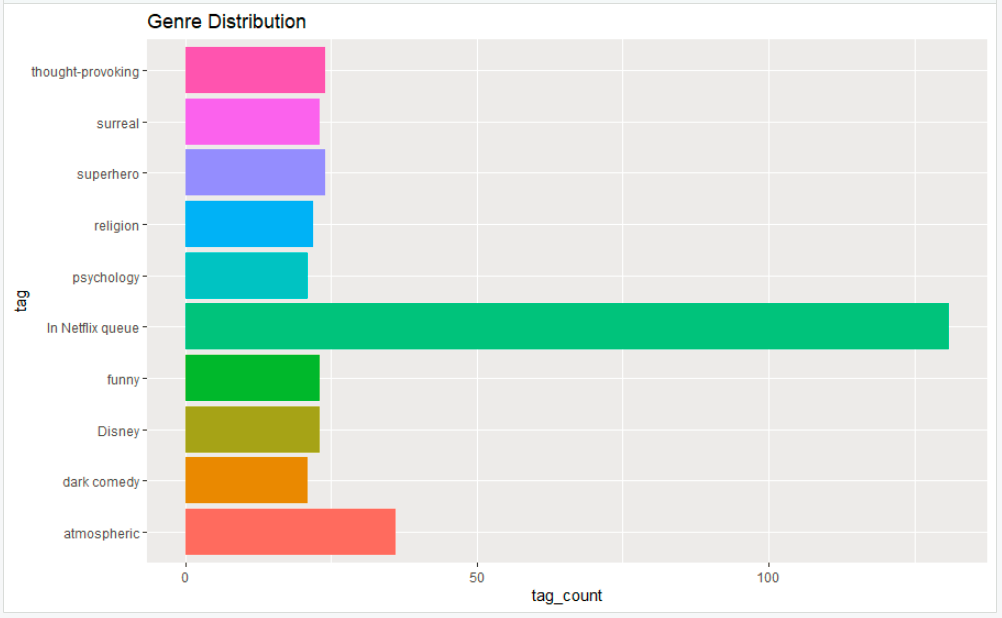


****

## **4.5 Horizontal Bar chart for Top 10 Tags for movies**

Bar chart showed that in Netflix queue and atmospheric are the top 2 tags.

****

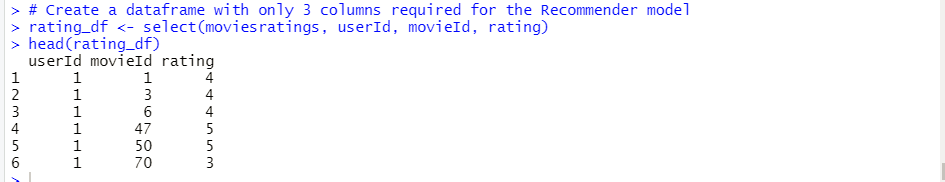
****

Based on the visualization above, ‘In Netflix queue’ appears to be the most used ‘tag’ in our dataset. This data may suggest that users involved in the data collection prefer to organize the movies they want to watch in the future by using the ‘queue’ feature on the Netflix video streaming platform. This pattern aligns seamlessly with society’s evolution of video streaming and digital content delivery.

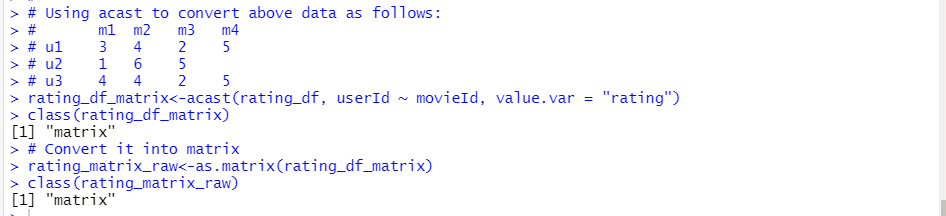
# **5.0 Modeling**

## **5.1 Data Preparation for Modeling**

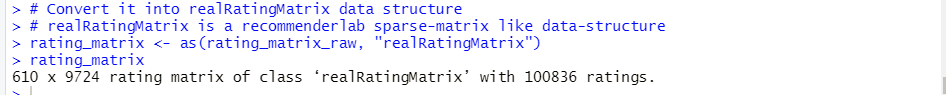
**Step-1**: We create a dataframe with only the 3 columns required for the Recommender model.



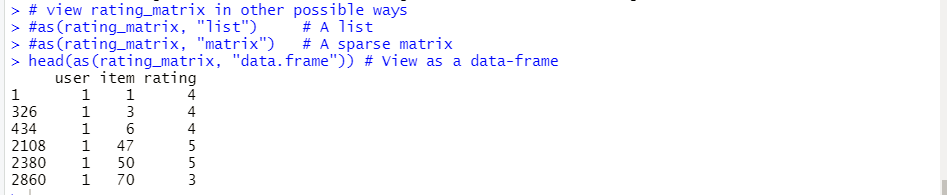
**Step-2**: We convert it into a matrix.

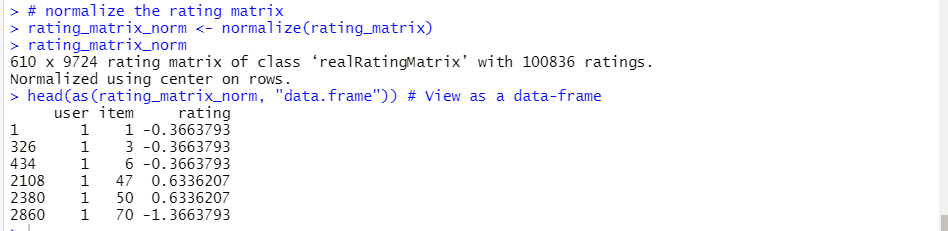


**Step-3**: We convert it into a “realRatingMatrix” data structure - a structure required by a Recommenderlab library.

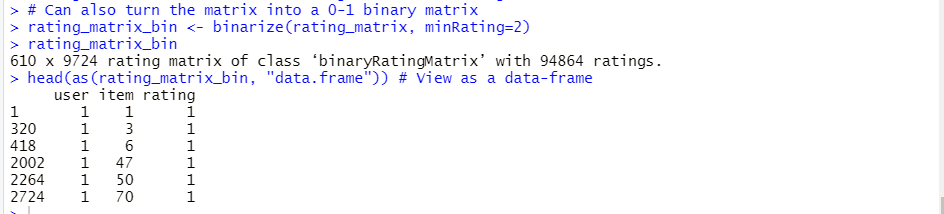


We view the rating matrix as a dataframe or a list:

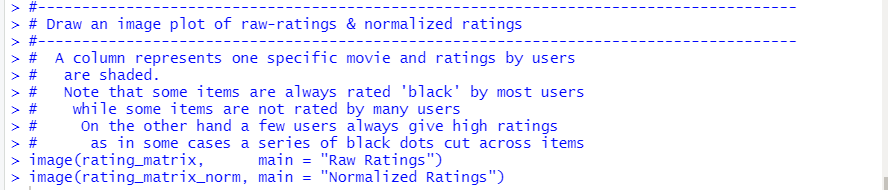


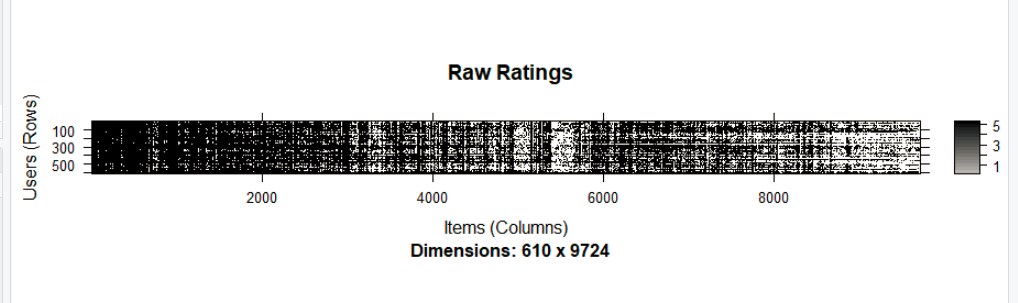
We can normalize the rating matrix using the normalize() function:

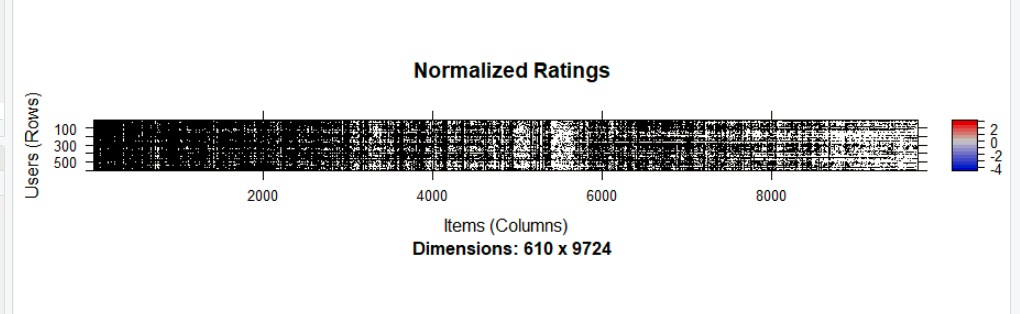
We can also turn the matrix into a 0-1 binary matrix using the binarize() function:



We can draw an image plot of raw-ratings matrix:





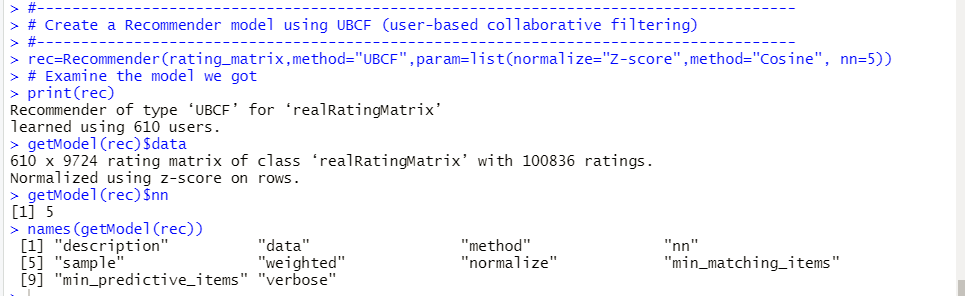


## 

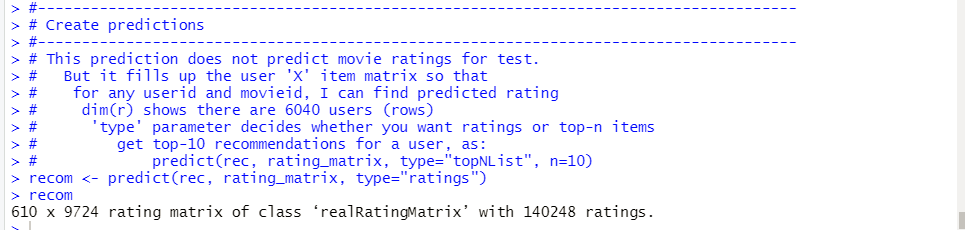
## **5.2 Recommender Model**

**Step-1**: We create a Recommender model using **User-Based Collaborative Filtering (UBCF)** with the following parameters:

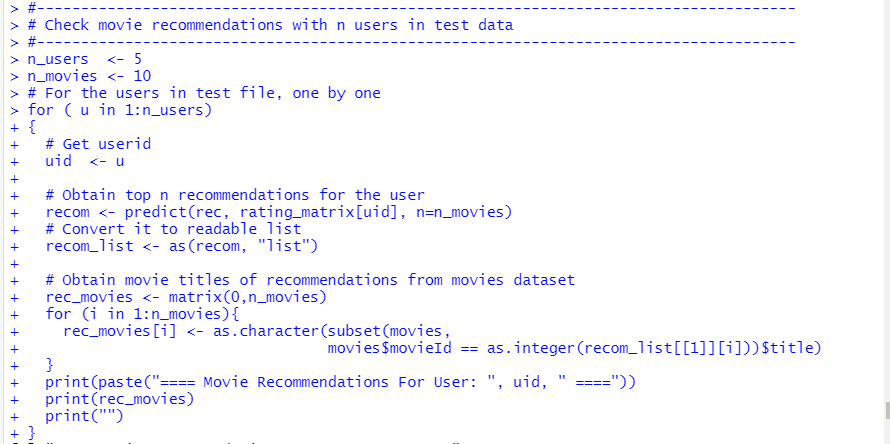
* normalize="Z-score" ## we use Z-score" as the normalize function
* method="Cosine" ## we use Cosine as similarity function
* nn=5 ## we use 5 as number of similar users



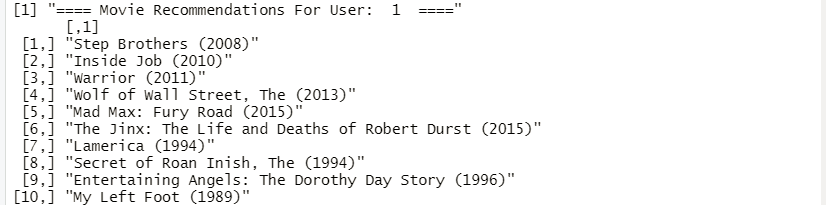
**Step-2**: We create the Prediction matrix:



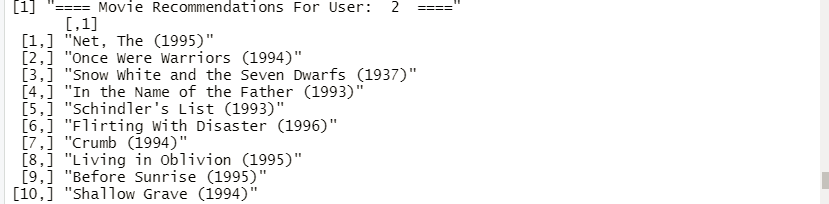
**Step-3**: We use a for-loop to get 10 movie recommendations for the first 5 users:



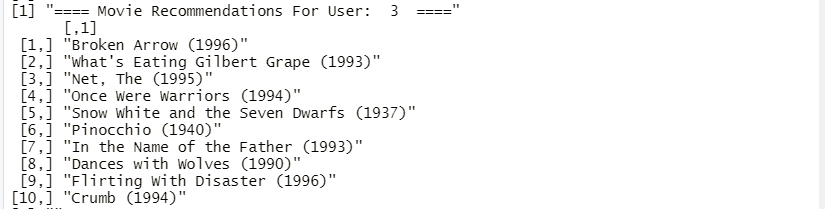
Movie recommendations for user # 1:



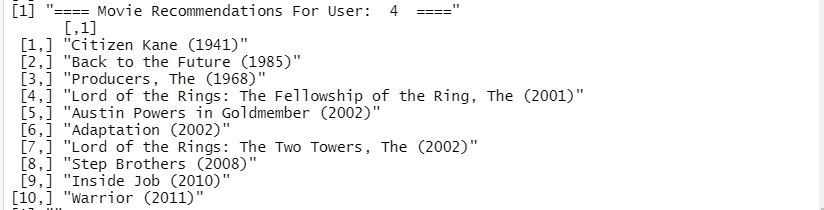
Movie recommendations for user # 2:



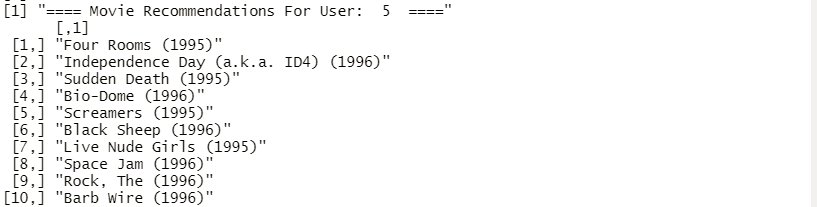
Movie recommendations for user # 3:



Movie recommendations for user # 4:

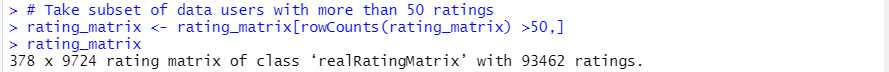


Movie recommendations for user # 5:



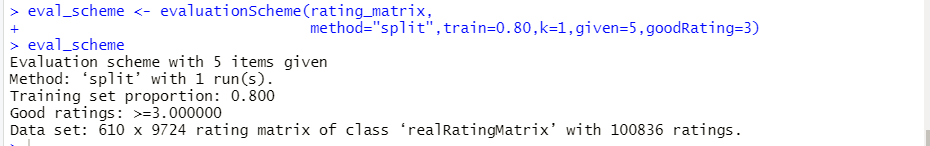
# **6.0 Evaluation**

**Step-1**: We take a subset of data with users having more than 50 ratings for evaluation. This subset of data has approximately 62% of users (378 or 610) and 93% of ratings (93,462 of 100,836).



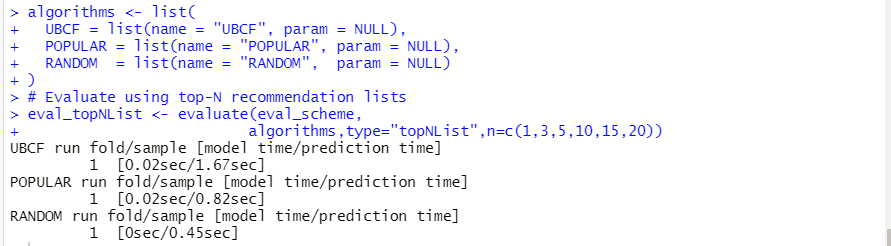
**Step-2**: We created an evaluation schema with the following parameters:

* method="split" ## use train/test split validation
* train=0.80 ## use 80/20 train/test split ratio
* k=1 ## with 1 run
* given=5 ## we use 5 times given schema
* goodRating=3 ## items with user rating >= 3 are considered positives

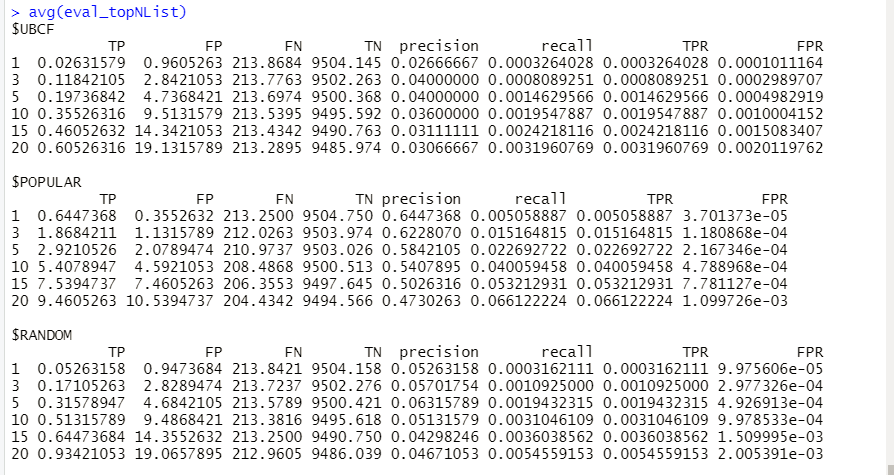


**Step-3**: We Call evaluate() with type = "topNList" to evaluate topNList recommendations:

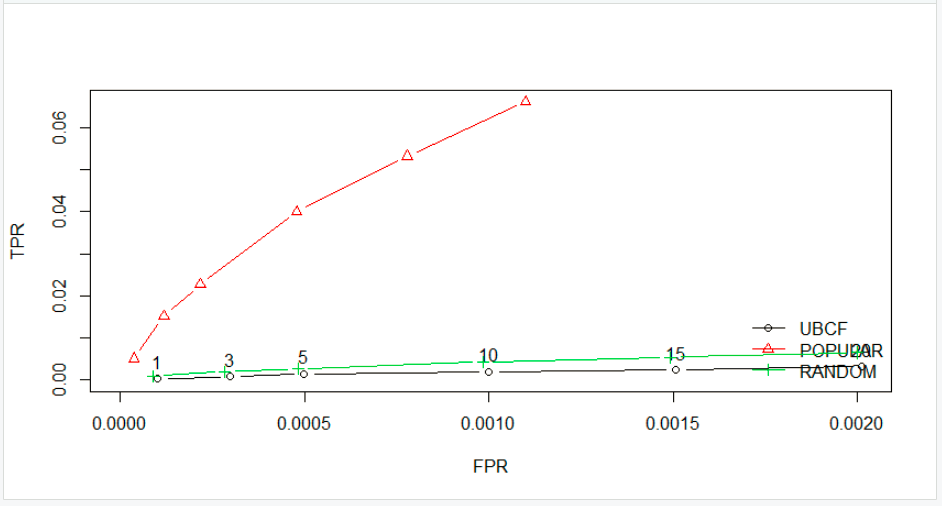
* UBCF (user-based collaborative filtering)
* POPULAR (based on item popularity)
* RANDOM (random recommendations)

****

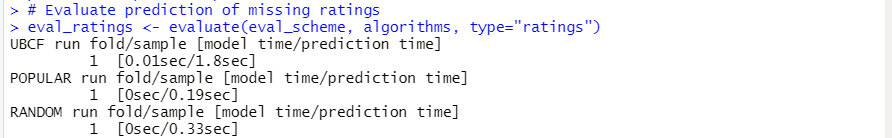
We print out evaluate results for topNList recommendations:

****

We plot the ROC curves for evaluate topNList recommendations:



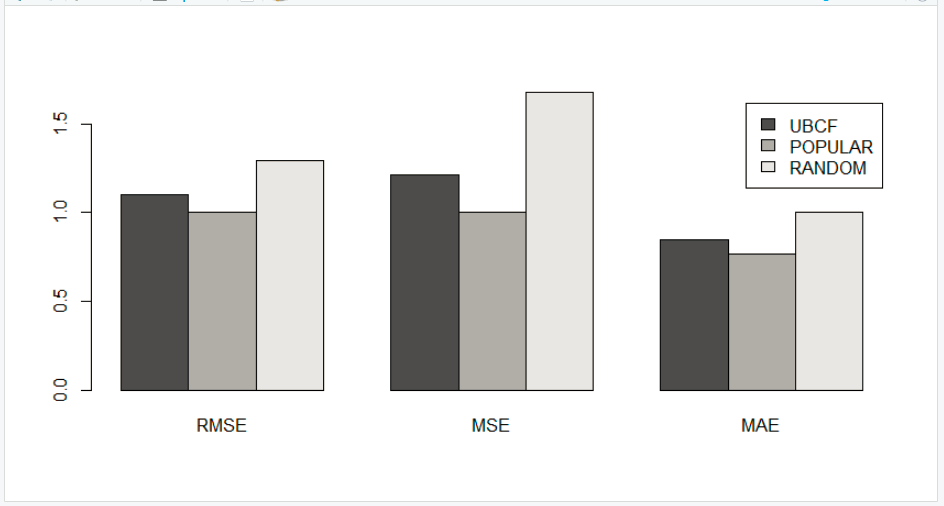
**Step-4**: Call evaluate() with type="ratings" to evaluate prediction of missing ratings:



We print out evaluate results for prediction of missing ratings:



We plot the charts for RMSE (Root-Mean-Square Deviation), MSE (Mean Squared Error) and MAE (Mean Absolute Error):



# **7.0 Deployment**

The deployment includes 6 steps from creating function, user interface, web server and connection to shiny app. Basically, a list of recommended movies will be presented on the web once the users type in the userID.

**Step-1**: To deploy the recommendation system in shiny app, a function will be created to receive input values from users.

In our case, users will type userID. Then the function will pick up userID, use it in the recommendation system and predict value. Next the predicted value will be translated into related movie titles and return the list of recommendation movies for specific users.

The code will be similar as following:

movie\_recommendation <- function(input,input2,input3) {

n\_users <- which(movies2[,2] == input)

for ( u in 1:n\_users)

{

# Read userid test data

#uid <- test[u,1]

uid <- u

recom <- predict(rec, rating\_matrix[uid], n=n\_movies) #Obtain top n recommendations for the user

recom\_list <- as(recom, "list") #Convert recommenderlab object to readable list

#Obtain recommendations from movies dataset

rec\_movies <- matrix(0,n\_movies)

for (i in 1:n\_movies){

rec\_movies[i] <- as.character(subset(movies, movies$movieId == as.integer(recom\_list[[1]][i]))$title)

}

print(paste("==== Movie Recommendations For User: ", uid, " ===="))

print(rec\_movies)

print("")

}

}

**Step-2**: The second step is the configuration of the user interface for receiving input values from users.

The sample code will be like below:

library(shiny)

shinyUI(fluidPage(

titlePanel("Movie Recommendation Engine"),

fluidRow(

column(5,

selectInput("select", label = h3("Choose Three Movies You Like"),

choices = as.character(movies2$title[1:1000])),

submitButton("Submit")

),

column(7,

h3("You Might Like These Too!"),

tableOutput("table"))

)

))

**Step-3**: The third step is the server.R component code. The input$select refers to the user ID that user typed in the user interface. recom() is the function that is made in the first step. After the server was set up, the r will auto pop out a screen or present a link like 127.0.0.1 that can be opened in the web explorer.

A sample code will be similar as following:

library(shiny)

library(proxy)

library(recommenderlab)

library(reshape2)

source("helpercode.R")

shinyServer(function(input, output) {

output$table <- renderTable({

recom(input$select)

})

}

)

**Step-4**: An shinyapps.io account will be created in this step.

**Step-5**: Installing devtools using code below. Then, authorizing shinyapps account, copying unique token from the token page in your account and running it in the Rstudio console. The app scripts and data files need to be in a new dedicated folder for the app. This can be ensured using getwd() command.

install.package(‘devtools’)

devtools::install\_github(‘rstudio/shinyapps’)

**Step-6**: Deployment of the app.

library(shinyapps)

# **8.0 Discussion**

The model and deployment need to be further improved and more function be provided to users.

The dataset was too big that further improvement of the model could be done by searching for package chopping it into several parts for analysis and intake of modeling. More modeling techniques will be employed to further improve users’ experience.

The code of the application can be found in GitHub at <https://github.com/skvuong/movieRec>

# **9.0** **Reference/Citation**

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. <https://doi.org/10.1145/2827872>